Evolving Variability Tolerant Logic

James Alan Hilder

Degree of Ph.D.

September, 2010

Intelligent Systems Group,
Department of Electronics
THE UNIVERSITY *of York*

Abstract

Intrinsic variability occurs between individual MOSFET transistors caused by atomic-scale differences in the construction of devices. The impact of this variability will become a major issue in future circuit design as the devices scale below 50nm. In this thesis, the background to the causes and effects of intrinsic variability, in particular that of random dopant placement and line-edge roughness, is discuss. A system is developed which uses a genetic algorithm to attempt to optimise the dimensions of transistors within standard-cell libraries, with the aim of improving performance and reducing the impact of intrinsic variability in terms of the effect on circuit delay and power consumption. The genetic algorithm uses a multi-objective fitness function to allow a number of circuit characteristics to be considered in the evolution process.

The system is tested using different standard-cell libraries from open-source and commercial providers, with developments and alterations to the system that have been made throughout the course of the experiments discussed. Comparisons of the performance with other optimisation techniques, hill climbing and simulated-annealing, are discussed. The optimisation process concludes with the use of e-Science techniques to allow for detailed statistical analysis of the evolved designs on high-performance computing clusters. The observed results for two-input logic gates demonstrate that the technique can be effective in the reduction of statistical spread in the delay and power consumption of circuits subject to intrinsic variability.

The thesis finishes with the investigation of larger circuits which are assembled from the optimised cells. A proposed design methodology is introduced, in which the processes of logic design are broken into small blocks, each of which uses techniques from evolutionary computation to improve performance. This includes an investigation into the application of a multi-objective fitness function to improve the performance of logic circuits evolved using Cartesian Genetic Programming, which produces designs for logic multiplier and display driver circuits which are competitive with human-produced designs and other evolved designs. These designs are assessed for their variability tolerance, with the multiplier circuit demonstrating an improvement in delay variability.

Contents

List of Figures

List of Tables

Declaration

The work discussed in this thesis has been developed with Dr. James A. Walker, as part of the nano-CMOS project. Except where explicitly stated otherwise, the bulk of the work discussed in the experiments is setup and developed by myself, with the assistance of Dr. Walker. The development of the MOTIVATED system, described in Chapter 4, has been equally split between the myself and Dr. Walker. Many of the results discussed have previous been described in the published works, detailed below in chronological order. The first-named author for all these papers can be considered the person who conducted the majority of the work in constructing the experiments and writing the papers. All the papers listed have been the subject of a peer-review process, with the exception of the first conference paper which was accepted as a late-breaking paper.

Conference Papers

- James A. Hilder and Andy M. Tyrrell. *An evolutionary platform for developing next-generation electronic circuits*. In proceedings of Genetic and Evolutionary Computation Conference (GECCO), Late-breaking papers. *London, UK, 7–11 July 2007.* Pages 2483–2488. ACM.

- James Alfred Walker, James A. Hilder and Andy M. Tyrrell. *Evolving variability-tolerant CMOS designs*. In proceedings of the 8th International Conference on Evolvable Systems (ICES): From Biology to Hardware. *Prague, Czech Republic, 21–24 September 2008.* Pages 308–319. Berlin, Germany: Springer.

- James A. Hilder, James Alfred Walker and Andy M. Tyrrell. *Optimisation of variability tolerant logic cells using multiple voltage supplies*. In proceedings of the IEEE Workshop on Evolvable and Adaptive Hardware (WEAH), part of the IEEE Symposium Series on Computational Intelligence (SSCI). *Nashville, TN, USA, 30 March–2 April 2009.* Pages 17–24. IEEE Press.

- James A. Hilder, James Alfred Walker and Andy M. Tyrrell. *Optimising variability tolerant standard cell libraries*. In proceedings of the 11th Congress on Evolutionary Computation (CEC). *Trondheim, Norway, 18–21 May 2009.* Pages 2273–2280. Piscataway, NJ: IEEE.

- James Alfred Walker, James A. Hilder and Andy M. Tyrrell. *Towards evolving industry-feasible intrinsic variability tolerant CMOS designs*. In proceedings of the 11th Congress on Evolutionary Computation (CEC). *Trondheim, Norway, 18–21 May 2009.* Pages 1591–1598. Piscataway, NJ: IEEE.

- James A. Hilder, James Alfred Walker and Andy M. Tyrrell. *Designing variability tolerant logic using evolutionary algorithms*. In proceedings of the 5th International Conference on Ph.D Research in Microelectronics & Electronics (PRIME). *Cork, Ireland, 12–17 July 2009.* Pages 184–187. IEEE Press.

- James A. Hilder, James Alfred Walker and Andy M. Tyrrell. *Use of a multi-objective fitness function to improve cartesian genetic programming circuits*. In proceedings of NASA/ESA Conference on Adaptive Hardware and Systems (AHS). *Anaheim, CA, USA, 15–18 June 2010.* Pages 179–185. IEEE.

- James Alfred Walker, James A. Hilder and Andy M. Tyrrell. *Measuring the Performance and Intrinsic Variability of Evolved Circuits*. In proceedings of the 9th International Conference on Evolvable Systems (ICES): From Biology to Hardware. *York, UK, 6–8 September 2010.* Pages 1–12. Berlin, Germany: Springer.

Journal Papers

- James Alfred Walker, Richard Sinnott, Gordon Stewart, James A. Hilder and Andy M. Tyrrell. *Optimizing electronic standard cell libraries for variability tolerance through the nano-CMOS grid*. Philosophical Transactions of The Royal Society A: Mathematical, Physical & Engineering Sciences. 2010. Vol 368. Pages 3967–3981.

- James Alfred Walker, James A. Hilder, Dave Reid, Asen Asenov, Scott Roy, Campbell Millar and Andy M. Tyrrell. *The Evolution of Standard Cell Libraries for Future Technology Nodes*. Accepted for publication in Genetic Programming and Evolvable Machines.

Hypothesis

The impact of intrinsic-variability within standard-cell designs can be diminished through the optimisation of transistor dimensions using a Multiple-Objective Evolutionary Algorithm. This method can then be combined with other established evolutionary techniques, to create a complete design process for intrinsic-variability tolerant logic circuits, without the need for human designs and intervention.

Acknowledgments

There are many people whom I wish to thank for their knowledge, support and encouragement throughout my study for this work. Firstly, I must thank my colleague and mentor Dr. James A. Walker, without whom I sincerely doubt that the results presented herein would have been possible - and I also give the greatest of thanks to my supervisor, Prof. Andy Tyrrell, for all the ideas, support and experience he has offered throughout the last four years. I must also mention Prof. Jon Timmis, Mr. Tim Clarke and Dr. Stephen Smith, for their knowledge, support and understanding both before, during and after I began this work.

I would then like to mention everybody else on the nano-CMOS project, and everybody in the Intelligent Systems Group at York, all people I consider friends, with whom it has been a priviledge to have worked with throughout this project. I thank Mr. Thom Blake, Mr. Geoff Burgess and Mr. Gerrard Huck, for being supportive and inspirational housemates with whom I have had the pleasure of living during the last four years. My family, especially my mum, dad and brother, who have always been there when needed, and all my friends. Finally, saving the best till last, I thank Beth, for keeping me fed, watered and sane whilst writing this, without your support I could not have finished this.

Chapter 1

Introduction

1.1 Background

The semiconductor industry is a huge global industry, with annual sales now in excess of \$250 billion, built on principles of ever increasing power, capacity and speed [6]. For semiconductor companies to produce more powerful microprocessors and more capacious memory chips, it is necessary to periodically shrink the size of components that make up these circuits - the MOS transistors that are assembled in pairs to form the building block logic and memory cells. For many years this reduction in the size of transistors, named device scaling, happened with relative ease, with many technological breakthroughs in the lithographic and fabrication processes that were needed to create smaller devices. Individual transistors have shrunk from the 10-μm devices that populated Intel's 4004 microprocessor to 32nm in the cutting-edge devices of today [1]. Given the atomic radius of a silicon atom is 0.11nm, it is clear that transistors are now approaching a size where they can be considered in terms of their atomic volume instead of traditional dimensions.

In the construction of a transistor, dopant atoms are implanted into the silicon lattice. Whilst the processes are tightly controlled, it is not possible to perfectly control the exact quantity and resting position of these dopants. As a consequence, no two transistors will be exactly the same, which results in differences in the electronic characteristics of each device. Whilst historically these differences have been small enough to be ignored, in recent years the extent of the variations and their random nature has grown to a point whereby transistor variability is now one of the major limiting factors that is curtailing the progression of scaling. Conventional methods to simulate transistors and circuits, and work-flows for creating designs,

[1]The figures actually represent the expected DRAM half-pitch - half the distance between cells in a block of dynamic RAM - rather than minimum transistor channel length, which is generally slightly shorter.

do not cater for the effects of variability; within a few years transistors will be scaled to a point where device failure rates and impaired performance will demand drastically different techniques in the design on circuits. Variability can be modelled, but it requires vast three dimensional statistical simulations that consider the electronic flow at an atomic level, which consumes colossal computational resources.

In this thesis, attempts to develop methods which will reduce the impact of this unavoidable variability are made. A system to attempt to optimise circuits through automatic adjustment of transistor dimensions within a conventional software simulation package is introduced. The optimisation system is based on ideas that come from the world of evolutionary computation, in which algorithms inspired by Darwinian models of evolution are used to accelerate the search process.

1.2 The Nano-CMOS Project

The research described in this thesis has formed part of a four-year EPSRC funded project entitled *"Meeting the design challenges of nano-CMOS Electronics"*, which is herein known as the nano-CMOS project. The project includes research groups from Glasgow, Manchester, Edinburgh and Southampton Universities and also the National E-Science Centre in Edinburgh, in addition to here at the Intelligent Systems Group, within the Department of Electronics at the University of York. The project secured four years of funding from the EPSRC, beginning in October 2006, with substantial further backing from several key industrial partners including Fujitsu, ARM, Wolfson Microelectronics, Freescale, National Semiconductor and Synopsys. The overall goal of the research project is to design accurate models for the simulation of the next generations of MOS transistors devices, focussing primarily on devices in the 45nm - 18nm range. Statistically-based modelling techniques are necessary in the extraction and evaluation of device parameters, as individual transistor elements contain inherent stochastic parameter fluctuations which cannot be removed by refining the manufacturing process.

Device variability is caused by both unavoidable intrinsic parameter fluctuations and also microscopic differences which can occur at many different stages within the silicon manufacturing process. Accounting for this device variability, through the use of new tools and techniques within the design tool-chains, will add a significant complexity to the overall design process, and will require the coordination of many teams of device experts and the tools they use. A detailed overview of the background to device fluctuations, their causes and the methods used to model them, along with suggested methods of overcoming their implications, is covered in Chapter 2.

The nano-CMOS project is led by the Device Modelling Group within the Electronics Department at the University of Glasgow, where sets of statistical models which include intrinsic parameter fluctuations have been developed for devices based on 45nm, 35nm and 18nm technology nodes. The simulations have used both in-house and commercial tools. Libraries of transistor compact models, which can be used in traditional software simulation tools such as SPICE, have been extracted from the analysis of the simulation results. The Microsystems Technology Group at the University Glasgow have worked to develop integrated circuit & device simulators and circuit level compact models using these results. A team at the Electronic Systems Design Group at the University of Southampton have attempted to incorporate the variability statistics higher up the design tool chain by means of extracting behavioural models for standard cell libraries, and a team from the Mixed-Mode Design Group at the University of Edinburgh are developing techniques for using the models to perform noise simulation and circuit level. The variability-aware compact models have been made available on a computing-grid to the Advance Processor Technologies Group at the University of Manchester, who are investigating the impact of device variability on existing design styles (including a large-scale multiprocessor architecture) [21].

A central focus of the whole project is the close involvement with the e-Science community and the use of cluster-based and Grid computing resources. Grid computing is a special form of parallel computation technology which provides the framework to share processing load across many computers, often distributed at different locations, which are networked together. This allows distributed groups of users to collaborate by sharing not just computing resources, but also designs, work-flows and data-sets, allowing new ways of performing collaborative research. At the heart of the nano-CMOS project is the development of the prototype of a '*nano-CMOS Design Grid*', in which each project partner has attempted to grid-enable their code and work flows to allow for the secure and efficient interactions between sites necessary for the progress of the overall project. This has been achieved through the nano-CMOS portal[2] which is the web-based hub for initiating simulations and transferring results, and the use of distributed file-servers based on the Andrew File System (AFS).

1.3 Objectives

The project proposal expressed the ISG's intended involvement in the project by suggesting three key questions to be answered, all relating to methods in which evolutionary algorithms and associated ideas could be used to assist or improve the modelling of next-generation de-

[2]Accessed through the secure server at `https://www.nanocmos.ac.uk/`.

vices, be it within the creation of the models, or in using the models to create improved performance of devices and circuits.

- How can evolutionary techniques be used to limit the effects of parameter variations?

- How can evolutionary techniques be used within device models to alleviate parameter fluctuation problems?

- How can parameter variation datasets be best used by evolutionary techniques to improve system performance?

Over the course of the four year project, the research at York has primarily focussed on addressing the first question above, although all three questions have been addressed in some capacity, as is discussed in the conclusions of this thesis. Two separate routes have been investigated in an attempt to find solutions to the first question. Dr. James Walker has focussed on investigating if the use of evolutionary techniques, notably a reworking of Cartesian Genetic Programming, could be used to design novel topologies of logic cells, which offered improved performance over conventional designs when subjected to the effects of transistor variability. I have attempted to reduce the impact of variability through the optimisation of transistor dimension within conventional and evolved standard cells through the use of a Genetic Algorithm. This has been followed with an investigation to see if other established evolutionary techniques can be combined in a design flow to create completely novel variability tolerant circuits. As such, the following statements can be considered the hypothesis which the research presented in thesis attempts to prove:

The impact of intrinsic-variability within standard-cell designs can be diminished through the optimisation of transistor dimensions using a Multiple-Objective Evolutionary Algorithm. This method can then be combined with other established evolutionary techniques, to create a complete design process for intrinsic-variability tolerant logic circuits, without the need for human designs and intervention.

1.4 Structure of Thesis

The thesis is organised as follows. Chapter 2 provides a detailed overview of the problem of transistor variability. This commences with a description of the MOS transistor and its physical operation, including the processes involved in creating large scale integrated circuits. An analysis of the different causes and consequences of device variability is then given, focussed on the stochastic variations caused by atomic level differences. This is followed by descriptions of the systems used to model variability and simulate transistor-based circuits, emphasising the tools that are used in this thesis. The chapter concludes with a look a the consequences of variability on logic design and what potential solutions exist to surmount the issues.

Chapter 3 gives an outline of the field of evolutionary computation, looking at the history and operations of the main types of evolutionary algorithms, including a detailed analysis of multi-objective optimisation techniques which are used in this research. A summary of some of the major developments of evolutionary techniques applied to field of electronic engineering is given at the conclusion of the chapter.

Chapter 4 introduces the framework of tools that have been developed to combine the variability-aware compact models, described in Chapter 2, with the multi-objective evolutionary algorithms discussed in Chapter 3. This includes a detailed description of the Simple Genetic Algorithm, which is used to optimise transistor dimensions within standard cell libraries.

The application of the algorithm is covered in Chapter 5, which discussed the results obtained when applying the algorithm to different sets of publicised libraries of cells. The developments made to the system throughout the course of the experiments are discussed, with comparisons to other optmisation techniques given.

Chapter 6 proposes the development of evolution inspired algorithms further into the logic design framework, by means of combining the optimisation techniques discussed in Chapters 4 & 5 with a system for creating logic topologies at a block-level. This aims to create the beginning seed of a system by which large logic designs can be completely evolved and optimised without the need for traditional design methods. Concluding remarks and observations are given in Chapter 7.

Chapter 2

Transistor Variability

2.1 Introduction

Complimentary Metal-Oxide Semiconductor (CMOS) form the backbone of almost all modern digital circuits. The first example of integrating multiple components on a monolith of semiconductor was demonstrated by Jack Kilby in 1958 [83]. The first CMOS based devices were assembled by Fairchild Semiconductor in 1963 [174], with the first commercial CMOS integrated circuits, the 'CD4000' series, released by RCA in 1968 [107]. The vast majority of very large scale integration (VLSI) [1] circuits are assembled from CMOS-based logic, including microprocessors, programmable hardware such as FPGAs, memory and many other types of circuits. Designs are build using complementary, symmetrical pairs of p-type and n-type MOSFETs arranged to create logic and memory functions.

The semiconductor industry itself is built upon the principles of CMOS devices increasing in complexity and performance with each product generation. The physical sizes of die (the functional block of the integrated circuit as fabricated on silicon) are fundamentally limited by the physical properties of the silicon wafer and economic constraints; thus in order to achieve increasing complexity, the dimensions of individual transistors have to periodically shrink. Transistor have now shrunk to a point where their construction can now be considered in terms of atoms instead of conventional dimensions, however it is not possible to guarantee the atomic layout and structure of any two transistors will be identical. The atomic variations between transistors results in significant differences in their operating characteristics, which creates new problems in the design, simulation and operation of large circuits which are not

[1]VLSI is herein used to describe all integrated circuits with upwards of one hundred thousand devices. This includes ultra large scale integration (ULSI), with tens of millions of devices per chip, and gigascale integration (GSI), with billions of devices per chip [113].

currently solvable using conventional design techniques.

In this chapter the causes, effects and potential solutions to the problem of atomic-level variability in transistors are considered. The chapter begins with a background overview of the MOSFET transistor, the key building block in created large scale integrated circuits. The structure and basic operation of the MOSFET are reviewed based on the traditional circuit models and theory. The process of utilising MOSFETs to create integrated circuits is then discussed, with the basic building block cells required to make CMOS logic gates and an overview of the steps in the manufacturing process of a CMOS integrated circuit, from wafer through to packaged device. A review of the history and proposed future of transistor scaling is then discussed, followed by a detailed analysis of the different types of transistor variability which occur in modern and future devices, focussed on the atomistic variabilities on which the work in this thesis is based. The following section is an overview of the techniques used to simulate circuits incorporating the effects of variability. The chapter concludes with a synopsis of the specific problem areas of device variability in modern logic and potential solutions to overcome the issues.

2.2 The Metal-Oxide Semiconductor Field-Effect Transistor

The Metal-Oxide Semiconductor Field-Effect Transistor (MOSFET) is a four-terminal device, available with either an n-type or a p-type channel. It is often used in pairs to form digital operations, and it is also used extensively in modern analogue circuits as current sources, voltage-controlled resistors and for power switching. Unlike a bipolar junction transistor (BJT), in which the output current is controlled by the input current at the base terminal, the FET controls output current based on the input voltage at the gate terminal. Field-effect transistors exist as both Junction FET (JFET) and Insulated-Gate FET (IGFET); the MOSFET is a type of IGFET where the metal gate electrode is electrically insulated from the substrate via a thin layer of silicon oxide.

The schematic symbols for the main types of MOSFET transistors, along with their association voltage and current definitions, is illustrated in Figure 2.1. The substrate connection in a FET, often labelled as *B* or *bulk*, in most often connected to the source; in many devices this connection is made internally, hence both the schematic and packaged device are often seen with only three terminals.

In a MOSFET transistor, the drain current I_D is controlled by the gate-source junction voltage V_{GS}. This relationship gives the transconductance, g_m, using the following formula:

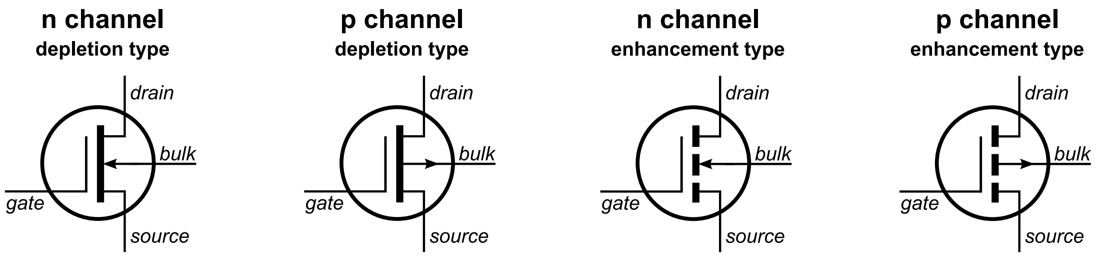


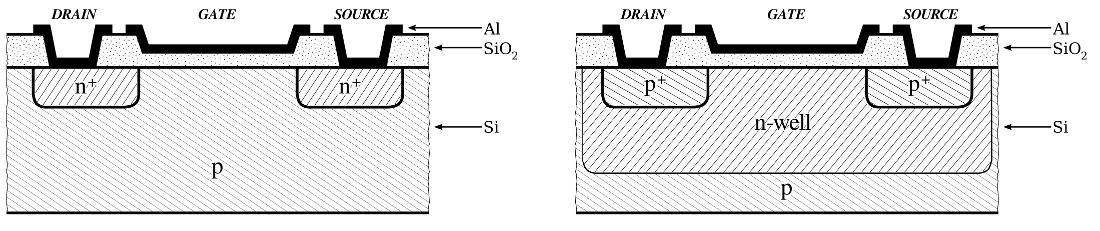
Figure 2.1: Four-terminal schematic circuit symbols for enhancement and depletion mode MOSFET transistors

$$g_m = \frac{\Delta I_D}{\Delta V_{GS}} \tag{2.1}$$

Transconductance g_m is measured in *mho* or *Siemens*, although with I_D typically being measured in the range of μA or mA, it is more common to find values represented in terms of millisiemens (mS). MOSFETs are available which cover a vast range of power handlings, with large discrete devices able to sink several hundred Watts, and the very low gate-leakage makes them suitable for numerous applications [79].

2.2.1 Structure of Field-Effect Transistors

The structure of a typical NMOS and PMOS transistor illustrated in Figure 2.2. In a fabricated CMOS circuit, a lightly doped p^- silicon is used as the device substrate, allowing a n-channel device to be directly formed with two heavily doped n^+ regions; a p-channel device requires the creation of an additional lightly doped n^- well, as can be seen in Figure 2.2b. The heavily doped regions form the Source and Drain connections, with the gate electrode separated by a thin layer of oxide.



(a) NMOS Transistor (b) PMOS Transistor

Figure 2.2: Physical cross-sectional structure of a NMOS and PMOS transistor, based on illustrations from [154, 160]

Early MOS transistors were built with a metal gate electrode (typically aluminium) on top of an oxide insulating layer on top of a semiconducting substrate. More recent designs replaced the metal gate electrode with one made from a polycrystalline silicon material which can tolerate the high temperatures in the annealing process better (although the name MOS has been retained). Some of the most recent design have an insulating layer with a high dielectric constant (known as high-κ models) have reverted back to using a metal layer as the gate electrode. To improve the performance in terms of speed and power consumption, the scaling of individual components is reduced allowing it to operate at a faster speed or with lower power wastage.

2.2.2 Function of Field-Effect Transistors

The basic function of a MOS transistor can be evaluated by considering the behaviour of a n-MOS device under different conditions. With all terminals grounded, the source and drain electrodes are separated by a pair of back-to-back P-N junctions, resulting in a very high resistance; the gate and substrate form a capacitor with the SiO_2 acting as a dielectric. If a negative voltage is applied to the gate, negative charges will accumulate in the poly-silicon gate and increased positive charges will be attracted to the channel. If either the source or drain are then given a positive-bias, only a negligible current will flow between them; this is the leakage current of the device.

When a positive potential is applied between the gate and source (V_{GS}), holes are repelled away from the substrate-oxide interface which results in a depletion region. An increase in the gate voltage results in negative charges being attracted to the channel; once enough exist the channel will change from being a p^- region to a n region and is said to be inverted. The gate-source voltage at which this inversion takes place (i.e. when the concentrations of electrons under the gate equals the concentration of holes deep in the p^- substrate is the transistor threshold voltage, V_T. In the simplest approximation, when V_{GS} is below V_T no current flows between source and drain - the device is in its cut-off region. If V_{GS} is above V_T, the channel electrically joins source and drain and current can flow freely [15].

Clearly this is an over-simplification; in reality when V_{GS} is close to V_T a gradual change of charge occurs in the channel, resulting in small current flow from source to drain; the device is considered to be in ‘weak inversion’ and working in the sub-threshold region.

MOSFET as a capacitor

In a condition where the channel is present, the accumulated negative charge is proportional to V_{GS} and is dependant on the oxide-thickness t_{ox} with the transistor operating as a capacitor. The gate capacitance per unit area is defined as follows:

$$C_{ox} = \frac{\epsilon_{ox}\epsilon_0}{t_{ox}} \tag{2.2}$$

in which ϵ_0 is the permittivity of free space, $\approx\ 8.854 \times 10^{-12}$ Fm^{-1}, and ϵ_{ox} is the relative permittivity of SiO_2, ≈ 3.9. The total capacitance of the device can be calculated by multiplying equation 2.2 by the device area:

$$C_{gs} = WLC_{ox} \tag{2.3}$$

This allows the charge in the device channel to be calculated as follows:

$$Q_T = C_{gs}\left(V_{GS} - V_T\right) \tag{2.4}$$

MOSFET as a resistor

An increase in V_D allows a current to flow between drain and source through the channel. When V_D is of a small value well below V_T, the charge-density in the channel will not increase significantly. The device effectively operates as a resistor of length L and width W, which relates the drain-source current I_{DS} and voltage V_{DS} based on the gate voltage as follows:

$$I_{DS} = \mu_n C_{ox}\left(V_{GS} - V_T\right)\frac{W}{L}V_{DS} \tag{2.5}$$

MOSFET in linear region

As V_D increase, the charge density profile across the channel changes. It is generally assumed to be a linear gradient between source and drain, which results in a revised equation to relate I_{DS} to V_{DS}; the current follows a linear relation to V_{GS} and a quadratic relation to V_{DS} [15]:

$$I_{DS} = \mu_n C_{ox}\left(V_{GS} - V_T - \frac{V_{DS}}{2}\right)\frac{W}{L}V_{DS} \tag{2.6}$$

MOSFET in saturation region

Once V_D is increased the point where the drain-gate voltage V_DG equals threshold voltage V_T, the charge density in the channel adjacent to the drain becomes zero and consequentially the current I_D approaches its maximum value; the transistor is said to be in saturation region. The charge concentration in the channel can be approximated as being constant beyond this point, which allows a new current expression to be derived. Given that $V_{DG} \geq V_T$ and $V_{DS} = V_{GS} + V_{DG}$, we can define a condition $V_{DS_{sat}} = V_{GS} - V_T$. Replacing V_{DS} with $V_{DS_{sat}}$ in equation 2.6 yields the saturation region current equation:

$$I_{DS} = \mu_n C_{ox} \left(V_{GS} - V_T\right)^2 \frac{W}{2L} \tag{2.7}$$

In reality a further increase in V_D results in an effect named channel length modulation, in which the channel length is decreased and the pinch-off region is increase. To account for this effect, a revised saturation region current equation incorporates a modulation factor λ which is inversely proportional to channel length L, as shown:

$$I_{DS} = \mu_n C_{ox} \left(V_{GS} - V_T\right)^2 \frac{W}{2L} \left[1 + \lambda \left(V_{DS} - V_{DS_{sat}}\right)\right] \tag{2.8}$$

When V_{DS} increases well beyond $V_{DS_{sat}}$ the device can enter a condition known as breakdown, in which current can increase dramatically with V_{DS}. The causes of device breakdown differ between short and long channel devices; in the former it is a breakdown of the parasitic bipolar transistor formed whilst in the latter it is a breakdown of the drain-body p-n junction [42].

Body-Effect

All the previous equations have assumed the substrate connection is connected to the source; this is certainly the most common form of operation, but is not a prerequisite condition. Where the source and substrate are at different potentials, a number of second-order effects exist which are named '*body effect*'; this can be modelled as an increase in threshold voltage, as shown in the following formula derived by Gray and Meyer, in which V_{SB} is the source-substrate potential and $V_{T,n0}$ is the threshold voltage with zero V_{SB} [69].

$$V_{T,n} = V_{T,n0} + \gamma \left(\sqrt{V_{SB} + 2|\phi_F|} - \sqrt{2|\phi_F|}\right) \tag{2.9}$$

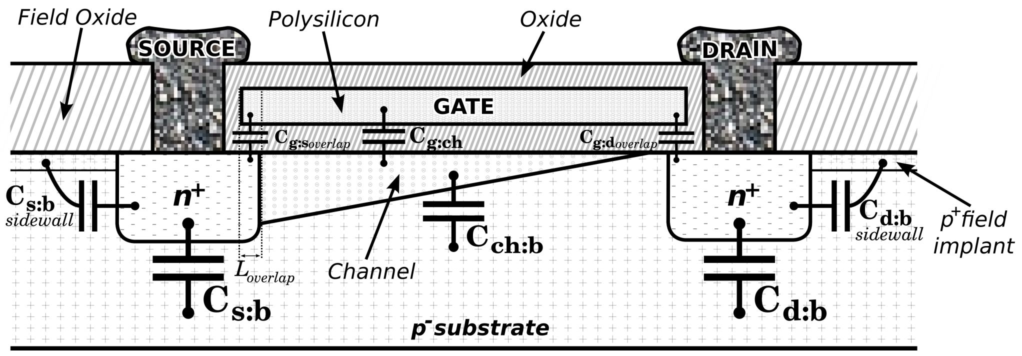


Figure 2.3: Cross-sectional structure of a NMOS transistor in saturation mode highlighting the parasitic capacitances [160, 15]

Parasitic Effects in a MOS Transistor

A MOS transistor in the saturation region includes several parasitic capacitive effects, which are illustrated in the cross-sectional model shown in Figure 2.3. The most significant effect occurs from the capacitors formed between the gate and source of the device; these are depicted in the diagram as $C_{g:ch}$ and $C_{g:s_{overlap}}$. The capacitor formed between the gate poly-silicon and the channel can be approximated to a linear capacitor dependant on the device area and oxide thickness, which roughly equates to: [136]

$$C_{g:ch} \approx \frac{2}{3} W L C_{ox} \tag{2.10}$$

There is an additional contribution from the capacitance that occurs in the narrow overlap between the gate and the n^+ doped source region, which results from horizontal spreading of the dopant during the fabrication process. Considering the length of the overlap to be $L_{overlap}$ the resulting parasitic capacitance is equal to:

$$C_{g:s_{overlap}} = W L_{overlap} C_{ox} \tag{2.11}$$

Equations 2.10 and 2.11 can be combined to give a gate-source capacitance $C_{g:s}$ as follows:

$$C_{g:s} \approx W \left(\frac{2}{3} L + L_{overlap} \right) C_{ox} \tag{2.12}$$

A similar overlap exists between the gate and drain regions resulting in a gate-drain capacitance:

$$C_{g:d} = C_{g:d_{overlap}} = WL_{overlap}C_{ox} \tag{2.13}$$

The other parasitic capacitances occur between at the interfaces between the source, channel and drain regions with the substrate and field-implant regions. The most significant are $C_{s:b}$, and $C_{d:b}$, formed by the depletion regions which occur at the reverse-biased p-n junctions at the source and drain. If L_{eff} is the defined as the length of the n$^+$ source and drain regions, the capacitances can be calculated as follows [160]:

$$C_{s:b,d:b} = \frac{WL_{eff}}{(1+V_{D,S}/\Phi_b)^{MJ}} \times C_J \tag{2.14}$$

where Φ_b is the working potential of the bulk, and MJ and CJ are process-dependant constants. The capacitances between the source and drain regions with the field-implant regions are considered to be *side-wall effects* and can be calculated as follows:

$$C_{s:b_{sidewall},d:b_{sidewall}} = \frac{W+2L_{eff}}{(1+V_{D,S}/\Phi_b)^{MJ_{sidewall}}} \times C_{J_{sidewall}} \tag{2.15}$$

In addition to the parasitic capacitances, there are additional resistances and leakage currents which must be considered. There are ohmic resistances at the source and drain connections, typically of the order of 10Ω for VLSI CMOS, which become significant in high-drain current applications. Leakage currents exist between the source & drain regions and the substrate bulk, which can be approximated by reverse-biased diodes using the Shockley ideal diode equation [160]:

$$I_{BD,BS} = I_s \left[e^{\left(\frac{qV_{BD,BS}}{kT}\right)} - 1 \right] \tag{2.16}$$

where q is the elementary charge, 1.602×10^{-19}C, k is the Boltzmann constant, 1.381×10^{-23}JK^{-1}, T is the temperature in Kelvins and I_s is the reverse saturation current of a P-N junction.

2.2.3 Structure of CMOS Integrated Circuits

Almost all modern VLSI (very large scale integration) circuits are assembled from CMOS based logic, including microprocessors, memory and many other integrated circuits. Most devices are built upon complementary, symmetrical pairs of p-type and n-type MOSFETs to create logic functions. CMOS is widely used in modern devices as it possesses low static power-supply drain and high noise immunity, with significant power only being drawn when the transistors are in a transitional state (switching between on and off states). This allows CMOS logic circuits to be built with far lower power consumption (and consequentially less heat production) than other traditional logic designs such as TTL.

Early CMOS devices were significantly slower than equivalent TTL (transistor-transistor logic) devices, although the low-power consumption led to their inclusion in early digital watches where battery life was more important than processing speed. For many years CMOS logic devices were built to be tolerant of a wide range of supply voltages, allowing them to interface easily with TTL with its industry-standard 5 volt power supply. By the early 1990's the desire to increase circuit density and speed led to a drop in CMOS supply voltages and the continued reduction of the geometric dimensions of devices. The most modern CMOS devices have been built to operate from supply voltages of 1 volt or less, with gate lengths for individual transistors now well under 100nm [154].

Much of the current research and development within the semiconductor industry is concerned with the development of System-on-a-Chip (SoC) technologies, which involves the integration of high-performance CMOS log and embedded memory (both SRAM or DRAM for convention, volatile random-access, alongside EEPROM or FLASH as long-term non-volatile storage), along with analogue CMOS components for base-band functions and RF-BiCMOS for mobile-radio and tuner functions. BiCMOS combines bipolar devices with CMOS devices, allowing for increased speed and drive currents in specific applications. To allow for both the high-integration density and high performance, it is necessary to continually scale down the size and operating voltages of the CMOS components.

2.2.4 Dielectric Materials

The traditional dielectric material used in the creating of CMOS devices has been silicon dioxide (SiO_2). As devices have become smaller, the thickness of this dielectric layer has steadily decreased, which naturally leads to an increase in gate capacitance; this has the effect of improving drive current and overall device performance. However, recent technology nodes have seen the width of the dielectric layer reduce to below 2nm, which has resulted in a dramatic increase in leakage current. This is due to quantum tunnelling, a quantum mechanical phenomenon observed where a particle passes through a barrier despite have a lower mechanical energy than the potential energy of the barrier [137]. The effect of this increased leakage current results in a increased overall power consumption of the device, and consequentially a reduced device reliability.

The gate-oxide layer of a MOSFET device can be modelled as a plate capacitor, the value of which is dependant on the area of the layer, the thickness of the layer, and the relative dielectric constant of the material κ. Ignoring any quantum mechanical and depletion effects, the capacitance of the layer is given in Equation 2.17, in which A denotes the capacitor area, t is the thickness of the oxide layer and ϵ_0 in the permittivity of free space ($8.854 \times 10^{-12} Fm^{-1}$).

$$C = \frac{\kappa \epsilon_0 A}{t} \tag{2.17}$$

To increase the gate capacitance, the options are either increasing the dielectric area, reducing the thickness of the dielectric, or increasing the dielectric constant of the material. Increasing the area negates advantages obtained in the shrinking of the technology; reducing the oxide-layer thickness results in unmanageable leakage current as described before. Therefore modern devices have turned to attempting to overcome these limitations with the use of a high-κ dielectric material. In such devices, the layer thickness is often increased over its SiO_2 equivalent as a means of reducing leakage current (and increasing device reliability) whilst maintaining a similar overall capacitance.

2.2.5 The CMOS Fabrication Process

In order to understand the causes and effects of transistor variability that occurs in CMOS, it is necessary to have a degree of understanding of the fabrication process that occurs. This section briefly describes the steps taken to convert raw materials in a packaged integrated circuit.

Growth of Silicon Ingot

The first stage in the fabrication process is the growth of a large, cylindrical ingot of single-crystal silicon, based on a method discovered by Jan Czochralski in 1916. High-purity silicon is melted in a quartz crucible in a controlled environment, with an argon atmosphere. A seed crystal is dipped into the molten silicon, and with a high degree of precision is slowly rotated and retracted, in a process commonly known as Czochralski pulling [43]. A tightly-controlled addition of dopant atoms, typically boron or phosphorus (although arsenic and antimony are also used) can be added to the molten silicon mix to create n-type or p-type doped silicon. Throughout the history of semiconductor fabrication, the ingots have periodically got wider and longer, with most major manufacturers currently producing 200-mm and 300-mm wide ingots, and 450-mm fabrication plants likely to appear around 2012 [2] [2]. A number of technical problems need to be overcome for each step in wafer-width, with 450-mm ingots set to be 3 times heavier than their 300-mm equivalents, approaching 1000kg per ingot, and twice as

[2]The 2009 ITRS roadmap has set the date for introduction of 450mm wafers between 2014 and 2016, noting the transition is critical to meet the demands of 30% cost reduction per die and 50% cycle time improvement [3]

long to grow and cool, resulting in significantly increased process times. Other techniques to Czochralski pulling are used for certain types of devices, such as the float-zone technique for power devices and the Bridgman-Stockbarger technique, used in gallium-arsenide and other non-silicon semiconductor growth [16].

The grown ingots, which can be in excess of 1 meter in length, are then sliced into wafers using multi-wire wafer saws or electrical-discharge machinery techniques [162]. The resultant wafers are generally between 275μm and 1000μm in thickness, with 300 mm ingots generally being sliced to 775μm thickness wafers, and forthcoming 450 mm ingots likely to be 925μm in thickness [2]. The wafers are polished to ensure an even width and cleaned with weak acid solutions to remove any unwanted particles and damage caused by the sawing process.

Oxidation

The first step in processing the cleaned wafer is that of the growth of the layer of SiO_2 on the surface; the oxide layer grows both into the silicon and as above the original surface, with approximately 44% of the total oxide-width formed below the surface. The oxide layer is most often accomplished through a process named thermal oxidation, in which controlled high temperatures (typically between $700\,^{\circ}$C and $1300\,^{\circ}$C) promote the growth of the SiO_2. This takes place in an oxidation furnace, in which the wafers are placed in quartz 'boats' with the oxidising agent diffusing over the surface of the wafer at atmospheric pressure. The oxidising agent is either O_2 in dry oxidation, or H_2O (steam) in wet oxidation; dry oxidation is preferred as it results in fewer defects [154, 1, 16].

Diffusion and Ion-Implantation

Diffusion and Ion-Implantation are the two processes in which dopant atoms are introduced into the silicon wafer; diffusion was used extensively in IC fabrication in the past but for high-density CMOS circuits ion-implantation is now more commonly used. In the diffusion process, impurity atoms which are at the surface of the wafer are moved into the bulk of the material under high temperatures, between $800\,^{\circ}$C and $1400\,^{\circ}$C, in a diffusion furnace.

In the ion-implantation process, ions of a particular dopant are accelerated by an electric field and physically lodge within the silicon substrate. The typical depth of penetration varies between 0.1μm and 0.6μm, dependant on the angle and velocity at which the ions strike the wafer. Ions are implanted off-axis from the wafer to ensure they experience collisions with lattice atoms, avoiding a 'channelling' of ions deep in the silicon. The process of ion-implantation causes damage to the crystal lattice of the silicon, which results in certain numbers of the implanted ions being electrically inactive. These inactive ions are recovered through the process of annealing, in which the wafer is heated to roughly $800\,^{\circ}\mathrm{C}$, which allows the implanted ions to move to electrically active locations within the lattice, although the annealing process can lead to some unintended diffusion of the dopants; this can be minimised through careful optimisation of the annealing time and temperature [112, 16].

Ion-implantation is generally preferable to diffusion in the creation of modern silicon integrated circuits as it allows a greater precision in the control of doping concentrations as the current of the ion-beam can be accurately measured during implantation; doping concentrations can be controlled to within $\pm 5\%$. It also has the advantage of being a room-temperature process (except for the annealing process), and the ability to implant through a very thin oxide layer; the diffusion process requires that the surface be free of SiO_2. The final advantage is a greater control over the depth-profile of implanted dopants; if needed, a concentration peak can be placed well below the surface of the silicon [154].

Deposition

At different stages in the fabrication process, thin films of dielectrics, semiconductors and metals have to be deposited on the silicon wafer. Several different techniques for deposition exist, including sputtering, evaporation and chemical-vapour deposition (CVD). The process of sputtering involves placing the wafer on an anode within a vacuum, with a cathode coated in the material to be deposited. Positive ions are bombarded against the cathode by means of a strong DC, radio-frequency or magnetic field, which results in the target material being dislodged onto the wafer through the process of direct momentum transfer. In the evaporation-deposition process, a solid material is heated within a vacuum until it evaporates. The evaporated molecules then strike the cooler surface of the wafer, condensing into a solid film on the surface [16].

The CVD process operates by way of reacting a silicon-rich gas such as SiH_4 with an oxygen-rich precursor which results in a chemical reaction, depositing SiO_2 on the substrate. The process can be occur at low-pressure in LPCVD, or at high-pressure in plasma-enhanced (PECVD) systems. Unlike the thermal oxidation process, CVD can occur at relatively low temperatures and does not consume any of the silicon from the surface of the wafer; the process of CVD can also be used to deposit other dielectric materials on the surface, including the polycrystalline silicon frequently used in modern devices and silicon nitride (Si_3N_4) [154].

Photolithography

Photolithography is the process that creates the complex patterns used to isolate the positions of different device areas on the completed silicon wafer; it refers to the complete process of taking a circuit image, invariably created on a computer, to creating a *photomask*, and transferring it to the wafer. The first stage of the process is the creation of a transparent quartz plate which contains the required pattern, called a *reticle*. The reticle is coated with a UV-light absorbing layer, typically iron-oxide, then further coated with a thin layer of electron-beam resist - an organic polymer which chemically changes when exposed to energetic particles. A computer controlled electron-beam exposes the resist layer selectively based on the desired patterns. The reticle is then developed in a chemical solution which removes the unwanted areas. Two main types of photoresist exist: positive-resist, in which exposed areas are removed, and negative-resist in which unexposed areas are removed. A plasma then etches off the uncoated areas of iron-oxide, leaving the patterned reticle. Many different reticles, often over a dozen, are needed throughout the fabrication process, each corresponding to the patterns required at different process steps. The reticle will typically contain the pattern for a single die as opposed to the entire wafer, and is scaled many times larger than the desired pattern to be created, being reduced and repeated several times to create the complete image across the wafer.

To transfer the image from the reticle onto the silicon wafer, the wafer is first coated with a thin (0.5μm) uniform coating of a UV-sensitive photoresist, deposited by spinning the wafer rapidly at $\approx$3000rpm. An ultra-violet lightsource is focussed through the reticle and a reduction lens onto the wafer, causing exposed regions to become acidified. Different photoresists and lightsources determine the minimum feature size that can be created; these are discussed in greater detail in Sections 2.3.3 and 2.5.2. The exposed wafers are developed in a Sodium Hydroxide solution, which etches the exposed photoresist away, then cured by baking at roughly $125\,^{\circ}\mathrm{C}$ to harden. Complete wafers are exposed in a die-by-die basis using a step-and-repeat process, which utilises laser controlled *mask aligners* featuring a very high degree of precision. The complete process of photolithography, including the techniques used, the chemicals used for photoresist and the exposure methods used to create the masks and features on-wafer are critical to the scaling process, and can themselves lead to variability in produced devices, as will be discussed later [16, 154].

Etching

Etching is the process of removing the unwanted material after the photoresist pattern has been formed. The etching process can be characterised by two desirable characteristics: *selectivity* and *anisotropy*. Selectivity is the ability of the process to remove just the desired area, leaving the masked areas and underlying substrate intact. Anisotropy is the ability to etch in one direction, and one direction only. Neither can be perfectly achieved; any etching process will result in some degree of removal of undesired areas and some degree of undercutting [16].

Originally etching was a wet process, using a dilute Hydrogen Fluoride solution to remove areas of SiO_2. Hydrofluoric acid has strong selectivity - it is very effective at removing the oxide whilst leaving the silicon substrate and photoresist-covered areas intact; unfortunately it etches isotropically, meaning it removes the oxide as quickly laterally as it does vertically, etching the oxide beneath the photoresist too much to cope with modern feature sizes. Other wet etching processes exist to remove other unwanted areas: phosphoric acid for metal areas, nitric acid for poly-silicon and phosphoric acid for silicon nitride, although for modern sub-micron processes a dry etching process is generally used to avoid the undercutting effect [154, 16]. Additionally, dry etching is a cleaner process, negating the need to carefully remove the acidic residue, it also offers greater compatibility with vacuum-processing techniques such as molecular-beam epitaxy, and is generally easier to automate [112].

Modern CMOS manufacture uses a plasma-based dry etching process named Reactive-Ion Etching (RIE). An etch-gas is pumped into a chamber, in which an RF voltage accelerates the electrons to high kinetic energies. Collisions between the electrons and neutral atoms creates ions and radicals; the ions are bombarded onto the wafer normal to the surface, ensuring an anisotropic etch, whilst the radicals simultaneously create a very selective isotropic etch. The resultant etch is thus a compromise of selectivity and anisotropy, allowing smaller feature sizes than could be manufactured with wet etches [154].

Metallisation and back-end processing

Once the semiconductor devices are fabricated, interconnections and contacts for wire-bonds are created through the process of metallisation. For silicon substrates, an aluminium alloy (typically 95% Al, 4% Cu and 1% Si) is deposited using the sputtering process. The aluminium is then patterned and etched using RIE, before being sintered at roughly $450\,^{\circ}\mathrm{C}$ to ensure a good ohmic contact with the silicon. A protective overcoat of silicon nitride is then deposited using chemical vapour deposition [16]. Pure aluminium is unsuitable for metallisation in VLSI CMOS because of electromigration effects (changing shape when subject to currents) and junction spiking; the silicon-aluminium commonly used to alleviate these problems leads to a high contact resistance, which is undesirable with the minute contacts formed in submicron devices; this has lead to the study and use of other materials, notably copper, as interconnects in modern devices [55].

The wafer now contains a quantity of complete die; the next step in the process is to divide up the wafer using a saw or scribe into individual circuits, mount the devices in the appropriate packages, and apply gold or aluminium bond wires between the aluminium contact points and the package leads; this is phase is known as back-end processing.

Fabrication steps for n-well Silicon

To illustrate how the above processes are combined in the creation of integrated circuits, a step-by-step outline is given in Figures 2.4 & 2.5. These diagrams show the steps to create transistors on the silicon wafer for the n-well silicon-gate CMOS process, one of the most frequently used in modern devices. A real-world GSI device would likely have many more steps and metal layers than are shown in the diagrams due to the scale of the device and complexity of interconnections.

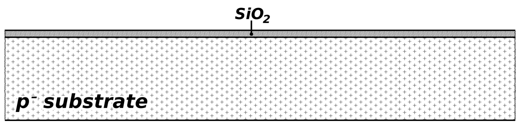
2.2.6 Transistor Sizing

The sizing of transistors within a logic or memory circuit is a key area of optimisation in CMOS design. The size dictates critical circuit parameters, such as delay, energy consumption and fan-out. The sizing determines the dynamic energy consumption through means of changing the switched capacitances, and also the leakage current of the circuit; it also affects energy consumption by determining the minimum operating voltage at which the circuit can correctly function [63].

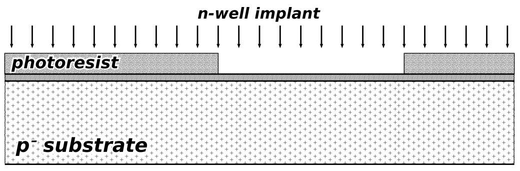
The three major sources of power dissipation are summarised by Chandrakasan et al in Equation 2.18 [39]. The first term is the dynamic or switching component, the product of loading capacitance C_L, supply voltage V_{dd}, voltage swing V and the clock frequency f_{clk}; p_t is the activity factor, the probability of a power-consuming transition occurring [3]. The second term is the leakage current $I_{leakage}$ which arises from sub-threshold effects and substrate injection. The final term is the short-circuit current I_{sc} which occurs when NMOS and PMOS transistors are simultaneously active, creating a direct path from supply to ground; this is generally a factor of static designs and is generally avoided in dynamic logic, except for when static pull-up devices are used, or a significant clock skew is present. It is clear that significant dynamic power improvement can be gained through the use of low-threshold devices and minimising supply voltages, however, this has the general trade-off of increasing sub-threshold currents causing a rise in static power dissipation.

Transistor sizing in a fabricated circuits is governed by a set of criteria known as design rules; these rules are a list of process-specific critical geometrical size and spacing constraints that must be adhered to when designing the lithographic masks, originating from limitations imposed by the lithographic process and physical constraints [163].

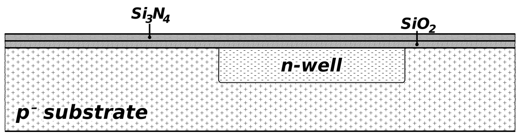
[3]The voltage swing is generally the same as supply voltage, thus switching power is generally proportional to ${V_{dd}}^2$.



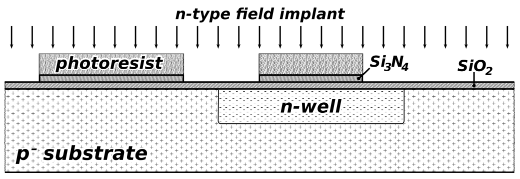
(a) The first step of the process is the growth of a thin SiO_2 oxide layer on the surface of a lightly p$^-$ doped wafer.



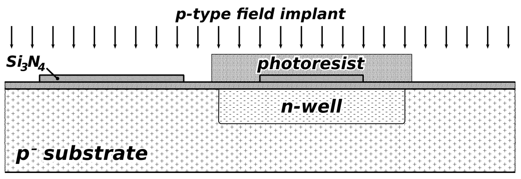
(b) The n-well is created by first depositing and developing photoresist around the area to be masked, then subjecting the wafer to a n$^-$ implantation. The photoresist is then removed and a high-temperature drives the implanted ions into the substrate.



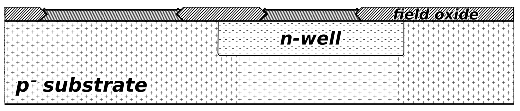
(c) A layer of silicon nitride is deposited over the entire wafer. Another layer of photoresist is deposited and developed on the silicon nitrite in areas where devices will reside, known as *moats*.



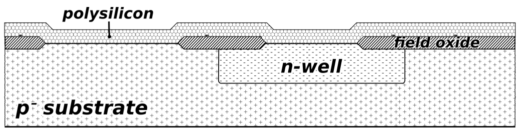
(d) The silicon nitrite is removed from the other areas. A global n$^-$-type field is implanted to these areas to ensure parasitic p-channel devices are not accidentally turned on via interconnect lines.



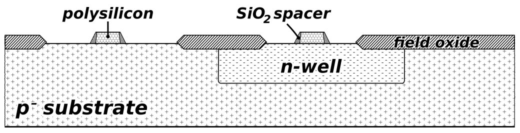
(e) The photoresist is removed then the process is repeated with a p$^-$-type field to ensure n-channel devices are not turned on.



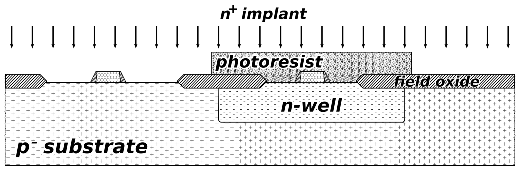
(f) A thick SiO_2 field oxide is the grown over the entire wafer in all areas except where Si_3N_4 resides. The oxide grows under the edges of the silicon nitride, resulting in the *'bird's beak'* shaped field-oxide regions.



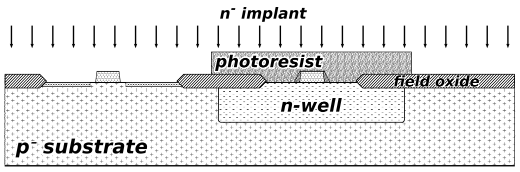
(g) The Si_3N_4 is removed and the whole wafer is covered with a very thin layer of SiO_2; this forms the gate oxide of the transistors. A thicker poly-silicon layer is then deposited over the entire wafer.



(h) The poly-silicon is patterned and etched where transistor gates and interconnect lines are needed. A SiO_2 layer is then grown over the poly-silicon and anisotropically etched, leaving thin oxide spacers on either side of each gate.

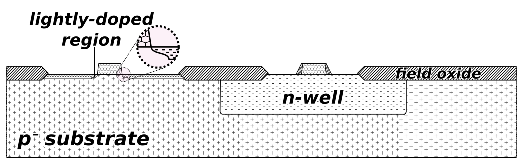


(i) To create the n-channel devices, photoresist is applied to mask unwanted areas, then n^+ ions are implanted into the substrate in areas not protected by photoresist, field oxide, poly-silicon and oxide spacers to create the source/drain regions.

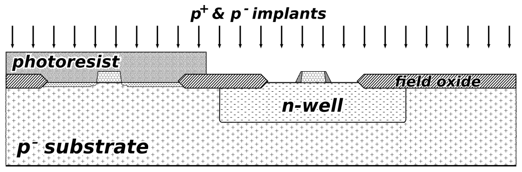


(j) The oxide spacers around the poly-silicon gate of the n-channel transistors are then etched away, followed by a lighter n^- implantation, to create a lightly-doped region.

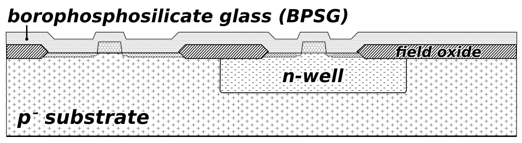
Figure 2.4: The early fabrication steps involved in constructing an n-Well CMOS Circuit (continued on next side).



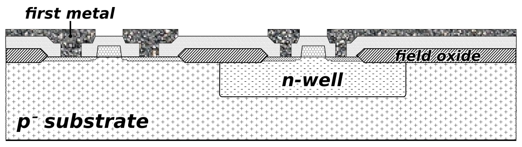
(a) The lightly doped drain and source regions created by the n^- implantation can be seen in this figure. This area of higher resistivity close to the poly-silicon gate helps to minimise impact ionisation. The photoresist is removed.



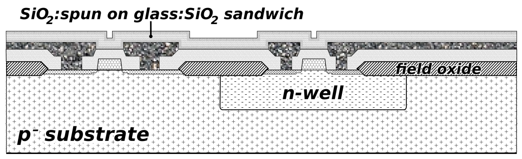
(b) The previous steps are then repeated to create the p-channel devices; photoresist is applied to mask the n-channel devices, an successive p^+ and p^- implantations are applied to create the p-channel drain and source with lightly doped regions.



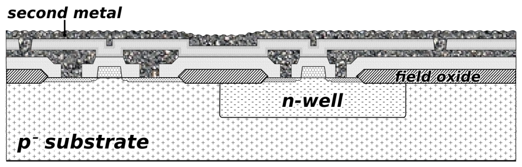
(c) With the transistors now created, the process of terminating contacts and interconnect the devices begins. A thick layer of oxide, typically borophosphosilicate glass (BPSG) is deposited.



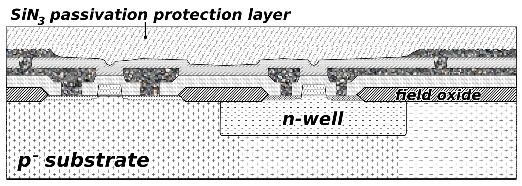
(d) The BPSG in areas where contacts are needed is etched away. A layer of aluminium is deposited on the surface, forming contact with the relevant poly-silicon areas, and unwanted aluminium areas are etched away.



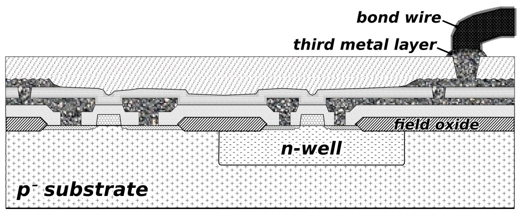
(e) A thick dielectric area is applied before the second layer of metals. This is typically a sandwich of SiO_2 applied with chemical-vapour deposition, followed by a spun-on layer of glass, followed by a final CVD SiO_2 layer, to minimise stray capacitance and ensure planarity.



(f) The vias (inter-metal connections) between the metal layers are isolated with photoresist and etched, before a second metal layer is deposited over the whole wafer.



(g) The second metal layer is selective etched to leave only the wanted interconnect regions. A thick layer, usually SiN_3, known as the passivation protection layer, is applied over the entire surface of the wafer. The areas where contacts are needed are etched from the SiN_3 and a final layer of metal is deposited and selectively etched.



(h) The wafer is split, individual die are packaged and wires are bonded from the relevant contacts to pins on the package. It should be noted that a modern IC may have many more metal layers, each separated by dielectric-sandwich layers, than pictured here, and that the bond wires, whilst often smaller than 15μm, are much larger than device channels.

Figure 2.5: The later fabrication steps involved in constructing an n-Well CMOS Circuit. Based on [16, 154].

$$P_{total} = p_t \left(C_L \cdot V_{dd} \cdot V \cdot f_{clk} \right) + I_{leakage} \cdot V_{dd} + I_{sc} \cdot V_{dd} \tag{2.18}$$

The aspect ratios of devices (W/L) determine the values of critical input voltages V_{IL},V_{IH} and V_I. Another import criteria in the layout and sizing of CMOS logic is the proportion of the aspect ratios between n-and-p CMOS pairs; it is quite common to employ higher ratios for PMOS transistors to compensate for the fact that $k'_n > k'_p$.

2.3 Transistor Scaling

The semiconductor industry, which underpins much of the worlds technology industry, is fundamentally built on a business model in which more computationally powerful and more energy efficient circuits are successively released. A more powerful circuit can be created by either increasing the operating speed of an existing design, or increasing the complexity of a design by way of increasing the number of functional blocks. The maximum operating speed of a device is determined by its physical dimensions of devices, with smaller transistors and interconnecting paths able to operate at higher frequencies than large devices. The number of functional blocks is also constrained by the device sizes; whilst it is possible to increase the die size, the increased likelihood of having imperfections within the enlarged die lowers the yield to a point where it is no longer financially viable[4].

2.3.1 Moore's Law

Gordon E. Moore, one of the co-founders of Intel, observed in his 1965 paper *"Cramming more components onto integrated circuits"*, that the number of components within an integrated circuit had doubled every year since their invention in 1958. He predicted that the trend would continue at the rate of high growth for at least a further 10 years, allowing single-wafer circuits with 65,000 devices by 1975 [121]. In his 1975 speech *"Progress in Digital Integrated Electronics"* he predicted a further steady growth, with the number of devices on an integrated circuit set to double every two years [122]. A colleague of Moore's, David House, noted that when combined with improvements in the operating characteristics of individual devices, such as operating frequency and power consumption, that the overall performance of an integrated circuit would effectively double at the faster rate of every 18 months. The prediction, commonly known as 'Moore's Law', has accurately stood the test of time (in many regards the actual speed of the development has been faster than 18 months) since 1975 [80].

[4]ULSI die are rarely larger than 300mm^2. The Itanium-2 at 596mm^2 is amongst the largest single die even produced, however the very high unit cost could partially offset the lower yield expected [77].

Moore's Law is illustrated to be an accurate reflection of growth in Figure 2.6. The figure shows the number of transistors on a selection of cutting-edge microprocessors and graphics processing units (GPUs) manufactured by two of the leading companies in each respective area, Intel and Nvidia. A curve showing the expected number of transistors following Moore's prediction of a doubling every two years is indicated. Certain devices, such as the HPC-oriented Itanium processor, tend to fall above the curve, though it should be noted these are generally very large (in terms of die area) and expensive devices. It can also be seen that the number of devices on a GPU has accelerated rapidly, to a point where cutting-edge GPUs now generally have more transistors than their microprocessor equivalents. GPUs are now regularly built on the most advanced technology-processes available, with the Nvidia GF100 'Fermi' chip being fabricated on TSMC's 40nm process [134]. As there is a trend for work traditionally done by the CPU to be off-loaded to the massively-parallel GPUs to increase speed and efficiency, it is likely that GPUs will be one of the major driving forces behind future transistor scaling [148].

Figure 2.7 illustrates the maximum clock speeds of selected range of Intel microprocessors at their release date. It is clear that from the mid-1980's to early 2000's Intel were successful in aggressively scaling clock-speed, achieving a doubling roughly every two years throughout this period (often referred to as the years of '*happy scaling*'). Since the mid-2000's, the clock-speed achievable from devices has plateaued rapidly, with the 3.8GHz frequency of the Pentium 4 HT-670, released in May 2005, still unsurpassed in retail devices. More recent performance enhancements have come not from ramping clock speed, but instead from enhancements to instruction-per-cycle-per-core (IPC^2) and the development of multiple core processors. Single-core processors are now used only in the lowest-end budget computers such as netbooks, with dual-and-quad core systems commonplace and hex/octal cores beginning to appear at the time of writing [77, 5].

In recent years the die size of leading-edge memory and logic product demonstrations in technical forums such as the IEEE International Solid State Circuits Conference (ISSCC) have grown at a rate of roughly 12% per year; the die growth has been necessary to accommodate the 40% to 60% more bits/capacitors/transistors per year in accordance with Moore's Law. Despite the die-size growth, semiconductor industries have managed a trend of reducing the leading-edge product cost by approximately 30% per year; to achieve this techniques to continuously enhance productivity and yield are necessary, through the processes of reducing feature size (at roughly 30% every two to three years) and periodically increasing wafer size [3].

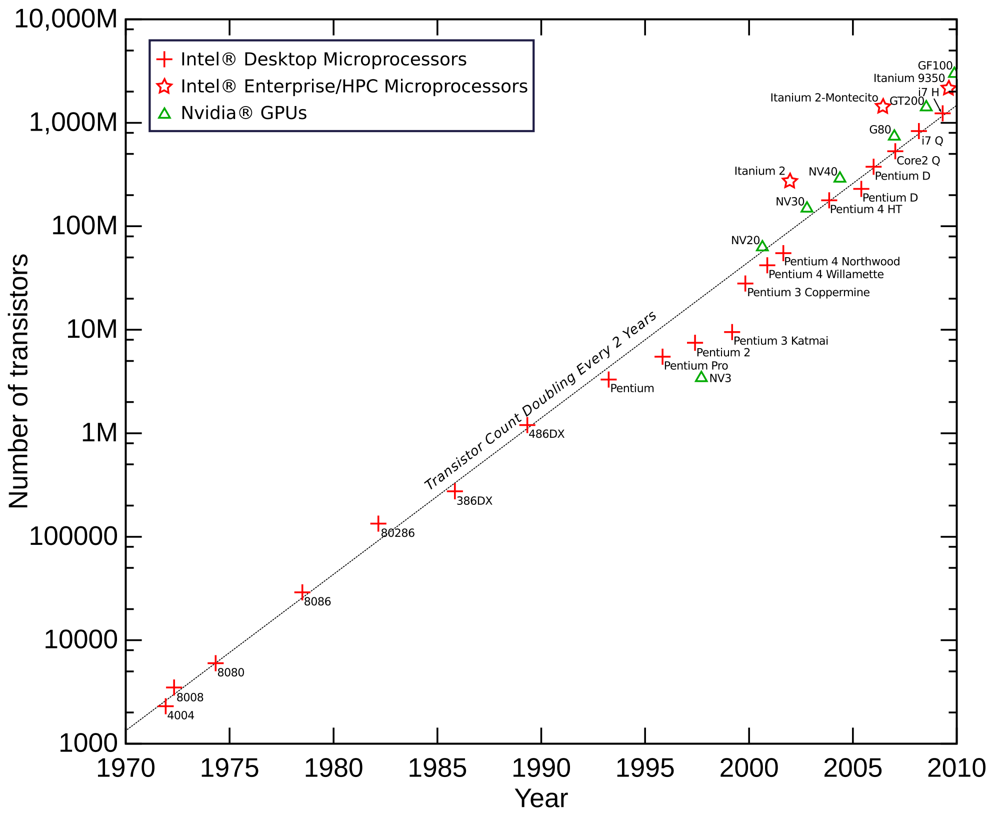


Figure 2.6: A comparison of the transistor counts in various Intel Microprocessors and Nvidia GPUs at different release dates, with a curve showing the projected counts following Moore's predictions of a doubling every 24 months [77, 5, 134]

2.3.2 Technology Roadmaps

Rules that propose the cyclical reduction scaling of devices were initially proposed by Robert Dennard in 1974; these rules have been followed relatively accurately for close to four decades [41]. Since 1992, the Semiconductor Industry Association, a trade association which represents the American semiconductor industry, has published roadmaps which plan to anticipate the evolution of the semiconductor market, and control the technological advances in IC production. Initially this roadmap was focussed on the American semiconductor market and named the National Technology Roadmap for Semiconductors (NTRS); since 1998 a closer involvement with semiconductor manufacturers in Europe and the Far-East led to the creation of the first global roadmap - the International Roadmap for Semiconductors (ITRS). Since 1998, the roadmap has been updated annually by the working groups affiliated with the SIA, which now contains close to one thousand companies [3].

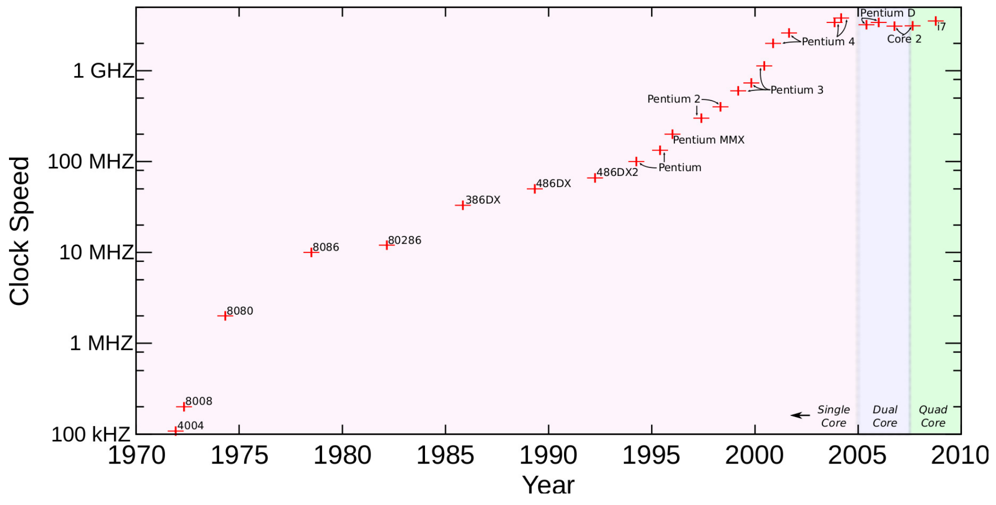


Figure 2.7: The maximum clock-speeds and core-counts of various Intel Microprocessors at different release dates [77, 5]

The SIA-ITRS published in 1999 projected an annual reduction of 11% in gate length every year through to 2015 (allowing the same drive current to be maintained whilst at a reduced operating voltage). The lateral size reduction leads to a similar reduction in the gate and junction capacitances which leads to a reduction in the gate delay of 10% per year [178]. Whilst the number of components on a single chip increases, the reduced operating voltage allows the total power consumption to either remain at a stable plateau or reduce. Some of the most significant elements to be found in the most recent release of the roadmap and the time of writing are given in Table 2.1, which outlines the progress that is intended by the major semiconductor manufacturers up to the year 2025. By this time, the major semiconductors fabricators aim to be achieving physical gate-lengths of under 8nm, and SRAM cell-areas under $0.02\mu m^2$, which compares to current leading-edge gate lengths of ≈25nm and SRAM cell areas of $\approx 0.45\mu m^2$ [3].

Table 2.1: Key Lithography-related Product Characteristics as listed in 2009 ITRS Roadmap [3]

	Short Term				*Long Term*			
	2009	**2011**	**2013**	**2015**	**2017**	**2019**	**2021**	**2023**
DRAM $\frac{1}{2}$ Pitch (nm)	52.0	40.0	32.0	25.0	20.0	15.9	12.6	10.0
Printed Gate Length (nm)	47.0	35.0	28.0	22.0	17.7	14.0	11.1	8.8
Physical Gate Length (nm)	29.0	24.0	20.0	17.0	14.0	11.7	9.7	8.1

2.3.3 Lithography Scaling

One of the key factors that dictates the progress of device-scaling is the available lithography processes which can be used. Lithography technologies have generally kept pace with integrated circuit development, and the advances predicted (or even dictated) by the industry-wide reliance on Moore's Law. Much research is done to ensure that the process of optical lithography can continue to improve in terms of minimum feature size and reliability to keep pace with technology-node targets. For devices larger than 130nm gate-length, most of the semiconductor industry relied on the exposure from a 248nm wavelength KrF laser to create the patterning of devices on integrated circuits. Numerous different enhancements such as phase-shift masking, sub-resolution assist and off-axis illumination exist which allowed the 248nm laser to produce sub-250nm designs; collectively these enhancements fall under the umbrella terms of Reticle Enhancement Technologies (RET) and sub-wavelength lithography. Of the RET techniques, two-mask phase-shift technology has proven the most effective, allowing the production of sub-30nm gate length patterning, albeit at heightened expense in mask-writing and inspection costs [154, 178]. Most manufacturers moved to a 193nm wavelength lightsource for their 130-nm technologies, retaining developed RETs. This wavelength has remained in used throughout the 90nm and 65nm technology nodes [33].

Optical lithography is constrained by the fundamental limits, which are described by the set of Rayleigh-scattering equations [31]. As the exposure wavelength scales proportionally with the minimum-linewidth, there is a strong motivation to shrink the wavelength as low as possible. To achieve this, a number of optical and non-optical lithography technologies have been proposed and researched, and are outlined in Table 2.2; some are currently used in actual fabrication, others are of an early-experimental natures and some are still theoretical. The important characteristic for each type of lithography are the wavelength of the lightsource, λ, the numerical-aperture NA of the lens used in the aligner, and the depth-of-focus DOF. The depth-of-focus indicated the range of distances from the focal plane at which the image is still sharp [154].

However, to achieve the sub-10nm gate-length goals proposed in the SIA roadmaps it is likely that a shift away from the 193nm lithography with two-mask phase-shift will be necessary. The most recent ITRS roadmaps predict that extreme-ultraviolet (EUV) lithography, using a 13 nm lightsource, will be the most likely light source for the next major generation of semiconductors, the 22 nm half-pitch node, although there are still several issues regarding the reliability of the light-source, the photoresist sensitivity and the availability of defect-free masks before this is ready for high volume manufacturing [155, 33].

Table 2.2: Historical, current and proposed future lithography technologies used in CMOS Production. The top three technologies are those which have been historically used. EUV is likely to be the next major technology in use, with the remaining technologies confined to laboratory tests [155, 33, 154, 178, 175, 3]

Technology	W_{min} (nm)	Exposure Mechanism	NA	DOF (μm)	Notes
356nm Hg UV-Lamp	250	Photons	≥ 0.5		Widely used throughout 1990's
248nm KrF laser	130-250	Photons	≥ 0.75		Used down to $\approx$ 130nm
193nm ArF laser	45-150	Photons	≥ 0.75	0.4	In current use, down to $\approx$ 22nm
Extreme Ultra-Violet	13	Photons	0.1	1.1	Most likely future technology
157nm F_2 laser	100-150	Photons	≥ 0.75	0.28	
126nm Ar_2 laser	22	Photons	0.5	0.5	
EBDW	22	Electrons	-	-	Electron Beam Direct Write
SCALPEL	0.24-0.16	Electrons	0.001	400	
X-ray	30	Photons	-	-	

2.4 Causes of Transistor Variability

Variability is an unavoidable reality of all CMOS integrated circuits due to the fact that it is not possible to create identical devices with perfect precision. It manifests itself as differences in the delay and power consumption characteristics of individual devices, and is exacerbated as a result of scaling circuits beyond a point where performance and power-dependant parameters can be controlled. The relative precision of individual device dimensions, and associated interconnects, reduces with the scaling of devices.

Traditionally, the significant variability in bulk-MOSFETs has been due to imperfections caused within the manufacturing process; such variability can be bound with known constraints, and provided that it is kept within practical ranges, is relatively easy to account for in the simulation and design processes. However, as technologies scale into the sub-50nm domain, atomistic constraints - problems caused by the precise atomic structure of the dopant-areas within a device - are becoming increasingly significant. The impact of such atomistic variations were small enough to be ignored in the days of 'happy-scaling', when the impact of manufacturing variations vastly outweighed their impact, but in future devices they will become a major obstacle. Many different advances and novel techniques have been developed, and will undoubtedly be made in the forthcoming years, to reduce the loss of precision that occurs in the manufacturing process, but the fundamental quantum-mechanical limitations cannot be overcome, and their impact will increase exponentially as devices shrink further [20, 129].

The methods for modelling and account for the two different types of variability are different. In this thesis we shall refer to the former - variability caused by the range of precision in the manufacturing process - *'deterministic'* variability, as it falls within known bounds and can be accounted for using relatively simple tolerance-aware models. The atomistic variability, which is stochastic in nature, is considered as *'intrinsic'* variability. Other authors have used the terms *'resolvable'* and *'fundamental'* to differentiate the types of variability, as the former can be fixed with technology improvements over time, whilst the latter requires a disruptive invention to circumvent [98]. The theory, models and experiments that are described in this thesis are concerned with intrinsic variability, although the deterministic variability which is caused by imperfections in manufacturing is discussed briefly below for comparative purposes.

2.4.1 Variability due to Manufacturing Process

Numerous variations occur due to imperfections introduced in the manufacturing process. One of the most significant contributors is patterning proximity effects and optical-proximity corrections (OPC) that occur through the many lithography steps involved in the fabrication process, particularly when RET techniques are used [38]. Additionally many other proximity effects associated with other manufacturing processes: stress, polishing (particularly chemical-mechanical polishing), annealing and droop exist, which add to variations in device characteristics. Over time, manufacturing improvements have seen the impact of such variabilities reduce to a level in which fully functional designs can be reliably created to operate within a set of bounds, which falls under the umbrella term of Design For Manufacturing (DFM) rules [118]. Automated software tools allow the model-based simulation of OPC, RET and etch effects, followed by circuit simulation of the resulting shapes, to be incorporated directly into the design flow, in a process named Desing Manufacturability Checking (DMC) [54].

2.5 Intrinsic Variability

This section discusses the different types of intrinsic variability that will exist in future MOSFET devices, paying particular attention to the atomistic variability which is used in the models discussed in this thesis. A 3D-model of a transistor illustrating the effect of random dopant placement, line-edge roughness, surface roughness and poly-silicon grain edge boundary effects is shown in Figure 2.8; these causes of these effects are now discussed in more detail.

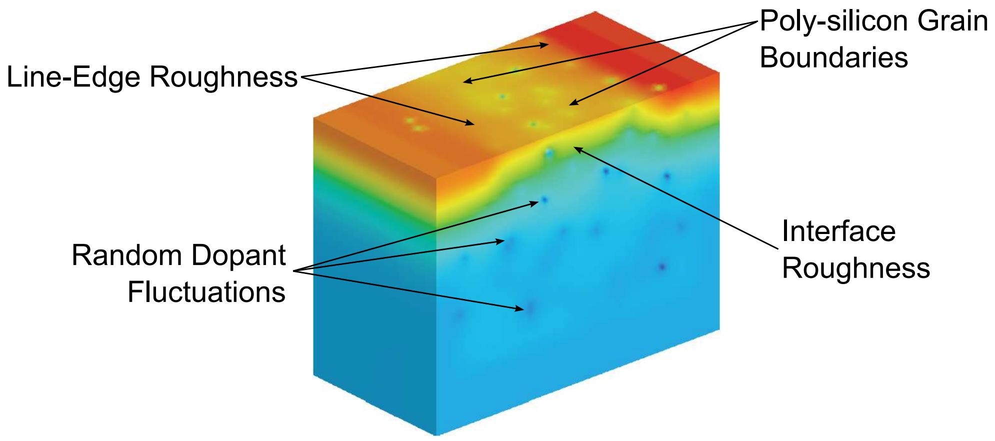
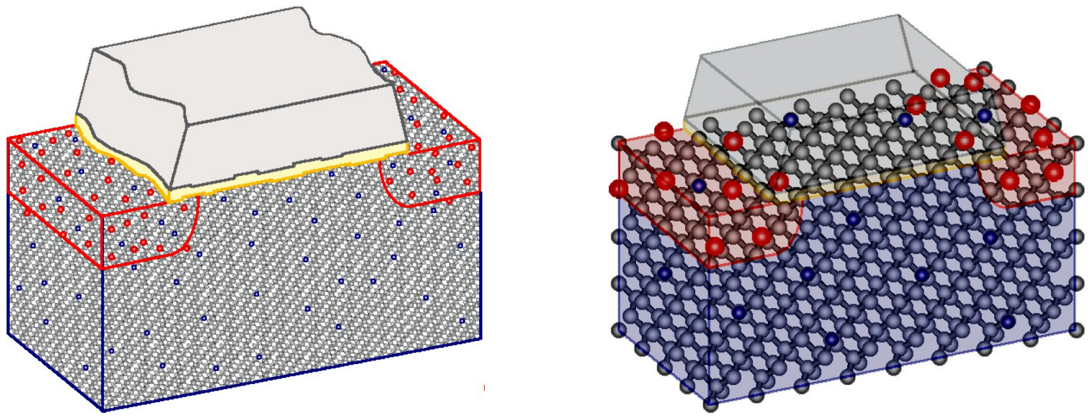


Figure 2.8: The main types of intrinsic variability illustrated on a 3D-simulated model of a 35nm MOSFET device [22]

2.5.1 Random Dopant Placement

Traditionally, transistor models have been based on assumptions of statistical averaging of the dopant concentrations, which results in smooth carrier concentration and potential profiles. This assumption is valid provided the number of dopants, and their specific position within the device, result in negligible differences between devices. However as devices scale well into the sub-100nm lengths, the number of dopant atoms within the active region of a device is relatively low. This results in devices which exhibit behaviour determined by the actual number and specific placement of dopant atoms; certain parts the device will become active before others, leading to a lower average threshold voltage than would be anticipated with a continuously doped device. Furthermore, significant variations of threshold voltage will be seen between individual devices on an intra-chip level, with the possibility of neighbouring devices exhibiting a substantial range of threshold voltages [138].

Where the dopants are considered on a per-atom basis with stochastic placement, these are named Random Discrete Dopants (RDD). Random dopants occur due to fabrication processes of modern MOSFET devices. The dopants atoms are implanted into the silicon at very high energies which leads to a scattering process; thermal annealing then allows implanted atoms to replace silicon atoms within the crystal lattice, a process which diffuses their position further. The nature of these processes mean it is impossible to precisely control the quantity and positions of the individual dopant atoms; every device will have its own unique distribution of dopants and with it a threshold voltage determined by the influence of dopant atoms on the potential. The atomic structures of simulated 22nm and 4nm with random dopants placements are illustrated in Figure 2.9; the 22nm device is representative of minimum sized devices that



(a) 22nm Device (b) 4nm Device

Figure 2.9: The atomic structures of 22nm and 4nm stylized transistors illustrating random discrete dopant placements [22]

will be created within the next five years, whilst the 4nm device, which has less than ten dopant atoms within the active region, is representative of the fundamentally minimum-sized devices that could be fabricated in silicon; beyond this scale it is highly unlikely that satisfactory functional devices could be assembled due to the lack of clearly defined regions.

The potential problem of random dopant placement and is consequences to threshold voltage variation have been acknowledged for nearly 40 years, being discussed in a 1972 paper by Hoeneisen and Mead, in which it was predicted that once devices reached channel lengths of 150nm, one-in-a-million devices would experience threshold voltage variations in excess of 20% [73]. A further detailed analysis of the effect on threshold and breakdown voltages caused by dopant quantity and placement was given by Keyes in 1975 [81]. The problems of dopant-placement induced variability were then confirmed in experimental observations on actual fabricated devices by Mizuno et al in 1993, in which the V_T variability in a fabricated transistor array with 8,192 devices with channel lengths of $L_{eff} = 0.5\mu$m and $L_{eff} = 0.3\mu$m was analysed. It was observed that the standard deviation of threshold voltages increased in the devices with smaller channel lengths, confirming that a portion of the overall V_T variability was independent to oxide-thickness variability and instead caused by the stochastic placement of dopant atoms [120, 119]. However it is within the last 10 years that the random dopants have become a *real* issue for semiconductor designers and manufacturers due to their impact on threshold voltages [144]. The number of dopant atoms present within the channel region a minimum-sized MOS transistor of is now typically around one hundred; in a set of devices simulated by Asenov et al, with effective channel length L_{eff} = 30nm and channel width

$W_{eff} = 50$nm, the number of dopant atoms within the channel depletion followed a Poisson distribution around a mean value $\bar{N}_d = 130$. At this scale and smaller, even assuming a perfectly uniform lithography, the randomness of the doping profile leads to significant variation in the actual effective channel length [20].

For contemporary bulk MOSFETs, which continue to be the prevailing design used in the 45- and 32-nm technology generations, random dopants are the dominant source of statistical variability [138]; the contribution of RDD to V_T variability has been demonstrated to be within the 60% to 65% range in fabricated 65- and 45- nm bulk MOSFETs [98]. Alternative technologies such as silicon-on-insulator (SOI) and FinFET devices (a non-planer double-gated design built on a SOI substrate [75]) are able to significantly reduce the impact of RDD variability as they have lower dopant concentrations near the interfaces [19]; such technologies add vast complexity to fabrication, and consequentially increase costs, and are largely reserved for specialist applications at the present time. Besides threshold voltage variations between devices, RDD also impact a number of other major device parameters, such as sub-threshold swing, sub-threshold leakage current and drain current.

2.5.2 Line-Edge Roughness

Line-Edge Roughness is defined to be any unwanted roughness in a semiconductor feature [155]. It primarily arises from the discrete molecular nature of the different photoresists that are used in the fabrication process. In the fabrication of a integrated circuit, the silicon wafer is spin coated with photoresist, which is subsequently exposed to UV light through a photo-mask, and baked to cure the photoresist. A developer solution is the used to remove either the exposed or unexposed areas of photoresist (the resist can be either positive, in which exposed areas are soluble to the developer, or negative, in which unexposed areas are soluble, resulting in an inverse of the image presented by the photomask). The primary cause of the roughness is because larger polymer aggregates in the photoresist take a longer time to be dissolved by the developer; the magnitude of the roughness has been shown to be closely related to the granule size and molecular weight of the resist [179, 138]. An illustration of line-edge roughness patterns created with negative photoresist is given in Figure 2.10.

Line-Edge Roughness has naturally always occurred, but has until recent technology nodes been so small compared to the actual linewidth as to create relatively insignificant variance; it has been demonstrated by Croon et al that LER has little affect on devices down to 80 nm gate lengths [51]. However, despite regular scaling down of linewidth, the edge roughness has remained relatively static, typically in the order of ± 5 nm, with occasionally greater differences depending on how the poly-line was formed [132]; these simulations have been corrob-

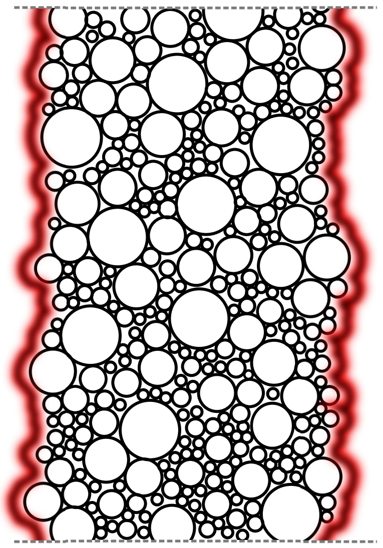
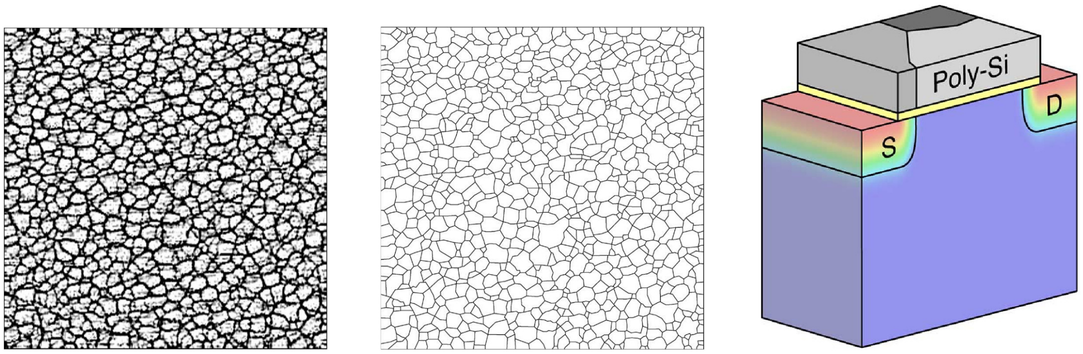


Figure 2.10: Line-Edge Roughness Patterns in Negative Photoresist [138]

orated by results of LER measured in physical devices using scanning electron microscopy (SEM) [176]. This results in a situation where LER will become increasing significant; various simulations have shown that if the magnitude of roughness remains at present levels, LER will have significant impact in sub-32 nm channel length devices and become the dominant form of stochastic variability at around 18 nm channel-lengths, outweighing the impact of RDD [51, 23]. This is a major problem for the semiconductor industry, as LER currently lags behind the requirements set out in the ITRS roadmap [155]; the current roadmaps set requirements for sub-1nm precision in LER before 2025.

Whilst LER is primarily attributed to the photoresist and lithographic process, Diaz et al identified that two distinct types of LER could be distinguished. That attributable to the resist/lithography they denoted as short-range LER, noting its high spatial frequency and short characteristic length of 1nm when caused by 193- and 248- nm lithography. Additionally, they noted a long-range LER with characteristic lengths in excess of 10nm, which could be attributed to poly-silicon surface roughness (see following sections) and stitching steps from the mask-creation process [56]. Additionally, varying light strength from the lithography light-source can contribute to long-range LER; this may become increasing important with future EUV light-sources as they have significantly less power than the current lasers used [155].



(a) Atomic-Force Microscopy image of Poly-Silicon Grains

(b) Corresponding traced image used as template for simulations

(c) Schematic of random grain boundaries in a simulated 30 × 30 nm MOSFET

Figure 2.11: Grain boundaries in poly-silicon and their reproduction in a 3D-simulation [36].

2.5.3 Surface Roughness

Surface-Roughness is vertical deviation of any realised surface when compared to its ideal form; in the context of MOSFET variability it is the asperity of the interfaces between the silicon substrate and the SiO_2 gate insulation. The shrinking of surface layers, in particular the oxide layer, leads to variations in the parasitic capacitances between the terminals resulting in addition V_T variability [123]. The roughness has the effect of limiting the effective mobility within the MOS devices, preventing devices from operating in the ballistic transport regime, in which electrons can transport with negligible resistivity; as a consequence surface roughness is one of the primary limiting factors of maximum on-current in sub-100nm devices [58], and the problem is exacerbated in Ultrathin-Body SOI MOSFETs due to the strong dependence between body thickness t_s and the low-field electron mobility [78].

2.5.4 Poly-Silicon Grain-Edge Boundaries

Polycrystalline structures are extensively in modern CMOS devices as gate materials, also finding use in on-chip rails and as emitters in bipolar devices in BiCMOS circuits. Poly-silicon grain boundary (PSGB) variability occurs due to the random arrangement of grains within the gate material due to this polycrystalline structure; polycrystalline silicon is rich in defects at the grain-boundary regions which can act as charge-traps [35]. Traditional analysis uses gradual channel approximation, which ignores the effects that grain boundaries have on the channel current. However, with the scaling down of gate-oxide thickness, implanted ions can penetrate through the poly-silicon and insulator into the device channel, resulting in localised stochastic variations on the lower surface of the poly-silicon [57].

Simulations at Glasgow on 30 $\times$ 30 nm MOSFET devices, as depicted in Figure 2.11(c), investigated the impact on V_T of different sized poly-Si grains. A peak in the magnitude of the σV_T was observed when the average grain size matched the channel length, i.e. 30nm. With smaller grains, a significant reduction in the standard deviation was observed, at the expensive of a steady increase in the average threshold voltage as a consequence of the increase in quantity of grain boundaries across the channel. With 30nm average grain size, the average V_T was roughly 370mV, with $\sigma V_T \approx$ 50mV. At 10nm average grain size, average V_T was roughly 580mV with $\sigma V_T \approx$ 30 mV. PSGB is likely the second most significant caused of V_T variability for devices between 35nm and 18nm channel lengths, behind RDD, however it is likely RDD and LER will become far more significant beyond these technology nodes [36].

2.5.5 Temporal Variability

In additional to the the previously described stochastic variabilities, further variations between devices can occur over time; these are currently modelled and characterised differently and shall be consider to be temporal, or dynamic, variability. Whilst traditionally a fabricated transistor could either be considered to be working or broken, it is now becoming necessary to consider and account for time-dependant instability and device degradation. Whilst the models discussed in this paper do not in include the any time-dependant breakdown, it will become increasing important to consider such failures in future circuit simulation and design; some of the techniques proposed in this thesis to reduce intrinsic-variability failures, such as redundancy and error-correction, may also be suitable to reduce overall circuit failures caused by time-dependant breakdowns. The three major sources of randomly distributed time-dependant failures are Negative-Bias Temperature Instability, hot-electron injection and electromigration [109].

Other significant sources of temporal variability on a VLSI circuit include ‘heat flux’, which concerns the spread of localised heat across an entire microprocessor die. Typically an area of cache-memory has a lot lower heat flux than an execution unit; these localised variations add stresses to the power-distribution grid resulting in time-dependant supply voltage variations. Degradation of a devices saturation current, due to hot-carrier degradation effects and oxide wear-out, occurs over a number of years. In previous generations this has been small enough to be accounted for using design margins; however this degradation is expected to escalate beyond the 32-nm technology node [33].

Another effect which should be mentioned here is soft-errors caused by radiation. Radiation from alpha-particles caused by radioactive impurities, and more significantly neutrons from terrestrial cosmic rays, hitting a integrated circuit, has the effect of create localised charge on nodes; this has the potential to flip the output of a memory cell or logic gate. Research by Hazucha et al has shown a steady 8% increase in the soft-error rate (SER) per technology generation through the 0.25μm, 0.18μm, 0.13μm and 90nm generations, and an 18% increase in SER when a 10% supply voltage reduction was used with 90nm technology [72]. When the increased rate of SER is combined with the doubling of cells per each generation following Moore's law, the overall probability of a single-bit soft-error is over 100 times more likely in a proposed 16 nm chip than in a 180 nm chip [33].

Negative-Bias Temperature Instability

The most pressing cause of temporal variability in CMOS devices is Negative-Bias Temperature Instability (NBTI). This primarily effects p-MOS transistors which are operating with negative gate voltage, although the same instability mechanism also affects n-MOS transistors when they are negatively-biased. NBTI is caused by the creation of interface traps and oxide charge when pMOS devices are supplied with negative-gate bias at elevated temperatures; this results in broken Si-H bonds at the interface between the silicon and insulator [13].

NBTI has become a significant reliability issue in recent years due to a number of factors: transistor scaling has increased the gate-electric field strengths; chips also now operate at higher temperatures and changes in fabrication have also increased the effects of NBTI, with nitrogen being added to thermally-grown SiO_2, and buried-channel p-MOS devices have been replaced with surface-channel devices. Whilst the existence of NBTI has documented since as early as 1966, it is only very recently that the various techniques to model it have begun to closely correlate with experimental data obtained from actual devices [146]. There is conflicting research with regards to the long term effects of NBTI; some research has suggested that full recovery occurs within a few seconds of stress-induced NBTI events, whilst other research has identified that NBTI damages is only partially-recoverable, with an element of permanent damage (and shift in threshold-voltage) caused by NBTI stress events [110]. Regardless of this discrepancy, given the frequency of device-switching in modern electronics, a significant V_T shift for a number of seconds will create significant problems for circuit design.

2.5.6 Interconnect Variability

The effects of atomistic variability are not limited to the transistor devices themselves; the interconnections found between devices also suffer from variability due to microscopic variations in their structure, resulting in deviation in the conductance and, most significantly, capacitance of the path. As the interconnecting paths reduce in size with transistor scaling, a similar increase in interconnect variability will be observed as is with device variability. Metal line width and thickness variations occur due to both manufacturing deficiencies and the discrete nature of the materials involved in the process, comparable to the line-edge and surface roughness variabilities found within devices [30].

For VLSI components to be successfully designed it is necessary to model interconnection variation and account for similar levels of variability as may be seen within the transistor device dimensions themselves, although the construction of parametric interconnect models is hampered by a rapid increase in model complexity and with it computation cost [103]. Whilst interconnect variability will have a significant impact on the design of forthcoming circuits, the tools to simulate or model it were not available for the experiments described in this thesis.

2.6 Modelling Variability

Basic modelling of MOSFET devices such as would be performed by hand utilises the large signal and small signal models. The large signal modelling is first utilised to find the DC conditions and operating point of the device. A linearised small-signal model, as illustrated in Figure 2.12 may then be used to simplify calculations once the DC operation point has been found, which will be valid only with small changes from the DC state. The large and small signal models offer a useful insight into the operation of a MOS device, but they ignore a large number of second-order effects that exist in the device, including channel length reduction due to overlapping regions, mobility reduction due to electric fields, drain-induced barrier lowering, velocity saturation and intrinsic parasitic resistances [15]. This renders them unsuitable for modelling the performance of VLSI circuits. For greater accuracy SPICE simulators have support for over 60 different device models of MOS transistor, with many different levels of complexity and many different fabrication technologies modelled [160].

2.6.1 The SPICE Circuit Analysis Tool

Circuit simulation programs operate by converting a text-based representation of a circuit, containing component types, values and interconnections (forthwith known as a *netlist*), then translate this into a set of non-linear differential equations. Depending on the software, these equations are solved using either the Newton-Raphson method, sparse-matrix techniques or implicit integration methods [11]. On of the most widely developed and used bases for circuit simulations is SPICE (“*Simulation Program with Integrated Circuit Emphasis*”). It is a well-established design tool and is widely used as a verification tool before designs are committed

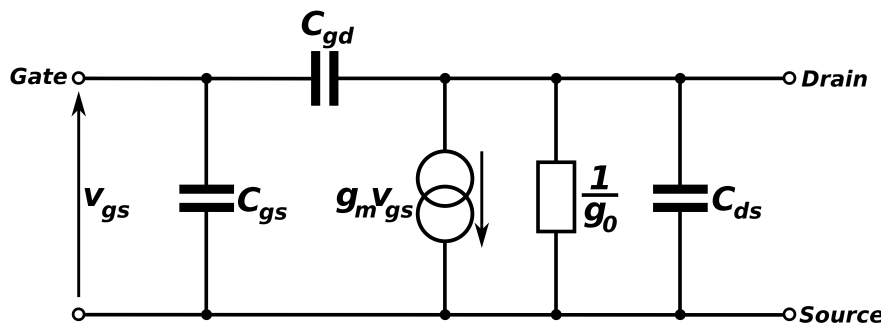


Figure 2.12: Small-Signal Model Equivalent Circuit for a MOS Transistor in Common-Source Configuration, based on illustrations from [160, 48]

to fabrication.

SPICE allows for a number of different types of circuit-analysis to be undertaken. The original version allowed non-linear DC analysis, small-signal steady-state sinusoidal (AC) analysis with noise analysis, and non-linear, time-domain transient analysis. Additional revisions added DC transfer-curve analysis and transfer-function analysis [128]. Additionally, small-signal analysis of sensitivity, pole-zero and distortion can also be calculated in more recent revisions, along with parametric sweeps to analyse circuits under changing operating conditions [12].

Background and History of SPICE

Computer-aided circuit analysis was first popularised with the Electric Circuit Analysis Program (ECAP), developed by IBM in the mid-1960's; this was the first time a system could automatically convert a nodal-description of the circuit into a set of equations and solve them. Subsequent improvements to the ECAP system included SPECTRE, NET, CIRCUS, CANCER and TRAC [28]. It is, however, the subsequent SPICE software which has received the most development, which can be attributed to its development at a public university with public-domain availability, and the inclusion of BJT and FET models, resulting in suitability and scalability at simulating both analogue and digital circuits [127].

SPICE is a general purpose analogue circuit simulation tool, originating from the Computer Aided Design group of the Electronics Research Laboratory at the University of California, Berkeley, in the early 1970's. The developers who had worked on the proprietary CANCER program redeveloped a new public-domain circuit simulator, first presented in 1973. By revising the CANCER code, they removed the proprietary constraints, and importantly added then-advanced device models for the BJT, JFET and MOSFET devices. Given the prevalence of BJTs in integrated circuits of the time, special emphasis was given to the BJT model in the original SPICE software. However, a key function of the original SPICE software was the possibility of using external device models via .MODEL cards, which would prove to be a key feature in allowing modern devices to be simulated [128].

The original version and its successor SPICE-2 were coded in Fortran-66, and used a process of nodal analysis to solve circuit equations [12]. The original version was tested on a CDC 6400 mainframe computer, with a memory limited to around 140 thousand octal words. This limited the size of circuits that could be simulated in the first version to roughly 50 nodes and 25 bipolar transistors [127]. SPICE-3, the base of the distribution discussed and used in this thesis, was developed in 1989 and is written entirely in C. The original release of Spice-3 (which shall be considered to be 'Berkeley SPICE') contains around 217,000 lines of C source code, spread across 878 separate files. It is regularly used to simulate circuits with several thousand nodes and active devices, generally limited by the memory-footprint of the computer [90].

SPICE functions by analysing the input file to determine all the circuit elements connected to each node, then applying Kirchoff's current-law to create a set of simultaneous equations for the circuit; the admittance of each branch being the known quantity and the voltages being the unknowns. The equations are formed into an admittance matrix, which is solved using the Newton-Raphson method [28].

Variants of SPICE

Several different packages exist which are based upon the Berkeley SPICE core but add a number of features and enhancements. Some of these are free-to-use, open-source variants, whilst many others are commercial designs. The Berkeley SPICE is limited in its input and output options, working in a ASCII environment; many of the alternatives add features such as interactive alteration of circuits, graphical post-processing and the ability to run on different computing platforms such as personal computers and clusters. Several of the different varieties of SPICE are listed in Table 2.3 [28, 12, 177].

For the work in this thesis, the NGSPICE simulator has been chosen. It is an actively maintained open-source, free to use variant of SPICE, which works correctly with the variability-aware transistor models that have been created (see section 2.6.1). The lack of licensing restrictions made it particularly suited to Grid-based evaluation; many of the commercial versions of SPICE, notably HSPICE, have licensing terms which work on a per-installation basis, which places obstacles in the way of a cluster or Grid-based platform where the software is to be distributed across many nodes. Throughout the duration of this thesis a number of revisions of

Table 2.3: Different distributions of SPICE-based Simulation Software

Name	Vendor	Licence	Platform[1]	Notes
HSpice®	Synopsys®	Commercial	W,L,S	[157]
Multisim	National Instruments	Commercial	W	
NGSpice	gEDA Project	Free (GPL)	W,L,M	First release 1999 [130]
PSpice®	Cadence®	Commercial	W,L	First released 1984[2] [37]
SmartSpice	Silvaco	Commercial	W,L	Multithreaded, 64-bit [150]
SpiceOpus	University of Ljubljana	Free	W,L	
T-Spice	Tanner EDA	Commercial	W	

[1] W=Windows, L=Linux, M=Mac/OSX S=Solaris
[2] PSpice is now part of the OrCAD and Allegro toolchains

Table 2.4: Revisions of NGSPICE used in this thesis

Version	Release Date	Notes
ngspice 17	30-08-2005	
ngspice 18	01-12-2008	Allows parametrical netlists, .lib statements
ngspice 19	23-04-2009	Bug fixes in memory management, revised BSim models
ngspice 20	16-11-2009	Added .measure statements, updated to BSIM 4.6.5
ngspice 21	21-06-2010	Latest release, updated BSIMSOI model

the NGSPICE software have been released, often adding significant functionality which has improved the operation or performance of the software described in this thesis; generally it has not been possible to repeat all results with the latest version due to the time taken to compute each set of results. A brief synopsis of the NGSPICE versions used is given in Table 2.4.

The BSIM Transistor Model

All variants of SPICE contain different models for field-effect transistors. The original version of SPICE used a model originally proposed by Shichman and Hodges, which is now considered to be the SPICE ‘Level 1’ model. This is a large signal model, which is solved using the formula shown in Equation 2.19, in which μ_O is the surface mobility of the n-channel device, C_{ox} is the capacitance per unit area of the gate oxide, W is the effective channel width and L is the effective channel length [149].

$$i_D = \frac{\mu_0 C_{ox} W}{L}\left[(v_{GS} - V_T) - \frac{v_{DS}}{2}\right] v_{DS} \tag{2.19}$$

A small-signal model can be derived from the above equation, which is suitable accurate for large MOSFET devices provided the length and width of the MOS device are greater than

10μm and the substrate doping is low. Once devices approached this size, a far more accurate model was required, which resulted in the development of the SPICE Level-3 model at Berkeley. This model adds narrow and short-channel effects to the basic large-signal model which requires a number of additional parameters to be considered. These include drain current, threshold voltage, effective carrier mobility, saturation voltage and channel length modulation; sub-threshold conduction is also present in the model. Despite containing 19 different parameters the model is not accurate enough to simulate modern devices; it is useful for devices down to 0.8μm in size [64].

At geometries below 0.8μm, the standard SPICE models fail to accurate reflect the observed characteristics of devices. In 1984, the BSIM1 model was introduced to address the need for a more accurate sub-micron model; the model contained 60 parameters and approached the modelling problem as a multi-parameter curve fitting exercise; whilst some of the parameters were related to device physics, it was largely a non-physical model. In 1991 an updated model, BSIM2, was released, to account for more accurate modelling of output resistance changes resulting from source/drain parasitic resistance, inversion-layer capacitance and hot-electron effects. This model upped the parameter count to 99, which results in significant expertise in designing and fitting models to specific devices. In 1994 a third model, BSIM3, was released, which returned the main function of the model to being based on device-physics rather than curve-fitting; as a consequence fewer parameters were needed (40 for BSIM3v2) and improved performance in modelling analogue circuits was observed. The subsequent BSIM3v3 simulation model includes modelling for all the relevant parasitic effects down to 0.15μm channel length devices, using extra parameters and additional values for the capacitances and resistances of metal lines to describe an individual transistor. This level of detail led to the BSIM3v3 model becoming the industry standard MOS transistor model [16]. A comparison of the performance and minimum channel lengths and oxide thickness for different SPICE models is given in Table 2.5.

The most recent series of BSIM models is BSIM4, originally released in the year 2000, which aims to address the physical effects observed in sub-100nm devices. It is essentially an extension of BSIM3, and continues the physics-based approach to modelling. It adds a more accurate model for intrinsic input resistance, a channel thermal-noise model and a gate direct tunnelling for multi-layer gate dielectrics, amongst 24 other total developments over BSIM3v3. The latest revision as of time of writing is BSIM4.6.5, released in September 2009 [124]. Due to the vast number of calculations and models necessary to describe the BSIM3 and BSIM4, they are not included here; they are covered in detail in [18, 64] for the BSIM3 series models and [124] for the BSIM4 series.

Table 2.5: Performance comparison of MOSFET Models available in SPICE [42, 16, 64]

Model	Minimum L (μm)	Minimum T$_{ox}$ (μm)	Subthreshold Accuracy	Small Signal Accuracy
MOS Level 1	5	50	Not modelled	Poor
MOS Level 2	2	25	Poor	Poor
MOS Level 3	1	20	Poor	Poor
BSIM1	0.8	15	Fair	Poor
BSIM2	0.35	7.5	Good	Fair
BSIM3v2	0.25	5	Good	Good
BSIM3v3	0.15	4	Good	Good
BSIM3v4	0.13	3	Good	Good
BSIM4v6	< 0.09	< 3	Good	Good

2.6.2 Simulation Techniques for modelling Variability

A range of techniques have been used to model the various effects of intrinsic variability in sub-micron transistors, which range dramatically in terms of the computational complexity. To simulate a device at this level, the self-consistent solution to two equations must be found: these are the Poisson equation, which computes the electrostatic potential between the fixed and mobile charge through the use of boundary conditions, and the transport model equation which models the electron and current density. Three different approaches to solving the equations are outlined here: the simplest is known as Drift-Diffusion simulation (DD), which represents a low order approximation of the Boltzmann transport equation (BTE). A method based on Monte Carlo statistical methods provides an indirect method of solving the BTE, improving on the accuracy offered by drift-diffusion. These two approaches are discussed in greater detail in the following section.

The most complex in terms of computational demand are techniques using quantum-mechanical approaches, most notably Non-Equilibrium Green's Function (NEGF) modelling. To solve Green's functions it is necessary to invert very large matrices containing Hamiltonian elements - it is only within the previous year that full 3D-simulations of variability in transistors using NEGF have been undertaken [111]. However, once channel lengths are scaled below $\approx$ 10nm, quantum effects begin to dominate and the Monte-Carlo approaches lose their validity [138].

Historically, a number of other methods have been used to simulate and model effects of intrinsic variability using a number of different techniques in both 2D- and 3D- domains. Diaz et al proposed an analytical model to estimate the effects of LER on drive current and off-state leakage in sub-100nm devices. Their model functions by partitioning a device into small unit cells, which each assume a constant gate length [56]. Many trials at combining both DD and Monte-Carlo simulations in hybrid techniques have been developed which use the Monte-Carlo techniques only in regions where it is mandatory to produce accurate results [86].

Drift-Diffusion Simulation

In the simulation of a steady-state NMOS device, the DD approach operates by self-consistently solving the steady-state current continuity equation with Poisson's equation. To solve these equations it is necessary to calculate the current-density (J_n), which is expressed as the superposition of the drift component ($J_{n\text{drift}}$), which is related to the electric field (E), and the diffusions components ($J_{n\text{diff}}$), related to the gradient of the electron density (n). The current-density equations are as follows [184, 138]:

$$J_{n\text{drift}} = qn\mu_n E \tag{2.20}$$

$$J_{n\text{diff}} = qD_n \frac{dn}{dx} \tag{2.21}$$

where D_n is the diffusion coefficient, q is the electronic charge, and μ is the mobility. The DD model makes many assumptions to obtain a closed system of equations from the BTE, ignoring non-local effects and assuming thermal equilibrium between the carrier and the lattice. This results in a drawback with the model: its failure to capture non-equilibrium carrier transport effects results in an underestimate of the on-state drain current. The electric field that exists within the active region of a submicron device can be very high, and often undergoes rapid variations over distances which are comparable to the carrier's mean free path. DD assumes that carriers can make instant responses to changes in the electric field, however in reality they have mass and require a finite time and distance to equilibrate with the field [86, 138].

Despite these shortcomings the DD is still effective when studying the sub-threshold regime of operation in a device due to the weak coupling between current and Poisson's equation in this region, and as such is suitable for the study of statistical threshold voltage variability in sub-100nm devices. To improve the validity of such simulations in very small devices it is possible to account for some non-local quantum effects through the use of density-gradient (DG) corrections, which add an extra driving force term to the current density equation [138]:

$$J_n = J_{n\text{drift}} + J_{n\text{diff}} + \text{DG} = qn\mu_n E + qD_n \frac{dn}{dx} + 2qn\mu_n \nabla \left(b_n \frac{\nabla^2 \sqrt{n}}{\sqrt{n}} \right) \tag{2.22}$$

in which b is a term which expresses the magnitude of the density gradient dependence. This driving force term can be thought of as a "quantum diffusion" current, which has the effect of pushing carriers away from the $Si{:}SiO_2$ interface. This adds to the model a number of quantum tunnelling effects, and also has the additional benefit of correctly capturing the effect of RDD on the potential distribution of the device.

Monte Carlo Simulation

In Monte-Carlo Simulation, the Boltzmann transport equation is solved in a stochastic manner, which avoids the limitations that exist in DD simulation. The BTE is virtually impossible to solve directly in all but the most trivial of cases, which results in the popularity of Monte Carlo methods to find solutions. The movement of carries within a device is simulated, which can be considered as a series of free flights and random scattering events at the end of each free flight. The scattering events include interactions between carriers and phonons, fixed impurities and other carriers. The free flight trajectories are calculated using Newtonian physics, with the scattering events adding quantum mechanical calculations. The methods used require a high quality source of random numbers, which are used to determine flight times and scattering events based on appropriate probability distributions.

2.6.3 Simulation of Intrinsic Variability in Future MOSFETS

The modelling techniques described in the previous section stress the computational demands for simulating the effects of RDD and other intrinsic variabilities in a single device. It is clear that simulation of a complete circuit, even of minimal nature such as a logic gate, is not practical using the methods outlined above. The problem is exacerbated when considering repeating the process many thousands of times as will be necessary within a evolutionary approach. In order to be able to achieve the goal of simulating variability-aware circuits, it is necessary to harness the techniques described above to create compact models which can be used with SPICE. This section describes the process undertaken by the Device Modelling Group (DMG) at Glasgow University to create libraries of such compact models.

The Atomistic Simulator

The "Atomistic Simulator" is a tool which has been developed and revised at the DMG in Glasgow over a number of years. The simulator functions by solving the nonlinear-Poisson and current-continuity equations used in the drift/diffusion approximation, discussed previously, in a 3D simulation domain. The simulator includes the density gradient corrections for holes and electrons which allows the effect of discrete dopants to be correctly determined. A detailed description of the design and operation of the Atomistic Simulator can be found in [138].

Before the introduction of the 3D Atomistic Simulator, simulations of RDD, LER and oxide-thickness variations had been studied using numerical simulations and idealised devices. In 2006, the Atomistic Simulator was used to create the first 3D models combining all of these effects, to simulate the fluctuations found in conventional MOSFET structures scaled to fit five successive technology generations proposed by the 2003 ITRS roadmap; these were 35-, 25-, 18-, 13- and 9-nm channel lengths. The simulations used density-gradient quantum corrections which were calibrated to a real 35nm MOSFET device [22].

For the simulation of poly-silicon grain boundary effects, the Atomistic Simulator has been enhanced to include potential pinning and doping non-uniformities along grain boundaries within the poly-silicon gate. This approach ignores mobility variations which are associated with fluctuations in the channel due to surface potential pinning in the poly-silicon; mobility variations have little effect of threshold voltage but do impact on the on-current of the device [36].

Creation of BSIM Model Libraries

The following stage in the process of generating a platform allowing the modelling of variability using standard software tools is the creation of libraries of models which reproduce the variability simulated by the atomistic simulator. The atomistic simulator is used generate sets of IV curves for each of the transistors it simulates. A BSIM4 [5] model based on the device being simulated, which is generally provided by the foundry which makes the device, is used as the base model for the library. For each simulated device, a carefully selected set of BSIM parameters are adjusted to match the extracted IV curves as closely as possible; this choice of chosen by hand and has evolved throughout the duration of the nano-CMOS project, whilst the actual fitting of curves uses a automated curve-fitting process. At the end of this process, a library of BSIM4 models which reflect the IV characteristics of simulated devices is created, which can then be used by the tool described in Section 2.6.4 to allow variability simulations using the SPICE simulation package.

[5]The actual models used have included version subversions of the BSIM4 series as new libraries and versions of the BSIM model have been created. The libraries in this thesis are based on BSIM4v4 and BSIM4v5.

Use of Grid Computing to Accelerate Simulation

Whilst the drift/diffusion techniques are substantially faster than Monte Carlo techniques and orders-of-magnitude faster that solving NEGF, the amount of time require to create statistically accurate libraries cannot be underestimated; ULSI devices with in excess of 10^9 transistors are now commonplace and thus it is necessary to know the variability that occurs deep in the tails of the distribution at 6σ and beyond to be able to model how devices such as SRAM will function [6]. The simulation of a single 35nm device was benchmarked on a single-core of an Intel Xeon 2.66GHz machine at ≈ 32.5 minutes, of which $\approx 95\%$ of the time was spent solving the Poisson's and density gradient correction equations, and the remainder solving current continuity [7]. The creation of the libraries used in this thesis required in total approximately 60 CPU years of work [138].

In order to facilitate this, the use of modern large-scale computing clusters and grids was essential. The calculations were primarily carried out on ScotGrid and an in-house cluster at the DMG, totalling approximately 1,000 high-performance cores based on Intel Xeon and AMD Opteron CPUs. A number of technical issues had to be overcome in the process of enabling the simulations to run on Grid-based systems, which are themselves a relatively new computational paradigm and not ideally suited to the tasks of job submission and data management when simulations of thousands of jobs in parallel are required; the problems faced by the DMG are similar to those faced by ourselves in the running of several thousand SPICE simulations in parallel and thus many of the lessons learned at Glasgow ultimately directed our decision to primarily focus on cluster-based simulations.

Job submission on ScotGrid was performed using the Globus software tool-kit, however this is limited to single-job submission and monitoring. To overcome this limitation the Ganga frontend, developed at CERN, allows multiple jobs to be submitted and monitored concurrently, albeit hampered by the deficiencies present in the Grid-middleware, most notably slow submission time, requiring in excess of one hour to submit a batch of 500 jobs. Additional problems occurred due to bugs in Ganga and the middleware and incompatibilities between updates of the different software stacks; this resulted in occasions where entire batches of jobs were lost, at significant expense in terms of time and resources [8]. In contrast, the local cluster based approach used Sun Grid Engine (SGE) with its inherent support of parallel submission and added ease in job-tracking and data-management due to the localised nature of

[6] Corresponds to 3.4 defective devices per million.

[7] The same simulation was carried out without the density gradient corrections and took only ≈ 2.5 minutes, however the corrections are needed for accurate simulation of such small devices [138]

[8] It is estimated that not less than 30% of all jobs submitted by DSG to ScotGrid over the course of the nano-CMOS project required resubmitting [138].

the resource [138].

Libraries used in the Thesis

Two different variability libraries have been used in most of the experiments in this thesis, with the earlier library being based on a Toshiba 35×35nm device, with only the effects of RDD simulated. This library only has single width devices, with larger widths assembled from parallel pairings of devices. This library was replaced by a later library based on 35×35nm high performance models, which include the effects of LER and surface-roughness in addition to RDD. They also feature higher accuracy and the inclusion of multi-width models, with the library containing devices which are 35nm (1×), 70nm (2×), 140nm (4×), 210nm (6×) and 280nm (8×) wide. Both libraries contain 200 NMOS and 200 PMOS devices for each width [9].

2.6.4 A tool for simulating variability: RandomSPICE

The process for creating libraries of BSIM compact models by the DMG has been described above. This section describes the tool, *RandomSPICE*, which is also developed by the DMG, that allows SPICE simulation of a given circuit based on randomised selections of transistors from the given libraries. The tool, written in Python, takes as its input a standard SPICE netlist, and produces a batch of equivalent netlists in which the MOSFET devices are replaced with models from the variability-aware libraries.

In the versions of RandomSPICE used in this thesis, the channel-lengths of the transistor devices are fixed to a single size; future revisions will include the ability to vary channel length (between a set of discrete values), however at the time of experiments the functionality was limited to single-length transistors. However, the channel width can be set to any integer multiple of the channel-length; again with future revisions sub-integer multiples may be possible [10].

As RandomSPICE and the libraries it used have been developed in parallel with the software in this thesis, certain developments which alter the way it operates have taken place throughout the experiments. In the earlier version of RandomSPICE, a device with a channel width four times the channel length in the input netlist would be replaced in each resultant output netlist with a parallel subcircuit of four devices, all of the same width as length. In later

[9]It is the opinion of the author that a larger number of devices would be needed to accurately model the behaviour of circuits containing transistors whose characteristics are at the extreme tails of distributions, however larger libraries were not available for the simulations in this thesis.

[10]In real-world fabricated devices, it is generally possible to create devices at sub-multiple lengths through the use of lithographies with RET. For example, one may find a library based on 32-nm minimum channel lengths which includes devices with 48-nm channel lengths and widths.

versions, models were created with channel widths at 1×, 2×, 4× and 8× the channel length, thus the same transistor could be replaced with a single device in the output netlist; this offered both increased accuracy of modelling and a significant reduction in simulation time due to the reduced node-count and netlist size of the output netlist [11]. For example, the following single-line extract from an input netlist represents a single NMOS device of width '4 units' (140nm) and length '1 unit' (35nm):

```
MN1 D G S B ATOMN L=35n W=140n
```

This line will be parsed by the RandomSPICE software which will detect that it is a MOSFET device (given by the initial 'M' character). It will keep the node connections intact, but determines which type of model to used from the 6th element (`ATOMN`). This tells the software to replace the device with the atomistic NMOS devices; it is possible to also use the token `UNIFN` in the user wishes to use the uniform (non-atomistic) devices for comparison purposes. This line will be replaced in each output netlist with, in earlier releases of RandomSPICE, with a randomised subcircuit such as the following:

```
XMN1 D G S B SUBMN1
.SUBCKT SUBMN1 SUBMN1_0 SUBMN1_1 SUBMN1_2 SUBMN1_3
M_SUBMN1_1 SUBMN1_0 SUBMN1_1 SUBMN1_2 SUBMN1_3 NCH56 L=3.5e-08 W=3.5e-08
M_SUBMN1_2 SUBMN1_0 SUBMN1_1 SUBMN1_2 SUBMN1_3 NCH123 L=3.5e-08 W=3.5e-08
M_SUBMN1_3 SUBMN1_0 SUBMN1_1 SUBMN1_2 SUBMN1_3 NCH17 L=3.5e-08 W=3.5e-08
M_SUBMN1_4 SUBMN1_0 SUBMN1_1 SUBMN1_2 SUBMN1_3 NCH192 L=3.5e-08 W=3.5e-08
.ENDS SUBMN1
```

The four model cards associated with the four transistors randomly chosen from the library (NMOS numbers 56, 123, 17 and 192 in the above example) would be appended to the end of the netlist. In later releases, with the ability to use libraries containing multi-unit widths, the above subcircuit would be replaced with a single transistor.

RandomSPICE allows operating parameters to be specified both through command-line interpretation and via XML; in this thesis it has been used exclusively through command-line interpretation as this was the only option in the earliest releases. The primary command line options of importance are outlines in Table 2.6.

[11]Each unique transistor in the output netlist includes a complete BSIM4 model card, which is ≈ 6,800 bytes and 300 lines long.

Table 2.6: RandomSPICE command-line options

Option	Definition
`-f <string>`	Specifies the input netlist filename
`-n <int>`	Specifies the number of output netlists to create
`-r <int>`	Specifies the random seed for model selection
`-d <string>`	Specifies path to the required compact model library
`-S <string>`	Specifies which type of SPICE (ngspice or HSpice)
`-u`	Forces uniform-only simulation

2.6.5 A Variability-Aware Design Flow For VLSI Circuits

One of the primary goals of the nano-CMOS project as a whole was to attempt to create a method for variability-aware analysis in digital circuits. Whilst it is possible to simulate VLSI circuits using SPICE, the number of transistors involved and the vast number of test criteria needed to analyse a circuit mean such simulations are neither realistic nor practical. As such, much on-going research by other project partners involves investigating methods by which the statistical variability models could be included in design tools used further up the hierarchical tool-chain, illustrated in Figure 2.13. This has involved investigating whether techniques such as statistical static timing analysis (SSTA) can be adapted to include the impacts of device variability, allowing larger circuits to be simulated using commercial tools.

The focus at York has been on developing tools that create and optimise the circuits which fit into the lower end of the hierarchy, focussing on circuit simulation software and standard-cells. Whilst some research has been carried out on larger circuits, it has still been within the domain of SPICE simulation using the extracted compact models.

2.7 Effect of Intrinsic Variability

Intrinsic variability has already had significant adverse effects on MOSFET scaling and integration. To date it has been seen to have negative impact on the yield and reliability of SRAM, it causes timing uncertainty logic circuits, and limits the progress in the scaling down of supply voltage with the knock-on effect of exacerbating the power dissipation problems that already exist in bleeding-edge microprocessors [36].

The observations that have already been made by the Device Modelling Group at Glasgow on 35 nm simulated devices reveal how significant the effects of intrinsic parameters can be; these results from pioneering simulations were later ratified with detailed experiments on actual devices. The random dopant fluctuations lead to sizeable differences between devices,

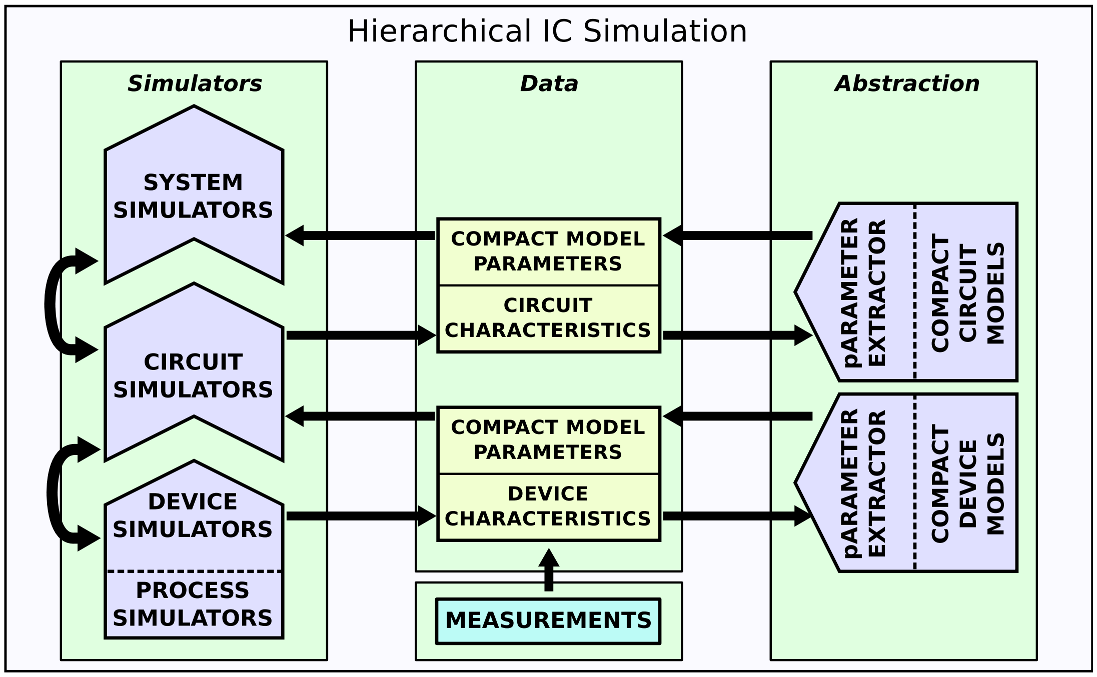


Figure 2.13: A hierarchical simulation methodology necessary to capture the impact of intrinsic variability on design [151].

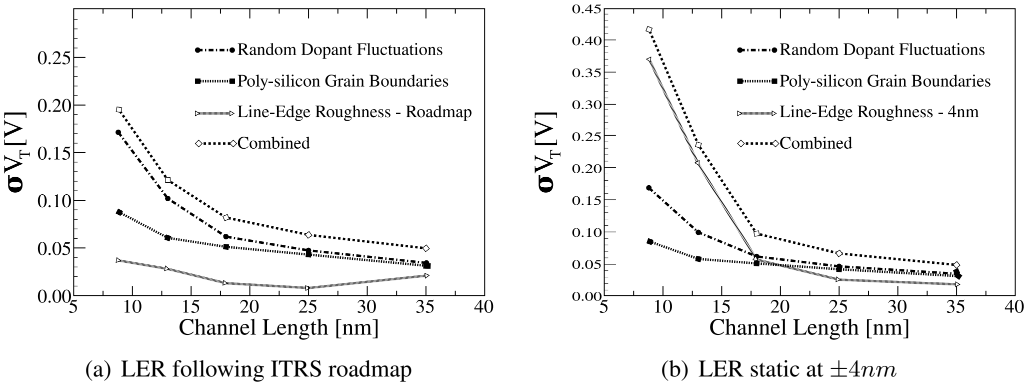
whilst the microscopic scale of the device means that the trapping of a single electron within the channel region can lead to drive current fluctuations in excess of 15%. Such fluctuations will lead to increased time-domain variability and increase noise levels, leading to uncertain operation in VLSI designs. When other intrinsic variations are also considered, including line-edge, surface and interface roughness, it becomes evident that 'identical' transistors at this scale will have significantly different geometries and pattern definition thus variations in gate-tunnelling, quantum confinement and surface/bulk mobility. Assuming a future device with fault-less manufacturing was possible, the intrinsic effects will still create devices with such significant variation that conventional design topologies would likely fail [22].

2.7.1 Impact on Threshold Voltage

The impact on threshold voltage caused by the primary causes of intrinsic variability in a set of 200 35 $\times$ 35nm devices as simulated by the atomistic simulator are outlined in Table 2.7. The most significant contributor to V_T variation at this channel length is caused by RDD [139]. It had traditionally been assumed that the distribution of V_T was symmetrical, following a Gaussian distribution. However, larger simulations based on 100,000 devices at 35nm and

Table 2.7: Threshold Voltage Variability caused by Intrinsic Parameter Fluctuations in a 35 × 35 nm Atomistic-Simulated Device [139]

Fluctuation	V_T	σV_T
Random Discrete Dopants	133 mV	33.2 mV
Line-Edge Roughness	126 mV	19.0 mV
Oxide-Thickness	122 mV	1.8 mV
RDD & LER Combined	126 mV	38.7 mV



(a) LER following ITRS roadmap (b) LER static at $\pm 4nm$

Figure 2.14: Threshold Voltage variation due to intrinsic parameter fluctuations considering different LER predictions [22]

140,000 and 13nm were later carried out by the DMG to more accurately assess the shapes of the distributions of parameter fluctuations due to RDD. These simulations demonstrated that the distributions are asymmetric, with the asymmetry increasing at the smaller channel length, confirmed by non-zero values for the skew and kurtosis of the distributions: the resultant distribution can be closely fit to a Pearson Type-IV distribution; this has significant consequences when the V_T of devices in the extreme tails of the distribution are considered [138].

The impact on V_T due to the different causes of intrinsic variability illustrated in Figure 2.14, which compares the predicted threshold voltage variation if improvements follow the ITRS roadmap for LER improvements, to that if LER remains at its current rate of $\pm 4nm$. If technological improvements in the mask cannot be made, LER will become the dominant cause of intrinsic variability once the channel length falls below $15nm$ [22].

One key consequence of V_T variability is the impact on delay. In digital circuits, a higher drain current results in faster capacitor charging and consequentially higher frequency of operation. The delay time for a switching MOSFET is indicated in Equation 2.23, in which C is the device capacitance, V_{DD} the supply voltage and I_D the drain current. In the expanded equation the effect of an increase in V_T can be seen; any increase in V_T will result in a reduction in I_D and lead to an increase in delay time [146].

$$t_d = \frac{C|V_{DD}|}{I_D} = \frac{2LC}{W\mu_{eff}C_{ox}(V_{DD} - V_T)^2} \tag{2.23}$$

2.7.2 Impact on Yield

Since integrated circuits were first created, there is a real-world reality that only a certain percentage of the die upon a wafer are functional. Historically, random imperfections and defects in the wafer and in the subsequent fabrication processes result in a certain number of failed die. The yield of a process is the percentage of the functional devices: [163]

$$Y = \frac{\text{Number of Working Die}}{\text{Number of Total Die}} \times 100\% \tag{2.24}$$

The impact of stochastic variations was already becoming a significant issue when the 90nm technology node was introduced around 2002. At this scale a standard deviation in threshold voltage of 40mV was typical, which statistically resulted in a threshold voltage difference of over 250mV between two identical transistors in a memory cell per million cells [161].

Circuit Optimisation

One method to improve the performance and reliability of circuits when subject to the effects of variability is the careful optimisation of device dimensions and topologies. Whilst traditionally transistors are sized to optimise for a number of criteria, such as minimal delay, low power and balanced rise/fall times, the techniques used do not account for the fluctuations that can occur. The optimisation technique described in the thesis attempts to identify, through the use of an evolutionary algorithm, if variability in circuit performance can be minimised through optimisation of transistor dimensions. The V_t variability that occurs at a single transistor level manifests itself as a spread in the characteristic delay, switching power and leakage currents through complete circuits.

Redundancy and Fault-Tolerant Techniques

An alternative technique is based on the assumption that certain transistors or cells will have problems - be it poor performance, intermittent instability or complete failure (such as stuck-on or stuck-off faults in logic and memory cells). The use of fault-tolerance techniques, redundancy and error correction techniques are well established in electronics and computation, although generally they have existed at a more macro-scale than will likely be needed if predicted levels of variability are realised in the sub-30nm nodes.

For many years microprocessor manufacturers have increased their effective yields by carefully verifying the functionality of devices in the testing phase and disabling processing cores and blocks of cache memory which are defective, selling the reduced-functionality resultant chip as a lower cost product, although to date the defective areas are largely a result of wafer and manufacturing imperfections. This process may still be an effective way of removing non-functional and substandard blocks that occur due to intrinsic variability in future devices, however it is likely that many more defective units will exist and thus a much finer scale of block-scrappage will be required, adding significant complexity to the design of the chip to allow for this and a significantly more thorough testing and validation process. Alternatively, redundancy, voting and error-correction techniques may be used more extensively throughout the chip, however the increase in transistor count and delay that these processes will add negates the benefits of the reduction in transistor scale.

2.7.3 Techniques to Limit Impact of Variability

The main semiconductor design and fabrication companies are having to adapt their design processes in an effort to overcome the effects of variability. Intel are confident that device scaling can be continued in line with Moore's law, allowing ICs with several billion transistors. However, within these devices a significant number of transistors will 'fail' in manufacture due to parameter variations with several more failing over the early lifespan of the device due to temporal variations. These faults may be either intermittent or permanent in nature, resulting in the need for dynamic on-chip self-test, error-detection and reconfiguration strategies or other methods to adapt to the variations. This will necessitate some form of on-chip redundancy resulting in an increase in the number of transistors needed to perform a function. For example, multi-bit error correction codes (ECC) can be used within RAM arrays to improve yields. Such strategies can be used successfully with bit-error rates as high as 0.01% but cannot cope with failure rates higher than 1% as is expected in sub-20nm length devices, thus new techniques for designing reliable systems on unreliable technology are needed [66].

2.7.4 Body-Biasing and Adaptive Supplies

One technique that has been used in design to minimise the impact of variability is the use of body-biasing techniques, in which devices have forward or reverse body bias applied. Applying a reverse body-bias can reduce the sub-threshold leakage, generally at the expense of operating frequency. Alternatively a forward bias can increase the frequency of the cell at the cost of a moderate increase in leakage power [33].

A similar technique is to use adaptive supply voltages. This can be on a chip-wide scale or a more localised scale, with different functional blocks operating at different voltages.

Material Techniques

One method already discussed to help reduce the impact of intrinsic variability is the use of different materials in the construction of devices. One of the most promising emerging technologies at withstanding device variability is the use of ultra-thin body (UTB) silicon-on-insulator MOSFETs. Such devices can tolerate a much lower dopant concentration within the channel region than conventional designs, thus lowering the impact of random-discrete dopant induced variability. Working UTB MOSFETs have already been demonstrated in laboratories with channel lengths as low as 6nm, however optimal scaling requires that the body-thickness is in the range of nanometres, where surface roughness at the top and bottom of the Si-SiO$_2$ interface will introduce local variations within the silicon body thickness. Samsudin et al used the 3D atomistic drift-diffusion simulator at Glasgow to create a sample set of 10nm channel length UTB-SOI MOSFET models, and observed that the effects of random dopants within the source and drain regions will be one of the critical sources of device mismatch. However, the simulated 10nm UTB-SOI based SRAM cells actually offered a better static-noise margin than similar results obtained using conventional 35nm MOSFETs, resulting in a more stable overall operation. When scaled down to 5nm - roughly the extent of current SIA roadmap predictions, the results suggest that other technologies may be needed to allow stable operation. Interpolation of the results suggest than SRAM may be functional to around 7.5nm, beyond which alternative technologies such a double-gates MOSFETs will be needed [143, 142].

2.7.5 A Special Case: Impact on SRAM

The one type of device that is most significantly affected by device variability is the SRAM memory cell, which is essential in modern devices when a high-clock speed register is needed such as in microprocessor cache memory. Device variability makes it harder to reduce the device operating voltage (V_{DDmin}) after reducing device scale, which in turn can lead to read and write failures, data retention failures and differential signal failures. As a consequence of the operative voltage the leakage currents are proportionally increased. Within the sense amp the device variability will cause increased input offset voltage in the differential bit-level sensing, and decreased accuracy within the timing tracking circuits. The consequences of all this are that whilst the area is reducing roughly 50% per node (at 65nm node, bit cell area is approximately 0.5μm^2 reducing to 0.25μm^2 in 45nm node), it is difficult to get the performance to scale with it; the V_{DD} is not scaling by the same amount and the leakage is increased [45].

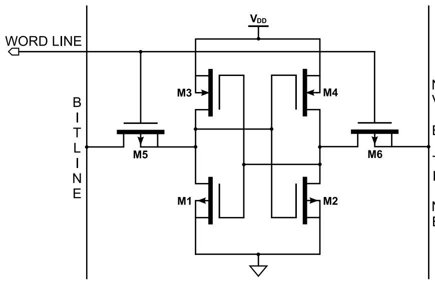
At the present bleeding-edge technology nodes, device variability caused by intrinsic parameter fluctuations have eliminated almost all of the available noise margin in SRAM cells that are based on standard MOSFET designs. In a typical modern system-on-a-chip, which may contain well in excess of 1 billion transistors, it is common for SRAM to occupy 50% of the overall chip area or greater. Thus the overall reliability of SOC designs is highly dependant on the reliability of the on-chip memory. As the stability margin on SRAM cells reduces, individual cells may either fail to hold their state or register a new state correctly. Those cells which are found on the tails of the statistical distributions will fail to work properly; as the distributions widen with technology scaling, increased cell density and V_{DD} reduction, new techniques to either increase redundancy in memory or increase stability margins will be essential [29, 65].

SRAM Construction

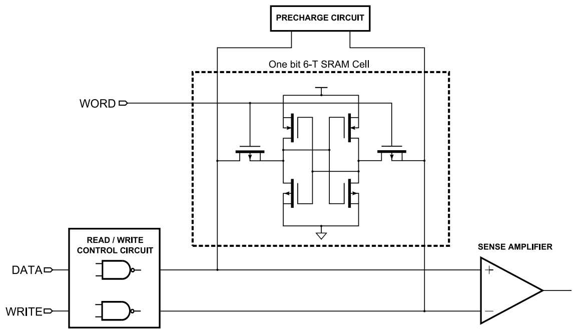
The typical SRAM memory bit uses 6 cross-connected transistors, arranged as illustrated in Figure 2.15a. Additional sub-circuits are used to address the correct memory bit and drive signal lines to usable values; these will typical consist of a precharge circuit, control logic and a sense-amplifier, as illustrated in Figure 2.15b. This arrangement, with a pair of n-FET pass gates driving cross-couple inverters, has been widely adopted as the industry standard embedded memory [65].

Alternative topologies and techniques to reduce variability

Whilst the mass-produced SRAM cells are typically based upon this 6-transistor design, a number of alternative designs have been implemented. Some of these techniques aim to overcome the limitations imposed by the various types of device variability and improve the reliability of SRAM cells. They include read-assist and write-assist features to help overcome the occurrence of read failure and write failure, alternative bit-cell layouts utilising different arrangements to the standard 6-transistor cell, and power management techniques to reduce the leakage current and control temperatures.



(a) Standard 6-T SRAM Schematic



(b) Block diagram of 6-T SRAM including precharge circuit, control logic and sense-amplifier

Figure 2.15: Schematic and block diagram of conventional 6-T SRAM

One such design is the 4T-2R SRAM cell, which replaces the PMOS devices with resistors; this design is significantly smaller in terms of area consumed, and overcomes problems attributed to NBTI, but it does not offer the high speed offered by 6T SRAM [154]. Li et al have developed a method for simulating the effects of transistor failure due to hot-carrier injection, time-depended dielectric breakdown and NBTI. They simulated a complete 6-T SRAM structure, combining a single memory-bit with the necessary precharge, write-control and sense amplifier circuits. Accelerated time-dependant transistor-failure models were then used to determine normalised lifetimes of all the transistors, and those determined to be the most-damaged were replaced with failed-transistor equivalent models in a SPICE simulation. Whilst the research was based around 0.25μm models and was concerned with time-dependant failure rather than the intrinsic-fluctuations covered in this thesis, some of the principles and methods used in the analysis are followed in the simulation of device generation with next-generation models [104].

This chapter has discussed the causes and effects of intrinsic variability on the MOS transistor and CMOS integrated circuits. This includes a review of the techniques to model variability through to the creation of compact models, which are used in the research within this thesis. The following chapter discusses the concepts and application of evolutionary algorithms, search techniques which are inspired by the ideas within Darwinian evolution.

Chapter 3

Evolutionary Algorithms

3.1 Introduction

This chapter discusses the field of biologically inspired algorithms, with special emphasis on evolutionary computation - computational procedures which are inspired by the principles and processes of natural evolution. The basic principles that underpin and describe all such algorithms are outlined followed by a more detailed review of some of the extensions to these basic principles in the most commonly used paradigms. The application of such techniques in the field of electronic circuit design and optimisation with examples of both extrinsic and intrinsic evolution is then covered with a look at important results from the past twenty years right up to the current state-of-the-art developments.

3.1.1 Biological Background

The foundations of evolutionary biology stem from the theory proposed by Charles Darwin in his 1859 work short-titled "*On the Origin of Species*", which provided alongside the root ideas of molecular genetics, the notion of 'survival of the fittest' [52]. The theory suggests that when a population of individuals has to compete for limited resources, those individuals which are best adapted to their environment will be the most likely to survive and reproduce: this phenomenon is named '*natural selection*'. In order for new features and characteristics to be introduced into a population it is necessary that new individuals can have some form of variation, which is caused by a second phenomenon known as '*mutation*'.

In the process of mutation random changes are introduced to individuals within a population with a certain probability, during the reproduction process, which results in new traits and characteristics in the off-spring. If these traits prove to be advantageous the process of natural selection will see the features reproduced, as the adaptation improves the likelihood of reproduction, whilst detrimental features will be gradually be discarded. Basic lifeforms such as plants and singled-celled organisms are able to asexually reproduce, however the vast majority of higher life-forms reproduce through sexual mating [40]. This means all individuals which are able to produce offspring with each other belong to the species, with new species arising once two sub-populations of a species have diversified to the extent such that sexual reproduction is no longer possible, as may occur through geographical separation. It is widely accepted that this fundamental diversification took place at early stages of life on the Earth, leaving no free ecological niches for new species to develop; barring a major-scaled ecological disaster, it is unlikely for entirely new species to evolve [160]. The process of evolution is thus a cycle of reproduction, mutation and selection, and these form the basis of all evolutionary algorithms [87, 26].

The Darwinian principles as discussed above describe the evolutionary process at a macroscopic level. Considering the microscopic level - the method in which information is transferred from parent to off-spring - is the field of molecular genetics. All living organisms contain an encoding which contains a chemical description of how to assemble the organism. The encoding is named a *genotype* and in the real world it contains the complete set of information on how to assemble the physiological and morphological appearance of the organism - its *phenotype* - in the form of a genetic encoding. All the information in the two parent genotypes can be inherited by the produced offspring, with a random amount of '*recombination*' between the two sets of genetic material and a further random mutation.

It is important here to distinguish carefully between the genotype and phenotype, as to where the processes involved in evolution take place. The mutation and recombination take place exclusively in the genome, whilst the selection process is exclusively the subject of the phenotype. Information is passed in one direction only, from genotype and phenotype, and never the other way - the skills and experiences learned by the parent throughout its life do not influence the genome and thus are lost upon its death. Ontogenesis, the self-organising building process which develops the phenotype from the information within the genotype is so complex it is far from being fully understood, however it is known that their is an inherent random element which results in no two phenotypes being exactly the same as each other, even if they share identical genetic material. In brief, portions of the genetic material, held in the structure of deoxyribonucleic acid (DNA), is copied into strands of messenger ribonucleic acid (RNA). The RNA is subsequently translated into one of 22 standard amino acids, which are concatenated to form a large variety of proteins [76]. These proteins have the ability to excite or inhibit the generation of other proteins; this allows the cells to be able to differentiate between one of hundreds of distinct morphological types, such as hormone secretion cells, nervous system cells, skin cells and blood cells, which combine to create the complex organs of the body [14].

The evolutionary process thus occurs when four conditions are satisfied: there is a population of entities, the entities have the ability to reproduce, there is variety in the reproduced entity's, and there is a variation in an entities survival chance that is associated with its environment. In the natural world variety is found within the chromosomes of an entity, resulting in variation in both structure and behaviour of the entity. These structural and behavioural differences result in variation in the survival and reproduction rates of entities, as those which can adapt better to their environment outlive and reproduce more than those which are less fit. Over a number of generations the population will contain more of the well-adapted entities [87].

3.1.2 History of Evolutionary Computation

Evolutionary computation is often considered a subfield of the computer-science study of artificial intelligence, in which a population is iteratively improved based on the primitive biological mechanisms that are found in evolution. The earliest examples of the application of Darwinian evolutionary operators of mutation and selection towards improved computational solutions can be found in the late 1950's, such as the work by Dr. George Box aimed at improving the productivity within the chemical industry, published in 1957 [34]. The lack of available computing platforms to develop such algorithms resulted in relatively slow growth until the mid 1970's. From this period onwards the field has seen steady [1] growth in terms of the number of publications and conferences. The majority of all implementations of evolutionary computation can be shown to descend from three independently developed, although strongly related, approaches. These are genetic algorithms (GA), evolutionary strategies (ES) and evolutionary programming [27]. The original objective of both genetic algorithms and evolutionary programming is the optimisation of parameters within a process; this differs from evolutionary strategies, which were developed with the goal of implementing artificial intelligence by simulating learning processes [160]. The different types of algorithm are discussed in detail in Section 3.2.

[1]For a while, the growth in the number of publications and conferences in the field was exponential [27]

3.1.3 Operation of an Evolutionary Algorithm

Almost all examples of evolutionary algorithms require at least five key functional operations to be specified. Firstly the data structure which represents the genotype must be determined. This must contain all of the relevant parameters that are required for expressing a candidate solution. Then, the mapping or developmental process which creates a phenotype from the genotype must be defined[2]. A process to determine a measure of how well each decoded phenotype performs at its given objective - the fitness function - must be assessed for all the individuals. A selection strategy for choosing members for the new population (and often a strategy for recombination of genetic material from multiple parents) must be implemented. Finally a method for introducing variations into the new population, normally by mutating the genotypes within the new population needs to be constructed. The operation principle of a typical evolution algorithm is demonstrated in the block diagram of Figure 3.1.

Genotype Definition

It is necessary for the genotype to contain all the variables and parameters necessary for the genotype-phenotype mapping process to completely develop the candidate solution. Generally the data types are chosen so that they allow an efficient coding and easy implementation of variation operations (crossover and mutation), such as binary bit-strings and integer or floating point arrays.

Initial Population

The first process in any evolutionary algorithm is the creation of an initial population. Most often this will be a set of randomised genotypes, although this is not always the case. In certain circumstances the first generation may be populated with ‘known’ solutions, which may be for example a set of genotypes which correspond to the current best human-created solutions in the case of an optimisation problem, or the termination-population of a previously run algorithm. The initial population could also be a group of clones of one single genotype, although in general it is desirable to have a level of diversity in any population.

[2]In some very trivial or simple algorithms, such as boolean functions, there may be no difference between genotype and phenotype and thus no mapping. Alternatively the mapping may be performed by a separate predefined structure, such as on an FPGA chip in intrinsic evolution where the genotype is passed directly to the external hardware.

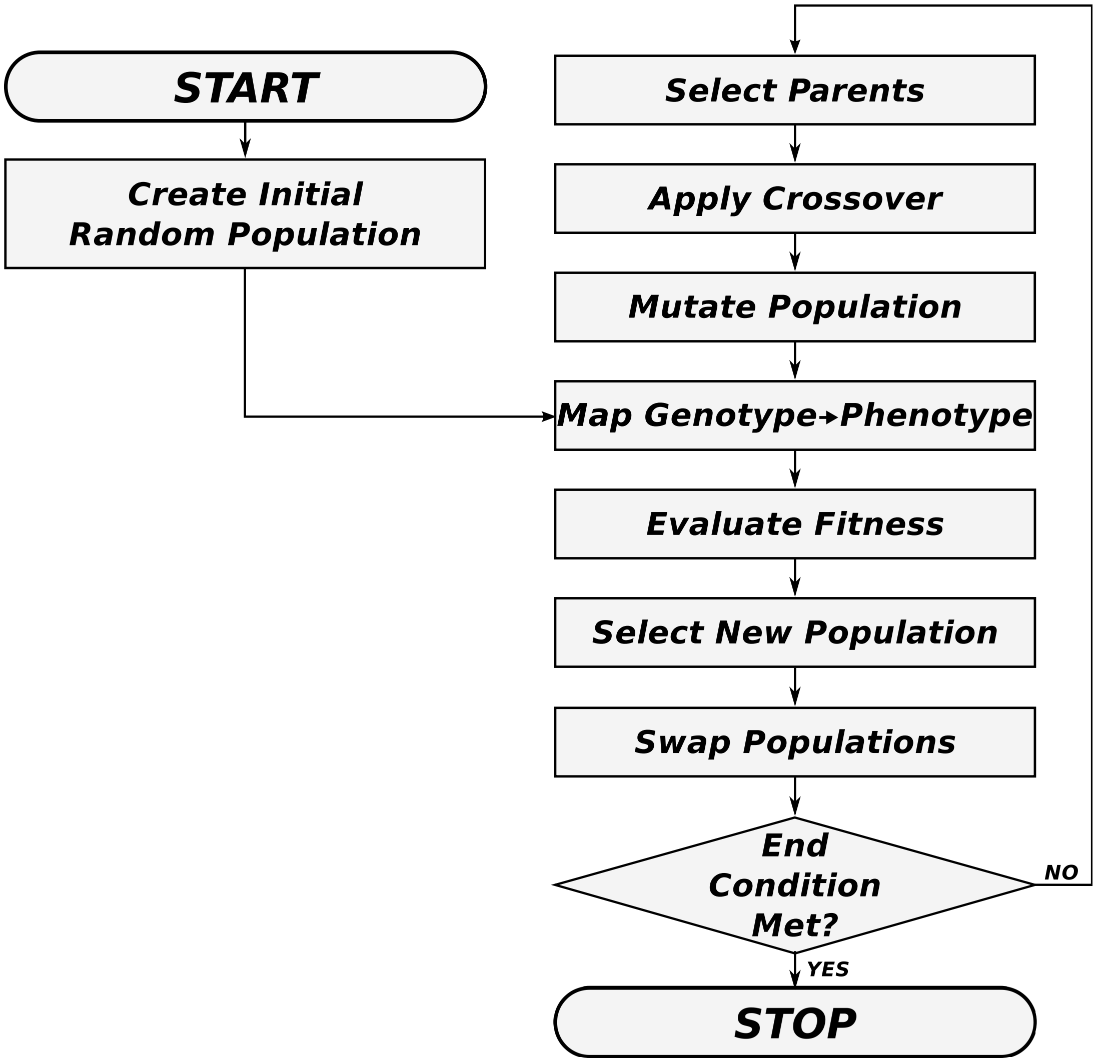


Figure 3.1: Block diagram of the main operations in a generic evolutionary algorithm.

Genotype $\rightarrow$ Phenotype Mapping

The Genotype $\rightarrow$ Phenotype Mapping process is the stage that reproduces ontogenesis in the natural world. It takes a genotype as its input, and produces a phenotype representation of an individual, being a state that is ready to be calculated by the fitness function. In many algorithms the phenotype may take the form of a standard data type or structure. In the case of extrinsic evolution of electronic circuits, the phenotype may be considered the SPICE netlist or hardware description language module, thus the mapping is the process of converting the genotype on which the algorithm operates into a netlist, ready for analysing by the simulation software.

The mapping may be either *fixed architecture* or *self-organising*. In a fixed architecture the structure of the phenotype is predetermined but the values are described within the genotype; this is the case for many parameter optimisation algorithms and most examples of intrinsic electronic evolution, where the mapping toggles the activation of certain function blocks on the hardware device. In a full self-organising system all aspects of the phenotype may be described from the genotype mapping; in the case of extrinsic evolution this may include the choice of components, connection topology and component values within a netlist [160].

Calculation of an Individuals Fitness

The process of evaluating an individuals fitness essentially contains the description of the problem to be solved and a method for rating each individuals effectiveness at solving that problem - the fitness function. This process takes the phenotype as its input an returns a single score, or a set of scores, as its output. Determining the appropriate fitness function for a problem is one of the most important and often difficult tasks as the process requires the implementation of a method for probing the individual to obtain results, and the subsequent fitness landscape can take on a vast number of different shapes. If the fitness landscape is scattered with many local optima there is a risk that the algorithm will prematurely converge and struggle to find the global optima.

The process of testing the individuals and calculating a score from the results is also generally by far the most time consuming in the algorithm, to the extent that in many cases all the other computation time can be considered negligible. Many problems will have multiple objectives to be optimised, thus it may be necessary for the fitness function to either combine and weight numerous scores into a final value using a mathematical function, or alternatively use one of a number of multiple-objective strategies that rank the entire population based on their performance across the whole gamut of objectives.

Selection Operation

The function of the selection strategy is to determine which members of the current generation should either directly survive, produce offspring, or die out. The evolutionary strategy may allow for survivors from a generation to pass through to the new generation unaltered - it is common practice in many strategies to preserve the fittest member of one generation, thus ensure every future generation will have a member at least as strong as any in the previous generation; such strategies are named *elitist* strategies. Parent selection strategies normally select two parents which are passed to the crossover algorithm, and from them a new offspring is created, containing characteristics of both. A number of different selection strategies have been implemented in different algorithms, and it is obviously possible to combine the strategies with different weights. A brief description of some of the most relevant strategies is given below:

- *Tournament Selection* is perhaps the simplest strategy to implement, and also one of the most computationally efficient to calculate. A group of q randomly selected individuals compete in a *tournament*; the fittest is promoted to the new population or as a new parent. The selective pressure of the strategy can be finely tuned by careful adjustment of the tournament size q. Generally all the members are available to compete for selection in all the tournaments, and the fittest is deterministically selected, although variations exist. A tournament between only two individuals is termed a *binary* tournament. Given that no sorting of the population is required the time complexity is $O(\lambda)$ making it suitable for use with very large populations sizes [27, 26].

- *Fitness Proportional Selection* maps the fitness scores across a fixed interval (typically [0,1]) using a linear, or otherwise, fitness scaling function, which allows for the distortion of different fitness regions. This strategy allows the promotion of specific regions of fitness and complex weightings to be implemented if they appear likely to improve the algorithm's performance, however, the most appropriate scaling function is likely to be highly problem specific [160, 26].

- *Rank-based Selection* converts all the fitness scores into a rank value and selects individuals based only on this rank score. Such a strategy is easy to implement and eliminates the need to create a fitness scaling such as used in proportional selection. Unlike in proportional selection, the fittest member of a rank-based strategy will always be assigned the same selection probability, which reduces the likelihood of an individual with a vastly superior fitness score engulfing the population. However, when vast population sizes are used, the act of ranking each and every individual can demand substantial memory and processing time [26].

- *Diversity-based Selection* is a method of selection designed to maintain a diverse population allow more of the potential search space to be explored. A metric for measuring the difference between individuals within the population is introduced and from this selection occurs, penalising crowded areas of the fitness landscape. With such a strategy the mean fitness and number of weak individuals and traded for the diversity of the population, reducing the likelihood of diverging towards local optima and stagnating [160].

- *Disruptive Selection* penalises members of the population with average fitness scores. One such function takes the form $u(x) = |f(x) - f(t)|$ where $f(x)$ is the fitness value for individual x, and $f(t)$ is the mean of all solutions at that point in time; both the very good and very weak individuals will be promoted ahead of average solutions. The use of such a strategy is highly problem dependent and would only work with a limited number of mutation strategies [26].

- (μ, λ)*evolution* strategies use a deterministic selection scheme, in which μ parents create λ offspring through recombination and mutation, with the best μ offspring selected to replace the parents. This is a non-elitist strategy thus the best individuals of future generations can perform worse than the best of the current generation.

- $(\mu + \lambda)$*evolution* is an elitist alternative to (μ, λ) selection in which the best μ individuals from the union of parents and offspring are selected for the future generation. This ensures that a monotonic course of evolution is guaranteed.

- (μ, κ, λ)*strategies* form a group including both the above strategies, in which κ defines the maximum generation lifespan of an individual. For (μ, λ) evolution $\kappa = 1$ and for $(\mu + \lambda)$ evolution $\kappa = \infty$ [27].

- *Multi-Objective Selection* strategies are used when multi-objective algorithms are run with multiple fitness values being produced. Whilst multiple objectives can be combined to create single scores in the fitness function, it is also possible to keep the values separate and this allows a far more diverse set of candidate solutions. The different multi-objective selection strategies are discussed in further detail in Section 3.3.

Mutation Operation

Mutation is one of the two ‘variation’ operators: it adds the element of random variation to the new population. It is generally usual for all the members of the new population to be subject to the mutation operation, aside from *elitist* schemes which preserve the fittest member(s) as is. Different strategies have been used for various data-types within the genotype. A bit-wise mutation, where randomly selected bits of the genotype are inverted, is one of the most simple operations and can theoretically be applied to all computed algorithms. However, when the genotype uses a non-binary encoding a bit-wise strategy can clearly have a wide range of consequential effects and in certain cases may lead to non-solvable solutions or other problem states.

Generally the mutation strategy will operate using the same data type as individual genes. When integer or floating-point data types are used, the mutation operation may alter a value to a new random value using some defined distribution centred on the original value. Alternatively the mutation operation may select a completely new random value (possibly within a defined range). The strategies have different consequential effects on the new population and the progress of an algorithm; limiting the mutation range will tend to improve the mean fitness score and gradually find local optima yet imposes more restriction on the search space than a wide-range or full-range strategy.

The distribution used within the mutation strategy has been subject to a body of research, where it has been noted that the best performing distribution cannot easily be predicted as it is dependent on the fitness landscape that results from the problem in hand. In a classical EP algorithm the mutation will use a Gaussian distribution. One alternative, such as that used in Fast Evolutionary Programming (FEP) [180], is the Cauchy distribution, which in its one-dimensional form is defined by the equation below:

$$f_t(x) = \frac{1}{\pi}\frac{t}{t^2 + x^2}, \ -\infty < x < \infty \tag{3.1}$$

The shape of the Cauchy distribution, illustrated in Figure 3.2, is initially similar to a Gaussian over the central region, but approaches the axis very slowly such that it has an infinite variance. This extra variance increases the possibility of an algorithm escaping from a local optima, and has been demonstrated to perform significantly better than a Gaussian strategy in multi-modal problems with complex fitness landscapes [181]. Other distribution functions for mutation strategies that have been investigated include the use of Laplace distributions by Montana and Davis [26].

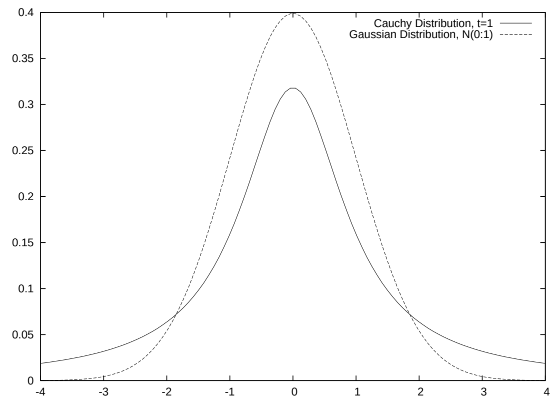


Figure 3.2: Comparison of the Cauchy and Gaussian Distributions

An important parameter of all evolutionary algorithms is the mutation rate (the probability of one bit or gene mutating). For bit-string based genotypes a very common choice is to mutate an average of one bit in each operation (a mutation rate of $1/l$ where l is the genotype length). In early algorithms mutation was often considered a minor or background operator with far greater emphasis placed on recombination; as a consequence very low mutation rates were often used. Schaffer et al researched the problem and found their optimal rates were in the range 0.05 to 0.1% [145], with further analysis on their results by Müllenberg arriving at the formula $N \times m \times \sqrt{n} = 1.7$, where N is the population size, n is the genotype length and m is the mutation rate [126]. However, more recent studies have revealed that the optimal size differs for every different problem and over the course of the optimisation. These studies have demonstrated that the use of much larger initial mutation rates which dynamically decrease over time, or the use of self-adaptive mutation rates which encode the current mutation rate as part of the genotype, can prove effective [25, 152].

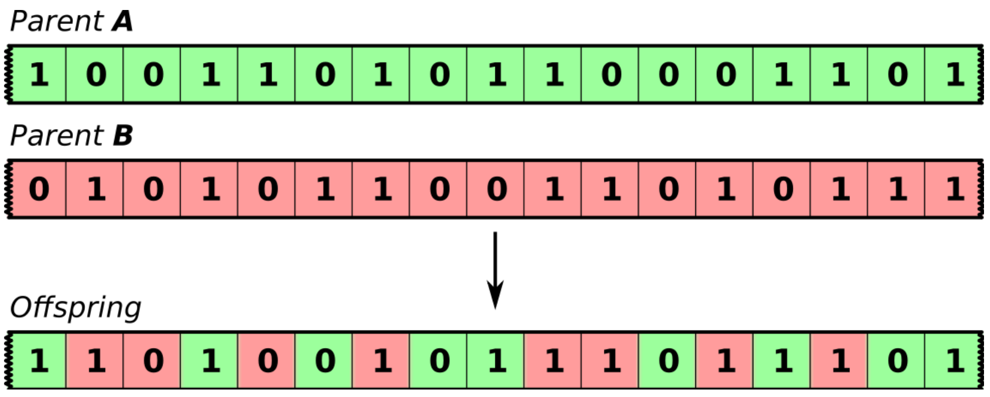
Recombination Operation

The second variation operator that is used in evolutionary algorithms is recombination or '*crossover*' - the way by which offspring genotypes are assembled from two or more parents. Numerous different implementations of crossover are possible dependent on the data types used in the genotype; different methods can have drastically different effects on the offspring, as illustrated in Figure 3.3. The proportion of parent string which undergo the crossover operation is determined by the *crossover rate*, although there may be additional rates depending on the recombination method [26]. The most simple form is a one-point crossover, in which the genotypes from two parents are combined into a single offspring genotype. A crossover position is randomly selected, with the resultant genotype formed by the concatenation of the substring of parent A up to the crossover point with parent B from the crossover point. Some of the most frequently used alternative crossover strategies are outlined in the list below [27]:

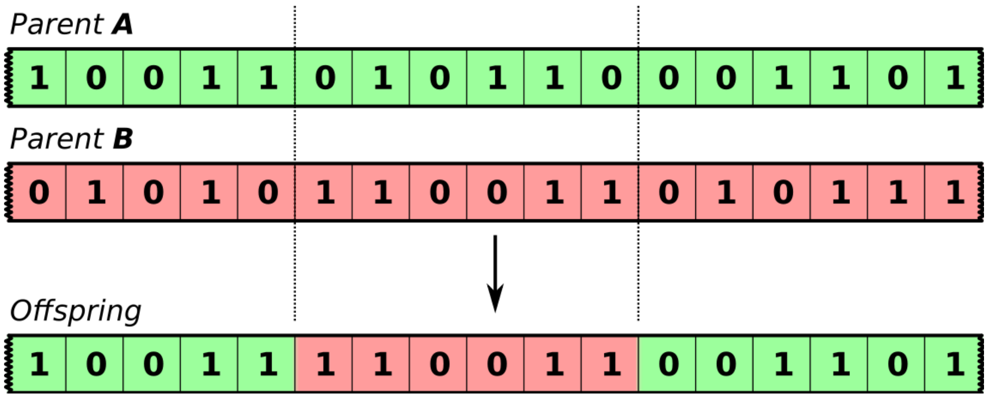
- *Uniform crossover* operations do not have predetermined crossover points; instead the decision to make a breakpoint is made independently at each position. In its simplest form, with a binary genotype, the crossover will randomly select each bit or gene of the offspring genotype from either parent A or parent B, as shown in Figure 3.3a. This can be neutrally weighted, or one parent can be given dominance at a predetermined rate. Alternatively the likelihood of switching the source parent can be below $p = 0.5$ at each position. Optionally an alternate offspring can also be created from the inverse selection, thus for each given gene parent A will contribute to either offspring C or D, and parent B the other [32].
- *N-Point crossover* arranges the genomes and splits them at *N* randomly selected crossing points. The offspring are then composed from alternating sections of the genome pair. Figure 3.3b shows an example of 2-Point crossover.
- *Arithmetical crossover* is used with integer or floating-point value genes and selects a distribution between the respective value from both parents using a chosen distribution; this can affect all genes or only a randomly selected number. In *guaranteed average crossover*, as shown in Figure 3.3c, the value is always the mean of the two values. In *geometrical crossover* the value is calculated by the square root of the product of the two parent values [114].
- *Multi-parent recombination* creates offspring from more than two parent genotypes, using adapted versions of crossover strategies such as those listed above.

Termination Criteria

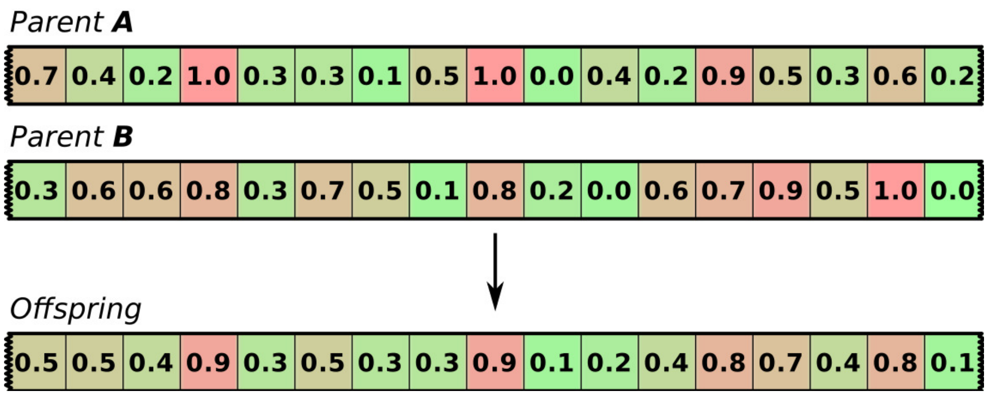
A final consideration to give to any evolution algorithm is if and when to terminate. For certain problems, there may be a single, correct solution, at which point the algorithm can terminate. An example of this is the creation of boolean logic functions where the only fitness objective is the functionality of the circuit - once a design is realised that perfectly realises the target truth table, the program can terminate. For many problems, particularly those focussed on optimisation, there may be no perfect solution, and as such termination will be desirable either after a predetermined amount of time (or number of generations), or when a certain quality



(a) Uniform Crossover



(b) 2-Point Crossover



(c) Guaranteed Average Crossover

Figure 3.3: Examples of some of the frequently used crossover operations in evolutionary algorithms

Table 3.1: The Implementation Differences between the primary forms of Evolutionary Algorithm [47]

Algorithm Type	Representation	Evolutionary Operators
Genetic Algorithms	Originally binary, also real-values	Mutation, recombination and selection
Evolutionary Programming	Real-values	Mutation and $(\mu + \lambda)$ selection
Evolutionary Strategies	Real-values and strategy-parameters	Mutation, recombination and $(\mu + \lambda)$ or (μ, λ) selection

of solution has been realised, or when a certain amount of time has passed since the last improvement. The setting of termination criteria is very problem specific, and care must be taken to balance allowing enough time for acceptable solutions to be found against allowing too much time such that the population spends much time in a state of stagnation. As computation resources become cheaper and more available there is arguably less need to concern about setting specific termination criteria, provided the algorithm can be terminated by a user without losing the data relating to the best solutions found to date.

3.2 Types of Evolutionary Algorithm

This section outlines the principle operation and differences between the main strands of EC algorithm: this covers the three original forms - the Genetic Algorithm, Evolutionary Programming and Evolutionary Strategies - along with the more recently developed fields of Genetic Programming and its descendant Cartesian Genetic Programming. The key implementation differences of the original forms are given in Table 3.1.

3.2.1 Genetic Algorithm

Genetic Algorithms (GAs) were established by John Holland, with the theoretical foundations and early applications explored in his 1975 book "*Adaptation in Natural and Artificial Systems*" [74]. The fundamental principles are based on the ability of a simple representation to encode a complicated structure, with simple evolutionary transformations improving the structure. In a GA these representations are akinned to what is named a *chromosome* within biology, and a group of chromosomes forms a *population*. Selection, mutation and cross-over operations are performed on the population with the aim of improving an individuals *fitness*, being the measure of how well it performs at the required function.

One adaptation of standard GAs that has proved effective in the field of optimisation within complex landscapes is the Parallel Genetic Algorithm (PGA). In a PGA, the global population is split into distributed subgroups, of which the best members are sent to neighbouring subgroups every k generations. If a sub-population is found to stagnate for a number of generations, one individual within it will switch from following the GA to attempting a local hill-climbing strategy (see Section 3.4.1) [125].

3.2.2 Evolutionary Programming

Another set of evolutionary computation paradigms is Evolutionary Programming (EP), first defined by Fogel in 1964, which differs from GAs in its top-down approach to optimisation. EP was created as an alternative approach to artificial intelligence, which progressively created organisms of increasing intellect over time as opposed to being reliant on heuristics. In a GA the concept of different building blocks within the genome is key, which comes with the implicit assumption that the fitness of a member can indeed be described as a function of different block parts of the genome. In contrast in EP each member is judged solely on its final fitness score and no effort is made to subdivide and extract building blocks from the genome, thus crossover or recombination operations do not appear in standard EP with variation coming from the mutation operator alone [26].

As first defined, EP involves a population of finite state machines, to which an 'environment' is presented as a sequence of symbols. The output symbol is observed and compared to the next input symbol, with the differences accumulated to provide an overall fitness score. Offspring machines are mutated through the randomised use of one of five mutation modes: change an output symbol, change the initial state, change a state transition, add a state and delete an existing state [62].

3.2.3 Evolutionary Strategies

Evolutionary Strategies (ES) were developed independent to GAs and EP by three students, Ingo Rechenberg, Hans-Paul Schwefel and Peter Bienert, in 1964, when working on a method to minimise drag on a flexible plate within a wind tunnel. They had employed an iterative search strategy in which individual folds were adjusted one at a time, however discovered that the search strategy always got stuck at a local minimum, a S-shaped fold, never realising the optimal form of a flat plate. By introducing small random changes, the local minimum could be bypassed - and combining this mutation with a decision to select the mutated design or step-back, ES were born [26]. The early ES were all $(1 + 1)$ strategies, such that there is just one parent and mutation, and the mutated individual is progress provided it scores as good as the parent or better. Later computer simulations into the theory led to multimembered strategies based upon continuous decision variables, with $\mu > 1$ parents and $\lambda > \mu$ children (see 3.1.3) [147].

3.2.4 Genetic Programming

Genetic Programming can be considered as specialisation of a GA, however the design of the representation and genetic operator leads to a conceptually different approach to solving EC problems. It is a domain-independent method that evolves populations of computer programs to solve given problems. The objective is to create an automatic system, accepting a high-level statement of a problem as its input and creating some form of computer program to solve the problem without human-intervention in the programming. A large number of randomly created programs forms the initial population, to which analogues of the Darwinian principles of survival-of-the-fittest and the operations of sexual recombination and genetic mutation are applied [26].

One of the first documented examples of a system designed to evolve computer functions was described by Nichael Cramer in 1985. This system extended the algorithmic programming language, PL, which allowed four basic operators: increment, zero, loop and goto, and added to these a set operator (which sets a first variable to the value of the second) and a block operator (which groups to other statements as arguments then performs them sequentially). This system was applied to evolve a two-input, one-output multiplier using a population size of fifty over thirty generations. Despite the very small test size the system produced the desired function 72% more often than a flat-fitness control sample[50].

The first runs of what is conventionally accepted as a GP were in 1987 with the system being first described in detail by John R Koza in his 1988 patent filing *Non-Linear Genetic Algorithms for Solving Problems*[94]. The programs that are generated within the algorithm were originally implemented in the LISP programming language which represents programs as data structures; more recently some of the syntax and tree-structure of LISP language has been retained whilst the programs themselves have been implemented in many other programming languages [26, 89].

A program in this sense is an entity which has data inputs, performs computations and produces data outputs, where computations can be arithmetic calculations, conditional computations or iterative and recursive loops. More elaborate programs can also store results in a memory cache, allowing sections can be divided and reused as subroutines with the passing of dummy input variables and return values. One form of code reuse has been given the name

Automatically Defined Functions [90]. Automatically Defined Functions (ADFs) are a method of exploiting modularity that may exist within a problem with the goal of improving GP performance. Evolved simultaneously with the main program, ADFs are subroutine blocks that can be reused within a solution. The approach was first suggested by Koza and Rice who observed significant performance enhancements in many different scenarios where the evolved solution had multiple instances of blocks of code [88].

Evolutionary Module Acquisition (EMA), described by Angeline and Pollack, is an alternative approach towards the implementation of ADFs which helps protect promising partial solutions and can speed acquisition time. Module acquisition in this manner allows promising partial solutions to be protected from the stochastic nature of the evolutionary operators. In this approach a randomly selected group of program elements are frozen from manipulation and the reproductive advantage this gives to the individual is evaluated. These frozen groups will be replicated "as is" into subsequent offspring. This is done by adding two new operators into the reproduction process: *compress* which selects a module to freeze from manipulation, and *expand* which has the opposite effect. These operators have no effect on the fitness of the evolved results, instead they help ensure that fit sections can be preserved in possible offspring [17].

An extension to the principle is the *atomisation* operator which represents a compressed module as a new component within the genotype, allowing additional compression of modules and the possibility of modules containing modules ultimately leading to a hierarchical arrangement. Both ADFs and EMAs have demonstrated the ability to improve algorithm performance in a number of situations where the problem can be split into modules and the principles of code reuse have been widely researched since their introduction [168, 17].

Cartesian Genetic Programming

Cartesian Genetic Programming (CGP) is a graph-based genetic programming paradigm within which the genotype is represented as a list of integers and functions are linked by connections, originally developed from a representation used for the evolution of digital building block circuits. The genotype is a string of integers which represent both the functions and interconnections between graph nodes, inputs and outputs [117]. One of the key distinctions between CGP and other forms of genetic programming is that CGP can encode non-connected graphs where it is not possible to walk between all of the pairs of nodes; however, at least one connected subgraph is always encoded. The presence of disconnected subgraphs (which can be altered or mutated without affecting the connected graph) implies a many-to-one genotype-phenotype mapping [115].

CGP itself has now been applied to a number of different fields in the field of Electronic Evolution. When applied to the task of evolving boolean 3-bit multiplier problems one of the evolved solutions used only 21 gates which is 20% less than conventional designs [115]. Other applications include analogue & digital filter design [177], and further afield to image processing and evolutionary art, artificial life and bio-inspired developmental models and molecular docking [116].

One important aspect of CGP which distinguishes it from other algorithms is the presence of redundant genes, which is analogous to junk DNA. Individual genes can themselves be turned on or off by mutation operations during the evolution, and it has been demonstrated that these "junk" genes offer beneficial roles within the evolutionary search space. Studies to evaluate the effect of the inherent redundancy within CGP have concluded that the best performance in terms of computational efficiency have occurred in scenarios where there are extremely high levels of redundancy [116].

Standard CGP does not include the use of ADFs (although it has outperformed traditional GP with ADFs on a number of occasions). Recently a new implementation named Embedded CGP (ECGP) has been developed which includes ADFs based on the EMA approach, allowing modules to be constructed and called by the main CGP code [168]. Another implementation of CGP has recently been used by Gadja and Sekanina with the specific goal of reducing the number of transistors in digital circuits using a gate-level evolutionary design, extending from earlier work by Miller et al, with early results highlighting the computation efficiency of the search method over Koza's GP method [67].

3.2.5 Hybrid Methods

Hybrid evolutionary systems are those which enhance a standard evolutionary method with some addition domain knowledge, problem-specific heuristics and possibly existing algorithms with the aim of improving performance (either of the existing algorithm, the evolutionary system or both). In the cases where an existing, non-evolutionary algorithm is to be hybridised, the technique will generally be to preserve the algorithms current encoding and positive features whilst incorporating relevant crossover and mutation operators. An alternative class of hybrid systems exist within which the EA acts as a component part of a larger system comprising of many other algorithms. One specific form of hybrid algorithm uses Lamarckian inheritance, whereby characteristics or knowledge acquired by individuals throughout their lifetime is in some form passed on to off-spring. Such algorithms are suited only to problems where a reverse phenotype $\rightarrow$ genotype mapping is possible, including any acquired characteristics.

3.3 Multi-Objective Selection

It is not generally possible to accurately rate the performance of a logic circuit, or indeed any electronic circuit, based on a single score. Circuit operating speed and power consumption are generally inversely proportional and these two primary characteristics can each be broken down further. The operational delay between an input-change and corresponding output change will vary based on the prior-state conditions, the output load and the relevant path between input and output. When considered as a whole circuit, numerous different delay characteristics can be considered. Likewise, the power consumption can be separated into static and dynamic power and will also vary between different input and output states and transitions. Whilst an exhaustive sweep of every possible combination of input-and-output transitions at all practicable loads is not a realistic simulation option (especially when several thousand circuits need to be simulated, as is the case within an evolutionary algorithm), it is clear that useful characterisation of a circuit performance must take into consideration a number of different circuit parameters. As such, the problem represents a Multi-Objective Optimisation problem, and its solution will require the use of a Multi-Objective Evolutionary Algorithm (MOEA).

3.3.1 Pareto Optimality

MOEA's are adaptations of traditional single-objective algorithms which provide a number of solutions, which provide trade-offs of performance between objective functions that mutually conflict. Whilst a traditional global optimisation problem will work towards a single solution, the aim of an MOEA is to find a set of trade-offs; rather than a single optimal solution, a set of results which are all considered equally optimal is provided. This set of results forms what is commonly known as a Pareto-optimal front, formally generalised by Vilfrado Pareto in 1893 [133], which can be formally mathematically considered through the following definition [47]:

$$\begin{gathered} x \in \Omega \text{ is } \textbf{Pareto-Optimal} \text{ w.r.t. } \Omega \text{ if and only if there is no } x' \in \Omega \text{ for which} \\ v = F(x') = (f_1(x'), ..., (f_k(x')) \text{ dominates } u = F(x) = (f_1(x), ...f_k(x)) \end{gathered} \tag{3.2}$$

In other words, discovered solutions to a problem are considered to be Pareto Optimal when their individual objective components cannot all be simultaneously improved. If, for a given solution x, there is no alternative solution x' in which fitness is improved in one objective without causing detriment in another. All such solutions form part of the Pareto-optimal front and are considered to be non-dominated. An example of Non-Dominated Front

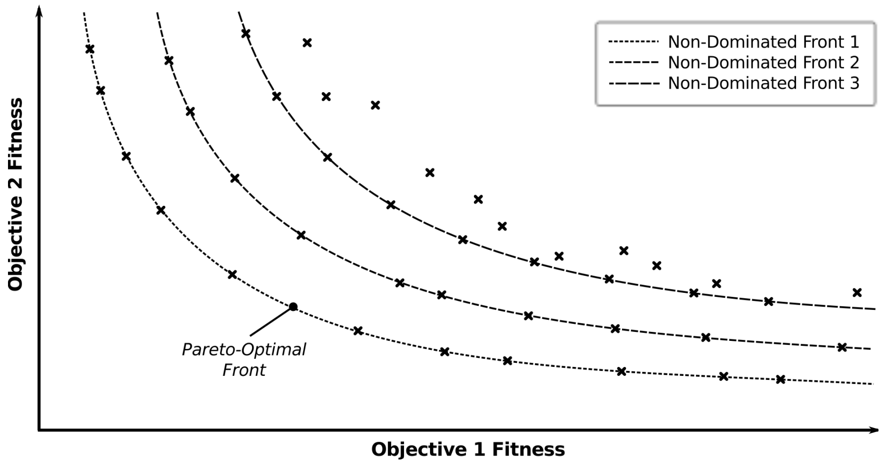


Figure 3.4: An example of individuals in a 2-objective problem. The top three Non-Dominated Fronts, including the Pareto-optimal front, are shown.

(NDF) highlighting the Pareto-optimal front is given in Figure 3.4.

3.3.2 Types of Multi-Objective Evolutionary Algorithms

There exist several publicised techniques and algorithms which are used to find solutions to multi-objective problems via evolutionary search. The earliest examples can be found in work published by Ito et al in 1983 which concerned insulation systems within piping networks, and the Vector Evaluation Genetic Algorithm (VEGA) as proposed by David Schaffer in his 1984 PhD thesis. VEGA was based on a simple GA, however the selection process was modified such that sub-populations were created at the end of each generation based on performance at each different objective function in turn, which were then shuffled to create the new population. Whilst this approach retained the individuals that excelled at individual objectives, well-balanced individuals that didn't excel at any objective would not survive the selection process [46]. Goldberg observed the shortcomings of VEGA and other MOEAs that had been created and pointed out the need to use the ideas of Pareto-optimality to create effective MOEAs, in which the population is ranked such that it moves towards the Pareto front [68].

This suggestion resulted in a number of new algorithms, which shall be considered first generation MOEAs, which can be characterised by the algorithms simplicity. Following comparative studies on these on algorithms, revisions were made which aimed to improve the efficiency, primarily through the use of elitism, and diversity of the algorithms, resulting in the second generation MOEAs. The primary operation of the most widely adopted first- and second- generation MOEAs are outlined in Table 3.2 [46, 47].

Table 3.2: Comparison of the features and characteristics of the most widely adapted first- and second- generation Multi-Objective Evolutionary Algorithms [46, 47]

	Name	Notes
First Generation	MOGA	*Multi-Objective Genetic Algorithm* ranks individuals based on the number of chromosomes in the population by which it is dominated, thus all non-dominated individuals are equally ranked.
	NPGA	*Niched-Pareto Genetic Algorithm* uses tournament selection between two individuals, in which each is compared to a random subset of the entire population. If either is non-dominated and the other isn't, the non-dominated individual is advanced to the next generation, whilst all other results are considered a tie with the selected individual determined through fitness sharing.
	NSGA	*Non-dominated Sorting Genetic Algorithm* ranks the entire population based on non-domination and assigns each rank a fitness value based on population size.
Second Generation	SPEA	*Strength Pareto Evolutionary Algorithm* stores an archive of previously found non-dominated solutions, which is added to at each generation. Each individual of the current population is compared to the strengths of all the archived solutions. The archive is periodically pruned to ensure its size remains below a certain threshold.
	SPEA-II	*Strength Pareto Evolutionary Algorithm II* enhances SPEA with a finer grained fitness measures which accounts for both the number of individuals that are dominated and those with are dominating. It adds a nearest-neighbour density estimation to enhance the efficiency of the search, and amends the archive truncation to ensure boundary solutions are preserved.
	PAES	*Pareto Archived Evolutionary Strategy* combines a historical archive of non-dominated solutions with a $(1+1)$ evolution strategy. To ensure diversity, each solution is given a coordinate position within an adaptive grid, with a procedure to ensure a balanced weight across all locations.
	NSGA-II	*Non-dominated Sorting Genetic Algorithm II* is an improved version of NSGA. It is discussed in detail in Section 3.3.3.

3.3.3 NSGA-II

The Non-dominated Sorting Genetic Algorithm-II addresses many of the criticisms that had been directed at the NSGA approach. It adds elitism to the strategy, which has been shown to improve performance. It is also faster to calculate, reducing the computational complexity from $O(MN^3)$ to $O(MN^2)$, where M is the number of objectives and N is the population size. Finally, it removes the reliance on the sharing parameter which preserved diversity in the original NSGA algorithm [53].

NSGA-II functions operates by ranking and sorting each individual in the population according to its non-domination level. One individual dominates another if, and only if, it scores better in at least one objective function and equal to or better in all others. It is therefore possible for a great number of solutions to fall within the same NDF, and as such there needs to be an additional function to determine which individuals in the same NDF should be given a higher probability to survive. This additional descrimination between individuals in the same front, referred to as niching, occurs through the use of a crowding distance measure, as described below.

NSGA-II has been used for many applications the field of evolutionary electronics, including the synthesis of CMOS operational amplifiers, electro-mechanical system design, antenna design [46] and development of transistor circuits on programmable hardware [160]. It has been selected as the multi-objective algorithm used in this project based on a number of comparative studies with other 2^{nd} generation MOEA studies, which suggest it is generally the best-performing or close to best performing when a moderate range of objective functions are used; other algorithms outperform it when few (under 4) or many (over 10) objectives are used, however for the range likely to be used in the project, the NSGA-II offers competitive performance [185, 47, 160]. The pseudo-code for the implementation of NSGA-II used in this thesis can be found in Appendix B.2.

Crowding Distance Measure

The crowding distance, C_{dist} is a measure of the density of solutions that exist within the vicinity of a particular individual p within the fitness landscape. It is calculated for each of the members within each non-dominated front respectively, based on the average distance to the nearest neighbours of p through each and every objective. The purpose of the crowding distance measure is to guide evolution towards a uniform distribution of results across the Pareto-optimal front, by means of promoting well-spaced individuals over densely packed ones. To achieve this C_{dist} is used as a secondary ranking for the individuals within a given

NDF [160].

3.4 Alternative Optimisation Algorithms

An important characteristic of evolutionary algorithms is the fact that only two of the key operational modules, namely the genotype $\rightarrow$ phenotype mapping, and the fitness function, need to be implemented in a problem specific way. All the other steps in the process are independent of the problem definition; no prior knowledge of how a candidate solution works, nor further information about the optimised system, is included within the algorithm. As such EAs can be considered to be 'black-box' approaches to solving optimisation problems; they are applicable to a vast number of optimisation tasks, although their performance above alternative strategies is never guaranteed.

There are other algorithms which fit the same black-box heuristics of EAs. The most simple are the exhaustive or brute-force search, in which every possible location within the search space is assessed in a linear fashion, and the plain random search, in which different possible locations within the search space are assessed entirely at random. The former strategy is practical in none but the simplest and smallest of search spaces, although it is easy to implement and has the benefit over all other strategies of absolutely guaranteeing finding the optimal solution within a finite, predetermined number of assessments; as such it provides a useful benchmark to which other algorithms may be compared. Aside these simplistic algorithms, there are a further set of approaches which take into account the fitness landscape and aim to advance solutions in a generation-by-generation; these are hill-climbing and simulated annealing.

3.4.1 Hill Climbing Algorithms

Hill-climbing advances the primitive techniques of exhaustive and random searches by including information about the search space. A hill-climbing algorithm takes an initial starting position within the landscape, and attempts to improve on the current position by evaluating the fitness of local neighbours, moving towards the state that appears to be the best. The simple hill-climbing algorithm examines potential new states in a fixed order and selects the first one that improves upon the current state. This clearly adds bias based on the ordering, so a more robust solution is found in the steepest ascent hill climbing algorithm (SAHC). In SAHC, all possible successors are evaluated with the one that offers the greatest improvement selected. A major problem for these basic algorithms is they only find local maxima; once a position is found in which all local neighbours are worse than the current state stagnation occurs. Additionally landscapes with plateaus, in which all local neighbours appear the same, and ridges, in which the target direction of the global maxima is along a narrow ridge, create situations in which the algorithm can get stuck [49].

There are many variations on the steepest-ascent hill climbing algorithm which aim to improve performance, particularly at avoiding stagnation at a local-maxima. Stochastic hill climbing adds a random element to the selection process with the probabilities of the different possible uphill moves weighted by their gradient; this generally converges more slowly than steepest-ascent but can perform better in certain landscapes. First-choice hill climbing generates successors at random until one is found to outperform the current state; this strategy is observed to work well in vast multivariate problems where a state has many thousands of potential successors. Random-restart hill climbing conducts a series of shorter hill-climbing searches from random starting positions with the aim of finding the global optima. Much as with evolutionary optimisation strategies, the success of a hill-climbing algorithm is very dependent - for landscapes with few local maxima, random-restart hill climbing can very rapidly find a good solution, however for a complex, spiked landscape, as is typical with an NP-hard problem, the algorithm will rapidly find strong local maxima but may miss the global maxima [140].

3.4.2 Simulated Annealing

Simulated Annealing (SA), invented by Kirkpatrick et al in 1983 [85], is an algorithm that essential combines hill-climbing with an element of random walk, to attempt to avoid the problem of getting stuck in local maxima, whilst being far more efficient that simply relying on random walk. The name stems from the metallurgy process of annealing in which metals are tempered by heating them to a high temperature, then gradually cooling them. The process is astutely described by Russell and Norvig using the following analogy: Consider a rough, bumpy landscape in which there is a ball, where the aim is to get the ball to reach the lowest point - a gradient descent problem. The process works by shaking the environment roughly, allowing the ball to fall and settle, then repeating the process with decreasing force until the ball has settled at the lowest point and doesn't get dislodged by the shaking [140].

Simulated Annealing is used in multi-variate combinatorial problems where the goal is to minimise the system energy - a global function that is based on all the variables. SA is an extension of Metropolis Monte-Carlo (MMC) simulation, in which information about the shape of the search landscape is learned through a process of initial random sampling followed by iterative small changes. When the small changes result in a decrease in energy, the new state is accepted. If, however, the energy value is increased, the new state is probabilistically accepted based on the Boltzmann Acceptance Criterion (BAC) $e^{(-\delta E/t)}$, in which δE is the increase in energy and T is the 'temperature' of the system, calculated from the percentage of steps that result in a rise in energy. A random value between 0 and 1 is chosen, with the new state accepted when the value is lower than the BAC; the allows the process to escape from local minima. In SA, successive iterations of MMC are run using progressively lower temperatures, with successive temperatures calculated from a cooling schedule. Popular cooling schedules include $T_{\text{new}} = T_{\text{old}} - \delta T$ and $T_{\text{new}} = T_{\text{old}} \times Q$ where Q is a constant below 1.0; the choice of the schedule, the initial temperature and the amount of cooling each iteration need to be carefully selected according to the landscape and parameters of the problem to solve [49]. It is noted that in addition to hill climbing and simulated annealing, there are numerous other optimisation algorithms that have been applied to similar tasks. These include, amongst many others, particle swarm optimisation, quantum annealing and tabu search; for reasons of time and space a detailed analysis and comparison of all these techniques has not been included here.

3.5 Application of Evolution Algorithms

Whilst various algorithms can be demonstrated to find optimal solutions to trivial problems, the potential for EC is far greater when solving non-linear multi-variable optimisation problems. It is particularly in problem areas with stochastic, temporal or chaotic components where EC demonstrates the most potential as conventional methods can either miss optimal solutions or use excessive computational resources, or in certain circumstance conventional methods simply do not exist to solve the problem at hand. In this section a number of different practical problems to which evolutionary computation methods have been applied are discussed, with special focus on the problems associated with electronic engineering design and optimisation.

3.5.1 General

Some of the most common optimisation areas in which evolutionary computation methods have been exploited are in routing, scheduling and packing. Routing problems are examples of combinatorial optimisation which have many real-world applications. One well known example is the Travelling Salesman Problem, which is finding the lowest cost route whilst travelling through all the nodes of a weighted graph. This problem is considered NP-hard and with such problems exact algorithms require exponential amounts of compute time as the number of nodes increases, thus methods to accelerate the discovery of near-optimal solutions, such as EAs, are implemented. Routing has many applications within the field of electronic engineering, scaling from the placement of component and tracks in PCB manufacture to the routing of tracks in semiconductor lithography [26, 87].

Scheduling problems try to best arrange a series of activities to fit within a time-frame whilst considering limited resources and other constraints. Scheduling algorithms have many applications in computer science, and are needed in the design of computer operating systems to efficiently allocate tasks and memory caches, particularly in multi-processor environments [61].

Packing problems generally try to determine the optimal fit of a number of various sized objects into fixed sized containers. Many real-world problems with EE have similar requirements, such as the optimal distribution of sub-circuits within a VLSI integrated circuit, or the optimal distribution of integrated circuits on a wafer in the silicon manufacturing process. Packing problems are also found in communications, for example the optimal distribution of communication channels for various users and bit-rates [26].

3.5.2 Uses within Electronic Circuit Design

A significant number of different applications of EC have been made in the field of electronic engineering; research within this area is often given the name of Evolutionary Electronics (EE). Examples of EE can be separated into extrinsic evolution, in which simulation software performs the analysis to create fitness scores, and intrinsic evolution, in which dedicated hardware, typically based on reconfigurable VLSI chips, is probed to obtain fitness scores; the two domains are illustrated in Figure 3.5. Examples of the creation of topologies and component optimisation for both digital and analogue circuits exist in both the extrinsic and intrinsic domains. A synopsis of notable examples is given in Table 3.3 [183].

3.5.3 Evolution of Analogue Circuits

Analogue circuit simulation in particular is a design area which has many times demonstrated the effectiveness of genetic algorithms. Unlike digital systems, which are generally systematic in their design thus well suited to automation, analogue circuit design has proven hard to create tools for, with most designs drawing heavily from intuition [183]. Despite the advent of digital electronic systems, analogue devices are still widely used in modern electronic devices both at discrete and integrated levels. Oscillators and clock-generators are necessary for all synchronous digital systems, filters and A-D & D-A converters are used extensively in communications and signal processing [11]. Traditional human circuit design methods general use a specific design principle, such as symbolic analysis, or are based on intuition and ad-hoc

Table 3.3: Examples of Evolutionary Algorithms in Electronic Circuit Design

Function	Author	Notes
Analogue Computation	Koza, 1997	Use of GP to evolve analogue computational circuits. The functions were square, cube, square-root and cube root. The circuits used BJ transistors and passive components.[91]
Digital computation	Koza, 1992	GP used to evolve boolean functions, multiplexers & parity functions[87].
	Miller, 1998	Evolution of binary multiplexers, adders and multipliers using CGP. The results included a design for 3-bit multiplier that is 20% more efficient in terms of gates than the best known conventional design[115].
	Langeheine, 2002	Optimisation on actual logic gates on FPTA chip. Correct DC functionality for all gates except X-OR was obtained[101].
	Hadjam, 2007	Multi-expression algorithm used in multi-island cluster approach for accelerating evolution of 3-bit multipliers and other designs[71].
Filters	Koza, 1995	Use of GP to evolve low-pass filter designs using SPICE small signal analysis[94].
	Koza, 1996	Inclusion use of ADF in filter evolution results in the emergence of ladder and elliptical topologies[90].
	Lohn, 1998	Butterworth and practical stethoscope low-pass filter circuits evolved using linear byte-code representation[106].
	Grimbleby, 2000	Use of hybrid GA to evolve bass-band filter problem, using GA to evolve topology and a conventional algorithm for values.[70]
	Koza, 2000	Evolution of generic low-pass filter with a free variable using GP[95].
	Fan, 2001	Bond-graph approach used to evolve topology and component values for high-pass, low-pass and band-pass filters[59].
	Chao, 2005	CGP used to evolve low-pass filter topologies using SPICE simulation[177].
	Wang, 2007	Two-layer GP approach used for evolving low-pass filter and voltage amplifiers[172].

Function	Author	Notes
Integrated Circuits	Koziel, 2003	Use of an adaptive EA to lower the power consumption of CMOS circuits during high-level synthesis, with the goal of reducing average and peak temperatures[96].
	Salomon, 2007	EA used to optimise dual-threshold CMOS circuits from ICSAS test suite at gate level[141].
Operational Amplifier	Kruiskamp, 1995	Synthesis of topology and sizing of 100-dB operational amplifiers using the DARWIN system. Building blocks input, 2nd gain and output stages were defined, from which the best arrangements and sizings were evolved[97].
	Koza, 1996	Automatic design of 5-dB audio range amplifier. Evolved topology contains an identifiable voltage gain and a Darlington emitter-follower output stage[92].
	Koza, 1997	GP used to evolve 96-dB gain operational amplifier. Analysis used SPICE DC-sweep and includes penalty for DC bias and linearity errors. Solution used 25 transistors and made use of ADFs and a parallel island model[93].
	Trefzer, 2006	Used multi-objective approach on FPTA with 11 different fitness criteria[160].
Oscillators	Aggarwal, 2006	GA used to invent oscillators using symbolic analysis, reproduced with SPICE and also with actual components[9].
Mixed Mode	Streeter, 2006	GP used to evolve analogue and mixed mode circuits for which patented human-designed solutions have been made since 2000. Circuits included variable capacitor and cubic signal generator[153].
Transistor Models	Aggarwal, 2006	Design of posynomial models for $0.18um$ technology NMOS transistor[11].
	Li, 2007	Extraction of CMOS device models parameters using an adaptive sampling based GA[105].

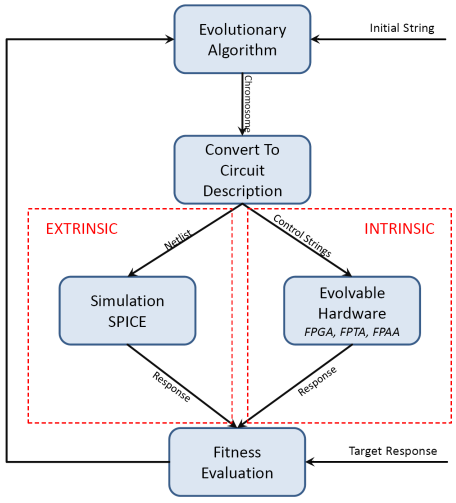


Figure 3.5: Comparison of the Principles of Extrinsic and Intrinsic Evolutionary Electronics

designs. There are exhaustive approaches which can use computational power to tackle the problems, but these rapidly become infeasible when complex active elements are introduced to the design, or when topological constraints are added; automated design and verification of such analogue circuits is not yet a practical option [9].

In a mixed-signal integrated circuit, with both analogue and digital stages, the design of the analogue stage generally consumes a large fraction of the design time whilst being a small portion of the overall circuit. Much of this time is spent after a candidate topology has been selected in the process of iteratively adjusting component values and transistor sizes in order to meet the design specifications, particularly when temperature and other constraints are considered [131].

For analogue circuit synthesis, the available computer-aided design tools almost invariably consist of circuit analysis and simulation packages. Many extrinsic evolution examples have involved the use of the SPICE circuit simulation package, described previously in Section 2.6.1, for the circuit analysis, which are parsed to create fitness values relating to the problem. Each candidate circuit is loaded into the simulation software, which will produce the results of one or more standard circuit analysis operations, including DC transfer, AC small-signal and transient analysis. The choice of analysis will depend on the target function of the circuit, for example basic filters will use an AC small-signal analysis whilst logic functions may use transient and DC analysis.

A major disadvantage of using such simulators within an evolutionary system is that, in general, they treat all circuit components as purely mathematical entities, ignoring physical limitations and interactions that may exist with real-world components. The evolutionary system must thus be aware of any such limitations, which requires checking for violating conditions within evolved circuits and penalising circuits which may simulate with high fitness but would not be realisable in real-world conditions. For example, in a circuit using bipolar-junction transistors, many simulators will allow base-emitter voltages to exceed the nominal 0.7V value and may allow collector current to exceed the maximum value. Whilst the simulator can reveal if these conditions are met, it will still allow the analysis to be performed regardless [183].

A major obstacle faced in the field of EE is the resource requirement when evolving large-scale circuits. Whilst EE algorithms are often able to find adequate solutions to smaller problems reasonably quickly, the large number of individuals and number of generations required to reach convergence in solutions for more complex problems can prove impractical. Consequentially, many efforts have recently been made to accelerate the EE process, including the use of custom simulation [82] and the use of cluster-based distributed algorithms [71].

Circuit Representation

In the extrinsic evolution of analogue circuits different methods of representing a circuit in a genotype have been implemented. Koza et al introduced the *circuit constructing tree* method allowing circuits characterised by cyclic graphs to be encoded into GP trees. The trees contain 'component creating functions' and 'connection-modifying functions', allow circuits to be develop from an *embryonic* circuit. Component creating functions insert a passive or active component into the developing circuit whilst the connection-modifying functions modify the developing topology. This allows, for example, the creation of a series or parallel composition of a particular component [87, 94, 183].

Alternatively a string representation is used to encode the circuit. This may be implemented by assigning each circuit component to a gene within the genotype string, thus the length of the string will determine the number of circuit components. Each gene will contain the information about the type of component (e.g. resistor, capacitor, inductor, transistor) and one or more component values[3] along with its connection nodes. In this manner the topology, circuit size and values are all encoded within the genotype. The use of a string representation is also typical in intrinsic evolution, where in some cases the genotype is passed directly to memory of the reconfigurable hardware board [183].

[3]Active components may require two or more values, for example a FET model may require at very least the device width and length.

Filters

Electronic filter design has been the target objective for the testing of many evolutionary methods and platforms. They are relatively easy to analyse and tend to use recurrent topologies thus are apt for testing ADFs and similar repetitive-structure forming methods. At the same time they are essential sub-circuits in most signal processing and communication tasks and strong designs can have commercial value. Koza used a GP genotype representation to describe analogue low-pass filter netlists which were evolved using SPICE. The fitness scoring function used by Koza has subsequently been used by others to compare efficiency of the evolutionary methods. Koza's group refined their initial tests by including the use of ADFs within the algorithm and later evolving generic filters containing a free variable, allowing one standard design to be applied to a range of target cut-off frequencies [94, 90, 95].

Since Koza's initial work, a number of different other techniques have been used to evolve analogue filter designs. Lohn used a GA with a linear circuit representation to successfully evolve low-pass designs of low and medium complexity, using a simple encoding system with few primitives [106]. Higuchi et al attempted to use intrinsic evolution for the creation of real-time adaptive filters. Aggarwal et al designed a system, named OptimFilt, suitable for evolving optimal coefficients for a transfer-function approximation of filters [10]. Khalifa et al included a nodal admittance matrix based fitness calculation for evolving the topologies and values for analogue low-pass and high-pass filters. By incorporating the fitness scoring function within the algorithm code a running speed far faster than equivalent systems running a version of SPICE, although the accuracy of the circuit analysis is lower. They aim to develop the problem to a multi-objective function including parasitic component effects [82]. Chao Wu adapted CGP to assemble circuits using a LISP-like language for the evolution of 'Koza' low-pass filters [177].

Oscillators

Human-competitive examples of sine-wave oscillators have been developed by Aggarwal et al which avoid the use of SPICE by using symbolic analysis, using the same method used in traditional design. The symbolic transfer function for an evolved circuit is parsed to assign a fitness score, using a first-order model for the active element. Whilst some accuracy is lost with the models used, a key advantage is all the circuits created are interpretable by humans (which is often not the case with circuits evolved and simulated in SPICE). Most of the circuits which are created in this method can still be analysed in SPICE, and the reasons behind those that don't can be resolved [11, 9, 84].

Operational Amplifiers

Noren and Ross used a genetic algorithm to optimise component value and transistor scaling within predefined operational amplifier topologies. The designs optimised included those based on BJT and CMOS transistor models, although the simulation used a basic square-law model rather than the industry standard BSIM3v3 which would add significantly to the processing time [131]. Whilst this produced results with transfer characteristic matching the specifications, it is still sizeable step to produce a commercially acceptable amplifier. Along with the required gain and transfer characteristic, many other factors need to be accounted for, such as bandwidth, slew-rate, noise, power consumption, CMRR, PSRR and overall circuit size. This requires a multi-objective fitness function to be developed, often requiring multiple analysis runs in SPICE. Multiple-objective operational amplifiers designs have been developed using both extrinsic and intrinsic methods [183, 160].

3.5.4 Intrinsic Evolution

A major recent field in the evolution of electronic designs is intrinsic evolution, the practice of directly evolving electronic circuits onto directly onto reprogrammable hardware, allowing the direct sampling of physical circuits without relying on software simulation. Numerous different devices ranging from available widely off-the-shelf ICs and development boards to custom-made ASICs, have been used to research intrinsic evolutions. One of the first examples of intrinsic evolution was by Thompson, with the evolution of a circuit capable of discriminating between two square waves (of frequencies 1KHz and 10KHz), utilising the underlying physical properties of the Xilinx 6126 chip. The field-programmable gate array (FPGA) used was programmed with a program which cannot be explained using conventional design methods, working instead directly on inherent electrical paths within the individual chip used, thus demonstrating that evolutionary methods can be used to produce designs which fall outside the scope of any traditional design methodologies [159].

Another type of device used for intrinsic evolution is Field Programmable Analogue Arrays (FPAAs), which contain multiple analogue functions, programmable using binary patterns. The arrangement and construction is similar to that of FPGA although the functions are considered in terms of analogue rather than digital performance. Zebulum et al used a Motorola MPAA020 switched-capacitor based device to evolve oscillator designs [183]. Another type of FPAA which has been used in evolutionary electronics is the Zetex TRAC (Totally Reconfigurable Analogue Hardware), which contains two parallel sets of ten interconnectable operational amplifiers having various connected components which may be switched on or off, allowing one of eight distinct circuit functions. Flockton and Sheehan used a test board containing 4 TRAC devices and an analogue interface allowing the programming of the devices, analogue signal generation and output signal detection to all be controlled by a PC. They designed a set of building blocks that could be implemented using the test board which could be connected together without risking damage to the components [60].

The Field Programmable Transistor Array (FPTA) is a device specifically created for the evolution and research of analogue circuits at transistor level. A FPTA was designed by Jörg Langeheine within the Electronic Vision Group at the University of Heidelberg. The work arose from Thompson's work evolving a tone-discriminator which exploited the underlying analogue core of the FPGA being used, from which the idea of substrate specifically for configuring transistor-based analogue circuits was drawn. The Heidelberg FPTA consists of a 16 x 16 arrangement of configurable PMOS and NMOS transistor cells in a checkerboard pattern. Each cell contains 20 transistors of different sizes, with lengths of 0.6,1,2,4 & 8 μm and widths of 1,2,4 & 8 μm. The transistors can be connected in parallel, allow any effective unit width between 1 & 15 μm allowing 80 possible size arrangements for each transistor cell. Another FPTA device has been developed by JPL with similar intentions to the Heidelberg device although including configurable capacitors and resistors in addition to transistors [100, 101, 102]. Amongst the circuits successfully evolved using the Heidelberg FPTA are the complete set of logic gates and other basic functions such as comparators. Trefzer used a multi-objective approach for evolving a complete operational amplifier on the device. This used twelve different fitness values, including DC offset, slew-rate, settling time, phase-shift and harmonic distortion in addition to gain. To evaluate all these values five different test modes were used. The evolved solutions included what could clearly be identified as a differential input stage [160].

Major drawbacks of custom designs such as the Heidelberg FPTA are the length of time and costs involved in the design, verification and manufacture of such an application specific chip, given its very limited potential market. Ideally one could create an FPTA using the very latest technologies, however it is unlikely that the most recent technology nodes would be available for such a run; the Heidelberg FPTA is manufactured using a 600 nm process and each device cost in excess of US $5K. Another significant issues is that with the nature of FPTAs relying on switching elements leading to parasitic non-linear resistance and capacitance. Whilst such effects may be exploited by the algorithm (such as in Thompson's frequency discriminator) such results are idiosyncratic and non-portable [158]. A key advantage of intrinsic methods is that all the properties of the silicon may be fully explored. Whilst simulation software only analysis the electronic properties of circuits, intrinsic methods will include any electromagnetic and thermal properties in the device [183]. This may result include properties which are unique to an individual piece of silicon, in a specific environment. This can however also be a disadvantage if the goal is to create generic, market-ready designs.

Intrinsic evolution generally requires a dedicated and often expensive arrangement of hardware in addition to the computing platform limiting the scope and distribution of simulations. Another disadvantage of intrinsic evolution is that the results may be hard to represent using traditional symbolic representations, and they may themselves be limited to the specific individual chip used (thus rendering them of little real-world value). In contrast, circuits which have been extrinsically simulated are generally already presented symbolically and can be implemented and reproduced using real world components [115]. Whilst custom made circuits such as the Heidelberg FPTA are potentially powerful analogue design tools, the time and expense involved in creating such devices limits their availability for research.

3.5.5 Evolution of transistor models

One particularly significant field where genetic algorithms have been used is in the creation of transistor models themselves. Without considering device level variability, transistor models will differ dependent on the semiconductor technology used in fabrication, with different substrate materials, doping concentrations, feature size and fabrication methodology all combining to lead to different IV characteristics, which must be modelled. The overall circuit behaviour is further complicated by interactions between the complex models of individual devices. Not all performance measures have closed-form expressions, and thus require the use of numerical and iterative functions to be determined. Aggarwal and O'Reilly used a GA to create transistor models allowing all the small signal parameters and certain large signal parameters to be expressed as a posynomial function of the transistor width, length and drain current. The models created are significantly more accurate than statistically fitted models [11]. Li utilised a GA with adaptive sampling to automatically extract complete parameter sets for BSIM3 models [105].

This chapter has discussed the history and implementation of the major fields of evolutionary computation, which special emphasis on the multiple objective algorithms that are used in the thesis. This includes a comparison of alternative search techniques applicable to similar optimisation problems, including simulated annealing. The following chapter introduces the software framework that has been *developed to allow for multi-objective algorithms to be used in collaboration with variability-aware transistor models and the SPICE simulation package.*

Chapter 4

A System for Evolving Logic

4.1 Introduction

This chapter describes the system that has been created for the purpose of evolving and optimising logic circuits using evolutionary algorithms. The goal has been to create a framework of tools that allows the creative use of different evolutionary algorithms, and other biologically inspired design and optimisation techniques, incorporating the variability-aware transistor models as part of the circuit evaluation process. By modularising the different functional elements as different 'tools' it has been possible to create a system which allows efficient reuse of code, enabling different algorithms to be effectively inserted into the 'evaluation loop' as required; completely separate algorithms for optimising standard-cell logic and designing novel logic topologies can be used with minimal changes to the rest of the code-base. The framework of tools has been named MOTIVATED, an acronym for ***M**ulti-**O**bjective **T**oolkit for **I**ntrinsic-**V**ariability **A**ware **T**ransistor-level **E**volutionary **D**esign*.

In this section, the different functional blocks that make up the toolkit are described in turn, with specific detail applied to the genetic algorithm used for optimising cell topologies, henceforth known as Simple Genetic Algorithm (SGA) [1]. The toolkit was developed and revised throughout the duration of the nano-CMOS research project by the author and Dr. James Walker; the author created the SGA optimisation algorithm, Dr. Walker created the CGP-based topology design system, with work on the rest of the toolkit shared between the developers.

[1] The choice of the name 'Simple Genetic Algorithm' was chosen at the time of implementation not due to the (lack of) complexity in the algorithm itself, but rather to differentiate it from the CGP-based approach at evolving logic topologies that had also been developed. It should be noted that whilst it is named as a GA based on the original implementation, by the final experiments it is much closer to an evolutionary strategy in function; this is due to the removal of crossover and switch from a tournament to a $(\mu + \lambda)$ selection strategy throughout the course of experiments.

4.1.1 Overview of MOTIVATED

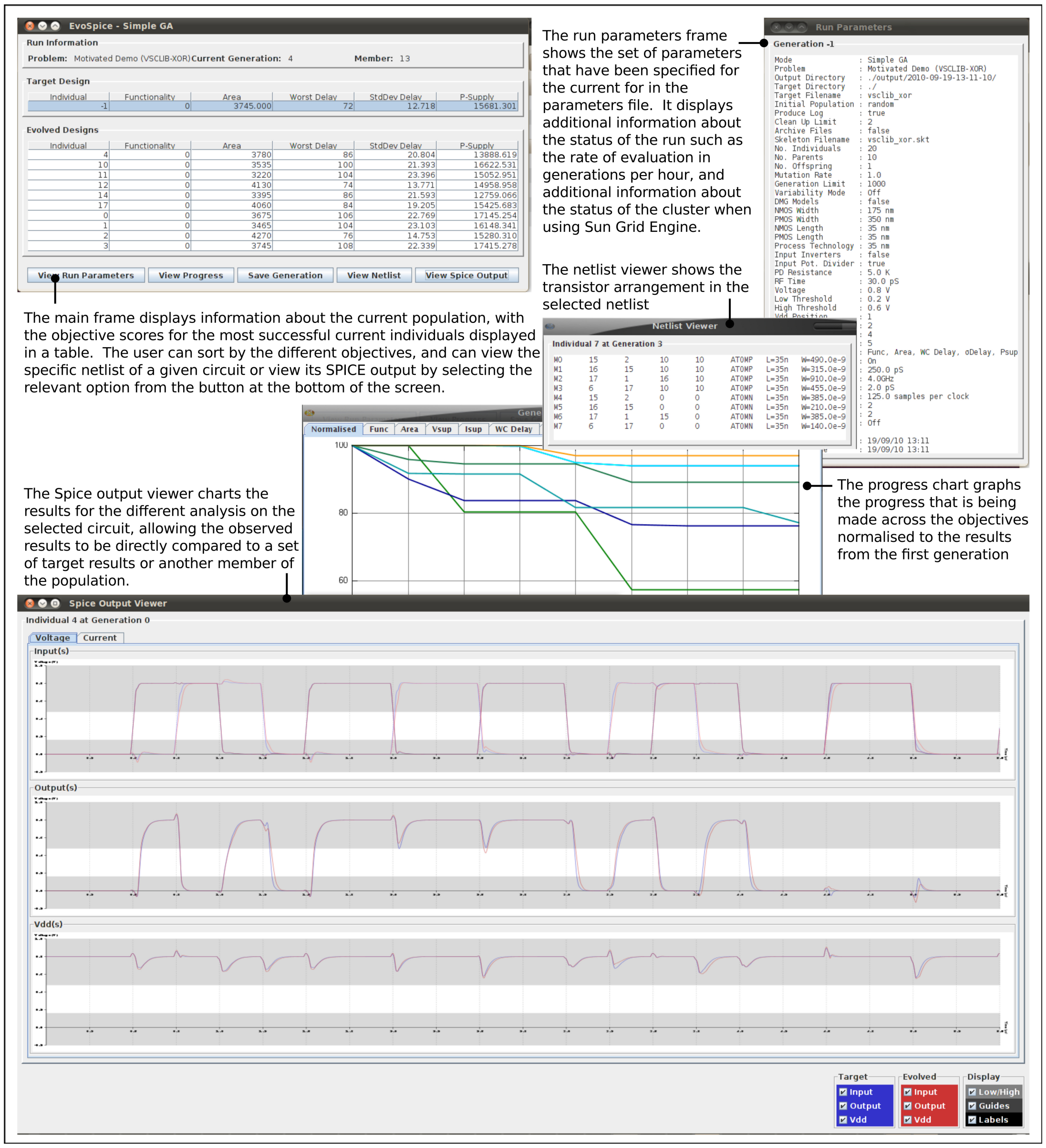
MOTIVATED consists of a number of tools, mainly written in Java, that allow for the different EC algorithms to be run. Originally it was written to run as a standalone application on a single machine, however later revisions have allowed a number of different modes of operation to be executed, largely to allow the time consuming SPICE simulations to be farmed onto local or remote computation clusters, a process described in Section 4.6. The tools include a graphical user interface, which allows the control and monitoring of evolutionary runs as they progress, including the option to view the SPICE output of different individuals in the population plotted to graphs. An set of screen captures of the different GUI components of the software is shown in Figure 4.1.

4.1.2 CGP-based Topology Design

Whilst the results in this thesis primarily concern the use of the MOTIVATED for the optimisation of standard-cell logic using the SGA system, brief mention is given here of the system created by Dr. James Walker for the evolution of novel logic-cell topologies based upon an modified form of Cartesian Genetic Programming, based on Clegg *et al's* representation [44]. Some of the circuits evolved with this algorithm are used later in this thesis in an example of completely-evolved digital design. With the algorithm, a fixed-length floating-point CGP genotype is decoded to a variable length phenotype, making use of the neutrality present in CGP [2]. The algorithm diverges from conventional CGP in the genotype $\rightarrow$ phenotype mapping, which translates the direct-graph CGP representation into a transistor-circuit topology; this mapping allows multiple connections including loops to be created within the phenotype circuit, whilst maintaining the feed-forward nature of the genotype.

The genotype is encoded with three floating-point, single precision values for each transistor, corresponding to the drain, gate and source connections. The two inputs to each transistor, and the two output-nodes for the genotype, are also encoded with a single floating-point value. The output nodes act as the root for the directed graph, such that two graphs can be extracted, one encoding the NMOS and the other the PMOS transistors. The encoding ensures that the decoded genotype follows CMOS design rules with complementary pairs of NMOS and PMOS devices, and also that all circuits are fully connected with no dangling connections. The detailed description of the decoding process, including the reasoning behind the design decisions, is discussed fully in [169].

[2]Neutrality in CGP occurs as nodes may not be connected to other nodes; neutrality has be demonstrated previously to be beneficial to the evolution process [117]. Additional neutrality occurs through the the quantisation of floating-point values into integers in the genotype [166]



Run Information

Problem: Motivated Demo (VSCLIB-XOR) **Current Generation:** 4 **Member:** 13

Target Design

Individual	Functionality	Area	Worst Delay	StdDev Delay	P-Supply
-1	0	3745.000	72	12.718	15681.301

Evolved Designs

Individual	Functionality	Area	Worst Delay	StdDev Delay	P-Supply
4	0	3780	86	20.804	13888.619
10	0	3535	100	21.393	16622.531
11	0	3220	104	23.396	15052.951
12	0	4130	74	13.771	14958.958
14	0	3395	86	21.593	12759.066
17	0	4060	84	19.205	15425.683
0	0	3675	106	22.769	17145.254
1	0	3465	104	23.103	16148.341
2	0	4270	76	14.753	15280.310
3	0	3745	108	22.339	17415.278

The run parameters frame shows the set of parameters that have been specified for the current for in the parameters file. It displays additional information about the status of the run such as the rate of evaluation in generations per hour, and additional information about the status of the cluster when using Sun Grid Engine.

Generation -1

```
Mode                : Simple GA
Problem             : Motivated Demo (VSCLIB-XOR)
Output Directory    : ./output/2010-09-19-13-11-10/
Target Directory    : ./
Target Filename     : vsclib_xor
Initial Population  : random
Produce Log         : true
Clean Up Limit      : 2
Archive Files       : false
Skeleton Filename   : vsclib_xor.skt
No. Individuals     : 20
No. Parents         : 10
No. Offspring       : 1
Mutation Rate       : 1.0
Generation Limit    : 1000
Variability Mode    : Off
DMG Models          : false
NMOS Width          : 175 nm
PMOS Width          : 350 nm
NMOS Length         : 35 nm
PMOS Length         : 35 nm
Process Technology  : 35 nm
Input Inverters     : false
Input Pot. Divider  : true
PD Resistance       : 5.0 K
RF Time             : 30.0 pS
Voltage             : 0.8 V
Low Threshold       : 0.2 V
High Threshold      : 0.6 V
Vdd Position        : 1
                    : 2
                    : 4
                    : 5
                    : Func, Area, WC Delay, σDelay, Psup
                    : On
                    : 250.0 pS
                    : 4.0GHz
                    : 2.0 pS
                    : 125.0 samples per clock
                    : 2
                    : 2
                    : Off

                    : 19/09/10 13:11
                  e : 19/09/10 13:11
```

The netlist viewer shows the transistor arrangement in the selected netlist

Individual 7 at Generation 3

```
M0   15   2    10   10   ATOMP   L=35n   W=490.0e-9
M1   16   15   10   10   ATOMP   L=35n   W=315.0e-9
M2   17   1    16   10   ATOMP   L=35n   W=910.0e-9
M3   6    17   10   10   ATOMP   L=35n   W=455.0e-9
M4   15   2    0    0    ATOMN   L=35n   W=385.0e-9
M5   16   15   0    0    ATOMN   L=35n   W=210.0e-9
M6   17   1    15   0    ATOMN   L=35n   W=385.0e-9
M7   6    17   0    0    ATOMN   L=35n   W=140.0e-9
```

The main frame displays information about the current population, with the objective scores for the most successful current individuals displayed in a table. The user can sort by the different objectives, and can view the specific netlist of a given circuit or view its SPICE output by selecting the relevant option from the button at the bottom of the screen.

The Spice output viewer charts the results for the different analysis on the selected circuit, allowing the observed results to be directly compared to a set of target results or another member of the population.

The progress chart graphs the progress that is being made across the objectives normalised to the results from the first generation

Figure 4.1: Screen captures of some of the key graphical-user interface components of the MOTIVATED system during the early stages of a SGA run.

4.2 SGA for Parameter Optimisation

The steps in the process of optimising circuits using SGA are illustrated in Figure 4.2. The initialisation of the algorithm follows a number of steps. First, the passed parameter file, described in Section 4.4 is parsed and verified, with default parameters added if unspecified in the parameter file. The parameters includes reference to a template netlist file, which is parsed and verified. The template netlist determines the necessary genotype length. The graphical-user interface, and classes which are responsible for handling RandomSPICE and NGSPICE, processing output data, calculating fitness, and writing the output and log files are initiated. If a previous saved population file is not specified, a population of randomised genotypes is created as described in Section 4.2.2.

Before the main algorithm loop begins, a sample netlist based on the target problem is evaluated, specified by a parameter in the parameters file. This netlist is evaluated using the same methods as all the other netlists, described in the following sections, with the exception that the functionality is always considered to be 0 (i.e. functionally correct). The logic state of the output of this netlist is sampled at the mid-point of each clock-cycle, and stored in a array; this is the logic state that all the subsequent netlists will be tested against - from these results a *target waveform* is created, which assumes zero delay in the transistion between states. The fitness scores extracted for this sample netlist have no effect on the operation of the algorithm, but provide a useful benchmark to which subsequent evolved circuits can be compared. Once these steps have all been initiated without problem, the main algorithm evolution loop can begin.

The first step of the main evolution loop is to save the current genotype an population archive file, which allows the algorithm to restarted at a later time if it is terminated by the user, through a system-error or because the end-generation is met. Each genotype is mapped into a RandomSPICE-compatible netlist, with extra components and the necessary SPICE analysis instructions added as described in Section 4.3.1.

The next stages of the evolution loop are the most time consuming, involving the use of the external RandomSPICE and NGSPICE software, and the post-processing of the SPICE output data; as a consequence the option of running these stages on an computation cluster has been implemented, discussed in Section 4.6. The RandomSPICE software is launched for each netlist, creating a predefined number of NGSPICE netlists for each individual, set by the `noRandomspiceRuns` option in the parameters file. NGSPICE is then called successively for each of these circuits, with the output data saved to a shared directory [3]. The output data

[3]In later revisions of RandomSPICE, the calling of NGSPICE for each netlist was available as part of the Python script. In later versions of MOTIVATED the SPICE output data is piped directly into the output parser to

is the read by a class which converts the tab-separated SPICE into the arrays of voltage and current values for each time step. These arrays are converted into a set of fitness scores for the different selected objectives using processes described in Section 4.3.2.

The following stage in the evolution loop ranks the all the individuals using the NSGA-II algorithm, described in Section 4.3.3. The fittest μ parents are the selected and λ mutated offspring for each parent are then created for the next generation, determined by the `noParents` and `noOffspring` parameters. The mutation operation is described in Section 4.2.4. The graphical user interface, described in Section 4.5 is updated with the results, and where appropriate the data files belonging to previous generations are archived or deleted to limit storage usage. The final process is to increment the generation counter, terminating the algorithm if the value set by `generationLimit` is reached; if not, the loop repeats for the next generation.

4.2.1 Template Netlist

The SGA takes as its input a plain-text file which is based upon the circuit topology section of a SPICE netlist. This template netlist contains a number of special 'marker' strings which are ultimately replaced with either numerical values or character strings through the decoding process, based upon values within the genotype. An example of the template netlist decoding procedure is illustrate in Figure 4.3, in which the widths of four transistors, and the values of two voltage sources, are encoded by the genotype.

Marker String

A marker string can be included at any place within the template netlist. It follows a string syntax, with the template-netlist parsing routine throwing a error and terminating the run if any invalidly formatted marker is found. The string always begins with a '\$' character and is followed by a fixed number of integer and floating-point values which correspond to the different parameters required by the token, terminating with a '\' character. The first number following the '\$' character determines the functional mode of the token; this must be an integer value between 0 and 2. This is followed by either 3 or 4 other comma-separated integer values, the quantity determined by the functional mode. For all modes, the second value is the *position marker* which corresponds to the gene within the genotype which is used to decode the value; the value must be greater than zero but does not have a strict top-bound and does not have to be unique. It is possible - indeed it is a 'feature' - that the same gene can be used to decode multiple marker strings within the netlist.

reduce disc access and increase efficiency.

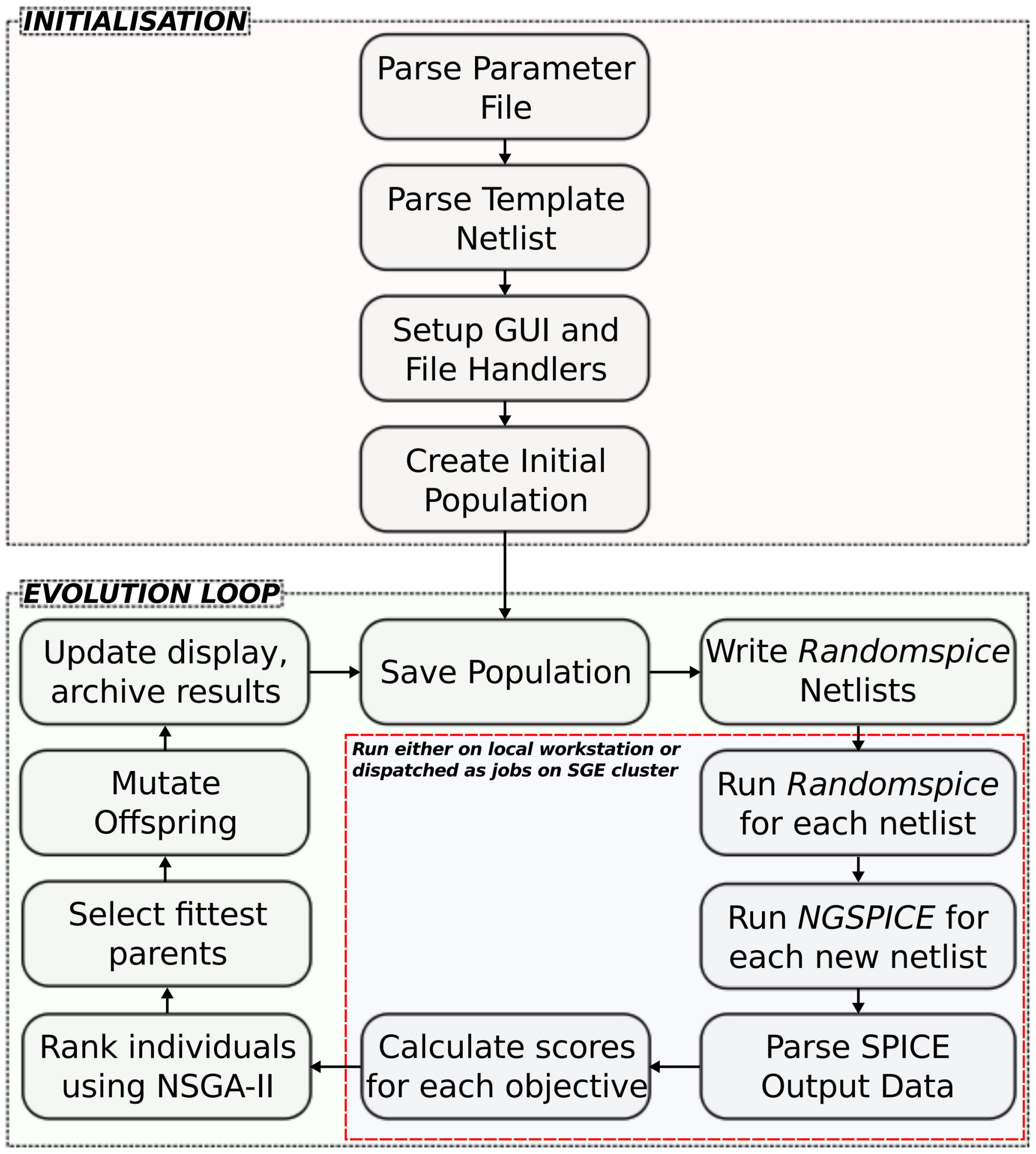
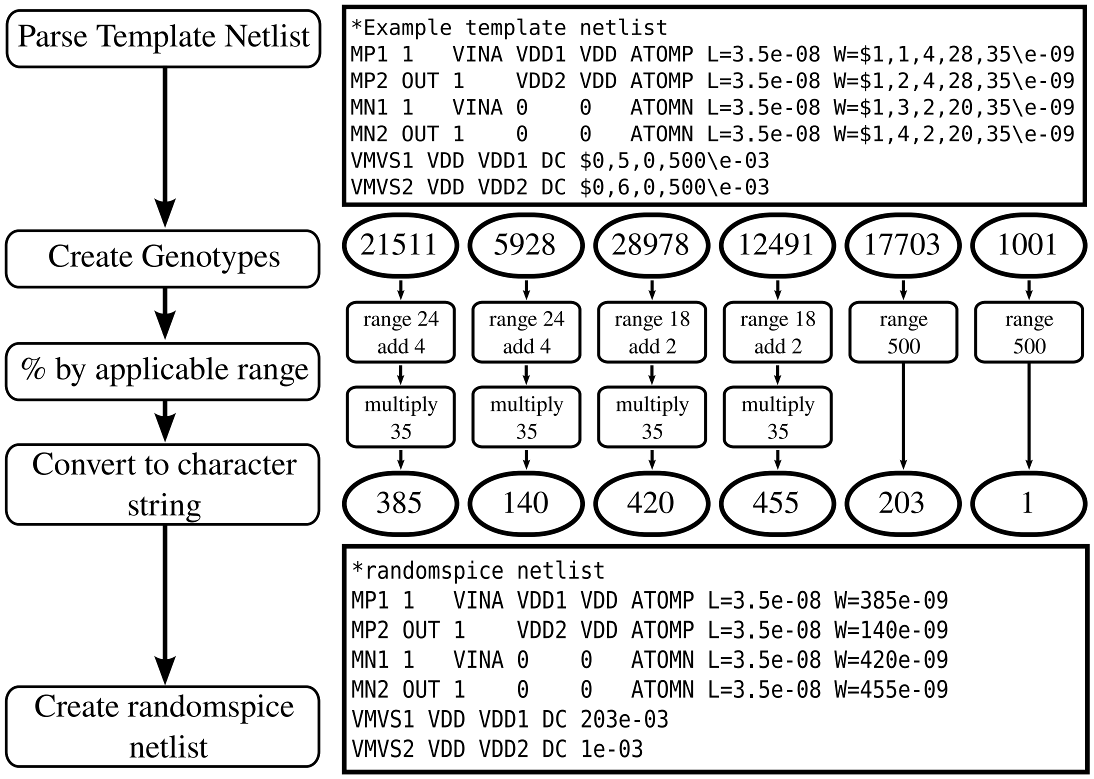

INITIALISATION
Parse Parameter File
Parse Template Netlist
Setup GUI and File Handlers
Create Initial Population
EVOLUTION LOOP
Update display, archive results
Save Population
Write Randomspice Netlists
Run either on local workstation or dispatched as jobs on SGE cluster
Mutate Offspring
Run Randomspice for each netlist
Select fittest parents
Run NGSPICE for each new netlist
Rank individuals using NSGA-II
Calculate scores for each objective
Parse SPICE Output Data

Figure 4.2: The steps in the workflow of the Simple Genetic Algorithm cycle, highlighting the initialisation phase and the evolution cycle.



```
*Example template netlist
MP1 1   VINA VDD1 VDD ATOMP L=3.5e-08 W=$1,1,4,28,35\e-09
MP2 OUT 1    VDD2 VDD ATOMP L=3.5e-08 W=$1,2,4,28,35\e-09
MN1 1   VINA 0    0   ATOMN L=3.5e-08 W=$1,3,2,20,35\e-09
MN2 OUT 1    0    0   ATOMN L=3.5e-08 W=$1,4,2,20,35\e-09
VMVS1 VDD VDD1 DC $0,5,0,500\e-03
VMVS2 VDD VDD2 DC $0,6,0,500\e-03
```

```
*randomspice netlist
MP1 1   VINA VDD1 VDD ATOMP L=3.5e-08 W=385e-09
MP2 OUT 1    VDD2 VDD ATOMP L=3.5e-08 W=140e-09
MN1 1   VINA 0    0   ATOMN L=3.5e-08 W=420e-09
MN2 OUT 1    0    0   ATOMN L=3.5e-08 W=455e-09
VMVS1 VDD VDD1 DC 203e-03
VMVS2 VDD VDD2 DC 1e-03
```

Figure 4.3: An example of a template netlist and its decoding procedure. The widths of the four transistors and voltages of the two voltage sources are encoded using Type 0 and Type 1 marker strings, which are decoded using to numerical values based on the genes within the genotype.

For markers where the functional mode is 0, the string will be replaced in the decoded netlist by an integer value. Two additional integer parameters, r_{low} and r_{high} define the linear-distributed range between which the decoded value v will fall such that $r_{low} \leq v < r_{high}$. The mode can be used to control, for example, circuit topology, by controlling which node a specific terminal is connected to. For example, in the following extract from a netlist:

```
MP1 0 $0,1,0,4\ VDD VDD ATOMP L=70n W=280n
MN1 3 $0,2,1,4\ 0 0 ATOMN L=70n W=140n
```

In the example, the gate-node of the PMOS transistor `MP1` will be assigned a value between 0 and 3, determined by the first gene in the genotype. The drain-node of the NMOS transistor `MN1` will be assigned a value between 1 and 3, determined by the second gene within the genotype. Thus the SGA can be used, through creative netlists, to manipulate circuit topologies, although as with all evolved-topology circuits that are to be simulated in SPICE, efforts must be made to avoid hanging branches and unsolvable loops.

For markers where the functional mode is 1, the string is replaced in the decoded netlist by a floating point value. The difference between mode 0 and mode 1 is the addition of an extra parameter which defines a constant by which the decoded value is to be multiplied. The constant can be any double-precision floating point value. This marker string is the type used extensively in the bulk of the experiments described in this thesis which concern the optimisation of transistor sizes within cell libraries. As RandomSPICE requires that transistor widths be unit-multiples of the base transistor size, this mode allows a transistor width to be specified within a bounded range as follows:

```
MP2 3 1 VDD VDD ATOMP L=35n W=$1,3,2,10,35\ n
MN2 3 1 0 0 ATOMN L=35n W=$1,4,1,10,35\ n
```

In the example, the PMOS transistor `MP2` will be assigned a width than is a multiple of 35nm, ranging from 70nm to 350nm. The NMOS transistor `MN2` will be assigned a width between 35nm and 350nm. The versions of RandomSPICE used in the thesis only allow for transistors to be multiples of a unit width, so the above example would be used with a 35nm library. Future released of RandomSPICE will allow an arbitrary value for the transistor width (accurate as long as it is above the minimum feature size); however there are still fundamental limits as to what transistor dimensions can actually be realised in a given fabrication process due to the lithography process used. Thus in future released, the following line might be a realistic option for optimising transistor widths:

```
MN3 3 1 0 0 ATOMN L=35n W=$1,3,2,20,17.5\ n
```

The above would allow for the transistor width to take a value between 35nm and 350nm at 17.5nm (half-feature size) intervals. The final functional mode of the marker strings, mode '2', allows a *comment* character (*) to be inserted into the netlist with a defined probability, otherwise the marker string is replaced with no value. When the comment character appears at the beginning of a line in a SPICE netlist, the line is treated as a comment and thus ignored. This mode takes three values: r, p, and s, which must be set to either 0 or 1. If s is zero, the comment character will be inserted when $n \bmod r > p$. When s is one, the comment character is inserted if $n \bmod r \leq p$. As multiple lines can use the same value from within the genotype, this allows for situations where two different lines within the netlist can be effectively toggled dependent on the gene value. For example, in the following netlist section:

```
$2,1,10,5,0\R1 1 0 10K
$2,1,10,5,1\L1 1 0 10mH
```

Both lines correspond to the first gene within the genotype. If this gene has the value 26, then the first line will begin with a comment (as $26 \bmod 10 > 5$) and the second line will begin as if the marker string wasn't present, as a result SPICE will see an inductor between nodes 1 and 0:

```
*R1 1 0 10K
L1 1 0 10mH
```

If however, the gene has the value 134, the opposite would occur (as $134 \bmod 10 \leq 5$), so SPICE will see a circuit with a resistor between nodes 1 and 0:

```
R1 1 0 10K
*L1 1 0 10mH
```

Through a combination of the above modes, skeleton netlists can be created which allow different blocks to be switched on an off. As SPICE netlists allow the formation of sub-circuits, in which complex sub-circuits can be represented with single lines in the top-most level, it is possible to use the mode 2 markers to choose between, for example, two different standard cell topologies, such as a high-speed XNOR gate or a low-power XNOR gate.

Table 4.1: Probability of number of genes mutated n_m when `mutationRate`= 1.0

Genes Mutated	1	2	3	4	5+
Probability	0.736	0.184	0.061	0.015	0.004

4.2.2 Genotype

The genotype in SGA is comprised of a fixed-length array of integers, using the standard Java data-type `int`. These are stored as 4-byte, two's complement values ranging from -2^{31} to $(2^{31} - 1)$. The length of the genotype is predefined by the number of unique position markers within the template netlist. The population is thus stored as a two-dimensional array of type `int`, whose length is determined by the number of parents and offspring specified in the parameters file, described in Section 4.4.

4.2.3 Selection Process

The algorithm uses a $(\mu + \lambda)$ strategy, with the value μ set by the `noParents` parameter and λ set by `noOffspring` in the parameters file. All the individuals in the population are ranked using the NSGA-II algorithm based on their set of objective scores, described in Section 4.3.2. The best μ ranked individuals from the population are promoted to the new generation, with each producing λ offspring, which undergo the mutation process, described below.

4.2.4 Mutation Process

The mutation process determines the number of genes to mutate n_m, by creating a random Poisson-distributed integer in which λ=`mutationRate`. The Poisson integer is created using the multiplication method described by Knuth [24] [4]. If the returned value of n_m is zero, it is changed to 1, ensuring that in every mutation operation at least one gene will be adjusted to avoid wasted processing cycles evaluating unchanged genotypes. The Poisson distribution is chosen to ensure that it is possible, if unlikely, for more than one gene to be mutated at any one time, to help avoid stagnation at local optima. For all the experiments described in this thesis the value of `mutationRate` has been set to 1.0. The results in the probability chart shown in Table 4.1, with the pseudocode for the mutation operation given in Appendix B.1.

[4]It is noted that this an inefficient method of creating random Poisson values when λ is a high value, however it is highly unlikely in the context of this algorithm to ever need such a large value, due to the small number of parameters to be optimised.

A for-loop is then started of length n_m steps, in which a gene is selected at random from the genotype, and replaced with a new random integer. Alternative schemes for the selection of the new value were tested, in which the new value is based on the current value added to a random Gaussian and Cauchy distributed value, described in Section 4.2.4, with the relevant σ^2 or γ values set using the `mutationWidth` parameter. However, these schemes were found to be less effective at evolving the discrete values for transistor dimensions in a number of early runs, thus the integer-replacement mutation has been used throughout the experiments described in this thesis.

4.3 Fitness Calculation

This section defines the methods that convert the template netlist into a complete RandomSPICE- or SPICE-ready netlist that can analysed, and from the output data retrieved assessed for performance at a number of different fitness objectives.

4.3.1 Circuit Test-Bench

The netlists discussed to date are effectively ‘black-box’ shells. In order to allow a standard system in which to evaluated logic circuits as part of the MOTIVATED cycle, the input netlists are populated with numerous extra components and control lines necessary to provide a test-bench for the circuits and enable SPICE to perform the correct analysis on the circuit. These comprise four key parts: a power supply stage, which provides the supply voltage and the voltage and current meters necessary to determine the power consumption of the test circuit, an input stage, which provides a realistic set of input signals to the test circuit, an output stage, which provides a suitable load to the test circuit, and the SPICE control stage, which includes all the analysis instructions needed for SPICE to produce the desired set of output scores based.

The test bench as used in most of the experiments within this thesis is shown in Figure 4.4. The test bench has seen many revisions and alterations throughout the project for a number of reasons. These include the desire to produce more accurate results, with the aim of reproducing the results observed from commercially available analysis tools such as those created by Synopsys and Cadence, and the need to adapt the test-bench to be compatible with the most recent versions of NGSPICE and RandomSPICE (see Sections 2.6.1 & 2.6.4).

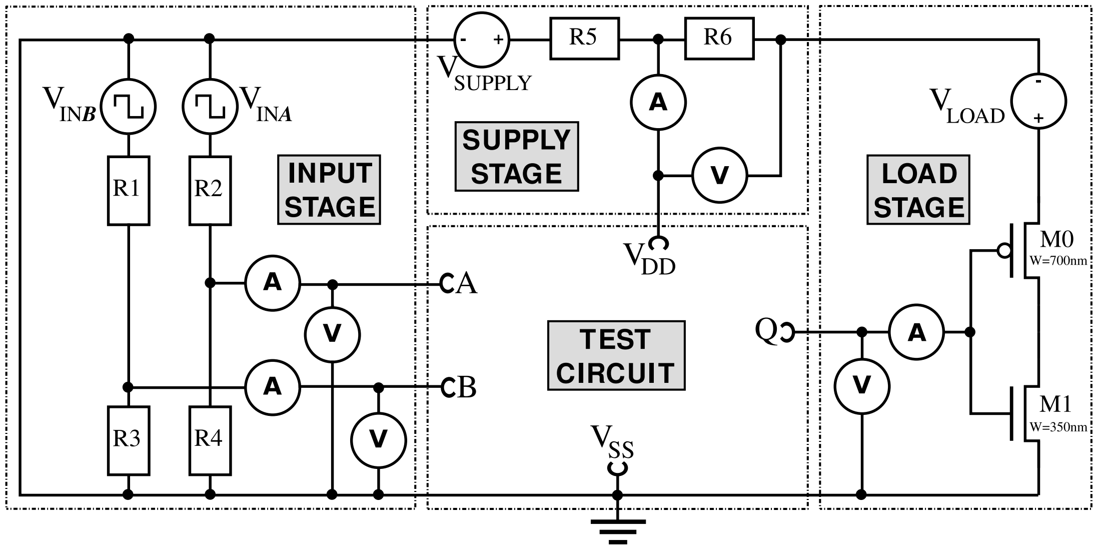


Figure 4.4: The test bench used in most of the experiments described in this thesis. The evolved netlist is connected to an input stage, a power supply stage an output load stage.

Power Supply Stage

The power supply stage contains a DC source rated at $V_{DD} \times 2$ connected to a potential divider. The potential divider allows the voltage-drop and current drawn by the test-circuit to be accurately measured using the voltmeter and ammeter. These calculations are used to create the power scores for the circuit under test.

Input Stage

The system allows a choice of input stages to be selected, specified by the run parameters. Originally, an input stage consisting of a potential divider, as shown in Figure 4.4, was used. For most of the later experiments this was replaced with a standard cell-library buffer which uses uniform transistors. This is based on the test arrangements used in the commercial Synopsys tools, and provides more accurate input slews for the performance analysis of the circuit.

Input Signals

After a study of the different analysis options available in NGSPICE, it was determined that the most accurate and efficient way to analyse the performance of 2-input logic circuits would be through the use of a single SPICE transient analysis, which covers the entire range of possible state-changes between the two inputs. Other possible solutions included the use of

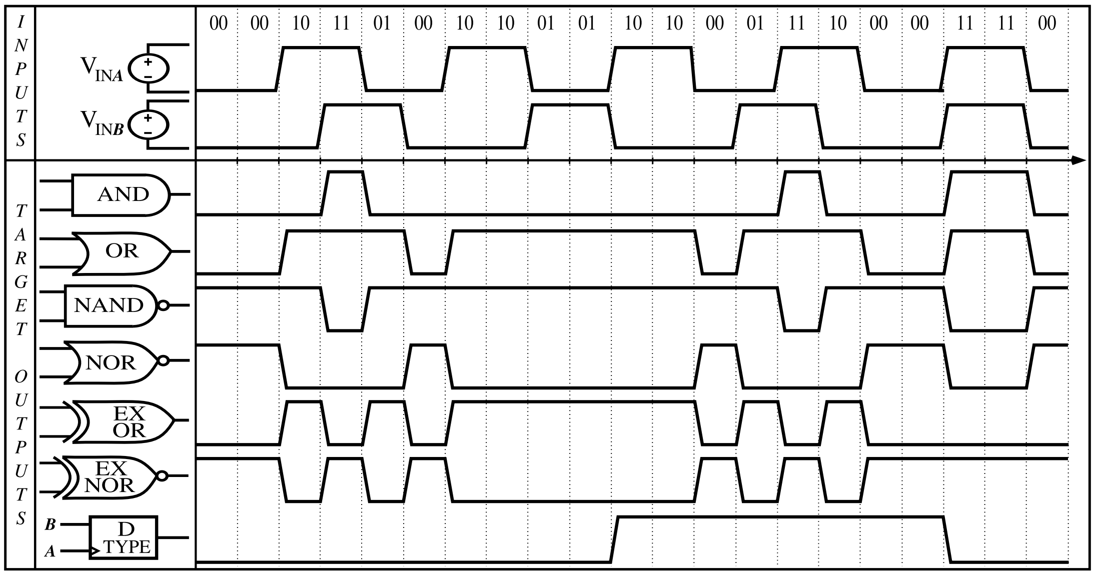


Figure 4.5: The input waveforms used for evaluating the performance of two-input logic circuits, and the associated target output waveforms.

several different sets of transient analysis, each covering one or two state changes; however, this was observed to take significantly longer to process through SPICE, presumably due to the considerable time taken to determine initial conditions for each analysis. This analysis is achieved using a pair of synchronous pulse-sources providing the input voltages to the two circuit inputs; the first of these source is held at a logic-low for two cycles, then at a logic-high for two cycles. The second source is held at a logic low for three cycles, the high for two cycles. This pattern allows the entire range of state-changes to be simulated over a period of 21 clock-cycles, as illustrated in Figure 4.5. The target outputs that are expected from the different primary logic gates are also illustrated in the figure.

Output Stage

In the output stage, the circuit under test is connected to a inverter, to provide a load for the circuit under test. Different values for the inverter allow different fan-outs for the circuits to be simulated. The voltage and current at the output is measured to create the circuit functionality, delay and slew assessments.

SPICE Control and Measurements

The SPICE control loop defines the measurements that are necessary to calculate the output scores. An example of the control loop is given below:

```
* SPICE Control Loop
.OPTIONS NOPAGE NOMOD
.control
ic v(1)=0 v(2)=0
tran 0.2ps 4200.0ps 0ps 0.2ps uic
linearize v(10) v(1) v(2) v(6) i(VMDD) i(VMI1) i(VMI2) i(VMO1)
print v(10) v(1) v(2) v(6) i(VMDD) i(VMI1) i(VMI2) i(VMO1)
.endc
.END
```

The `tran` statement established the transient analysis that is used; in the example shown, a 0.2pS sample period is used with a 5GHz clock rate. The `linearize` statement ensures that sample calculations are created at each time step. The `print` statement ensures that the output for the eight voltmeters and ammeters is written to the output file. NGSPICE writes the output in tab-separated columns of data with each line representing a new time-step.

4.3.2 Fitness Objectives

For each MOTIVATED run, a number of different fitness objectives can be selected, chosen through the use of a comma-separated list of integers for the `objectives` field in the parameter file (described in Section 4.4. For the earlier version of MOTIVATED there were 31 different standard objective scores calculated, listed in Table 4.2. The calculation methodology and reasoning behind these objectives are discussed in the following sections.

Circuit Functionality

The first objective, assigned number '0', evaluates if the circuit is functioning as expected, and is generally used on all runs. The `firstObjPriority` parameter, when enabled, weights this objective above all others; any circuit which fails to function correctly will always be ranked lower than one that does. The objective functions by sampling the voltage output of the circuit at the first sample point which occurs after half a clock-cycle has passed since a change of input state. The desired logic state at each clock-cycle is defined as illustrated in Figure 4.5. When a logic low is required, the observed output voltage must be below the value

Table 4.2: List of circuit fitness objectives MOTIVATED

Index	Name	Definition
0	**Functionality**	The count of transition errors
1	**Start-Mid Error**	The cumulative voltage error between the observed signal and the ideal output for the first half of each clock-cycle
2	**Mid-End Error**	As above for the second half of each clock-cycle
3	**Start-End Error**	As above for the complete clock-cycle
4	**Weighted Error**	As above but weighted so later samples have greater impact
5	**Cumulative Delay**	The sum of all transition delay periods
6	**Worst Delay**	The worst case transition delay
7	**StdDev Delay**	The standard deviation of transition delays
8	**Cumulative Slew**	The sum of all transition slew-rates
9	**Worst Slew**	The worst case transition slew-rate
10	**StdDev Slew**	The standard deviation of slew-rates
11	**Power Supply**	The cumulative power drawn from the supply
12	**Power Inputs**	The cumulation power drawn from the inputs
13	**Power Combined**	The above values combined
14	**Voltage Supply**	The cumulative voltage drop measured from the supply
15	**Voltage Inputs**	The voltage drop from the inputs
16	**Voltage Combined**	The above values combined
17	**Current Supply**	The absolute current measured from the supply
18	**Current Inputs**	The current measured from the input
19	**Current Combined**	The above values combined
20	**Weighted P. Supply**	As (11) but weighted so the later samples have greater impact
21	**Weighted P. Inputs**	As (12) but weighted
22	**Weighted P. Combined**	As (13) but weighted
23	**Weighted V. Supply**	As (14) but weighted
24	**Weighted V. Inputs**	As (15) but weighted
25	**Weighted V. Combined**	As (16) but weighted
26	**Weighted I. Supply**	As (17) but weighted
27	**Weighted I. Inputs**	As (18) but weighted
28	**Weighted I. Combined**	As (19) but weighted
29	**Number of transistors**	The count of MOSFETs in the netlist
30	**Transistor Area**	The sum of individual transistor areas in the netlist
+100	**Lowest Score**	The lowest score across all RandomSPICE runs
+200	**Highest Score**	The highest score across all RandomSPICE runs
+300	**Range**	Range of scores (highest-lowest) across RandomSPICE runs
+400	**Mean**	The mean score across all RandomSPICE runs
+500	**StdDev**	The standard-deviation of scores across RandomSPICE runs

defined in parameter `lowThreshold`: if it is, a score of 0 is given for that clock-period, otherwise a score of 1 is given. When a logic high is required, the output voltage must be above `highThreshold`. The scores are then summed over the whole 21 clock-cycles; a functionally correct circuit will have a score of zero, and a completely functionally incorrect circuit (such as a OR-gate when a NOR-gate is expected) will score 21.

Whilst not strictly necessary for the optimisation of circuits, an additional alteration was made for circuits which produced a static output, to improve the effectiveness of a metric when evaluating CGP-evolved topologies. Circuits which exhibit a steady-state output, such as stuck at 0V, V_{DD}, or an intermediate value in which no transistions from logic low to high or vice-versa were made, are given an overall score of 22. This alteration ensures circuits which demonstrate some measure of logic capability are promoted above those with no functionality whatsoever: for example, without this measure, a circuit which connected ground to the output would return a functionality score of 4 if the desired function was an AND-gate, whilst a OR-gate would return a score of 10, meaning the grounded circuit will be ranked higher than the OR-gate circuit. With the added alteration, the grounded circuit will score 22, thus the OR-gate circuit will be ranked higher, in recognition of the fact it does at least perform a function. Any circuit which, for whatever reasons, fails to produce SPICE output, is given a maximal score for functionality of $(2^{31}) - 1$. In general practice with both the SGA and the CGP algorithms this shouldn't happen, bar a system or network failure, however with previous designs of the CGP based topology design it was possible to create circuits which would cause SPICE to throw errors [167].

When optimising transistor dimensions, the topological arrangement of transistors remains constant, thus at first glance it might appear the functionality objective is unnecessary. However, if the clock-rate or other parameters are set such that extremely poor-performing arrangements of dimensions (such as all-minimum sized transistors) produce circuits which fail to change states before the measurement is taken, they will be penalised to a low-ranked position. Without this objective, such circuits would likely have a high-rank when transistor area and power are used as objectives.

Error Measurements

The second sets of objectives are considered error measurements: they measure how far the output voltages at each sample point are from the ideal value specified by the target waveform. At every sample point, the difference between the output voltage and the ideal target output voltage are measured. These values are summated across all the sample points to create an overall error score. Four different error score objectives have been created, which analyse different regions. The first measures the error between the start of the transition, the point at which the changing input voltage(s) cross the 50% threshold ($V_{DD}/2$), and half-way through the clock cycle - this is called the **Start-Mid Error**. The second measures the error from half-way through the clock cycle to the end of the clock cycle - this is the **Mid-End Error**. As the functionality test requires that the output waveform has crossed the relevant high or low threshold by half-way through the clock cycle, the **Start-Mid Error** will invariably much greater than the **Mid-End Error**, and is generally roughly proportional to the delay rate. The Mid-End error measures the later stage of the transition, an is a useful indicator of any problematic oscillation in the output. Two other error scores can be calculated: firstly, the **Start-End Error** is the error over the complete cycle, a summation of the previous two measures, and finally a **Weighted Error** linearly weights each step with its position in the clock-cycle: the first value is multiplied by 1, the second by 2, up to the final point which is multiplied by the number of samples in the clock cycle. This error value thus penalises heavily for circuits which exhibit severe ringing, or when the output is not pulled fully high or low.

The error objectives are primarily used in the CGP-topology system. In the SGA, the circuits generally all perform as expected and the delay, slew-rate and power scores (described below) are more effective indicators of circuit performance. However, when evolving topologies, arrangements exist which exhibit unwanted effects that many not be indicated by the mentioned objectives. One typical example of this is a topology which, due to the arrangement of devices between the output and supply, fails to produce a logic high output at V_{DD}, falling somewhere between `highThreshold` and V_{DD}. Despite this it can still pass the functionality test and produce respectable scores for delay, slew and power due to the way in which they are measured. Such a circuit would however score very poorly in the different error objectives described here.

Delay and Slew-Rate

The first set of objectives which are used to discriminate between different functionally correct circuits involve the calculation of delay times and slew-rates. These calculations are made in the clock-cycle in the analysis in which an output transistion for low to high or vice-versa is anticipated. The methodology for calculating delay and slew-rates is designed to reflect that used in commercial tools used in the semiconductor industry such as those developed by Synopsys and Cadence. The definitions of delay and slew-rate are illustrated in Figure 4.6.

The is calculated by counted the number of sample-points in an output transition, between when the input-signal crossing the 50% threshold ($V_{DD}/2$) and when the output signal crosses the threshold. This will result in an integer value, which, when multiplied by the sample-rate,

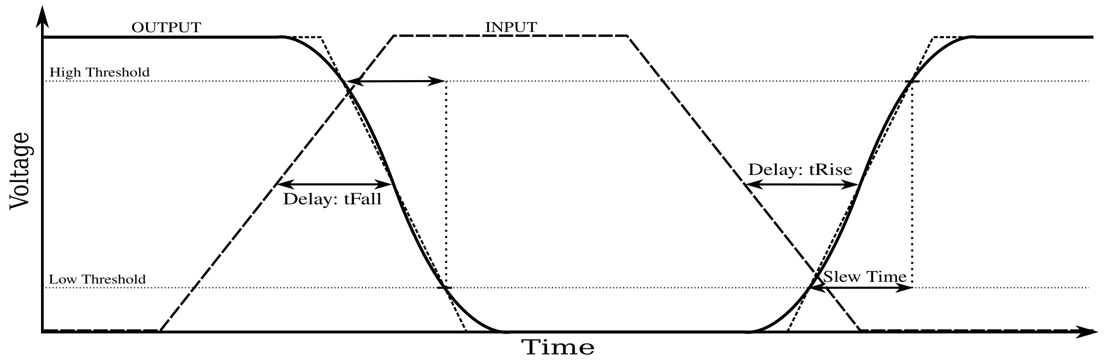


Figure 4.6: The calculations used for calculating the delay and slew-rates in a single clock-cycle.

describes the delay period for that specific transition. In the case where the output never crosses the 50% mark within 1 clock-cycle, due either to very poor circuit performance or an incorrect topology for the target output, the delay period for that given transition will be set equal to the clock-period, however by definition the circuit will also have failed the functionality test for the transition.

The slew-rate for a transition is the count of sample-points between the output signal crossing the `highThreshold` voltage and it crossing the `lowThreshold` voltage for a falling transition, and vice-versa for a rising transition. As with the delay calculation, if this condition is not reached within a complete clock-cycle, the given count is equal to clock-rate, but once again the circuit will have failed the functionality test.

The above measurements are used to calculate the delay and slew-rate periods for a single state transition. The actual objectives allow three different methods for combining these scores across the whole 21-cycle analysis. The first of these is the cumulative score, which combines all the delay-times or slew-rates into a single figure. The second is the worst-case, which compares all the given delay or slew-rate periods and returns the highest value. The final is the standard-deviation, which calculates the mean delay or slew-rate across all the transitions and from this the standard deviation. Of these options, the worst-case and standard-deviation are the two objectives which have been used in most the experiments in this thesis. The worst-case is a useful metric as it reveals to the developer the delay and slew that must be accounted for when combining designs. The standard deviation is equally important as having wide-ranging delay or slew rates can cause problems when many gates are nested; in general it is desirable to have balanced rise-fall and fall-rise times across all output transitions wherever possible.

Voltage, Current and Power

The bulk of the remainder of the objective scores are based on the voltages and currents drawn by the circuits. The voltage scores are based drop in voltage between the supply and the relevant voltmeters (shown in Figure 4.4; separate scores are available to measure the drop from the main supply and from the two circuits inputs. The difference is measured at each sample point and summated over the entire analysis. The current scores summate the absolute current measured at the ammeters. The power score is calculated by integrating the product of the currents and voltages across the entire analysis. The relevant scores from the supply and inputs are also summated into a combined score. For each of the power, voltage and current objectives, there is also an equivalent weighted options, which, as for the weighted error objective, multiplies the score at each sample position by its position within the clock cycle.

Whilst the power score provides a useful indication of power consumption by the test circuit, it was realised that for more effective results it would be necessary to migrate towards using a method for power measurement which matched that used by commercial tools. Such a system separates the power into dynamic power (the power consumed during the switching of states and through short-circuit currents) and static power (the steady state power consumed through leakage). This adaptation was made after many of the experiments described in this thesis and is discussed in more detail in the following chapter.

Transistor Area and Transistor Count

The final two objectives are transistor area and transistor count. These are calculated when the initial netlist are created as they are not dependent on SPICE analysis and are not dependent on variability. The transistor area is the summation of the areas of all individual transistors in the netlist, which are calculated from the product of the channel length and width parameters in the individual models. This objective is designed to be a rough approximation of the overall area that would be consumed in a fabricated circuit, although in actual fabrication many other factors will determine the realised size, such as the proximity between devices and the necessary interconnections; a circuit with one transistor will scores the same as a circuit with two transistors each at half width of that in the original circuit, whilst in reality the latter would require a greater area of silicon. Despite this, when evaluating topologically identical circuits, the transistor area metric does provide a worthwhile indication of the expected fabricated area. The transistor count objective is primarily for use with the CGP algorithm for evolving topologies, and functions by summing the number of MOSFET device lines present in the netlist.

When two circuits perform identically at all other objectives, it is clear that the design with fewer transistors is more desirable.

Variability Scores

All the objectives described so far consider the single-circuit scores. When RandomSPICE is used, multiple copies of the same circuit (using different variability transistor models) are evaluated in SPICE, thus a sets of SPICE output are returned. This means for each of the desired objectives, a score for each individual circuit is returned. These scores can be combined of five different ways, which are selected `objectives` field by adding 100, 200, 300, 400 or 500 to the objective indexes. The first two of these methods return the lowest of the scores, or highest of the scores, respectively. Generally, the highest is more likely to be a useful metric of a circuits performance, as it will give the worst-case example of the objective across the different variability-aware circuits. The third method returns the range - the difference between the highest and lowest score - for the objective. The fourth method returns the mean value, and the fifth the standard-deviation of the set of results.

4.3.3 Multi-Objective Ranking

Once all the objective scores are calculated, the entire population is ranked using an implementation of the NSGA-II algorithm. If the `firstObjPriority` parameter is enabled, then circuit functionality is considered to be primary rank and the NSGA-II rankings secondary to this. The process first sorts all the individuals into non-dominated fronts, then calculates the distances between individuals in each of the fronts and ranks those well spaced the highest. The returns a unique rank for each of the individuals in the population to the selection process. The NSGA-II algorithm is described in Section 3.3.3, and pseudo-code of the ranking algorithm used within MOTIVATED is given in Appendix B.2.

4.4 Run Parameters

When the main MOTIVATED system is initiated, it is passed a parameter file which defines all the parameters necessary for the required evolutionary run. The complete set of parameters is given in Table 4.3. The parameters given relate to the system setup, considered to be *system parameters*, also to the specifics of the evolutionary algorithm to be performed, considered to be *evolution parameters*, and the parameters to be used in the SPICE analysis, considered to be *circuit parameters*.

Table 4.3: MOTIVATED Run Parameters

	Parameter Name	Type	Definition
System Parameters	`mode`	int	The evolutionary algorithm to be run, 1=CGP, 2=SGA
	`targetPath`	string	The location where the template netlist is stored
	`targetFilename`	string	The filename of the template netlist
	`outputPath`	string	The location where output data is to be saved
	`debug`	boolean	Enables debug mode with verbose output logging
	`problem`	string	A title for the evolutionary run
	`extractionMode`	boolean	
	`spiceThreads`	int	The number of threads of SPICE to run in parallel for MP acceleration
	`screenPrint`	int	The number of generations between on-screen refreshes
	`cleanUpLimit`	int	The number of previous generations worth of output data to keep before deletion or archiving
	`archiveFiles`	boolean	If enabled, netlists and SPICE output data are archived to a single .tgz file per generation
	`populationFile`	string	If specified, the algorithm initiates with a previously saved population
Evolution Parameters	`objectives`	CS int	The comma-separated objective set to be evaluated
	`firstObjPriority`	boolean	
	`noParents`	int	The number of parents (μ) to preserve for the next generation
	`noOffspring`	int	The number of off-spring (λ) to produce per parent
	`mutationMode`	int	The mutation mode. 0=random value, 1=Gaussian distributed, 2=Cauchy
	`mutationRate`	float	The mutation rate - for SGA the number corresponds to the Poisson parameter λ in setting the number of genes to mutate per operation, for CGP it corresponds to the % mutation rate
	`mutationWidth`	float	The value of σ^2 when Gaussian and γ when Cauchy mutation is used
Spice Parameters	`generationLimit`	int	The termination generation for the algorithm
	`clockRate`	float	Specifies the clock-rate in GHz
	`sampleRate`	int	Specifies the number of SPICE transient analysis steps per clock-cycle
	`inputInverter`	boolean	Adds an inverter to the input-stage
	`inputDivider`	boolean	Adds a potential-divider to the input-stage
	`voltage`	float	Specifies V_{DD} for the circuit
	`lowThreshold`	float	Specifies V_{low} threshold for fitness calculation
	`highThreshold`	float	Specifies V_{high} threshold for fitness calculation
	`noRandomspiceRuns`	int	The number of RandomSPICE circuits to be created for each netlist
	`dmgModels`	boolean	True if RandomSPICE & intrinsic-variability aware models are to be used
	`modelLibrary`	string	Location of filename of model-library for RandomSPICE to use
	`uniformModels`	boolean	True if the uniform (non intrinsic-variability) models are to be used

4.4.1 System Parameters

The system parameters define the information needed to setup the system. The most important parameters are `mode`, which determines which of the two evolutionary systems to run: CGP for topology design, or the SGA for optimisation. A special mode, set by `extractionMode`, allows the extraction of detailed analysis of a single circuit using a large number of variability runs; no evolutionary algorithm or GUI is launched in this mode. The `spiceThreads` parameter allows the execution of RandomSPICE and SPICE to be parallelised across multiple threads, to improve performance on multi-CPU workstations. Improvements observed through increase this parallelisation are discussed in Section 4.6.6. The remaining system parameters are concerned with the paths and filenames of the input and output files and minor GUI settings.

4.4.2 Evolution Parameters

The evolution parameters are those which directly affect the evolutionary algorithm being run. The `firstObjPriority` setting is used to control if the circuit functionality objective is considered as the primary objective, meaning that in no circumstances can a circuit which has a worse (higher) functionality score than another can rank above it. The `noParents` and `noOffspring` parameters correspond to μ and λ respectively. The `mutationMode` parameter has three options: random value mutation, Gaussian-distributed and Cauchy-distributed mutation, with the relative widths of the latter distributions determined by `mutationWidth`, with the likelihood of mutation set by `mutationRate`. The `generationLimit` defines the generation number at which the algorithm will terminate.

4.4.3 Circuit Parameters

The circuit parameters govern the settings that are used in RandomSPICE and SPICE and the specifics of the additional components added to each template netlist in the formation of the final RandomSPICE or SPICE ready netlists. The `clockRate` parameter defines, in GHz, the clock-rate at which the input waveforms are set; the value chosen should be slow enough to ensure that the output of a circuit is likely to have changed state by half-way through the clock period, but fast enough such that a useful discrimination between the performance of different circuits can be evaluated, without needing to set the `sampleRate` unnecessarily high. The `sampleRate` defines how many analysis measurements are taken in each clock cycle, thus when multiplied by 21 indicates the total number of analysis steps each SPICE evaluation will undertake. The `voltage` parameter specifies V_{DD} for the circuit, with the thresholds at which circuit functionality and slew-rates are calculated set by `lowThreshold` and `highThreshold`. The `inputDivider` and `inputInverter` control if a potential-divider and/or inverter are added to the circuit inputs.

The `noRandomspiceRuns` parameter defines the number of unique netlists that will be created by RandomSPICE and simulated with SPICE in the creation of variability-aware output results. The `dmgModels` parameter is set to true if RandomSPICE is to be used; when false, the RandomSPICE stage is omitted, with circuits passed straight to NGSPICE. The `uniformModels` parameter is set to true if it is desired for circuits to be passed through RandomSPICE but the uniform (i.e. not intrinsic-variability aware) models to be used. If `dmgModels` is set to zero, or `uniformModels` is set to true, the `noRandomspiceRuns` parameter is ignored and only one circuit per individual is evaluated.

4.5 Graphical User Interface

The graphical user interface for the MOTIVATED system contains a number of different frames and tools allow the control and feedback information relating to the current run. The main frame of the system, shown in the top left of Figure 4.1, appears once all the needed files have been loaded, the target netlist evaluated and the main run started. At the top of the frame, the title of the problem and information about the current generation is shown, along with the current member that is being evaluated when run locally (this information is less accurate when run on the cluster as many individuals will be run in parallel, so it only displays the count which are known to have been completed). Below this is a pair of tables; the uppermost table contains the relevant scores for the target-netlist so they can be compared to the evolved results. The larger main table contains the list of promoted parents from the previous generation, initially ordered by fitness rank, with the fittest circuits at the top. The rows can be sorted by clicking on an objective column header, easing the process of locating the circuits which perform the best in any given objective.

Any of the individuals shown in either table can be selected, allowing one of the additional information frames to be opened through selecting the relevant button. One of these frames displays a stripped-down display of the netlist which includes only the transistors, their connections and their dimensions. Another frame charts all the different Spice analysis for the chosen circuit alongside the target circuit, allowing the observed results to be compared. The other main features of the GUI of note are the ability to display the graphically display the progress of each of the objectives, with the relative scores from the initial generation normalised to 100%, to see in which objectives progress is being made and where evolution is stagnating. The run parameters can also be displayed, including detailed information about the current status of the run when using the cluster and approximations of the run-speed of the algorithm. Finally the user can control certain aspects of the algorithm, such as allowing the operation to be paused/resumed and allowing the current generation to be saved for restarting at a future occasion (with different parameter settings if desired).

4.6 Use of High-Performance Computation to Accelerate MOTIVATED

The system described to this point allows the complete evolution cycle to be run on a single machine, albeit with parallel operation of the SPICE processing when multiple processing cores are available. The most time consuming stage of the algorithm is the analysis of the individual circuits in SPICE, with the creation of the randomised netlist in RandomSPICE also taking a considerable amount of time. The remainder of the process - the selection and mutation processes - consume a very small fraction of the overall compute time. As a consequence, a vast speed-up can potentially be realised if the RandomSPICE and SPICE operations can be distributed to cluster and Grid computer systems.

4.6.1 Background to Grid Computing

Grid-Computing is a form of high-performance computing (HPC) that distinguishes itself from conventional cluster and super-computer systems by means of the loose-coupled nature, with heterogeneous resources distributed geographically across a number of sites. Grid-computation systems are operated through a set of tools termed *middleware* which control the tasks of job creation, submission, monitoring and data movement, all protected with a set of security provisions. One of the main HPC resources that has been utilised as part of the nano-CMOS project is ScotGrid, the Scottish Grid Service. This is primarily hosted at Glasgow, with additional nodes in Edinburgh and Durham. The Glasgow site houses 310 worker nodes, which comprises 280 Opteron 280 2.4GHz cores and 680 Xeon E5420 2.5GHz cores, with 2GB RAM provided for each core. The system also includes 24 nodes dedicated to providing the Grid Server architecture and 20 dedicated storage nodes, providing 500TB storage [8].

4.6.2 The MAVEN Cluster

In order to accelerate the process of performing the SPICE calculations locally, a local cluster-computer has been built within the Intelligent Systems Group at the University of York. The cluster, named MAVEN, contains twelve quad-core processing nodes, each housing an Intel Q9550 2.83GHz CPU and 8GB of RAM; this results in 48 effective processing nodes as seen by the SGE system. Local storage on each processing node is provided with an Intel X25-V 40GB solid state drive. A head-node which doubles as a file-server exists as the access point to the cluster, also equipped with an Intel Q9550 and 8GB of RAM, also housing 4 Samsung 1TB hard disc drive RAID array. Initially the MAVEN cluster used the Kerrighed open-sourced single-system image cluster software to provide job submission, however this was later transferred to the Sun Grid Engine (SGE) [5] system.

4.6.3 The Sun Grid Engine System

SGE is an open-source batch-queuing system that controls the submission, scheduling, dispatching and management of jobs on cluster-computing systems. It operates atop a number of Unix-like operating systems, providing the management of resources across a number of local or distributed nodes. On the MAVEN cluster, SGE version 6.2u5 runs on the Ubuntu 9.4 operating system on all the nodes and file-server. Jobs are submitted to a job-queue on a per-user basis using simple command line functions. Once resources are available, the jobs are seamlessly distributed to the available nodes and run until termination or user-interruption [4].

4.6.4 Alterations to code to enable cluster operation

The alterations to the MOTIVATED code to enable it to operate on the MAVEN cluster using SGE are relatively simple, thanks to the nature of SGE's command-line based job submission system. As the RandomSPICE operations are already initiated through command-line style procedure calls in the standard single-machine MOTIVATED framework, only minor changes are needed to submit the operations as SGE jobs. To do this, the RandomSPICE jobs are wrapped into sets `.sh` shell-scripts. These scripts are then submitted as jobs on the cluster through an SSH connection, using the SGE `qsub` command. An extra thread is then started which periodically polls the SGE system to detect the status of the submitted jobs. Once all the jobs are completed, the MOTIVATED system can process the output data and extract the required fitness scores.

A later alteration to the system moved the post-processing of SPICE data, in which the fitness scores are calculated based on the SPICE output files, to be run as part of the submitted job scripts. This alteration reduced the amount of data-processing that needed to be done on the local system. As a result of this alteration, all the data needed for the locally run

[5] Now known as Oracle Grid Engine

MOTIVATED system is the sets of fitness scores, which are saved on the shared AFS file system. A seperate branch of the MOTIVATED code based was created which is responsible for cluster-based operation; the code and parameters are identical to that previously described with the exception of the routines where RandomSPICE and SPICE are launched and where the output-data is processed.

4.6.5 Alterations to code to enable Grid operation

In addition to the code alterations to allow distributed analysis of the SPICE netlists on the local MAVEN cluster, the systems has also been adapted to allow the RandomSPICE and SPICE applications to be run remotely on Grid-based high-performance computing (HPC) clusters. The infrastructure that has been developed to support job creation and submission as part of the nano-CMOS project as a whole comprises of a number of web services, each of which provides a particular category of functionality, that is to say a particular type of submission or job to a particular type of resource. These services all take the form of Apache Axis 2 Web Services, in which Apache RAMPART performs the encryption, time-stamping and signing of messages to provide the message-level security. Interaction with these services is implemented through a number of command-line client applications, written by the e-Science teams working on the nano-CMOS project, which are distributed in a single self-contained bundle.

Running a job on a Grid involves a two-step process, in which an application service is invokes which is responsible for the creation of the job, followed by a submission service which submits the job to a particular HPC resource; the submission service can be bypassed using direct Globus submission. The job creation and submission processes are closely linked to a data management system which exploits the distributed Andrew File System (AFS). To create a job, the user passes an input file to the application service, which contains the input file for the underlying application (RandomSPICE) wrapped with a set of additional information needed to run the job on the HPC resource. This information includes a reference to a relevant ‘template’ AFS directory where run data will be stored, and the number of sub-jobs, which can, dependent on the HPC resource, be distributed and executed in parallel. One the job-creation request has been received, the application service duplicates the template directory for each sub-job and creates a record of the new job in the data service, written in Job Submission Description Language (JSDL). Once these steps are complete, a Uniform Resource Identifier (URI) is returned to the user which acts as the unique identifier for the job. One a job has been submitted, the user invokes the submission client application, which provides the user with a list of available HPC resources on which the job can be run. Different software is then used to implement to the different types of HPC resource - the Java Commodity Grid Kit is used to submit to Globus resources, such as ScotGrid or a bespoke submission service for Sun Grid Engine clusters such as MAVEN [171].

Whilst the infrastructure for running the RandomSPICE and SPICE applications as part of the evolutionary loop within MOTIVATED has been implemented, due to a number of reasons its use has been minimal. Primarily, the overhead in time associated with the job creation and submission steps has largely negated the advantages of using the Grid when all but the largest population sizes are used. As most tests have evaluated populations with fewer than one thousand individuals, this time penalty results in a slower rate of evolution than could be achieved locally on the MAVEN cluster. However, the Grid-based system has proven invaluable in the process of performing a detailed evaluation of extracted designs, allowing analysis of circuits based on 100,000 different RandomSPICE simulations, which provide a far greater degree of accuracy in the statistical data retrieved, as described in Section 5.6.2.

4.6.6 Performance Observations

For reasons detailed in the previous section, the most effective performance of the MOTIVATED system was observed when run using the local MAVEN cluster. Due to the nature of the SGE system and the fact that as a shared resource it is not always possible to guarantee 100% usage, a certain performance improvement cannot be guaranteed. However, a set of small benchmark runs were evaluated on the system at a period when no external jobs were allocated. There are a number of important considerations to make to ensure efficient and effective use of the cluster-based system: firstly, it is necessary for the workload to be equally balanced across all the available nodes. Furthermore, it is desirable for all the individual jobs to terminate as close to each other as possible, as the MO ranking mandates that all results are available to compare and sort. A final consideration is that the workload is of a practical length; by this, setting a very short workload to each node, such as a single SPICE evaluation, will be inefficient due to a number of reasons: whilst the job submission process on SGE is rapid compared to that for Grid-submission systems, it can take a number of seconds from submission to the initiation of jobs; having multiple jobs start simultaneously adds delays due to the shared file system; the polling process adds a slight delay between the completion of the final job on the cluster and the acknowledgement of completion on the remote system on which MOTIVATED is running. As a consequence, it is generally desirable to have more than one SPICE netlist evaluated by each available node, thus a multiple of 48 is generally chosen for the population size.

The time taken for NGSPICE to process a netlist on a single node can be broken in two parts; firstly the relatively static time taken to load the software, read the netlist and calculate the initial conditions, secondly the time taken to perform the transient analysis and write the corresponding output data to the file-system. Typically the first of these processes generally takes around half a second on the systems used [6]; a precise figure is hard to extrapolate due to the various loads on the filesystem. The time taken to perform the transient analysis is roughly linear in line with the number of steps to be calculated in the analysis, determined from the combination of the sample-rate, the clock-speed and the length of the transients.

A set of performance benchmarks were carried out to evaluate the difference in performance between running RandomSPICE, NGSPICE and the data handling on a local workstation to the performance when running on Maven. For these tests, two different test circuits were optimised, first a four-transistor NAND gate and secondly a nine-transistor XOR gate from the VSCLIB standard-cell library (described in Section 5.2.2). For each individual, RandomSPICE creates 10 netlists which are evaluated in NGSPICE. The tests were carried out at three different sample rates for SPICE transient analysis, firstly taking 50 samples per clock-cycle (a 10pS sample-rate based on the 2GHz clock-frequency, for a total of 1050 data points), then with 100 samples per clock-cycle (5pS sample-rate, 2100 data points) and finally at 200 samples per clock (2.5pS sample-rate, 4200 data points). The test were carried out using single, dual and quad-SPICE threads on the local workstation (which has a quad core Intel Q6600 CPU) and the using all 48-logical nodes on the MAVEN cluster, with overall population sizes of 48, 96 and 192 (i.e. one, two and four RandomSPICE runs per node respectively). On both systems other processes and jobs were minimised to ensure maximum CPU availability, and the tests were repeated three times with an average result taken. Each algorithm was terminated after 10 generations, with the average time to evaluated one individual calculated by dividing the total time the algorithm was running by the number of individuals evaluated.

The results for the benchmarks are shown in Figure 4.7, with the relative performance increases above single-threaded operation on an overall and per-core basis illustrated in Figure 4.8. Across the runs operating just on the workstation, a speed-up of between 1.46× and 1.82× ($\mu = 1.60\times$, $\sigma = 0.15$) was observed when moving from single-threaded to dual-threaded operation. When moving to four threads, the speed-up measured was between 2.72× and 3.50× ($\mu = 3.17\times$, $\sigma = 0.37$) the single-thread performance, demonstrating that the evolutionary loop is very scalable through parallel threading of the RandomSPICE and SPICE operations. The largest improvement in performance was actually found with the smaller circuits and lower sample rates; it is suggested that this is likely due to the hard disc access being the bottleneck, with smaller reads and writes to the disc needed with smaller circuits.

[6]This figure is roughly the same on both the 2.66GHz Intel Q6600-based local machines and the nodes in the MAVEN cluster.

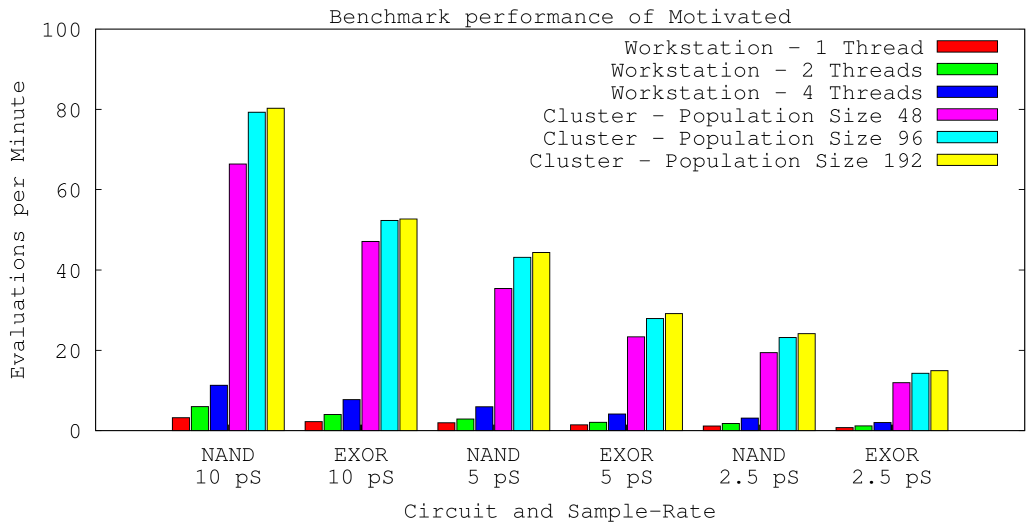


Figure 4.7: Benchmarks of the performance of MOTIVATED at evaluating different circuits on a single workstation and on the MAVEN cluster

When moving to the Maven cluster, the performance speed up over the single-threaded workstation ranges from 15.8× to 21.0× ($\mu = 18.35\times$, $\sigma = 1.95$) when a population size of 48 is used (equivalent to 1 RandomSPICE run per compute node), rising up to 18.3× to 24.7× ($\mu = 20.92\times$, $\sigma = 2.36$) when a population of 96 is used. A much smaller increase in performance is observed when the population size is again doubled to 192, with a performance speed up of 19.2× to 25.0× ($\mu = 21.93\times$, $\sigma = 2.52$). When the cluster computation is compared to the four-threaded single workstation performance, a speed-up of between 6.82× and 7.77× ($\mu = 7.27\times$, $\sigma = 0.31$) is observed, with a greater improvement found when using the smaller NAND circuit than the EXOR circuit, and also when the lower sample rates are used. This again points towards a data-storage (or network transport) bottleneck, which is probably a side-effect of all the worker nodes sharing the fileserver resources.

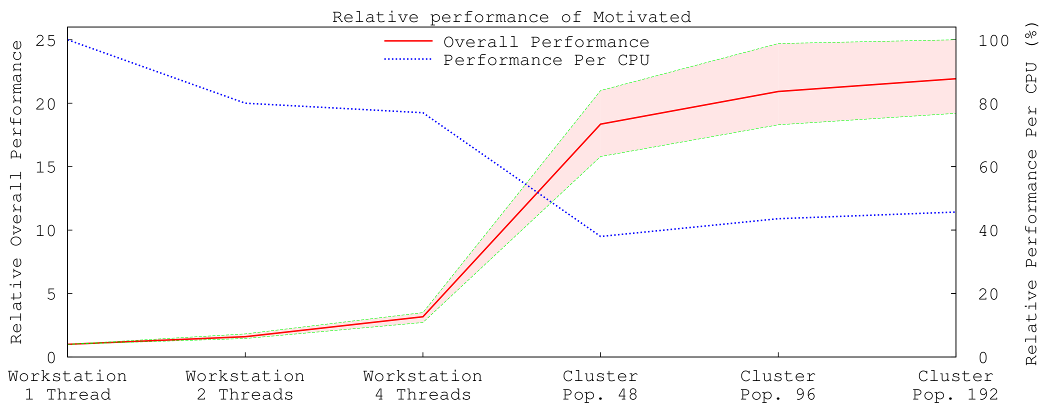


Figure 4.8: The relative performance improvement when MOTIVATED is run both multi-threaded, single workstation mode, and on the MAVEN cluster, across all the benchmark runs, weighted so the single-threaded performance = 1.

This chapter has discussed the algorithm that will be used for optimising the transistor dimensions with logic circuits, and the set of tools that allow the algorithm to be used in conjunction with RandomSPICE and NGSPICE, both locally on a workstation and on a HPC cluster based on Sun Grid Engine. The following chapter details the experiments that have been carried out to attempt to optimise logic circuits using the system.

Chapter 5

Optimising Standard Cell Libraries

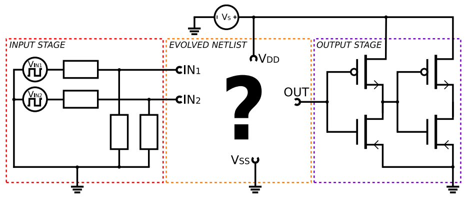
The chapter describes the experiments that have been carried out with the aim of optimising standard cell libraries, and the results that have been achieved. The results are given in approximate chronological order of when the experiments were carried out; it is important to point out that the system used to evolve circuits itself, and the model libraries used, have themselves evolved over the period of the experiments, guided by observations made from the results, and performance, and also by the progress made at other nano-CMOS project partners, notably the DMG responsible for creating the transistor models.

5.1 Early Results: Evolving Transistor Topologies using the SGA

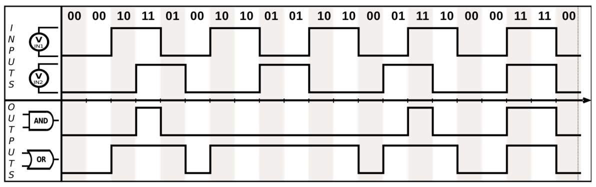
The first experiments that were carried out using the SGA system actually attempted to evolve circuit topologies, as opposed to just optimising transistor dimensions within known designs. The results from these experiments, and similar parallel experiments into evolving topologies using an early CGP-based implementation, guided many of the decisions that have been made in future revisions to the MOTIVATED system, described in the previous chapter, and so are included here.

5.1.1 Methodology

In the experiments, an early implementation SGA system was launched to evolve AND and OR gate circuits simultaneously. As the algorithm setup and parameters predate the system described in the previous chapter, the main differences are outlined below.



(a) Test Bench



(b) Inputs Pulses and Target Outputs

Figure 5.1: The test bench, input waveforms and target outputs as used in the early experiments in evolving 2-input logic topologies

Template Netlist

The template netlist used for the circuit evolution contained 6 NMOS transistors and 6 PMOS transistors. The genotype contained 60 genes, with 5 genes corresponding to each of the 12 transistors in the netlist. Of these 5, the first determined if the transistor was enabled or disabled; if the integer value was even, the transistor line would be commented out in the produced netlist. The next 3 genes corresponded to the drain, gate and source connections of the transistor, each being able to take a value between 0 and 10 (with 0 corresponding to the ground node, 3 to V_{DD} and 4 to the output connection of the circuit). The final gene determined the channel width of the transistor, with the length set to 35nm and the width being an integer multiple of 35nm between $1\times$ and $8\times$. The actual template netlist used is given in Appendix C.1.1.

Test Bench

The test-bench added to each netlist, with the relevant voltage sources, input and output circuits is shown in Figure 5.1a. The input voltage sources are connected to 1KΩ potential dividers, and provide pulsed square-waves with no rise\fall slew, as shown in Figure 5.1b, which also shows the two target output waveforms for the AND gate and OR gate. The output of the circuit is connected to a buffer (a nested pair of inverters), which was chosen to provide a realistic loading.

Evolutionary Parameters

The experiments predated the use of the NSGA-II algorithm to calculate fitness. Instead, fitness was calculated using a 2-step ranked fitness function. The primary fitness function corresponded to the functionality of the circuit, with the secondary fitness calculated from the maximum cumulative error between the output traces and the target waveform. For circuits

Table 5.1: The run parameters used in the early experiments to evolve AND and OR gate topologies using SGA. Population, mutation and crossover parameters were chosen based on observations from prior experiments.

Parameter	Value
Population Size	2000
Genotype Length	60
Transistor Count	12 (6 pairs)
Target Functions	AND, OR
Mutation Rate (%)	5
Crossover Rate (%)	10
Tournament Size	4
RandomSPICE Runs	20
Supply Voltage (V_{DD})	0.8
Clock (GHz)	5
Sample Duration (pS)	4

where the functionality score was tied, the secondary fitness function would be used to discriminate between individuals. Scores were calculated for each individual for performance as both AND-gates and OR-gates. For each individual tested, 20 RandomSPICE netlists were created and analysed.

At the end of each generation, a new population was created, with half the members chosen based on their AND-gate score and the other half based on their OR-gate score. The algorithm used tournament selection to determine which individuals would be promoted to the future generation, without the use of elitism (i.e. all the members in the new generation were subject to the evolutionary operators). A simple form of crossover was included, in which 10% of the new population was created from 2 tournament-selected parents, using a uniform crossover method. The parameters used in these experiments are shown in Table 5.1.

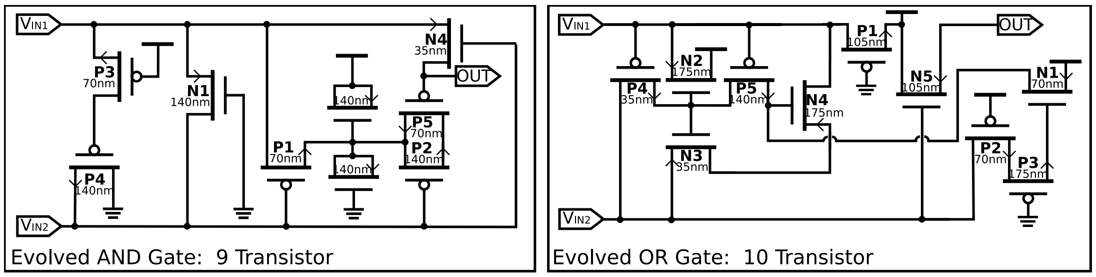
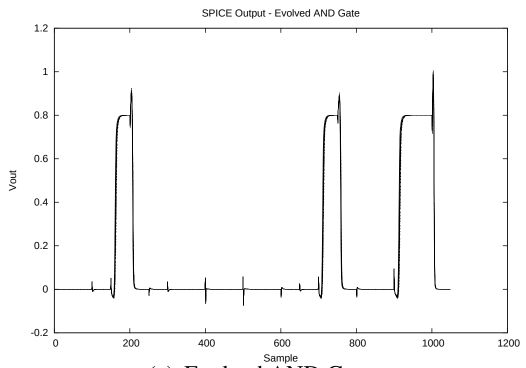
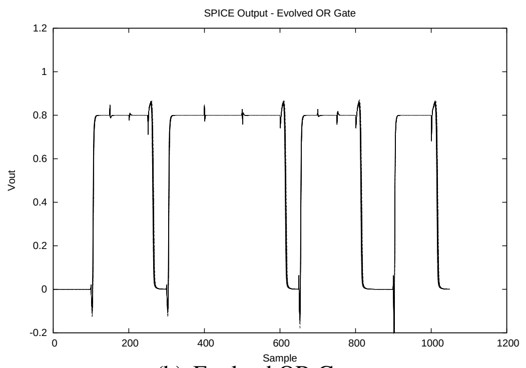


Figure 5.2: Schematics of the evolved AND and OR gates



(a) Evolved AND Gate



(b) Evolved OR Gate

Figure 5.3: SPICE output voltage from designs shown in Figure 5.2

5.1.2 Results

The algorithm was run for approximately twenty days on a single-core workstation, terminating at the end of the 85^{th} generation. At this point, 170,000 different individuals had been evaluated, which equals a total of 3.4 million SPICE evaluations (roughly 2 per second). The first functionally correct OR-gate design appeared in generation 26, with the first functionally correct AND-gate appearing shortly after in generation 29. The topological arrangement of the fittest designs, which are illustrated in Figure 5.2, first appeared in generation 50 for the AND-gate and 62 for the OR-gate; the improvements in later generations upon this design were changes to the individual transistor widths within the arrangement, up to the fittest designs extracted in generation 85 which are shown in the figure. The output traces from which the fitness scores were calculated for these two circuits are shown in Figure 5.3.

5.1.3 Observations

The details of these early experiments have been included here not because they produced realistic real-world designs, but rather because they taught a lot about the potential pit-falls that could occur in creating variability-aware standard cell designs, and as a consequence heavily influenced the future direction in the development of the MOTIVATED system. One of the key things learnt was the need for a far more effective method for determining the fitness of a circuit; the evolved designs actually produced better fitness scores than conventional standard-cell designs using the given test-bench and scoring criteria, however in reality were woefully performing designs, due to vast power consumption caused by short-circuit currents. Both the topologies evolved with the SGA, and also topologies evolved using a CGP representation (described in [167]), produced designs which contained source or drain connections of transistors to input rails, feedback from outputs to inputs, and other arrangements which do not reflect conventional CMOS design philosophies. However, as the fitness function looked solely at output voltage, such arrangements were allowed and not penalised. The following points were concluded from the experiments which ultimately resulted in the MOTIVATED system described in the previous chapter:

- To evaluate circuits accurately, a fitness function that takes into account the power consumption at both supply and inputs, and the delay of the circuit, is needed.
- The fitness function thus needs to be able to simultaneously optimise for parameters which are generally inversely proportional.
- The method for including variability in the designs needed to be carefully considered.
- The CGP system is more efficient at evolving topologies, although alterations needed to be made to ensure circuits were arranged from complementary pairs of transistors.
- The SGA system did prove effective at optimising the transistor values once functionally-correct topologies had been found.

Thus, following these experiments, it was decided to focus on evolving topologies using a revised design of the CGP-system, and use the SGA for optimising dimensions existing (and evolved) designs. The revised test-bench, multi-objective fitness function and selection strategies described in the previous chapter were implemented, with both algorithms unified into the single characterisation architecture that became MOTIVATED.

5.2 Optimising Transistor Widths within 2-Input Combinatorial Logic Cells

This section describes the experiments and results obtained when the MOTIVATED system, using the SGA, was applied to the problem of optimising transistor dimensions within a open-source standard-cell library. The experiments described in the section were the first in which the MOTIVATED system with its multi-objective fitness function based on NSGA-II were used, having been developed from observations on the early results described in the previous section. Whilst the experiments described here were based on the system described in Chapter 4, they predate the construction of the MAVEN cluster and with it the SGE based method for distributing the workload; as such all the experiments described in this section were processed on a single, quad-core workstation.

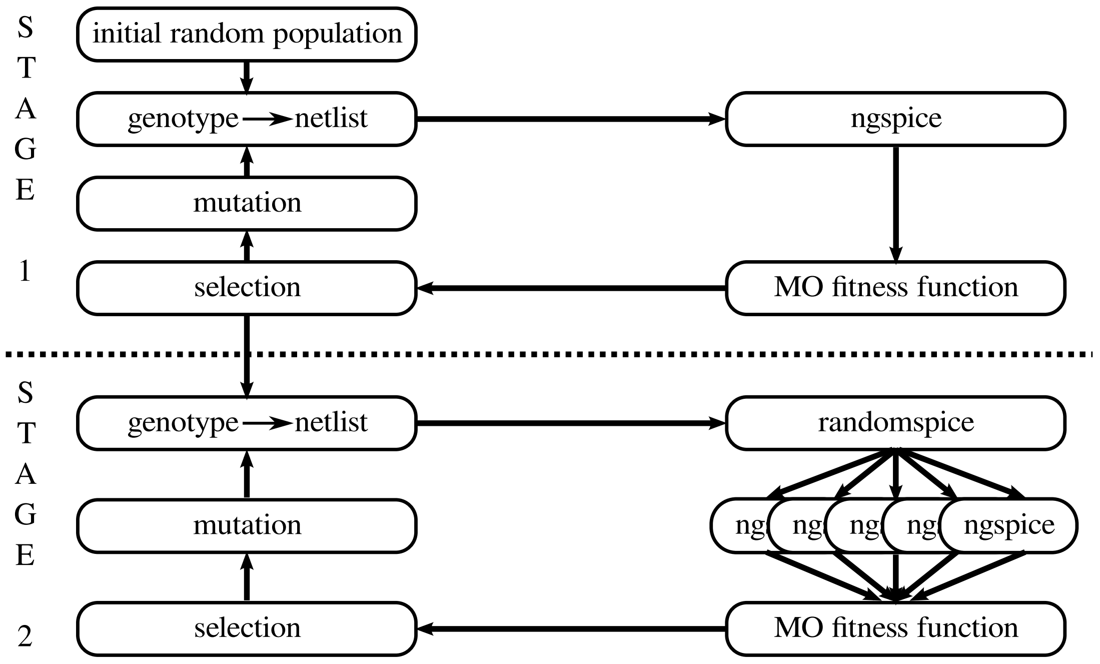


Figure 5.4: Two stage cycle use to optimise circuits in MOTIVATED

5.2.1 Two-stage Cycle for Incorporating Variability

Based on observations of the time taken to run early simulations, it was realised that there was neither the computational resources, nor time available to follow a similar methodology, at the time the experiments were carried out. As a consequence, it was decided to attempt to see if a two-stage process could be used successfully. The two stage process, illustrated in Figure 5.4, operates as follows: initially, the algorithm optimised the transistor widths with the goal of finding high-speed, low-power and balanced delay times, using the uniform BSIM transistor models that the variability models are based on (specified by using the `uniformModels` parameters). After a predetermined number of generations, the algorithm retains the current population and switches to using the variability-enhanced RandomSPICE models, with the objectives switched so that variability-tolerance accounted for. The motivation for the two stage process is to reduce the time needed to produce the initial fronts of circuits. The variability-aware circuits take much longer to evaluate, both because of the number of circuits evolved in RandomSPICE, and also because the each of the RandomSPICE circuits take a lot longer to process, due to the increase in number of nodes and number of models that have to be accounted for.

Table 5.2: Fitness objectives in the two stages of evolution

Objective	Uniform Stage	Variability Stage
1	Functionality	*Worst-Case* Functionality
2	Circuit Area	Circuit Area
3	Supply Power Score	*Worst-Case* Supply Power Score
4	Max delay	*Worst-Case* Max delay
5	σ delay	*Worst-Case* σ delay
6	Max slew-time	*Std-Dev* Max delay
7	σ slew-time	*Std-Dev* σ delay

5.2.2 Methodology

Objectives

The objectives for each of the two stages in the experiments are shown in Table 5.2. During the first stage of the algorithm, known as the *uniform* stage, the objectives include the worst-case transition delay times and slew-rates to help promote the fastest circuits, and also the standard-deviation of the transition delay times and slew-rates to identify the most balanced circuits. The worst-case power supply scores helps to promote the circuits with the lowest power consumption. The first is run for a predetermined number of generations, then the entire final population is passed to the second-stage of the process, the *variability* stage.

For the variability stage, circuit functionality and six circuit performance objectives are used with the multi-objective strategy. Circuit area is the combined total of the transistor widths; this value remains constant and is not affected by variability. The overall worst-case delay score (the worst switching speed across all transitions in all of the RandomSPICE runs) represents the delay-time of the cell that must be accounted for when it is to be included in larger circuits. As a measure for the extent of variability, the standard-deviation of the worst-case delay across all of the randomised MOSFET runs is used as an objective. It is desirable to keep the switching-delays as constant as possible between the different input transitions over all random runs, so the worst-case of the delay standard-deviations is used as another objective. Finally as a indication of the power requirements, the overall worst-case summation of the power scores at each sample point is used. This value is best considered as being proportional to the energy used in driving the load through the 21-clock cycles in the most power-demanding of the variability runs.

Test Circuits

The experiments cover a set of seven different two-input logic circuits from a standard cell library (SCL). The circuit layouts have been translated from the **vsclib** library, which are designed for a standard bulk 130nm process [135]. The model library used with RandomSPICE is based on a 35nm $\times$ 35nm Toshiba transistor, with the effects of RDD added; the libraries available at the time these experiments were carried out did not include the additional effects of LER or PSGB.

The test circuits comprised of six standard two-input logic gates and a D-type flip-flop. Given that only device widths were being evolved, it was anticipated that the smaller logic circuits would serve as a test of the effectiveness and reliability of the system, whilst the exclusive-gates and flip-flop should provide a large enough search space to evaluate algorithm performance. Each of the circuits was converted to a template netlist, in which one gene per transistor is used to assign a width to the device. The allowed ranges for the transistor widths were set to between 2 and 20 units (70nm - 700nm) for the NMOS transistors and 4 to 28 units (140nm to 980nm) for the PMOS transistors. These ranges are chosen to reflect the design rules for the allocated space available to transistors within the chosen SCL library [135]. The schematics for each of the test circuits is shown in Figure 5.5, whilst the corresponding template netlists that are used can be found in Appendix C.1.2.

Run Parameters

The parameters used for the runs are shown in Table 5.3. A larger population size is used for the more complex tests. Each promoted parent produced 4 mutated offspring; the mutation rate chosen means there is 74% likelyhood of a single gene being mutated, as discussed in Section 4.2.4. Clearly for the smaller circuits, particular the NAND and NOR gates with only four genes in the genotype, this corresponds to a significant mutation in each cycle. In the variability stage of the algorithm, 50 RandomSPICE output netlists are evaluated for each individual. The input signals followed a $4GHz$ clock-frequency, which was measured to be slow enough to ensure all transitions would be complete within the required half clock-cycle, even in the worst-performing circuits. The `sampleRate` parameter was set to 50 samples per clock, corresponding to a 4 pS interval between samples.

5.2.3 Results

For each of the test runs, the algorithm was run using the uniform models for a predetermined number of generations as listed in Table 5.3 then stopped. The number of generations in this

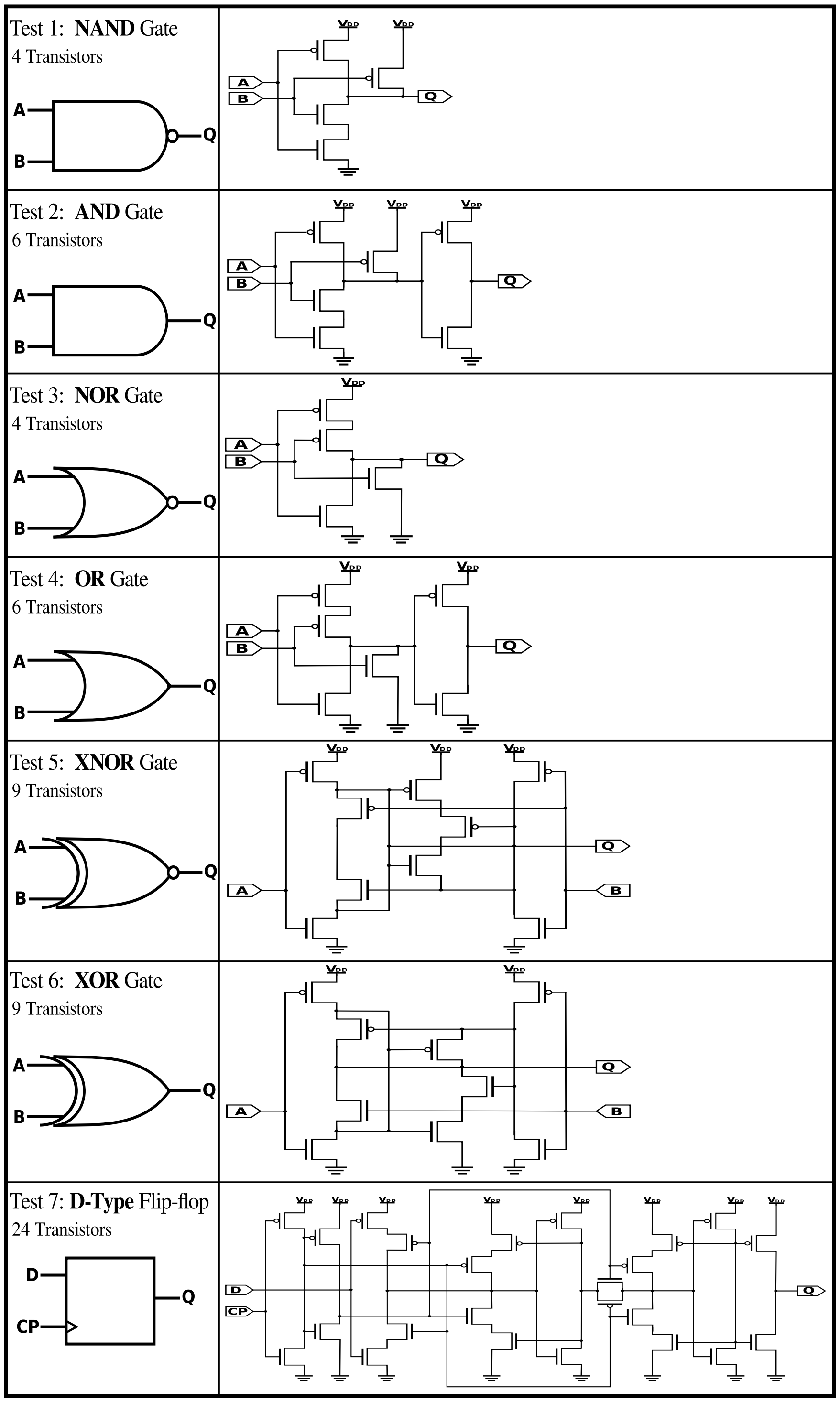


Figure 5.5: Schematics of the seven test circuits from the VSCLIB SCL for optimisation using the SGA [135]

Table 5.3: The run parameters used in the experiments

Parameter	Tests 1&3	Tests 2&4	Tests 5&6	Test 7
Population size	100	100	200	400
Number of Parents	20	20	40	80
Genotype length	4	6	9	24
Mutation rate	1	1	1	1
V_{DD} (V)	0.8	0.8	0.8	1.0
Clock (GHz)	4	4	4	4
Sample Period (pS)	5	5	5	5
Uniform Stage Generations	200	300	500	1000
Variability Stage Generations	50	50	50	50
RandomSPICE Circuits	50	50	50	50

stage was chosen based on previous work and the length of the genotype. As is shown in Table 5.4, the number of possible arrangements of transistor width grows rapidly with the size of the problem circuit; for the NAND and NOR gate, the number of circuits tests after the uniform stage is close to 9% of the total number of possible combinations within the search space. For the AND and OR gate this figure drops to below 0.028%, beyond which only a tiny fraction of the total number of combinations can be assess.

Uniform Stage

A comparison of the results seen in the initial random population, compared with the population at the end of the uniform stage, are illustrated in Figures 5.6 & 5.7, which shows each of

Table 5.4: The number of generations for the uniform stage of evolution, and the corresponding number of possible combinations within the search space for each of the test circuits

	Circuit	Transistors	Generations	Evaluations	Possible Combinations
1	NAND	4 *(2N,2P)*	200	20,000	225,625
2	AND	6 *(3N,3P)*	300	30,000	1.072×10^{8}
3	NOR	4 *(2N,2P)*	200	20,000	225,625
4	OR	6 *(3N,3P)*	300	30,000	1.072×10^{8}
5	XNOR	9 *(4N,5P)*	500	100,000	1.273×10^{12}
6	XOR	9 *(5N,4P)*	500	100,000	9.672×10^{11}
7	D-Type	24 *(12N,12P)*	1000	500,000	1.319×10^{32}

the circuits selected as a parent at the end of the initial generation, and final generation of the uniform stage, along with a marker for the circuit based on the reference cell-library design [1].

The first column in each of the two charts plots the worst switching-delay against the supply power. The second column plots the worst-switching delay against the standard-deviation of all switching delays, to evaluate how well balanced the circuit switching is. It is clear from the results of the uniform stage that the smallest logic gates, the NAND and NOR, are already well optimised, with the reference design falling very close to the optimal power-delay front. However, when moving towards the larger circuits, a greater improvement is observed in the extracted Pareto optimal-front for power-delay: for the exclusive gates, the algorithm has produced designs which outperform the reference design. This was most significant in the case of the XOR design, where designs have been found with a 15% improvement in delay and at a lower power consumption than the reference design. At first glance the results for the D-Type flip-flop appear to indicate the greatest improvement, however, as will be discussed shortly in Section 5.4.4, the result is misleading.

From this final generation of parents, three circuits were chosen which demonstrated potential improvements over the reference design for each of the test circuits. The first circuit chosen is considered the *high-speed* circuit - this is not necessarily the overall lowest-delay, but rather the circuit which demonstrated the best balance of low-delay and low standard deviation across delays: for some circuits the absolute lowest-delay would come at such a power penalty that it was deemed unsuitable for selection. The second circuit chosen was the one which best fit the criteria of minimising circuit power consumption whilst operating at a similar delay ($\pm 5\%$) to the reference design; this is considered the *balanced speed-power* circuit. The final circuit extracted is the one which scores the lowest supply-power score altogether, this is the *low-power* circuit. It is important it note that these designs were extracted by hand, through analysis of the different objective scores in the final generation, as opposed to using an automatic or rule-based system of extraction.

Variability Stage

The variability stage of the algorithm was then initiated using the final population of the uniform stage. After 50 generations the run was terminated. The circuit which demonstrated the overall lowest standard deviation of worst-case delay time, whilst still performing close to or better than the reference SCL design in terms of both power and delay (no more than 5% greater in terms of delay or power) was selected from the final population by hand. This

[1]The reference cell-library design used transistor dimensions proportional to those suggested in the VSCLIB designs for bulk 130-nm process

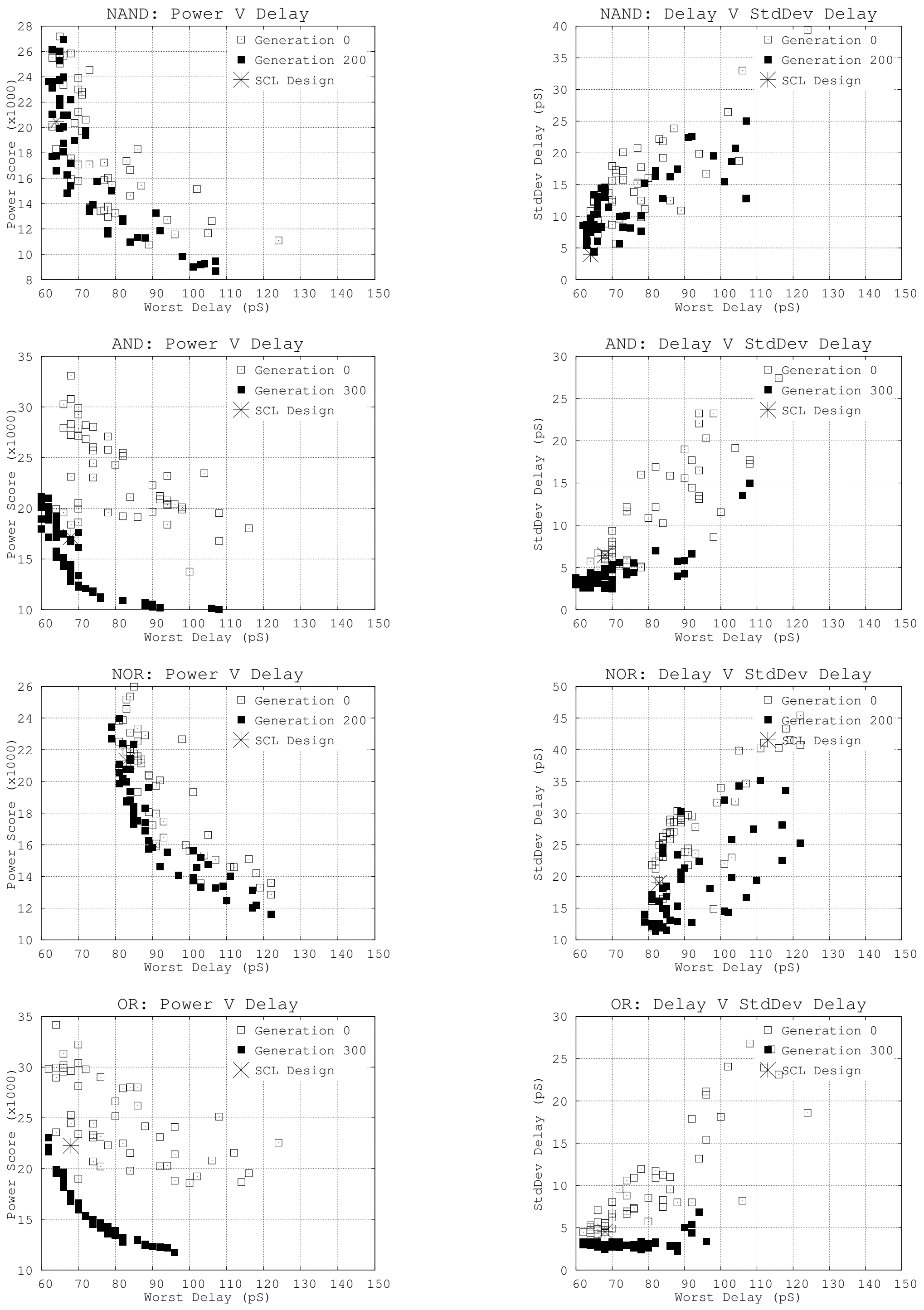


Figure 5.6: The state of the populations before and after the uniform stage of the algorithm for the NAND, AND, NOR and OR gates, including the reference design scaled from the VSCLIB library.

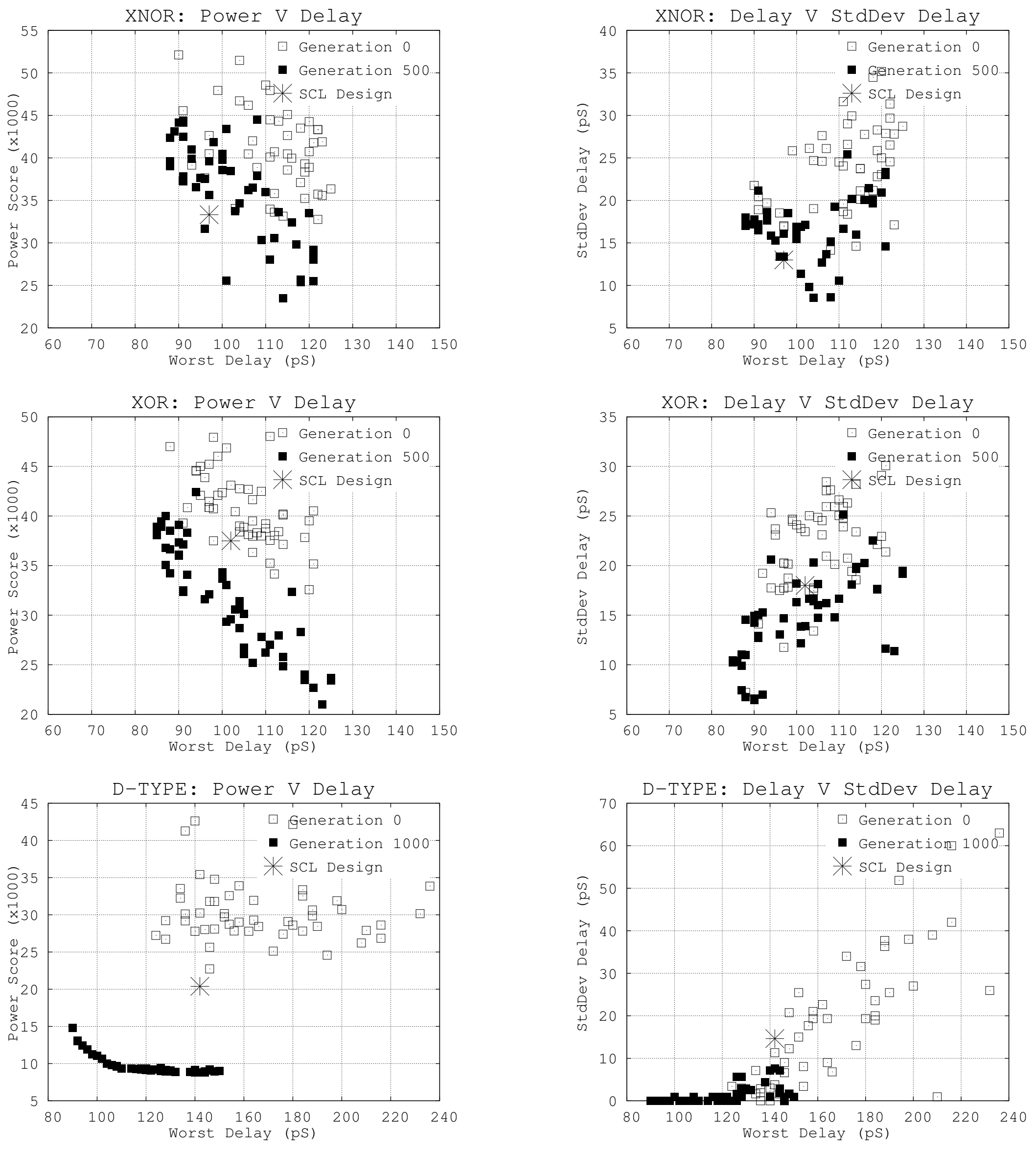


Figure 5.7: The state of the populations before and after the uniform stage of the algorithm for the XNOR, XOR and D-Type flip-flop, including the reference design scaled from the VSCLIB library.

circuit is considered the *variability-tolerant* circuit within the results.

In addition to the best extracted variability-tolerant design, each of the three extracted designs from the uniform stage, alongside the reference design, were then tested using the RDD-aware variability models for 50 RandomSPICE runs, to assess their variability tolerance. The speed-power plots extracted from all of these runs are illustrated in Figures 5.8 & 5.9. From these results it is observed that some of the extracted designs (notably the balanced NAND gate, high-speed XOR gate and both the high-speed and balanced D-Type designs) perform significantly better than the reference design in terms of variability-tolerance. The largest improvements again appear to be found in the larger circuits, particularly the D-Type, where the high-speed design appears to operate over 30% faster than the reference design, whilst consuming 25% less power and showing significantly less variability in delay times when the RDD-aware models are used. Detailed results for the scores from these tests can be found in Tables A.1 & A.2 within Appendix A.

5.2.4 Observations on Results

The *variability-tolerant* circuits for both the NAND and NOR gate runs are very similar in layout and score to the respective *high-speed* design evolved in the uniform stage. They consume more power than the reference designs, with both designs scoring roughly 12% higher on power-score, but demonstrate far less deviation in delay. In the XOR run the *variability-tolerant* circuit is both faster and more power efficient whilst also improving delay variation. After the second stage of the XNOR run, a circuit performing faster than the previously extracted *high-speed* circuit was found, which also improved on the delay variability score, albeit at a higher power consumption than the reference design.

The results did demonstrate that the SGA system could be used to optimise circuits, both using uniform models for traditional objectives of low-power and high-speed, and using variability models for creating dimension arrangements which minimised the impact of variability on the ranges of delay and power consumption. However, they also highlighted a number of issues with the system, not least the compromises that were essential to be made in order to complete the evolution runs in a sensible time-frame through the use of the two-stage process. The observations on performance cemented the need to distribute the problem of SPICE evaluation on HPC systems, particularly when the RandomSPICE and variability models were used.

The *variability-tolerant* D-Type circuit demonstrated the largest improvement over the reference design in terms of power score, worst-case delay and standard-deviation of worst-case delay, which appears to suggest larger circuits may benefit the most from optimisation.

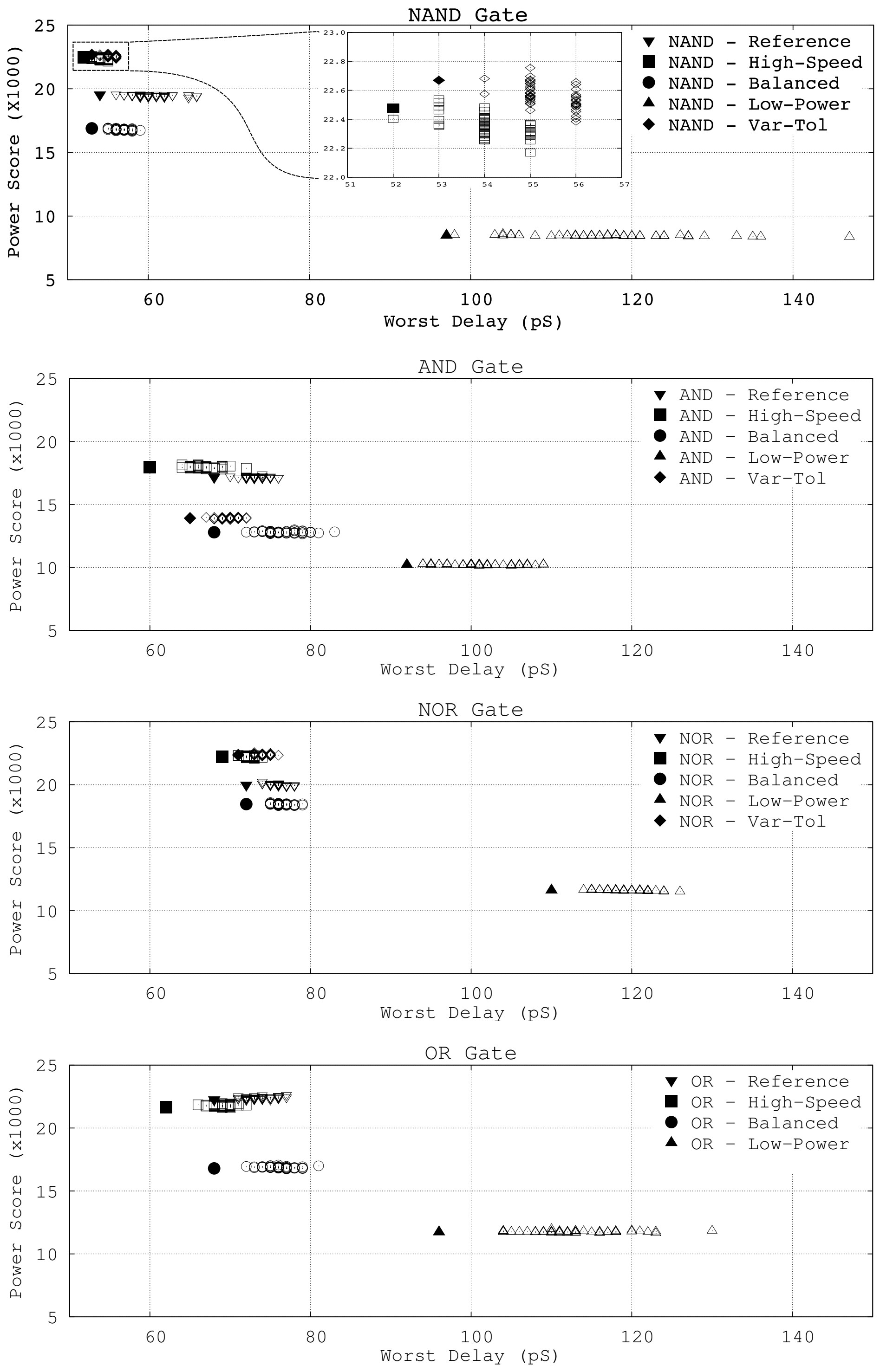


Figure 5.8: Variability within the optimised VSCLIB NAND, AND, NOR and OR designs

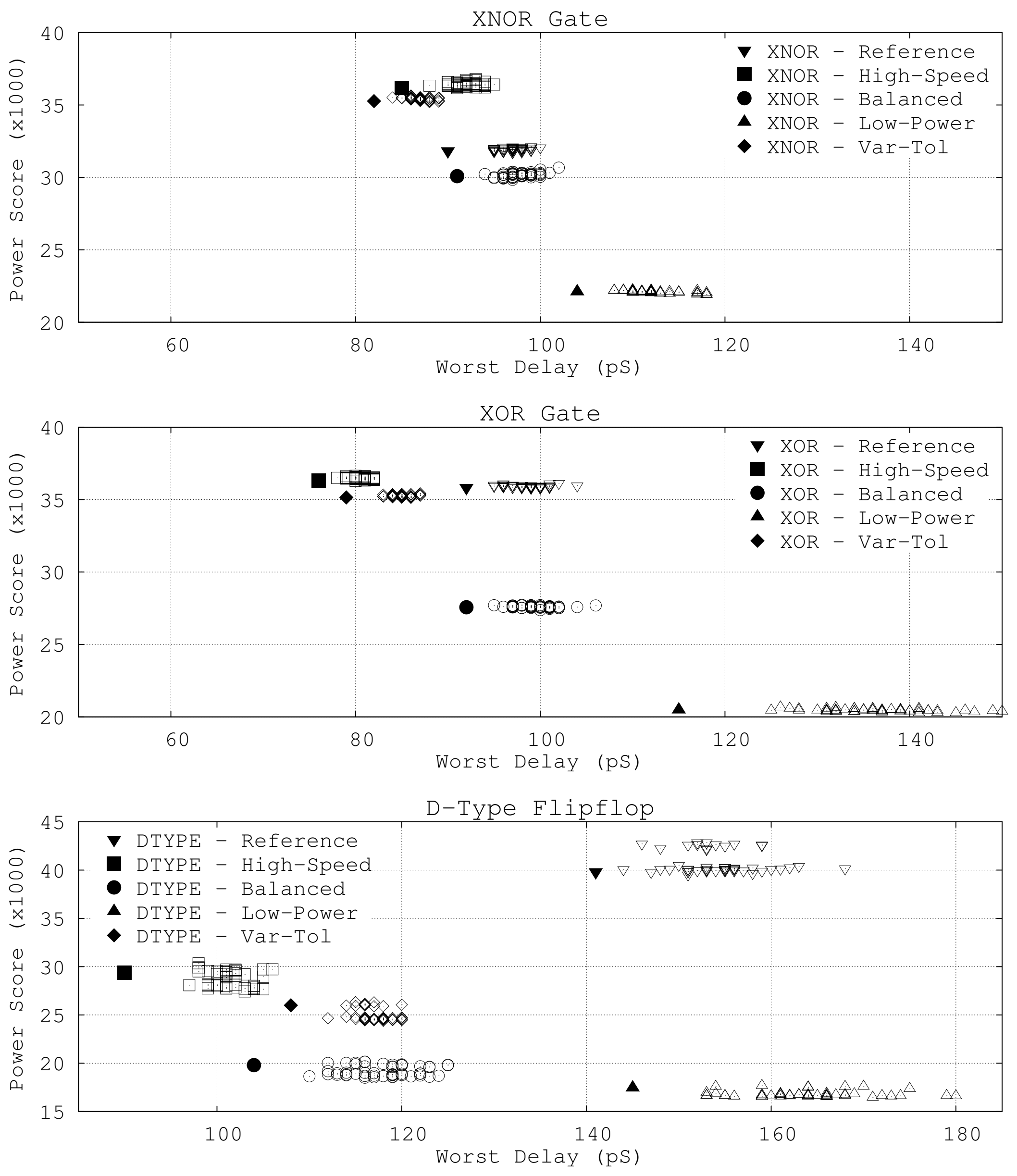


Figure 5.9: Variability within the optimised VSCLIB XNOR, XOR and D-Type designs

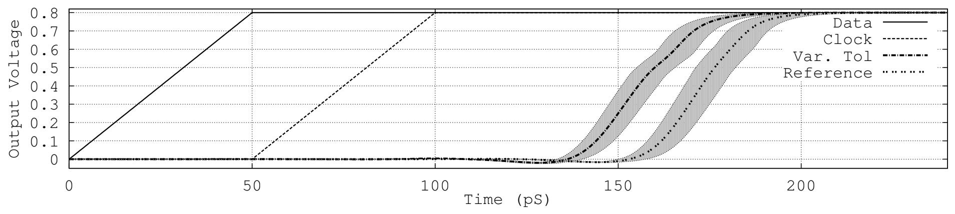


Figure 5.10: Output traces from D-Type circuits

Indeed, initial analysis on the output circuit appeared to confirm this, with the variability output traces of the *variability-tolerant* D-Type circuit when switching to logic high are illustrated in Figure 5.10, alongside the reference case. In the figure the best and worst case output are marked either side of the mean output values across 50 RandomSPICE runs. However, whilst the algorithm proved effective at optimising the circuit to fit the test criteria well, it was later discovered that an underlying flaw existed in the test-setup for the D-Type flip-flop, as is discussed in Section 5.4.4.

5.3 Comparison with Other Optimisation Algorithms

At this point, a number of tests were conducted to evaluate the performance of the SGA system when compared with other optimisation algorithms. The comparisons were made using using the MOTIVATED system with the same circuit test-bench, SPICE parameters and objectives as given in the previous set of tests. Variability was not considered in these tests due to the time taken to run variability-aware experiments on a single workstation, so the comparative tests were based on the optimisation of circuits for power and delay using uniform models only, and are compared with the results at the end of the uniform-stage in the previous sets of experiments. It is acknowledged that this is undesirable as other optimisation algorithms may indeed perform well at the variability stage. Only two test circuits from the VSCLIB were used in the comparison tests: firstly the NAND gate, and secondly the XOR gate.

5.3.1 Steepest-Ascent Hill Climbing

The first set of tests was based on a steepest-ascent hill climbing algorithm. To allow this algorithm to be used with the current MOTIVATED system, it was implemented as a adaptation of the mutation operation within the SGA framework. To allow this, the population at each generation consists of μ parent and λ offspring, where mu is set to the number of objectives [2] and where λ is $2\times$ the number of genes within the genotype. In the replaced mutation operation, each of the offspring has one gene adjusted a single unit up or down from the parent's value respectively, such that all λ offspring represent the entire set of possible solutions with one transistor value shifted. These are then assessed by SPICE and ranked using the multi-objective fitness function, using the same method as before, with the highest ranked individuals being promoted to the next generation. A memory of previously visited circuits is stored, with the algorithm terminated once a generation in which all of the new individuals have previously been assessed is found.

The algorithm was used with two test circuits: the NAND gate and the XOR gate from the VSCLIB, as used in the previous experiment and illustrated in Figure 5.5. For the first test, the algorithm was launched with an initial population in which every NMOS transistor was set to 4-units of width (140nm) and every PMOS transistor was set to 8-units of width (280nm) [3]. For the NAND gate circuit, the algorithm terminated after 19 generations. The six parents from the final population were all found in the final population of the equivalent run in the SGA algorithm: for this circuit, the SAHC had proven far more efficient. However, for the XOR circuit, the algorithm terminated after 42 generations, and the extracted parents from the final population were less effective that those found in the SGA run in all objective scores with the exception of circuit area. These results indicate that the landscape for the NAND gate circuit is relatively smooth, despite the number of dimensions, thus very suited to the SAHC algorithm; in contrast the landscape for the XOR circuit clearly contains many local optima.

5.3.2 SAHC With Random Walk

The SAHC algorithm was adapted so that it both began with a randomised population, and so that in each generation an extra randomised parent circuit was included in the population, to add a periodical random walk to the algorithm. The algorithm was once again applied to the NAND and XOR circuits, although this time 5 runs for each test were carried out, with the algorithms terminated after 357 generations for the NAND circuit and 848 generations for the XOR circuit; these values were chosen so that an equivalent numbers of circuits would have been assessed in both the SAHC run and the SGA run described previously.

As expected, in all five runs for the NAND gate circuit, results were found which matched those from the SGA algorithm, confirming that both the extracted results are indeed the best that can be acheived for the given objective scores, and also that the underlying shape of the search space for the NAND circuit is devoid of many local optima. For the XOR circuit,

[2]Excluding the functionality objective

[3]As there is algorithm is purely deterministic there was no need to do more than one run.

there were significant differences observed across the five runs, with the best observed results across all the SAHC algorithm runs producing better results than the SGA in certain fitness objectives *(area and σ delay)*, and worse in others *(power)*. Given both were given a similar length of circuit evaluations, it is hard to determine categorically which is the more efficient, although the results do suggest that the SGA may be more effective in larger circuits, with more complex fitness landscapes, whilst the SAHC clearly outperforms it in smaller circuits with simple landscapes. A detailed comparison of all the results between the SGA, the SAHC and the SAHC with random walk can be found in Table A.3 within Appendix A.

5.4 Optimising Using Multiple-Voltage Sources within 2-Input Combinatorial Logic Cells

This section describes the another set of experiments and results obtained when the MOTIVATED system, using the SGA, was applied to the problem of optimising transistor dimensions within a open-source standard-cell library. These experiments, carried out shortly after those described in the previous section, amended the skeleton netlist by different supply voltages to drive each of the pairs of CMOS transistors within the netlist. As with the experiments described in the previous section, all the experiments described in this section were processed on a quad-core workstation.

The motivation behind this experiment stems from recent developments in the manufacture of microprocessors and other ULSI circuits is the use of multiple voltage supply rails. In the simplest implementation a pair of supplies is used, one at a higher voltage than the other [99]. Functional blocks that require higher drive or lower delay will generally use the higher voltages, with all other blocks connected to the lower voltage. This implementation can reduce power requirements in sections of the device whilst keeping up to target specifications for drive and speed; it is proposed that future designs might benefit from the use of different supply rails at a finer circuit level, so in these experiments each PMOS transistor can be connected to an independent supply, with the value of each supply evolved alongside the transistor widths.

5.4.1 Methodology

The evolutionary process repeats the two-stage process that was described in the previous section, and uses the same versions of MOTIVATED, RandomSPICE and NGSPICE as used in the previous section, utilising the RDD-aware model library based on a 35nm Toshiba device. The experiments are again based on the two-input logic circuits found in the VSCLIB cell library.

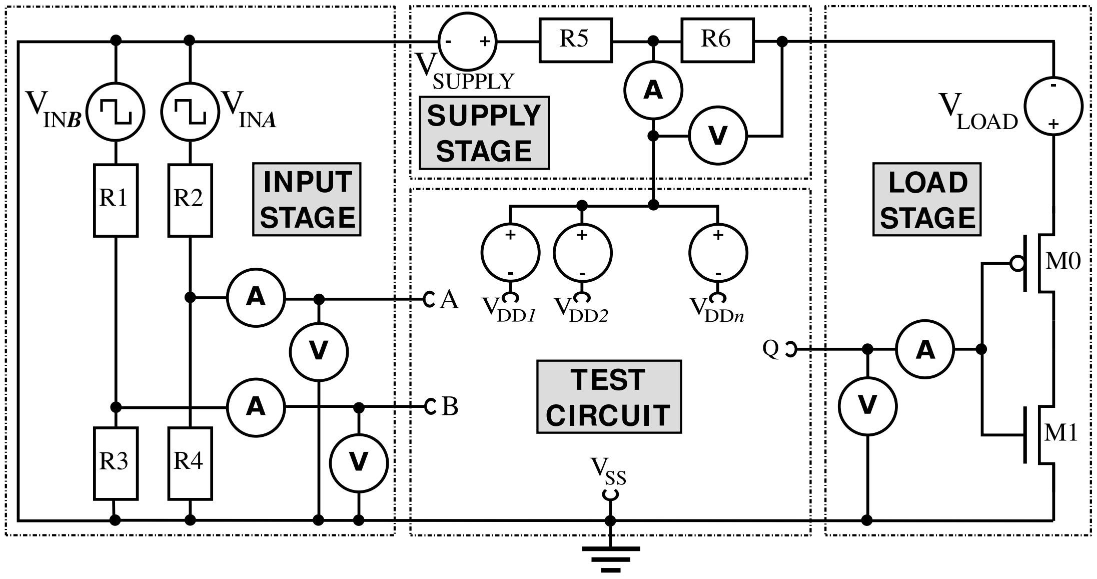


Figure 5.11: Test arrangement for optimising multiple source voltage circuits

The differences between the two experiments are found in the template netlists, which include an additional voltage supply, as illustrated in the adapted test-bench of Figure 5.11. The template netlists for the experiments can be found in Appendix C.1.3. The fitness objectives used in the experiments are the same as those shown in Table 5.2.

Test Circuits

Three different logic circuits from the VSCLIB standard-cell library were chosen, based on the observations from the experiments described in the previous section. The test circuits comprised the 2-input XOR-gate with 2 supply sources, the 2-input XNOR gate with 3 supply sources and the D-type flip-flop with 8 supply sources [4]. These three circuits are used for three different sets of tests. In the first set, the transistor widths are all static, based the scaled-down dimensions found in SCL reference design, and only the value of the voltage sources are determined by the algorithm. These tests are denoted by **MVS** in the tables and figures.

In the second set of tests, denoted **WO**, the multiple voltage sources are omitted with the transistor widths being determined by the algorithm. Each of the circuits is converted to a SPICE netlist within which each of the transistors is assigned a width between 2-20 units for NMOS transistors and 4-28 units for PMOS transistors, where each unit is a $35nm$ width

[4]The chosen netlists include the D-Type flip-flop, as the experiments predate the detailed analysis on the circuit in which the problems with the test-benchmark were discovered.

Table 5.5: The run parameters used in the multiple-voltage source experiments

Parameter	X-OR Gate	X-NOR Gate	D-Type
Voltage Supplies	2	3	8
Transistors	9	9	24
Genotype length (**MVS**)	2	3	8
Genotype length (**WO**)	9	9	24
Genotype length (**MVS+W**)	11	12	32
Population size	200	200	200
Number of Parents	50	50	50
Mutation rate	1	1	1
V_{DD} (V)	0.8	0.8	0.8
Clock (GHz)	4	4	2.5
Sample Period (pS)	2	2	2
Uniform Stage Generations	300	300	300
Variability Stage Generations	20	20	20
RandomSPICE Circuits	50	50	50

transistor. In the final set of tests, denoted **MVS+W**, both the transistor widths and multiple voltage sources are determined by the algorithm. The template netlists for the **MVS+W** circuits can be found in Appendix C.1.3.

Run Parameters

The parameters used in these experiments were adjusted slightly from those described in the previous section, as shown in Table 5.5. Due to time constraints, fewer generations were used in both the uniform and variability stages, and a smaller population size was used. To attempt to improve the accuracy of results, a higher sample rate was used. This compromise was deemed necessary to allow the future experiments to be carried out, which likely offered more practical designs, whilst still gauging the potential performance of the MVS technique.

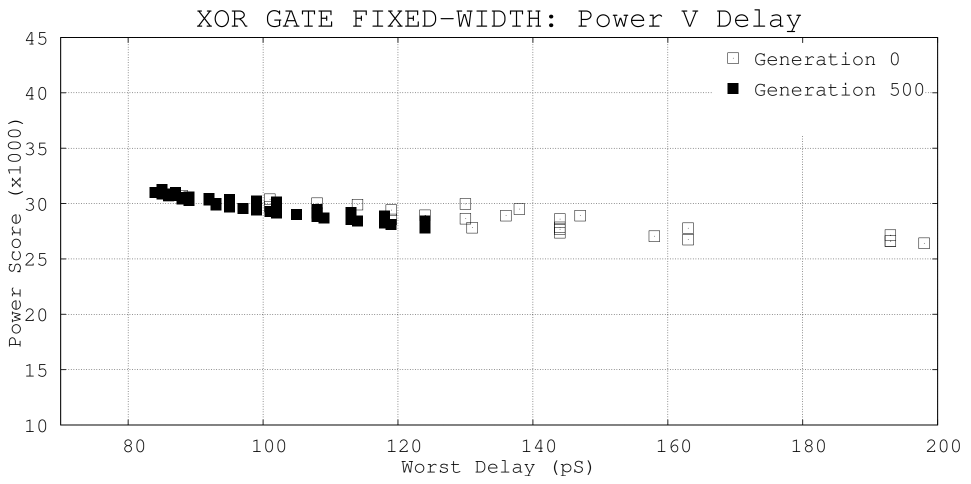
5.4.2 Results

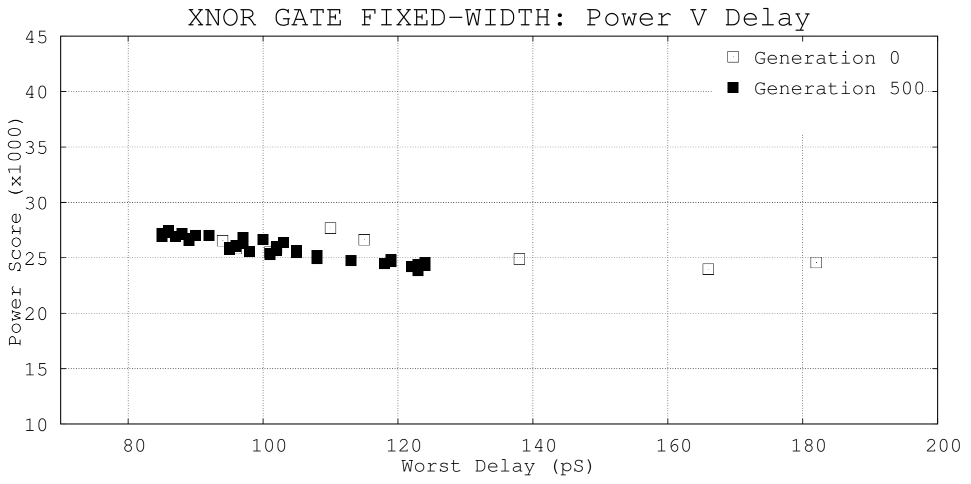
For each of the three test designs the algorithm was run using the uniform transistor models, starting with an initial random population. The results for the fixed width (**MVS**) circuits, in which only the voltage sources are evolved, are illustrated in Figure 5.12, which compare the power and delay performance of the initial random populations to those at the end of the uniform stage. The XOR and XNOR results suggest that the benefits of adjusting the voltage sources within the reference design have limited benefit in improving circuit power consumption or delay. Of the evolved designs, those with the lowest delays generally had close to zero voltage on the evolvable voltage sources, and those circuits which demonstrated an improvement in power score over the reference design did so with a delay time significantly worse than the reference design. The D-Type, which has many more connections to the supply and thus more scope for optimisation via evolution of supply voltages, demonstrated more promising results, with arrangements which appeared performed as fast as the standard reference design whilst consuming less power, due to the reduction in supply voltage in non-critical parts of the circuit.

Uniform Stage

The results for the uniform stage of the algorithm optimising widths only (**WO**) are shown in Figure 5.13. Given the setup for this particular set of circuits is very similar to those described earlier, it is not surprising that similar results were seen: these results show that the initial population contains some very poor-performing circuits, including several circuits which fail to operate at the required speed, however by the end of the stage the algorithm has successfully optimised the population of circuits so that all the promoted parents are functioning correctly, with many individuals operating at faster speed and lower power than the reference design from the cell-library, particularly so with the D-type circuit. These results once again illustrate the significant potential for the use of evolutionary methods in the optimisation of transistor dimensions within a standard cell, with all three tests suggesting the conventional methods use to design the reference circuits may not be resulting in the optimal balance of circuit delay and power consumption.

The final set of test results from the uniform stage are illustrated in Figure 5.14, which compares the initial to final population when both transistor widths and supply voltages are evolved simultaneously (**MVS+W**). These tests show the worst performing initial population of the three tests, due to the wide range of possible transistor dimensions and supplies, and the impact that these values have on the circuit functionality. By the final generation the evolved populations have vastly improved and show an easily identifiable power-vs-delay pareto front. The final population of these tests include both circuits which perform faster, and circuits which require less power, than the corresponding final populations of the supply-voltage only and width-only tests.





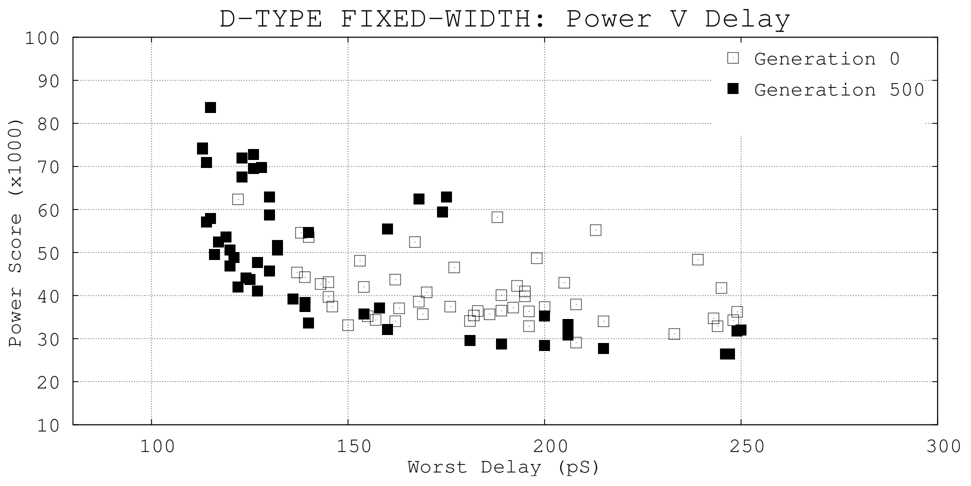
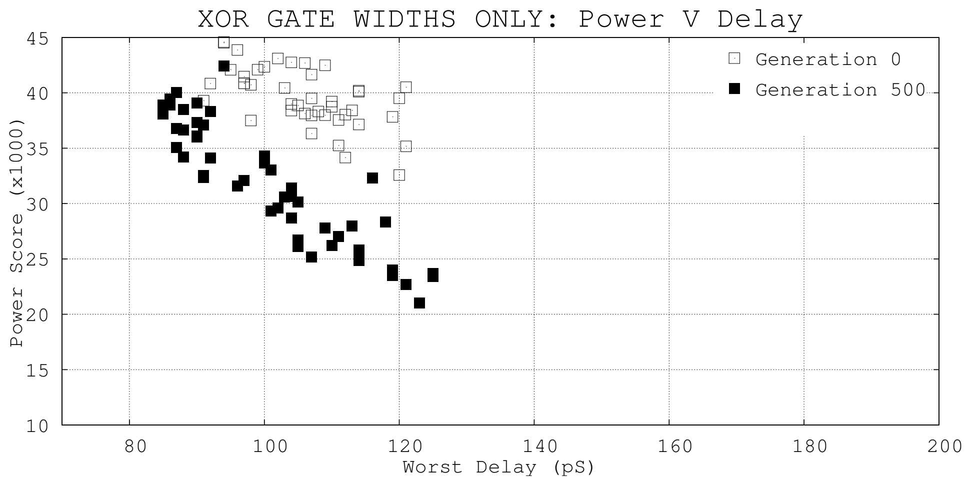
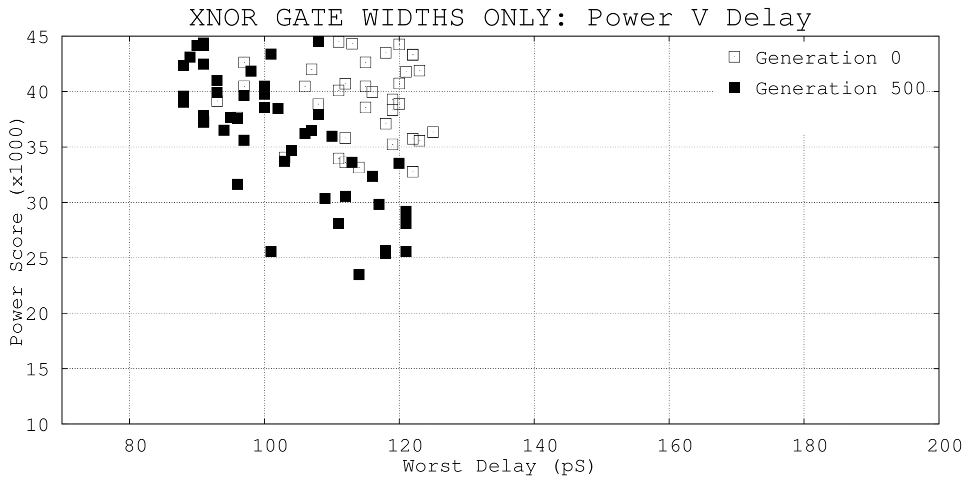


Figure 5.12: The state of the population before and after the uniform stage for the evolved voltage-source (**MVS**) circuits





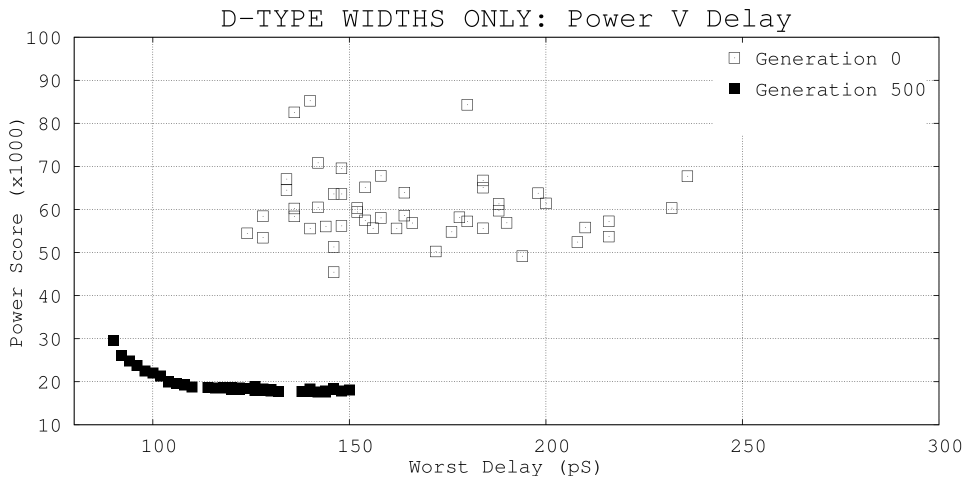
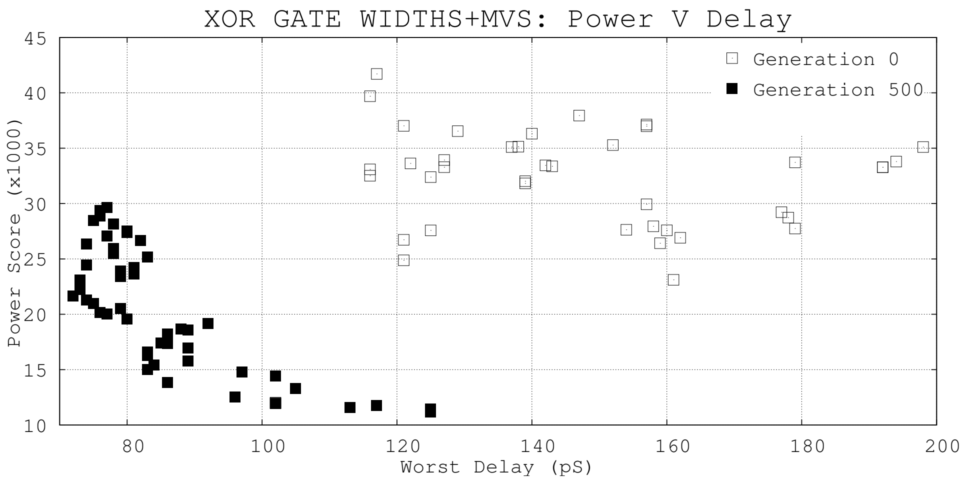
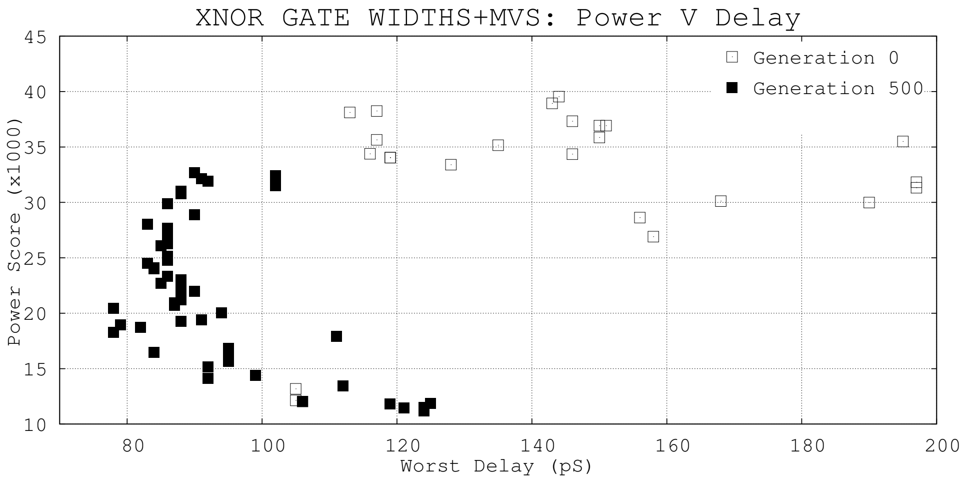


Figure 5.13: The state of the population before and after the uniform stage for the evolved transistor width (**WO**) circuits





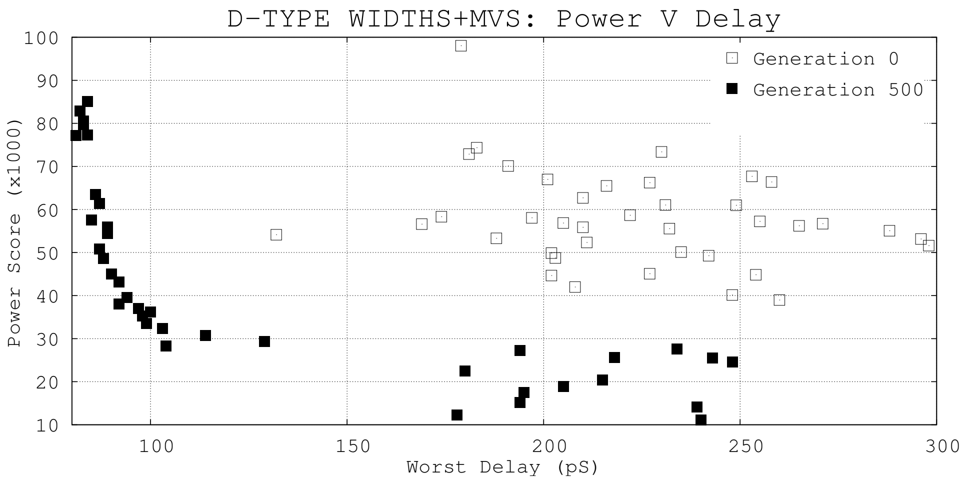


Figure 5.14: The state of the population before and after the uniform stage for the evolved width and voltage-source (**MVS+W**) circuits

Variability Stage

For each of the nine experiments, the algorithm was then restarted using the final population from the uniform stage, and run for twenty generations with each individual creating 50 variability-aware netlists. At the end of this short variability-aware run, two circuits were chosen from the final population which demonstrate potential improvements over the reference design. The first circuit is the one which best fits the criteria of minimising circuit power consumption and area without severely affecting circuit-delay and variability tolerance (larger circuits will typically switch faster and consume more power); this is considered the *low-power circuit*. The second circuit is the one which demonstrates the best balance of low-delay and low standard deviation across delays; this is considered the *high-speed* circuit. The power score against delay for each of the designs is illustrated in Figure 5.15, in which each of the fifty RDF-aware outputs is plotted with a outlined shape, along with the same circuit using the uniform-models plotted with a solid shape. A detailed breakdown of the scores for each of these circuits can be found in Table A.4, in Appendix A.

5.4.3 Observations

From the variability results it is clear that arrangements optimised for low-power consumption show significant variability in delay time when the RDF models are used; the low-power arrangements typical contain very narrow transistors which will vary the most between random circuits. In contrast, the arrangements optimised for high-speed are far less variable. When compared to the results for optimising dimensions alone, it appears that extending the evolution to optimise the supply voltages for the transistors within the cell can extend the power and delay front producing circuits that are faster or lower in power than those which are optimised using transistor widths alone. However, this clearly adds a significant degree of complexity to the actual fabrication of larger circuits. From a performance point of view, these experiments once again underlined the need to migrate towards an HPC solution for optimising with variability models.

5.4.4 The D-Type Problem

In both the standard optimisation runs and those using multiple-voltage sources, some of the best results were observed when optimising the D-Type flip-flop function; in some cases the results were so dramatically improved over the reference design that suspicions were aroused about the suitability of the test setup. Upon further analysis of the circuit, it was realised that whilst the evolved designs functioned as expected under the given test-conditions, this did not

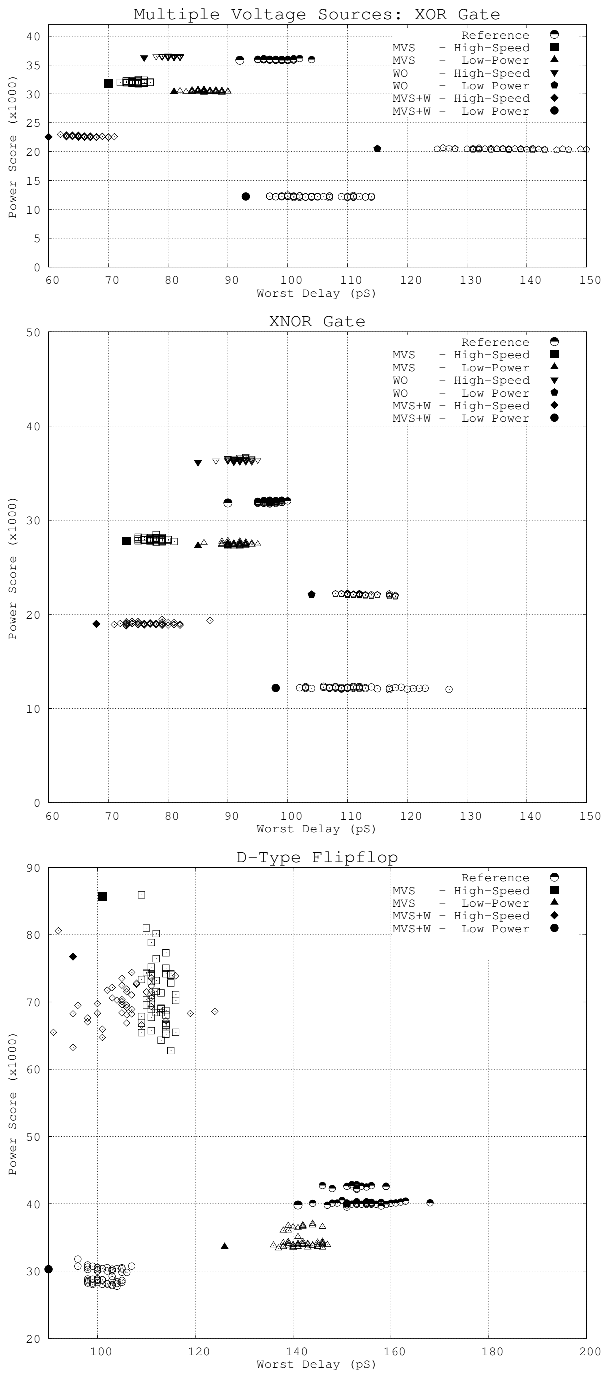


Figure 5.15: Variability within the extracted circuits of the multiple voltage-source experiments. In each chart, the scores for each of the RDD-aware results are plotted using an outlined shape. The same circuit when evaluated with uniform models is plotted using a solid shape.

mean they functioned fully as a D-type memory cell, due to limitations in the test-setup. The reason for this is that in the used test bench, both the clock-input and the D-input to the flip-flop can change simultaneously, whilst in a realistic test the D-type input must change state before the clock-input does. As a result, whilst the circuits optimised did perform very well at the given test, they failed when applied to a more accurate test based on a standard D-Type timing cycle. The result underlined the pitfalls that can occur if any inaccuracies exist in the testing setup when evolving circuits: the algorithm worked exactly as intended, however a human error in the setup produced misleading results. Whilst an amendment to the test setup to allow accurate analysis of flip-flops and memory cells is possible (and has now been created in work that post-dates this thesis), it was decided to focus on two-input logic circuits for the remainder of the experiments.

5.5 Optimising CGP Evolved Designs

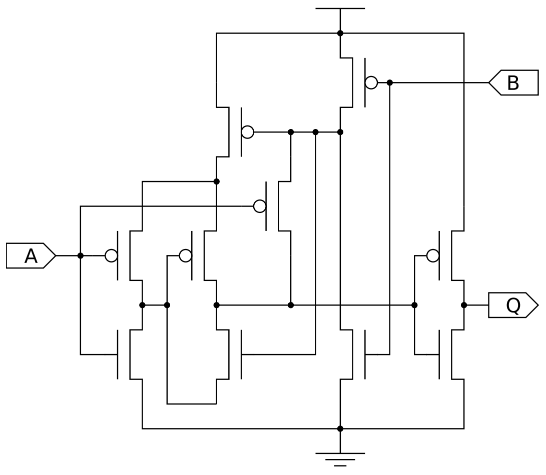
A final set of experiments carried out using the MOTIVATED system prior to SGE-cluster implementation being created involved optimising the transistor widths in designs that had previously been evolved using CGP.

5.5.1 Using CGP To Evolve Standard-Cell Topologies

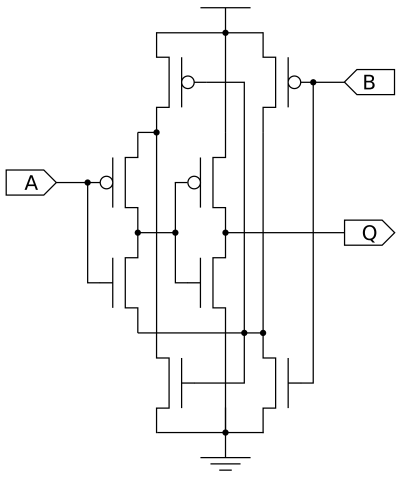
As part of his research in the nano-CMOS project, Dr. James Walker has devised a system for evolving standard-cell topologies, based on an adapted representation of previous work he had conducted in Cartesian Genetic Programming, as previously discussed in Section 4.1.2. As a result of this work, two novel topologies for XOR and XNOR gates had been extracted. Using the same methodology and settings described in Section 5.2, the optimisation process was applied to the new topologies to see how they performed in terms of power, speed and variability tolerance.

5.5.2 Evolved Designs

The best designs that were extracted from the CGP runs are illustrated in Figure 5.16. The evolved XOR design uses 10 transistors, and the evolved XNOR uses 8 transistors. Both of the evolved designs are structurally and characteristically similar to conventional CMOS designs, however differ from any designs found in the VSCLIB and NANGATE cell-libraries, and are considered as fair as the public-domain literature is available to be novel in their topologies. During the CGP evolution process within MOTIVATED all the transistors within



(a)



(b)

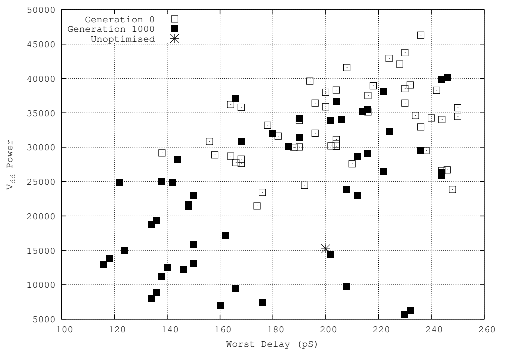
Figure 5.16: CGP-evolved designs for the XOR gate (a) and the XNOR gate (b).

the circuits were given static width dimensions of 5-units for NMOS (175nm) and 10-units for PMOS (350nm). The goals of the optimisation process using the SGA was to optimise these dimensions to extract circuits which improved performance in terms of delay, power and variability tolerance. The template netlists for these circuits used by the SGA can be found in Appendix C.1.4.

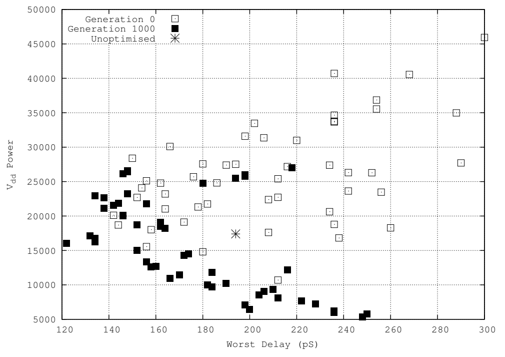
5.5.3 Results

The MOTIVATED system was used with the SGA to optimise the two evolved circuits with uniform models, using exactly the same parameters as in the previous runs on the VSCLIB described in 5.2 in order that the two sets of results can be directly compared. A comparison of the initial and final populations at the end of the uniform stage of the evolutionary run are shown in 5.17. For both circuits a marked improvement in the best power and delay scores can be seen when compared to the unoptimised design (with all NMOS transistors at 5 units wide and all PMOS transistors at 10 units wide); however, when compared directly to the VSCLIB results from Figure 5.7, the pareto fronts are less defined with a greater scattering in the output, particularly in the case of the XOR design. The algorithm was then restarted using the variability models as before. At the end of this run, three circuits were extracted: a high-speed design (**HS**), a design balanced between high-speed and low-power (**SP**) and a low-power design (**LP**).

The fitness scores for these three extracted circuit are compared with the equivalent scores



(a)

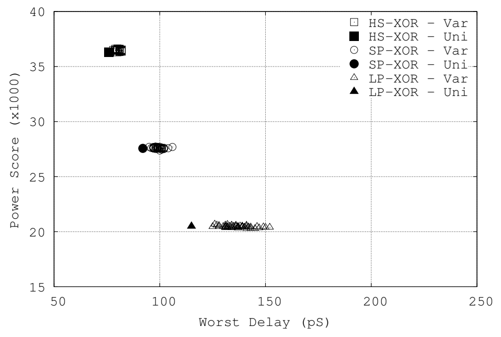


(b)

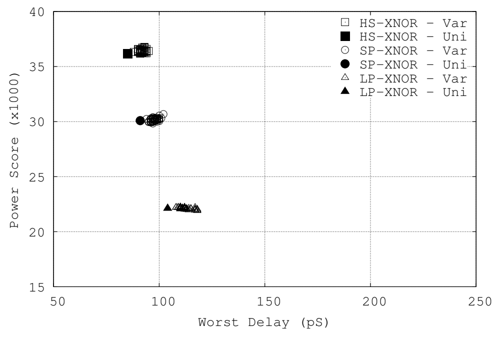
Figure 5.17: The worst case delay and power statistics of the population at the initial generation and after the uniform-stage of optimisation for the CGP-evolved XOR (a) and XNOR (b) standard cell designs.

based on the SCL designs in Table A.5, found in Appendix A. Although the optimised evolved designs for the XOR and XNOR are much slower than the SCL designs, they are also significantly lower in power consumption, and are smaller in transistor area (this is not surprising for the evolved XNOR, as it contains one fewer MOSFET than the SCL design, so it naturally has an advantage). The effects of variability on the worst-case delay scores and power scores for both the SCL designs and the CGP-evolved designs are compared in Figure 5.18. The optimised VSCLIB SCL designs show less variability than the CGP evolved designs, both terms of power and delay. However, this is partly a consequence of the VSCLIB designs being larger and consuming more power, both of which have been shown to reduce the effects of variability [22].

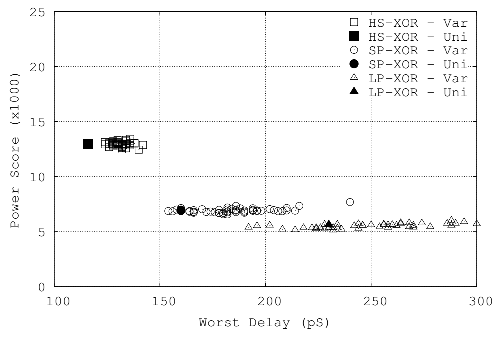
The range of delays across the variability runs is far greater in the CGP evolved designs than the SCL designs; at the greatest extreme, the low-power evolved XOR design ranged between 190 pS and 322 pS delay across RandomSPICE circuits, which is in stark contrast to the range of 80 pS to 86 pS delay observed for the high-speed VSCLIB design. These experiments thus revealed that whilst the CGP-evolved designs offered benefits over the SCL designs if low-power circuits are required, they performed poorly when variability was considered, whilst the conventional designs actually demonstrated a far better degree of variability-tolerance. This does, however, emphasise both the importance that topology choices have in created variability-tolerant designs, and the sacrifices in terms of delay-variability that have to be made when low-power cells are needed.



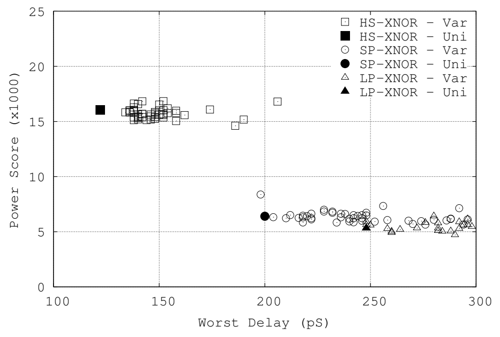
(a) VSCLIB XOR



(b) VSCLIB XNOR



(c) CGP-Evolved XOR



(d) CGP-Evolved XNOR

Figure 5.18: Comparison of the impact of RDD variability on the worst-case delay and power statistics between the conventional VSCLIB designs (top row) and the CGP evolved designs (bottom row) for the high-speed, balanced and low-power XOR and XNOR gate extractions.

5.6 Detailed optimisation using HPC Cluster

The experiments described in the previous sections highlighted the strong points and the flaws in the SGA optimisation system. On the positive note, when the test parameters were carefully setup, improvements in the performance of standard cells could be observed, both in terms of the traditional delay and power characteristics, but also in terms of variability tolerance. However, if there are any problems in the test arrangement, the evolution could produce misleading results, as was seen in the studies of a D-Type flip-flop. Furthermore, the two-stage system that was needed to produce results on a single workstation was seriously compromised, as it primarily optimised the power and delay objectives with the variability tolerance effectively tagged on as a extra goodwill measure once the main phase of optimisation was complete. To genuinely optimise cells for variability tolerance, it would be essential to include variability in the objectives from the outset, and in order to achieve this additional computational resources would be essential.

The goal for this experiment were to revisit the task of optimising a standard-cell library, however this time optimise using the variability models throughout the process (i.e. a single stage process), extracting designs that exhibit improved variability tolerance alongside both higher speed and lower-power designs than the reference design from the library. By including variability scores as objectives from the offset, the multi-objective algorithm will be promoting designs which exhibit such a tolerance throughout the duration of the algorithm. To achieve this, the MOTIVATED system was revised as discussed in Section 4.6 to allow the RandomSPICE, NGSPICE and score calculations to be all processed in parallel on the MAVEN cluster, as illustrated in Figure 5.19.

5.6.1 Methodology

In addition to the changes to the workflow to allow the HPC operation described above, a number of other differences were made between these experiments and the previous based on experience from the previous tests and advances in the available technology. The models used by RandomSPICE in these tests were based on bulk 35nm Toshiba devices, and included the effects of LER and surface roughness in addition to RDD. The standard cell library from which cells were selected for optimisation was changed to one which was closer to the 35nm dimensions used by the variability library.

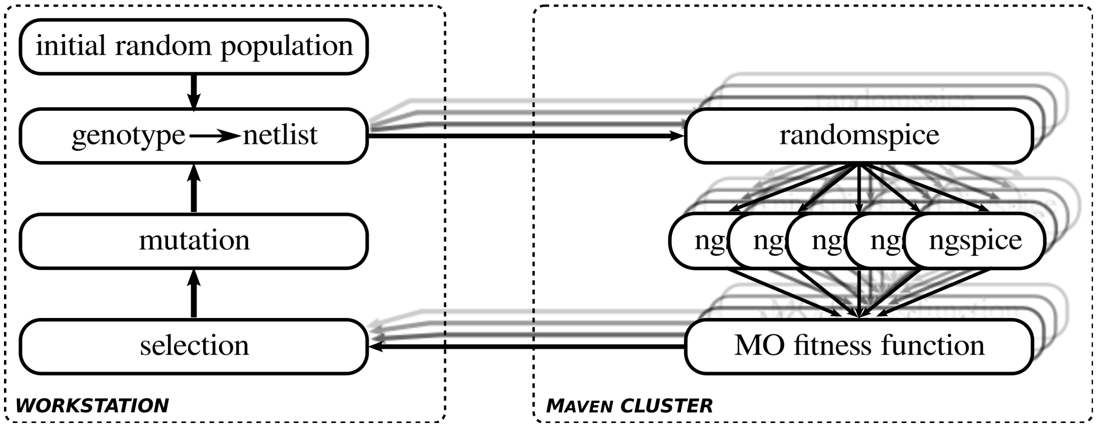


Figure 5.19: The revised single stage cycle to optimise circuits using the MAVEN HPC cluster for variability analysis.

Test Circuits

For these tests cells from the fourth release of the Nangate 45nm Open Cell Library was used [7]. This is an open-source standard-cell library which is designed for university research in developing workflows and new algorithms, and contains 38 different functions which range from buffers up to S–R flip-flops, at a range of drive strengths. Nangate is a fast-growing multinational developer of Electronic Design Automation (EDA) software with headquarters in Denmark and USA.

Four test circuits were chosen from the library for optimisation. These are a buffer, a NAND gate, an OR gate and an XOR gate. The first two were chosen as they are extensively used in many circuits, and, based on the previous experimental data, are generally already heavily optimised for power and speed [5]. In contrast, the previous experiments on the VSCLIB suggested there was significant room for improvement in the OR and XOR circuits, both with and without the consideration of variability, so it was desired to see if this was a side-effect of the reduction process in scaling from a 130nm library down to 35nm device, or whether it was a trend which would be repeated in the more modern cell library.

Objectives

Based on the observations from previous experiments, six objectives were selected for use in the optimisation process, which are shown in Table 5.6. The first priority is, as with all the

[5]This is to say that the reference designs have generally fallen on the optimised pareto-front for power vs delay. Whilst faster designs may be extracted, these will come at a power penalty, and vice-versa with low power designs.

Table 5.6: Fitness objectives used in the single-stage optimisation run

Objective	Function	Variability Statistics	Index[1]
1	Functionality	Worst-Case	100
2	Worst-Case Delay	Mean	406
3	Worst-Case Delay	Range	306
4	Worst-Case Delay	σ	506
5	Standard-Deviation Delay	Mean	407
6	Supply Power Score	Mean	411

[1] The index value is that used in the parameters file to describe the objective, see Table 4.2.

other experiments, the circuit functionality, so the worst-case result across of the RandomSPICE circuits is taken; as such, if any of the circuit fails the functionality test, that individual will be demoted below all fully successful individuals.

To promote the fastest circuits through each generation of evolution, the second objective is the mean-value of the worst-case propagation delays across the RandomSPICE runs. In addition to the mean value, the range and the standard deviation (σ) of the worst-case delays are included as the third and fourth objectives; there will likely be a degree of correlation between these scores as results which show an excessive range will likely also have a high standard deviation: having both objectives in the set will increase the likelyhood of a variability tolerant circuit being promoted to the next generation due to the increased number of fronts in which it will dominate other individuals. The fifth objective is the mean value, across all the RandomSPICE circuits, of the standard-deviation of all the delays within the test sequence. This will advance circuits which offer balanced switching times across all transitions. The final objective is the mean value of the power supply score across all runs. Based on both previous experiments and general logic design rules, it was decided the variation in delay is a more significant problem than variation in power, thus the objectives which aim to minimise variability focus on delay; the final objective serves to promote individuals which consume a low average power across all the RandomSPICE circuits.

Run Parameters

The run parameters used for this set of experiments are given in Table 5.7. Based on previous results and tests on the new circuits it was found that the clock speed could be boosted as high as 10GHz without causing excessive functionality failures; only the very worst performing circuits would fail the functionality test at these clock speeds and thus fall to the bottom ranks,

Table 5.7: The run parameters used in the single-stage optimisation run

Parameter	Buffer	NAND Gate	OR Gate	XOR Gate
Genotype length	4	4	6	10
Population size	80	80	80	80
Number of Parents	40	40	40	40
Mutation rate	1	1	1	1
V_{DD} (V)	1.0	1.0	1.0	1.0
Clock (GHz)	10	10	10	10
Sample Period (pS)	0.5	0.5	0.5	0.5
Number of Generations	100	100	100	100
RandomSPICE Circuits	50	50	50	50

and such circuits would not be desirable in future populations anyway. As a consequence of the increase in clock speed it was possible to reduce the sample period to 0.5pS, which ensured that sufficient discrimination between similar circuits would exist. To both speed up the generation rate and optimise the cluster-based workflow, only one offspring was produced per each parent, with a total population size of 80 [6]. This resulted in each node evaluating one circuit per generation, with the exception of the initial generation in which two would be evaluated. The mutation rate was unchanged from the previous experiments. All of the runs were set to terminated after 100 generations, with the initial and final populations saved for analysis.

5.6.2 Results

When the run was complete, the set of scores for the final population was compared the scores for the reference design. Unlike in the previous experiments, it was decided to extract a single circuit which demonstrated a potential improvement over the reference design which appeared to offer the most variability tolerance, without accepting a penalty in the delay or power over the refence design. Thus, whilst the decision process for extracting the circuits considered all of the available objective scores, priority was given to the two which concerned the variability in the propagation delay: the range of the worst-case delays and the standard-deviation of the worst-case delays. The motivation for extracted just one variability tolerant (with regards delay) circuit was to allow for a detailed and time consuming post-run analysis of this circuit to be made.

[6] At the time these tests were run, the MAVEN cluster had 10 functioning worker nodes, each with 4 cores, thus appearing as 40 logical nodes to the SGE system.

Table 5.8: The relative improvement observed between the optimised design and the reference design for the mean worst-case delay and power scores.

Parameter	Buffer	NAND Gate	OR Gate	XOR Gate
Mean Worst-case delay (%)	12.6	1.4	11.8	9.0
Mean Power Score (%)	14.5	7.1	4.0	27.6

The circuit chosen for each test was the one which offered the best scores for these two values whilst scoring at least as good as the reference design in both power and delay. For all four experiments, circuits which met these criteria could be found: Table 5.8 shows the improvement in the mean worst-case delay and power scores between the extracted optimised design and the reference design for the four tests. As before with the VSCLIB, the improvement seen in the NAND circuit is marginal, which can likely be attributed to both the minimal length of the transistor chain within the cell [7], and the fact that the cell is already likely extensively optimised both due to its small size and heavy-use in larger circuits.

Detailed Variability Analysis using Scot-Grid

Whilst the results from the final population appear to indicate a degree of variability tolerance in the extracted designs, this is only measured over 50 different RandomSPICE simulations. In order to more deeply analyse the results, a much larger statistical study of each circuit is necessary. Whilst it would be possible to do such detailed simulations on the MAVEN cluster, it was decided to exploit the greater HPC resources offered through the e-Science ties in the nano-CMOS project. This allowed access to the Scot-Grid cluster, on which all the necessary infrastructure to run large scale statistical Spice simulations, using RandomSPICE and NGSPICE, was already in place. Using this resource, each of the four extracted circuits were evaluated using 100,000 RandomSPICE simulations, with the data stored on the distributed AFS filesystem. For these tests the sample-rate was increased to a more detailed 0.1pS per sample. This process, which would have taken months on a single machine, or days on the MAVEN cluster, was completed in a few hours [8].

The results from the detailed analysis of the extracted circuits are shown in Figures 5.20 to 5.23. In these plots, the scatter clouds for the sets of RandomSPICE circuits from the ex-

[7]The NAND-gate has a chain length of 1, meaning that no transistor drains are connected to the gate of another transistor. The buffer and OR gate have chain lengths of 2, the XOR has a chain length of 3.

[8]The round-trip time from submitting the jobs to receiving locally all the completed data was under 8 hours, although it is not known what the utilisation level of the Scot-Grid system was, or what other jobs were being run or queued simultaneously.

tracted, optimised design and the original reference design from the cell library are compared, with the worst-case delay score plotted on the (x-axis) and the power score plotted on the (y-axis). Included in the plot are histograms which show the distributions of worst-case delay and power score across all the runs. From the scatter plots, it is clear that in all four tests a circuit has been extracted which outperforms the reference design both in terms of worst-case delay and power score.

From analysis of the delay histograms, it can be seen that in all tests that the spread in delay distribution is narrower - considerably so in the case of the buffer and OR gate. Perhaps of even greater significance is that the extent of the tails of the distribution, the outlying circuits which perform particularly poorly, has been greatly reduced in the buffer, OR and XOR designs. This is particularly important as it is the outliers that exist in the distribution that will affect the overall timing of a larger circuit, and reduce yield if not correctly accounted for. The best improvement in this regard is seen in the OR gate: the mean worst-case delay is improved 11.8% between the reference and optimised design, reducing from 17.5pS to 15.6pS, however an even greater improvement is seen in the worst-performing circuits. In the optimised designs, the worst-case delay ranges from 14.6ps to 17.2pS, a 2.6pS range, with all circuits falling within 10% of the mean value. In the reference design, the range is much wider at 16.2pS to 21.2pS, with a 5.0pS range: five circuits in the 100,000 test cases have a delay that is over 20% slower than the mean value.

These observations are further supported by box and whisker plots for the worst-case delay in the four cells, which are illustrated in Figure 5.24. The bottom and top of the box in these plots indicates the inter-quartile range for the delay data across the RandomSPICE results (the location of the 25^{th} and 75^{th} percentile). The whiskers indicate the extent of $1.5\times$IQR, with outliers beyond this range indicate with crosses on the plots. The plots clearly show the extent to which the extreme outliers in the distributions for the buffer, OR and XOR gates have been greatly reduced. This is confirmed in Table 5.9, which shows the percentage improvement in the range of the worst-case delays from the reference designs to the optimised designs. As suggested from the earlier plots, the NAND gate offers no improvement in IQR and only a modest improvement in the overall range, however the results suggest a significant improvement in tolerance in the buffer and especially the OR gate designs, which is close to doubling the reference design in terms of delay-variability tolerance.

5.6.3 Observations

This final set of optimisation experiments display clear improvements in both the operational characteristics of propagation delay and power consumption, in addition to a significantly im-

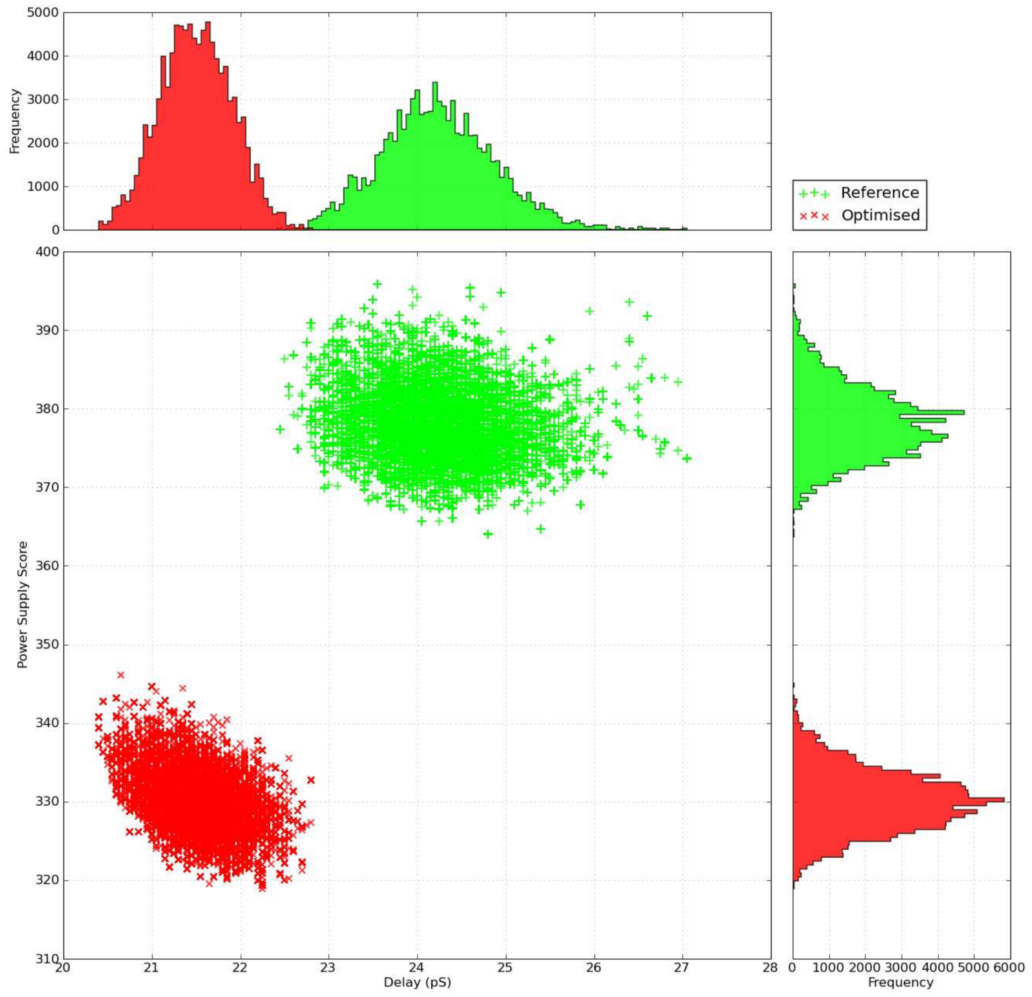


Figure 5.20: Detailed analysis of variability in the reference and optimised buffer designs, plotting the output from 100,000 RandomSPICE simulations. The histograms show the distributions for the worst-case delay (x-axis) and the power score (y-axis).

Table 5.9: The relative improvement observed between the optimised design and the reference design for the inter-quartile range (IQR) and overall range of the worst-case delay scores across 100,000 variability circuits.

Statistic	Buffer	NAND Gate	OR Gate	XOR Gate
IQR (%)	54.5	0.0	87.5	10.0
Range (%)	91.7	10.8	96.1	32.1

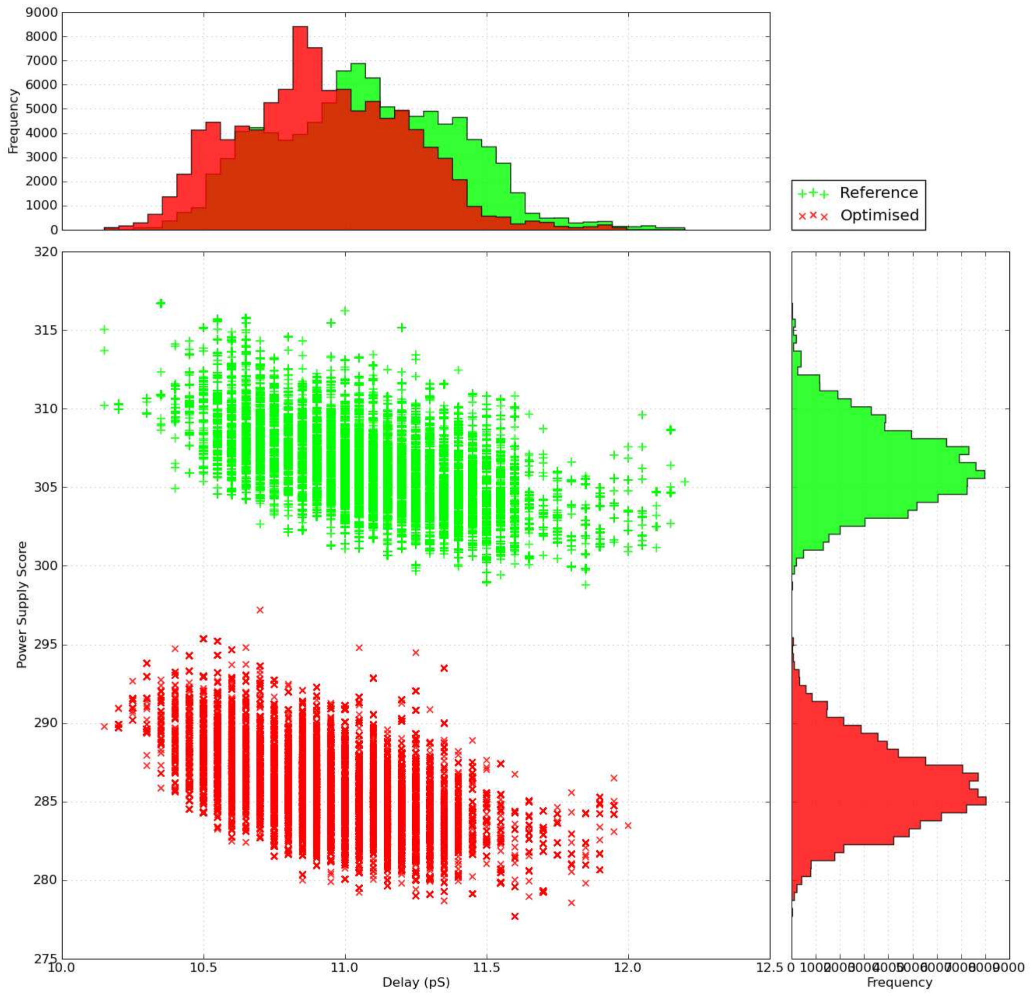


Figure 5.21: Detailed analysis of variability in the reference and optimised NAND gate designs, plotting the output from 100,000 RandomSPICE simulations. The histograms show the distributions for the worst-case delay (x-axis) and the power score (y-axis).

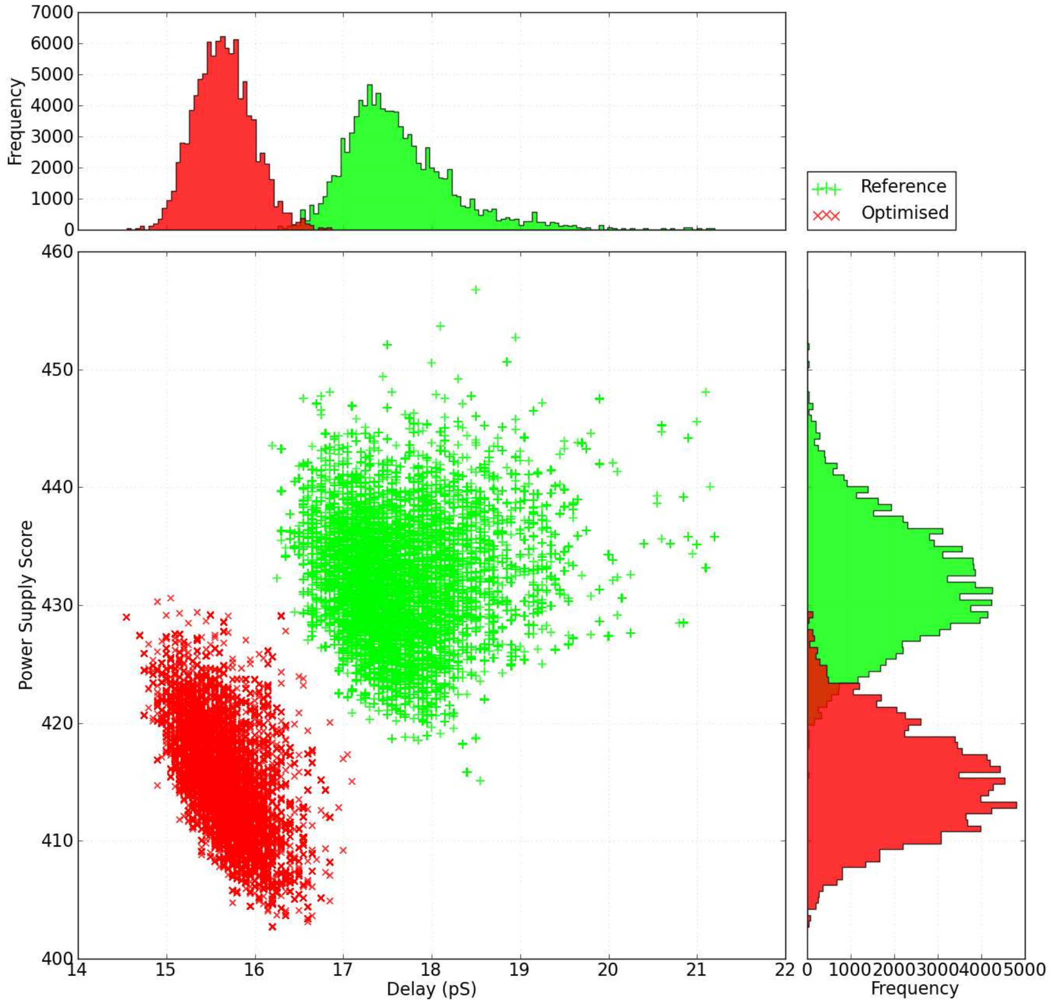


Figure 5.22: Detailed analysis of variability in the reference and optimised OR gate designs, plotting the output from 100,000 RandomSPICE simulations. The histograms show the distributions for the worst-case delay (x-axis) and the power score (y-axis).

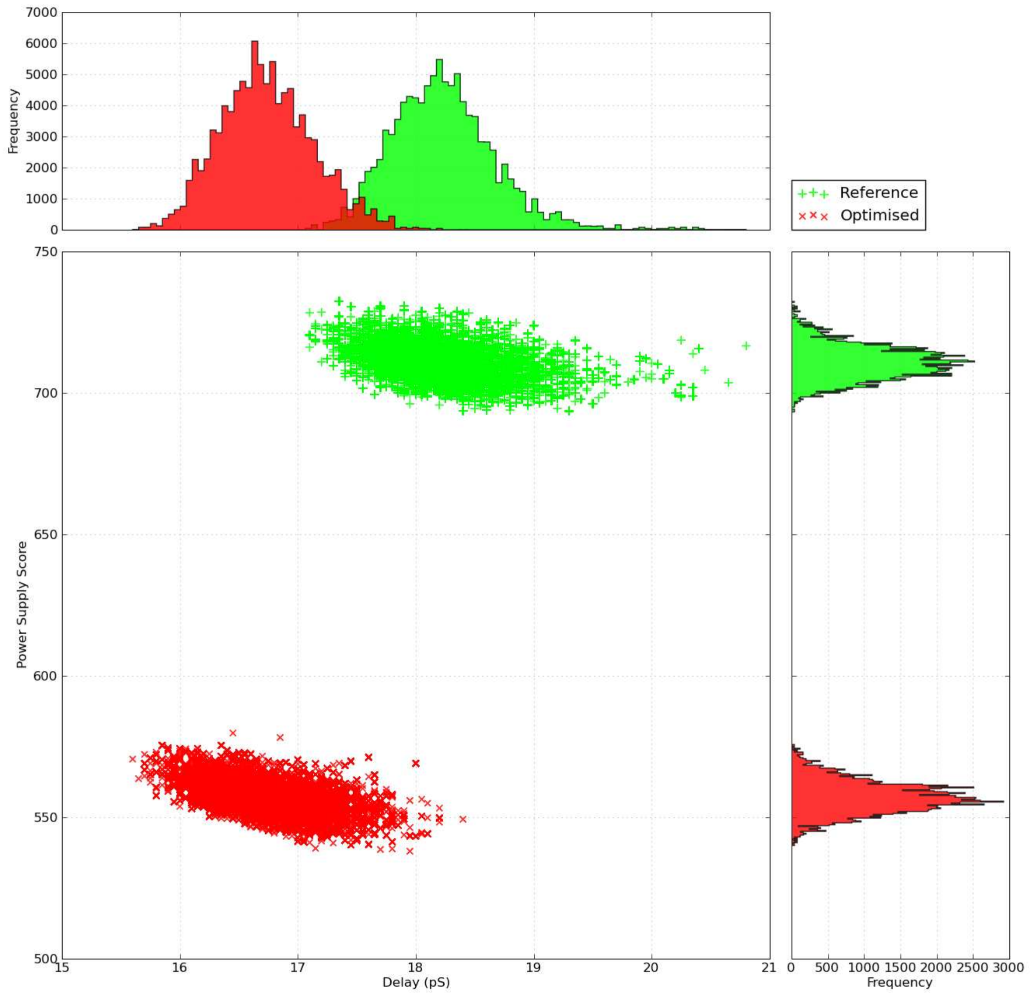
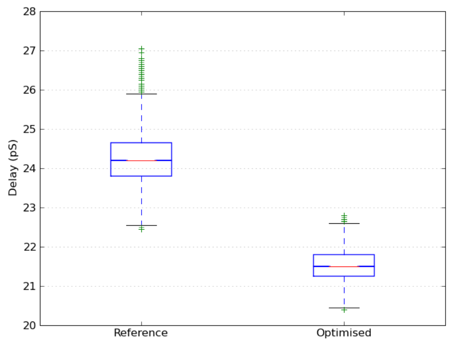
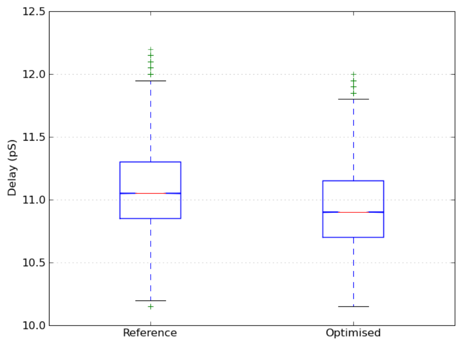


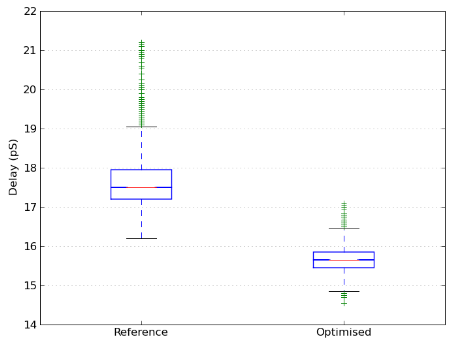
Figure 5.23: Detailed analysis of variability in the reference and optimised XOR gate designs, plotting the output from 100,000 RandomSPICE simulations. The histograms show the distributions for the worst-case delay (x-axis) and the power score (y-axis).



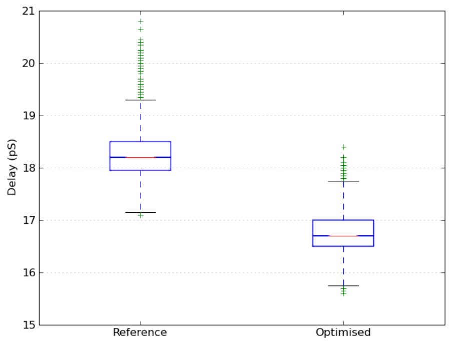
(a) Buffer



(b) NAND Gate



(c) OR Gate



(d) XOR Gate

Figure 5.24: Comparison of the mean worst-case delays in the reference and optimised cells for the buffer, NAND, OR and XOR gates. In the box-whisker plots, the line in the center of the box denotes the median value, with the upper and lower edges of the box denoting the inter-quartile range (IQR), the middle 50% of the distribution. The whiskers denote the extend of 1.5×IQR, with the crosses outside this range denoting the outliers in the distribution.

proved variability-tolerance, over the reference designs. As the reference designs are based on a cell library which is close in transistor length to that of the library used, this demonstrates the potential power of the MOTIVATED system in the field of transistor dimension optimisation. With variability-tolerance being the primary goal of this set of experiments, the fact that the extracted circuits demonstrate improvements in both delay and power over the reference designs in addition to improved variability-tolerance is very promising.

In this chapter, the experiments into optimising transistor dimensions within standard-cell libraries have been discussed. Initially the SGA was tested to evolve circuit topologies. From the results of these experiments the MO-system was introduced, with the algorithm used to optimise widths and independant voltage-sources within the VSCLIB SCL. Following these results, the HPC-based system was introduced, allowing more detailed optimisation with variability considered from the start of the process. This allowed the detailed optimisation of cells selected from the NANGATE SCL, with promising results. In the following chapter, a system which enables all stages of the process of generating variability-tolerant logic circuits is proposed. To evolve the topologies of the larger circuits at a cell-level, an adaptation to the existing CGP algorithm is discussed, in which the multi-objective algorithm used within these experiments is employed to improve the extracted designs. Two of the extracted designs are then assessed for variability tolerance and compared to conventional designs.

Chapter 6

Optimising Larger Circuits

In this chapter, an approach is introduced which proposes a system to combine different evolutionary computation techniques, with the aim of creating larger variability tolerant logic circuits from scratch. This involves the use of three different evolutionary algorithms: the simple genetic algorithm discussed previously for optimising both standard cells, a CGP-derived techniques for evolving the topology of standard cells, and a different adapted version of CGP for evolving logic circuits at a block-level. The block-level designs are then optimised again using the simple genetic algorithm to select the most appropriate standard cells from the optimised libraries. The basic steps of the work-flow are outlines in Figure 6.1. With this approach, it is intended to demonstrate that evolutionary techniques can be combined into a work flow with the aim of sidestepping the need for human design in the creation of logic circuits.

The motivations for these experiments are two-fold. Firstly, there was a desire within the nano-CMOS project to investigate larger circuits than standard-cells, however the tools necessary to investigate much larger circuits with variability using VHDL or a similar language were not going to be available within the time-frame of the research, thus circuit simulations would be limited to the SPICE-based analysis, and with it a moderate limit on the maximum size of circuits that could realistically be simulated. Secondly, it was realised that through the earlier research, methods to evolve topologies of standard-cells and then optimise those cells for variability-tolerance were in place. It was known from previous research by Dr. James Walker *et al* in CGP that the algorithm could be used to generate novel logic topologies for circuits such as adders and multipliers based on the combination of boolean-logic functions provided by the standard cells. It was hypothesised that combining the existing CGP system with the NSGA-II multi-objective strategy used by MOTIVATED would result in circuits with improved characteristics in terms of the count of gates, depth of gates and count of transistors over those evolved with conventional CGP. Additionally, the ability of the SGA to selectively

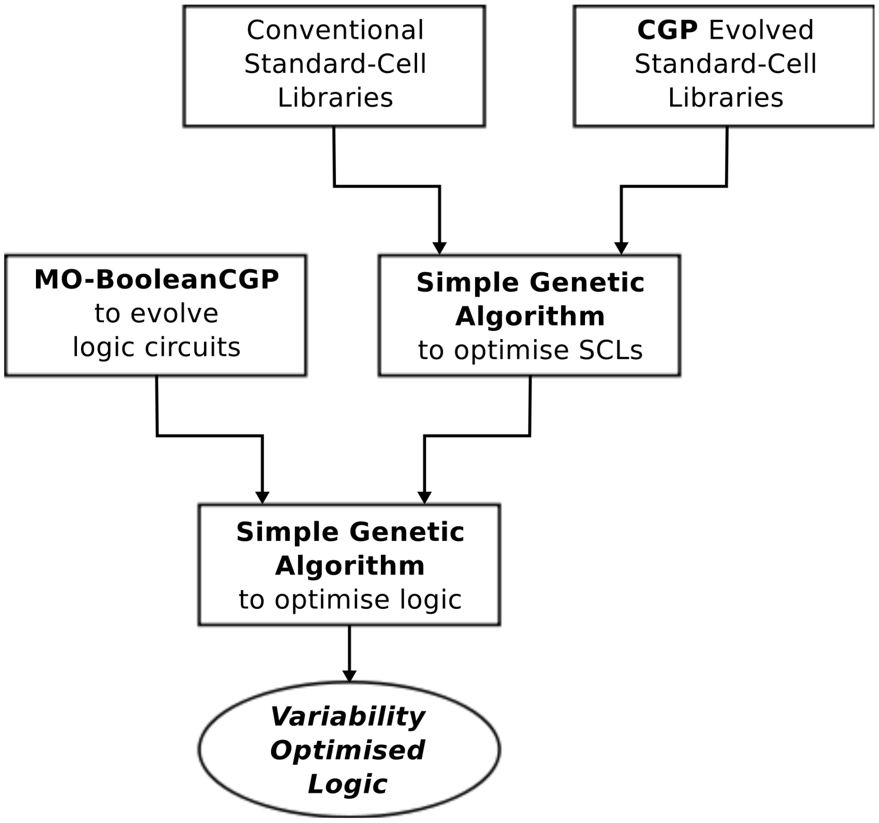

Conventional Standard-Cell Libraries
CGP Evolved Standard-Cell Libraries
MO-BooleanCGP to evolve logic circuits
Simple Genetic Algorithm to optimise SCLs
Simple Genetic Algorithm to optimise logic
Variability Optimised Logic

Figure 6.1: Flow diagram of the tools used to create evolved, optimised logic circuits. If the conventional SCLs are not included in the workflow, a completely evolved path exists in which no human-designed topologies are needed.

enable and disable functional blocks based on entries within the genotype, would allow the existing MOTIVATED system to further optimise the evolved logic functions by selecting specific cells from a library of optimised standard-cells.

6.1 Evolving Block-Level Topologies

The CGP algorithm used to create block-level topologies builds on previous work by Miller and Walker, adding a multi-objective fitness function to improve the performance and efficiency of the extracted circuits. As this form of CGP evolves logic circuits at a Boolean-logic level, the algortihm will herein be named MO-Boolean-CGP, in order to differentiate it from the previously described algorithm for creating transistor-level topologies through an adapted CGP.

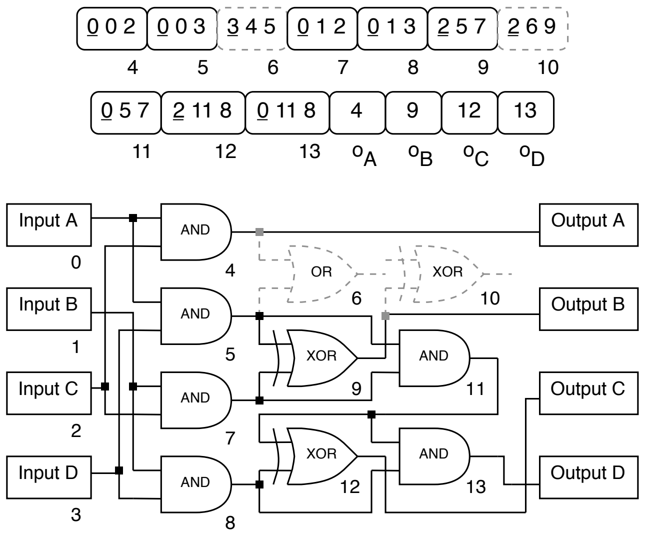


Figure 6.2: Example of a CGP genotype and its mapping to a 2-bit multiplier circuit [165]

6.1.1 Conventional CGP To Evolve Logic Circuits

As originally described by Miller and Walker, a typical CGP genotype, and its corresponding decoded phenotype and logic circuit representation are illustrated in Figure 6.2. In the example illustrated, the genotype decodes to represent a functionally-correct 2-bit multiplier circuit. The first element of each node represents the nodes functionality *(e.g. 0 = a logical AND gate)*, where the remaining two elements of each node represent the inputs to the logic block; the first n values representing the circuit inputs, and the other values representing existing decoded logic blocks. The final k elements of the genotype (4 in the given example) represent the circuit output connections, each having a single input. The genotype$\rightarrow$phenotype mapping does not mandate that all nodes are connected to each other, resulting in a bounded variable-length phenotype, with genes that are entirely inactive, and have a neutral effect on the resulting fitness. This neutrality is one of the important characteristics of CGP, and has been shown to be very beneficial to the evolutionary process in several research papers [182]. The complete genotype$\rightarrow$phenotype decoding procedure is covered in more detail in [165].

Whilst various mutation and recombination strategies are possible with CGP, MO-Boolean-CGP is entirely mutation-based without any form of crossover [44]. A point-mutation operator is used in which a given number of genes, from any position within the genotype, are changed to a different value; the new value is chosen at random but will fall within the valid range of values which depends on the gene's actual position within the genotype. For example, if the gene selected for mutation is one which determines the nodes functionality, the new value will within the range of $0 : 6$, as the function set comprises the set of six standard 2-input logic gates plus NOT function. However if the gene selected for mutation is an input gene, the new value will fall within a range covering the program inputs, and the output label of any previous node.

6.1.2 Use of Multi-Objective Fitness

Whilst the above standard CGP strategy has successfully been used to automatically create number of functionally correct logic circuits before, including 4-bit adders, 4-bit multipliers and 8-bit parity circuits, the algorithm has generally terminated once a functionally correct circuit is found, thus the evolved circuits are often far larger in terms of gates and path lengths than optimal designs. To attempt to create more efficient results, the algorithm has been adapted such that it operates in a two-stage process. In the first stage, the algorithm operates as above with fitness calculated solely in the terms of bit-errors based on the target truth-table. Once a functionally correct circuit is found, a second stage is initiated, in which all functionally correct circuits are ranked used a multi-objective strategy.

The multi-objective chosen is again based on the NSGA-II, allowing reuse of the code created within MOTIVATED. For each circuit evaluated the primary fitness measure is the circuit functionality; circuits which are not functionally correct are not evaluated further and not included in the fronts used by NSGA-II. For circuits which pass the functionality test, four fitness scores are calculated which are equally weighted within the fitness calculation. The first of these objectives is the number of logic gates used; minimising the number of gates will generally result in the most compact circuit schematics. In other research to optimise designs based on CGP and other evolutionary algorithms, number of gates has been fitness criteria to which circuits are optimised. Whilst having a minimum number of gates is a clear goal in terms of creating compact, space-efficient schematics, and is an interesting optimisation target, there are a number of reasons why it is not necessarily the best target for an optimised circuit. It must be remembered that not all gates are equal; exclusive-gates (XOR and XNOR) contain more transistors than AND and OR gates, which themselves contain more transistors than NAND and NOR gates. The number of transistors impacts on the die-area, power consumption and maximum operating speed in a fabricated design. Logic effort theory reveals how NAND gates themselves are preferable to NOR gates in terms of delay, due to the lower input-capacitances [156]. For these reasons we look at not just the number of logic gates, but also the number of transistors, and also the lengths of the paths between input and output.

The secondary objective is the number of transistors used, based on the logic-gate arrangements found within the VSCLIB open-source standard cell library. Each inverter is considered to be 2 transistors, each NAND and NOR function is 4 transistors, each AND and OR gate is 6 transistors and the XNOR and XOR gates are 9 transistors each. Whilst there will be a degree of correlation between gate-count and transistor-count, there are important differences which may affect which design is considered more useful; a minimised number of gates will generally result in a more compact gate-level schematic and thus might be preferable if the design is to appear in print, however the minimised number of transistors will result in a more-compact fabricated design and thus may be more efficient in actual fabrication. Whilst this gate-count may appear to be a relatively redundant objective, there are a significant number of examples of papers within the EC field which assess performance of circuits solely on gate-count. The reader is referred to [164] and [115] for examples of this practice.

The third objective is the longest gate-level path length between input and output. For this objective, all gates are equally weighted, with the resultant scores being the sum of gates in the longest path between an output and any of its inputs. This objective effectively attempts to minimise the width of the printed schematic, assuming gates are spaced in uniform rows between input and output. The final objective is the longest approximated delay between input and output based on the switching delays within the cell library; the approximated score is based on the count of transistor gates that are traversed in the longest path between input and output. It is acknowledged that a more accurate score could be calculated from the logic-effort formulae, however this would significantly add to processing time for each evaluation. As with the previous objectives, there will be a degree of correlation between these two objectives, however minimising the former will produce the most compact-width circuits when drawn as a gate-level schematic whilst the latter will minimise the worst-case switching delay in an actual fabricated circuit.

Most previous published work based on the CGP algorithm for evolving digital circuits has made use of a $1 + \lambda$ selection process, and as such has generally used a very small population size; for effective use of a multiple-objective fitness function it is necessary to use a significantly larger population, in order to arrange resultant circuits into Pareto-fronts and extract a range of circuits from the primary front [53]. For this reason, the population size was set to 100.

6.1.3 Test Circuits

Five different combinatorial logic target circuits have been evolved, as shown in Table 6.1. The first two circuits tested are a 2-bit full-adder and multiplier; such circuits are fundamental building blocks for larger scale logic designs in arithmetic logic units, and have been widely re-

Table 6.1: The 5 test circuits evolved with MO-Boolean-CGP

	Test Circuit	Inputs	Outputs
A	2-Bit Full-Adder	5	3
B	2-Bit Multiplier	4	4
C	3-Bit Full-Adder	7	4
D	3-Bit Multiplier	6	6
E	Hex to 7-Segment Display Driver	4	7

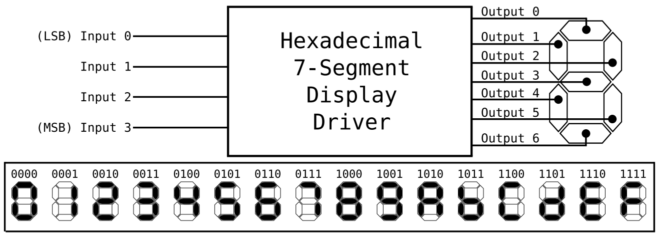


Figure 6.3: Hexadecimal Seven-Segment Display Driver layout and outputs

searched in both conventional digital design and in the biologically-inspired and non-standard computation fields.

To evaluate algorithm performance on more complex circuits 3-bit full-adders and multipliers are also investigated; conventional designs for circuits will use tens of logic gates thus they represent a greater challenge for the algorithm to find solutions for but also provide greater scope for design-improvement in the second-stage of the algorithm where working designs are improved.

The final test design is a hexadecimal 7-segment display driver, as illustrated in Figure 6.3. This circuit was chosen as it has a greater number of inputs and outputs than the previous circuits with little correlation or pattern between the outputs; as such it represents a challenge for the MO optimisation stage to see if any parallelism can be extracted from gates, which can be shared between the different output paths. To evaluate the circuits the truth-table for the required function is encoded into an input and output file which can be directly read by the CGP algorithm.

6.1.4 Methodology

The MO-Boolean-CGP algorithm was set to evolve the test-circuits, allowing solutions to be built using a complete set of two-input logic gates (NAND, NOR, AND, OR, XOR and

Table 6.2: The evolution parameters use by the MO-Boolean-CGP Algorithm in the different test-cases

Parameter	2-bit Adder & Multiplier	Other Circuits
Number of runs	20	20
Population size	100	100
Number of nodes	3	3
Genotype length (genes)	300	300
Mutation rate (%)	2	2
End Generation	20,000,000	50,000,000

XNOR) alongside a single-input inverter function. For each of the test-cases, twenty runs were executed for a predetermined number of generations as described in Table 6.2, with the 5 best individuals at the end of each run saved for analysis. The end generation was set artificially high to ensure that the algorithm had entered a phase of stagnation; the value was determined on shorter runs conducted previously. Also recorded was the generation and fitness scores of the first circuit in each run that was functionally correct, as this result is where conventional CGP run would terminate. The mutation rates and genotype length were chosen based on the optimal values from previous work by Walker using conventional CGP.

6.1.5 Evolved Results

The results for the first functionally correct circuit in each run are shown in Table 6.3. These results illustrate the outputs that could be expected for a standard-CGP algorithm, without the multi-objective optimisation stage, whereby once a functionally correct circuit is found the program terminates. The table shows both the best extracted result alongside the averaged results for all the runs. It is clear from the figures that whilst the CGP algorithm is effective at finding functionally correct results, there is little optimisation and some of the results are large in terms of transistor count and circuit area; every run found a functionally correct circuit well in advance of the predefined end-generation.

Table 6.4 contains both the best observed, and average scores, for the test-circuits at the termination of the algorithm. Each algorithm was left running for more generations after a solution was found to observe progress. Naturally a significant improvement in all scores was noticed by the end of each run, with certain populations reaching an optima and stagnating well before the end-generation. With the smaller circuits the same set of results appeared in most of the runs; the larger circuits had more varied runs with more diverse populations towards

Table 6.3: Observed results for the MO-Boolean-CGP experiments before Multi-Objective Optimisation

		2-Bit Adder	2-Bit Multiplier	3-Bit Adder	3-Bit Multiplier	Hex to 7-Seg Display Driver
	Generation	317	111	3,576	4,974	10,257
	Gate Count	16	13	28	41	41
Best	Transistor Count	92	64	192	225	223
	Longest Gate-Path	5	4	7	7	8
	Longest Transistor-Path	16	10	20	20	18
	Generation	3,942	282	20,019	292,195	39,070
	Gate Count	23.2	16.8	38	57.9	67.5
Mean	Transistor Count	134	78	223	326	384
	Longest Gate-Path	7.3	5.5	8	9.7	11.1
	Longest Transistor-Path	21.6	12.4	25.2	28.2	32.7

the end of the run. To generate the schematics for the extracted circuits that are shown in the following figures, an evolutionary algorithm was used which directly converts the output from the MO-Boolean-CGP program to a scalable vector graphics. This algorithm is described in Appendix A.1.

2-Bit Adder

After the MO optimisation, all the runs for the 2-bit Full-Adder discover circuits which contained 10 gates, 60 transistors and a gate-path length of 5. In addition, 3 of the runs discover a circuit utilising 13 gates and 63 transistors but with a shorter gate-path length of 4, as is shown in Figure 6.4. Both these circuits better those discovered using conventional CGP alone, although working solutions are discovered quickly by the standard algorithm and given the relatively simple function it is possible that running the conventional algorithm several times may discover the same results.

6.1.6 2-Bit Multiplier

In the case of the 2-bit multiplier, all five runs found an identical optimised solution, shown in Figure 6.5. This solution was found after 698,000 generations in the slowest run and after an average of 240,000 generations over all the runs; the solution has 7 gates and 35 transistors with a longest-gate path of 2 and a longest-transistor path of 6; no improvements in any of the objectives were found over the course of the algorithm. As this solution was not bettered in

Table 6.4: Results for the MO-Boolean-CGP experiments after Multi-Objective Optimisation

		2-Bit Adder	2-Bit Multiplier	3-Bit Adder	3-Bit Multiplier	Hex to 7-Seg Display Driver
	Gate Count	10	7	15	28	22
Best	Transistor Count	60	35	90	148	125
	Longest Gate-Path	4	2	5	5	4
	Longest Transistor-Path	11	6	12	13	11
	Gate Count	10	7	15.6	31.64	26.95
Mean	Transistor Count	60	35	94.6	177.3	146.5
	Longest Gate-Path	4.4	2	5.6	5.86	5.05
	Longest Transistor-Path	11	6	13.6	15.82	13.85

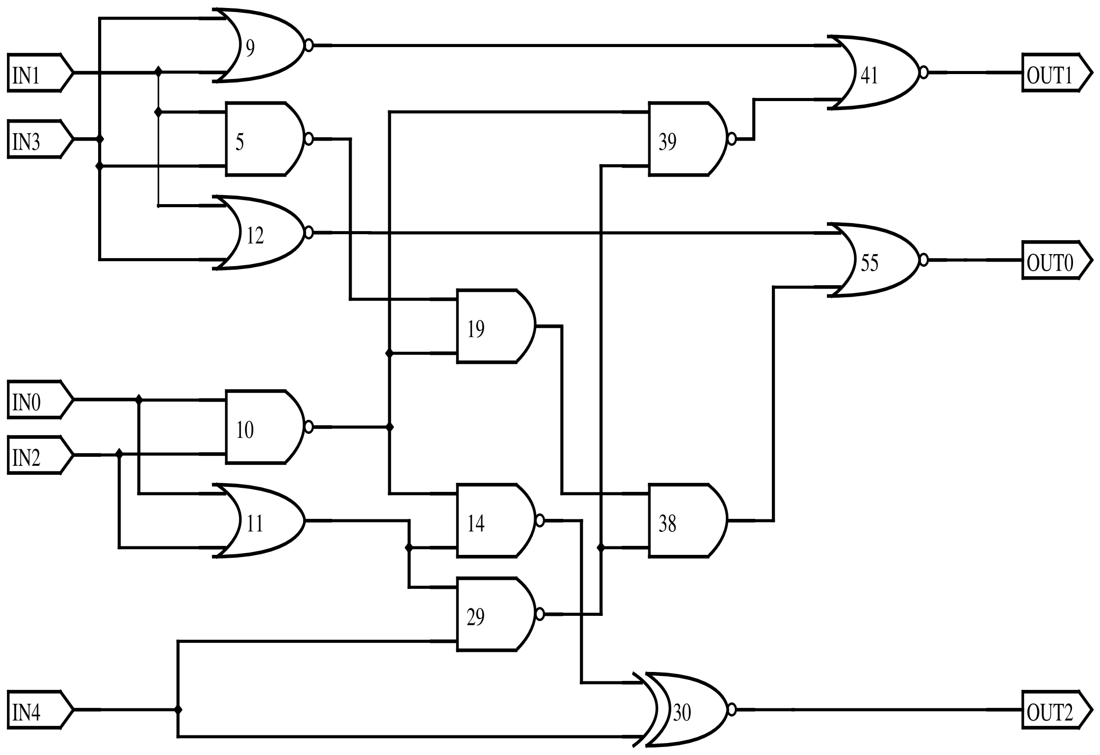


Figure 6.4: Evolved 2-Bit Adder design with minimum gate-path length

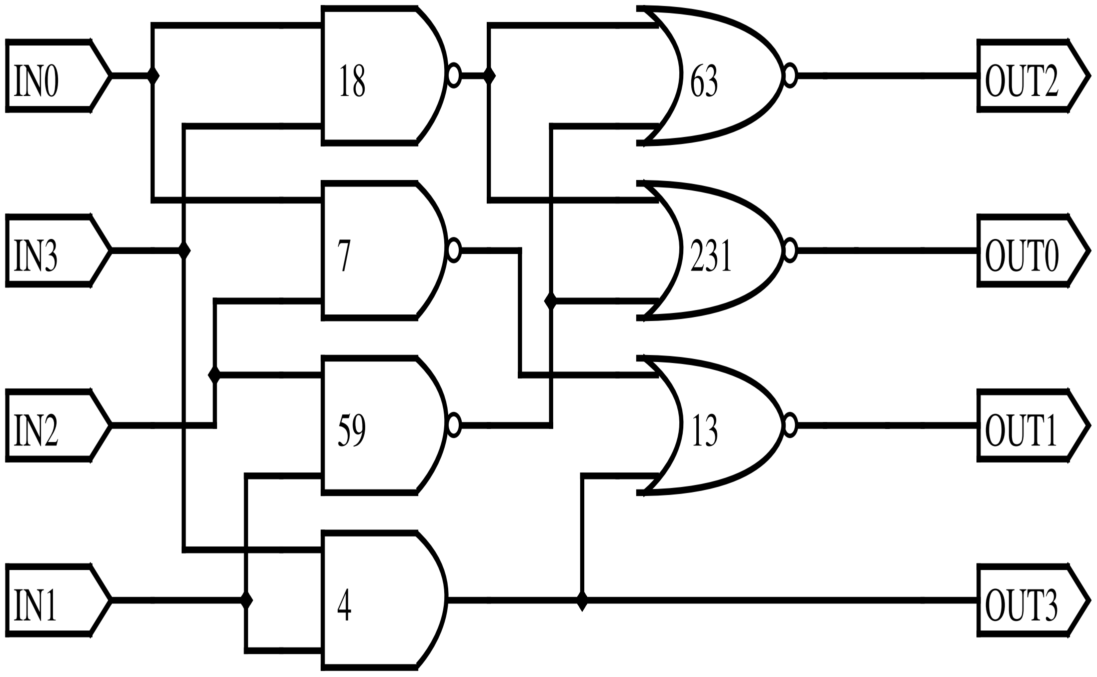


Figure 6.5: Evolved 2-Bit Multiplier design with minimum scores in all objectives

any of the objectives, the promoted population in each generation rapidly became filled with this solution.

6.1.7 3-Bit Adder

Most of the runs for the 3-bit adder circuit discovered circuits which utilised 15 gates and a gate-path depth of 5. One run also discovered the circuit illustrated in Figure 6.6, which utilised 18 gates and has a deepest gate-path depth of 6, but offers a lower longest transistor-path length score of 12. The observed improvement during the MO-stage is steady and regular until the circuits with less than 18 gates are discovered, at which point progress typically stagnates.

6.1.8 3-Bit Multiplier

The most compact 3-bit multiplier extracted after the MO optimisation utilised 28 gates, as illustrated in Figure 6.7. Another optimised design had a longest gate-path length of 5 gates, but used a total of 31 gates. The results show promise when compared with results from other researchers. A better result in terms of gate-count for this circuit has been extracted in

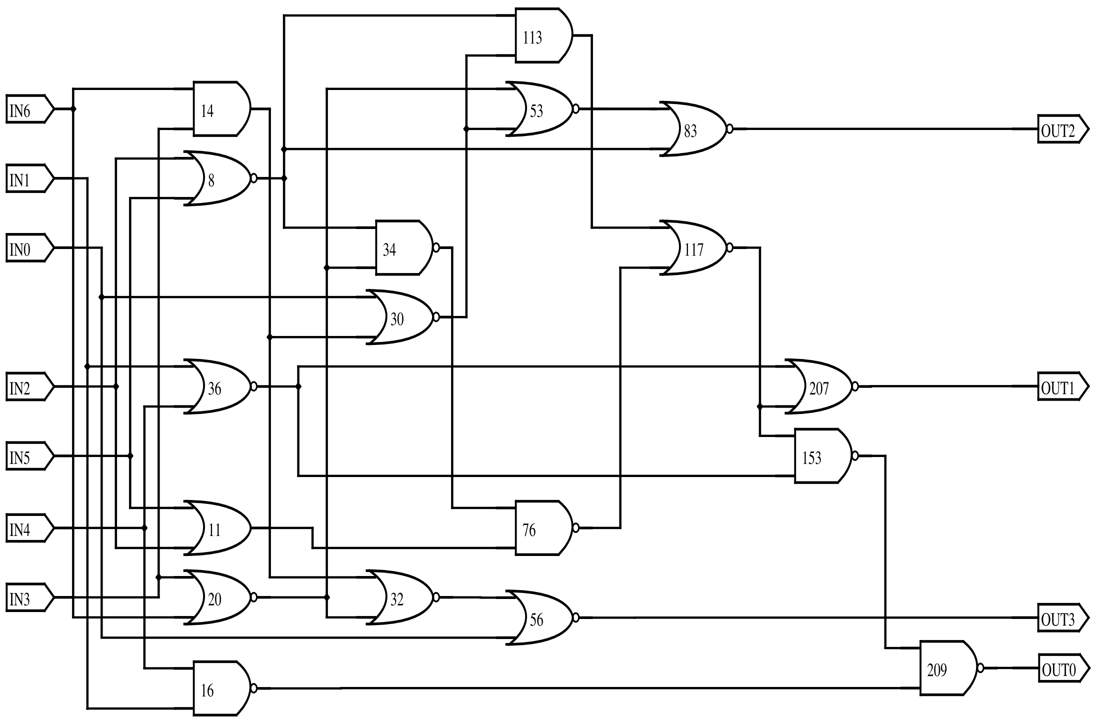


Figure 6.6: Evolved 3-Bit Adder design with minimum transistor-path length

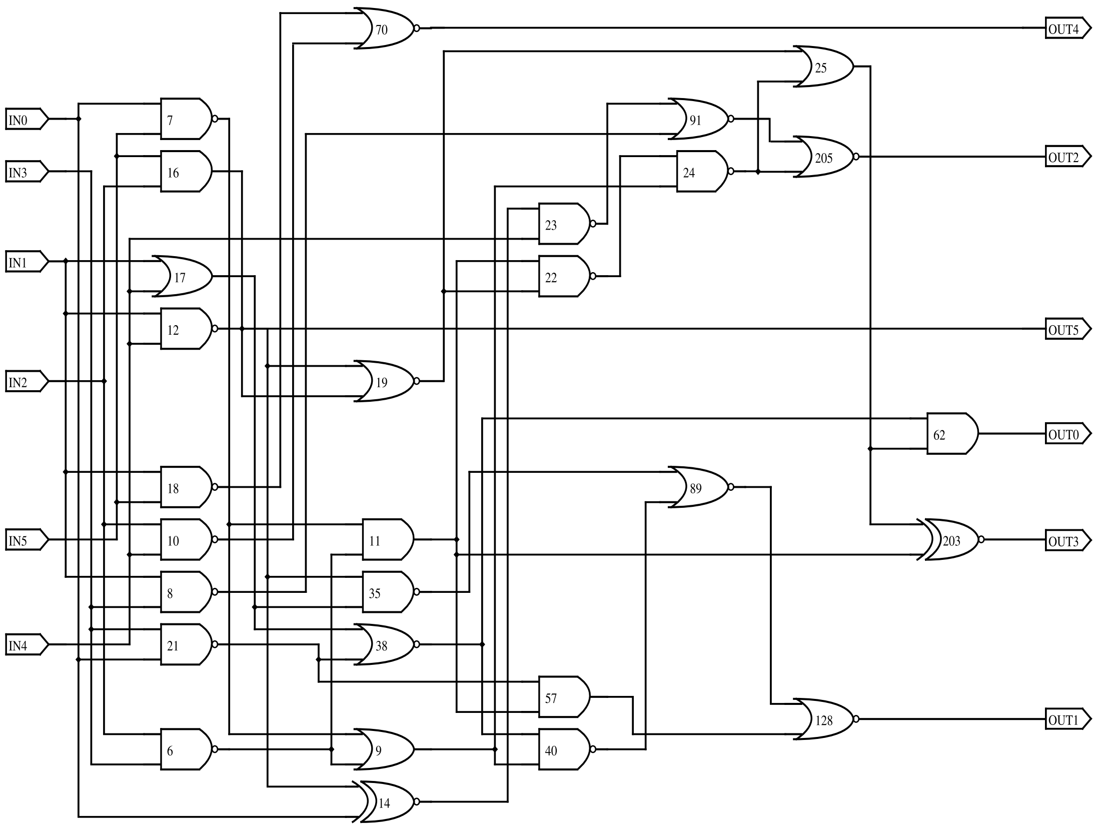


Figure 6.7: Evolved 3-Bit Multiplier design with minimum gate count

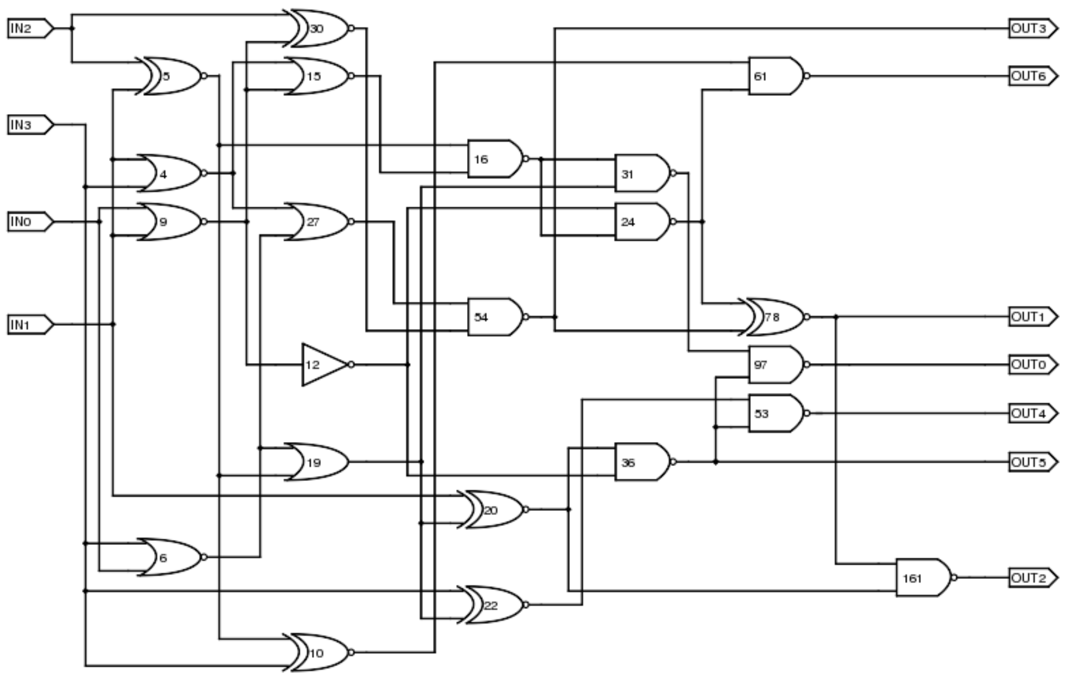


Figure 6.8: Evolved 7-Segment Display Driver with minimum gate count

work by Wang and Lee [173], however their evolved design featured a wider propagation-gate delay with a path length of 6 gates. The Wang and Lee method utilised an adapted version of CGP incorporating a Self-Adaptive Mutation Rate Control (SAMRC) to enhance algorithm performance. Wang and Lee observed a reduction in average gate-count from 52.97 down to 31.70 with the addition of a MO optimisation phase; similar results are observed here with a reduction from 57.90 to 31.64 average gates pre- and post- MO optimisation. It is noted that the results from Vassilev and Miller which produce a 3-bit multiplier from only 23 gates make use of non-standard gates such as one-inverted-input AND and use a much smaller library of gates for optimisation; however the produced circuit has a longest gate path length of 8 gates plus an additional two inverted input [164]. The most efficient human-designed topologies for this circuit use 30 gates [115].

6.1.9 7-Segment Display Driver

Of the larger circuits under test, the most consistent improvement through multi-objective optimisation was seen with the seven-segment hexadecimal display driver. Across the twenty runs, the initial circuit to matched the target functionality contained on average 67.5 gates and an average longest gate-path length of 11.1; the best observed circuits had a gate count of 41 and a gate-path length of 8. By the end of the optimisation phase, each run had individuals within the population with gate-counts under 30 and gate-path lengths of 6 or less; the most gate-efficient circuit had only 22 gates and is shown in Figure 6.8; another circuit utilised 26 gates but managed a gate-path length of just 4. The best solutions exploit shared gates between most of the output paths that would be hard to identify using traditional logic-design methods such as Karnaugh mapping and suggests that MO-CGP could be very beneficial in combinatorial-logic problems with several distinct outputs.

6.1.10 Observations on Results

The goal of the research on MO-Boolean-CGP was not to create an algorithm that is more computationally efficient than conventional CGP (or for that matter any other logic-building algorithm), but to demonstrate that the use of an MO could effectively improve the characteristics of a circuit, albeit at the cost of increased computational resources. In all of the runs the algorithm was run for a significant length of time, with 2 billion circuit-evaluations in each of the 2-input logic runs and 5 billion in all of the other runs, a significant computational demand and well beyond the point where stagnation had appeared in the populations. The basic CGP algorithm itself found working solutions for the problems in a much shorter time: for the 3-bit multiplier - generally the slowest of the test-circuits to find a solution - a solution was, on average, found after 30 million circuit-evaluations. In the worst-performing example, a solution was not found until 169 million circuit-evaluations had been performed. However, a key observation that holds true across all of the runs is that the post-MO result betters any of the pre-MO results in every objective.

For the smaller circuits, the evolved designs have been identical replicas of human-created designs. For the larger circuits, some of the designs have bettered published human-created designs in certain criteria, and matched or bettered the performance of designs created by other EC based algorithms. Previous work on CGP has identified that whilst circuits can effectively be created for smaller digital building block circuits, once circuits are scaled beyond a certain size it becomes exponentially difficult to evolve designs; whilst 3-bit and 4-bit multipliers and adders have been created, it is unlikely that a 16-bit or 32-bit multiplier or adder could feasibly be evolved in a practical time without the use of modular techniques such as ADFs; as the MO extension to CGP is reliant on the standard algorithm creating functional circuits before the MO section begins, it does nothing to reduce the difficulty in evolving such designs. The MO extension does, however, appear to effectively enhance and optimise populations in which functional designs have already been realised.

6.2 Variability in Evolved Designs

In this section, two of the evolved designs discussed in the previous section are examined at a finer detail using the SPICE simulation software, in a set of experiments which were carried out by Dr. James Walker. They evolved designs are compared to established conventional designs for the same circuits, are evaluated for both traditional metrics of power-requirements and propagation delay. In order to assess the circuit performance within SPICE, it is necessary to convert the standard-cell description of the circuits into a transistor based netlist. For this, the standard-cells from the reference VSCLIB library, described in Section 5.2, were used. In the process of creating the test netlists, each standard-cell within all of the test circuits is directly replaced with the transistor-level VSCLIB netlist for the cell as a subcircuit.

6.2.1 Methodology

Once the base SPICE netlists containing all the required standard-cells has been created, an input, supply and load stage are added to the netlist, as illustrated in Figure 6.9 [1]. The chosen arrangement allows the voltage and current that is found at the inputs, supply and outputs to be measured, with the load capacitors providing a suitable output load that ensure that realistic output scores are calculated. For these tests, the designs a given piece-wise linear sources, which approximate a transistor response, with a predetermined rise/fall time. In this arrangement, one input is held at logic high for a clock cycle, then switch low for a clock cycle, and high again for a final clock cycle, with the other three inputs all held at logic high. This process is then repeated for the remaining inputs (three for the multiplier circuit, four for the adder). A transient analysis is then used with NGSPICE to observe the relevant voltages and currents, which takes place over a period of 12 clock-cycles for the 2-bit multiplier circuit and 15 clock-cycles for the 2-bit adder due to the extra input.

[1]The test-bench for the adder circuit has an additional input due to the *Carry In* input to the circuit

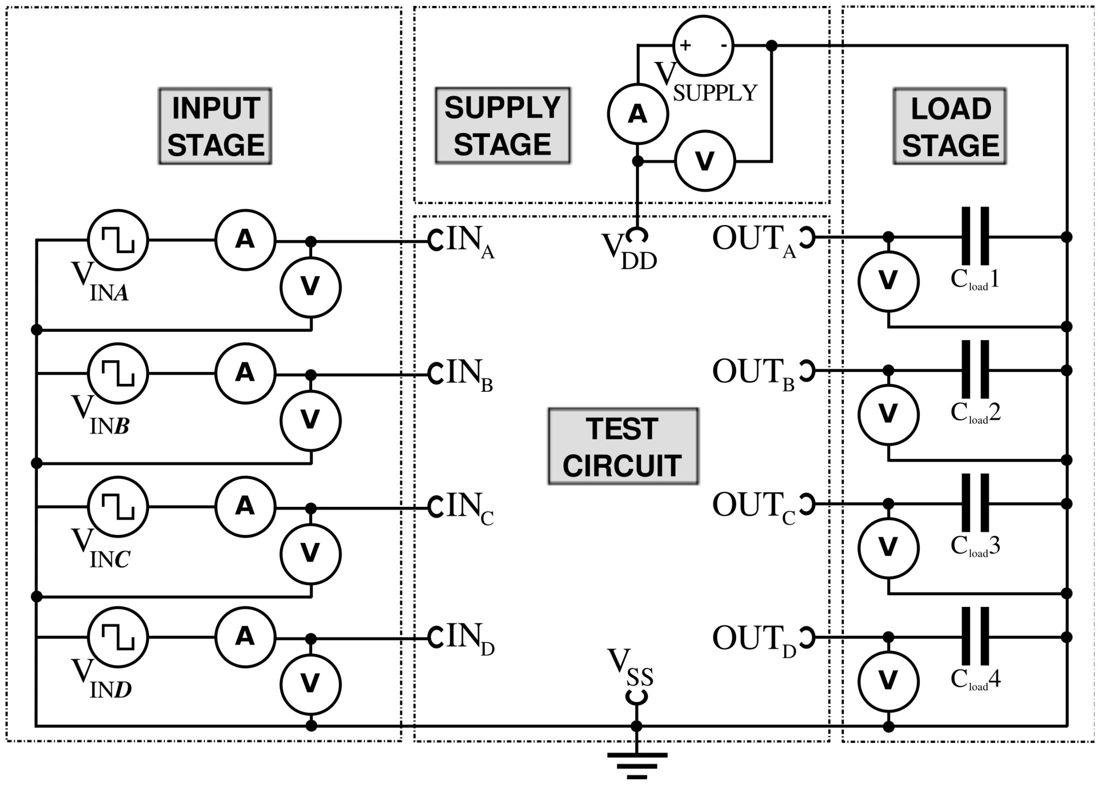


Figure 6.9: The Test Bench used to evaluate the 2-Bit Multiplier circuit in NGSPICE.

Delay and Power Consumption Calculations

These tests were carried out after the release of Version 20 of the NGSPICE software. One of the major updates that has been introduced in the newest releases of NGSPICE is the ability to use `.MEASURE` statements within the control loop of the analysis, which allow the calculations of propagation delay and dynamic power to be calculated directly by the NGSPICE software. This offers major advantages to the MOTIVATED system: it both massively reduces the amount of output data the needs to be processed and removes the need to write the routines necessary to perform the calculations, and as an additional benefit it ensures that the values calculated match those generated by the commercial HSpice software. As before, the propagation delay is the measure of the time taken from an input crossing the 50% V_{DD} threshold, to when the corresponding output signal crosses the same threshold. As before, the slowest overall propagation delay across all outputs and transistions is the value considered to be the most relevant, as it determines the maximum operating frequency at which the test circuit can reliably be used.

The dynamic power is defined as being the integral of the product of the supply voltage and current entering the test circuit, over the region in which the design is in a switching state. The switch region is defined as being the duration from which the input begins to switch (either rising or falling) up to the point where the slowest output has reached a stable state; as with delay, it is the worst-case dynamic power across all the outputs and switching states that is considered as the metric for evaluating power performance.

The above measurements are carried out with uniform models to extract set of non-variability scores for the power and delay in each test-circuit. In order to measure variability in the test circuits, a set of results using the Toshiba 35nm variability enhanced models through RandomSPICE is also carried out, with the same delay and power characteristics measured in the results. For each of the test circuits, 1000 RandomSPICE netlist are evaluated on the MAVEN cluster.

6.2.2 Results

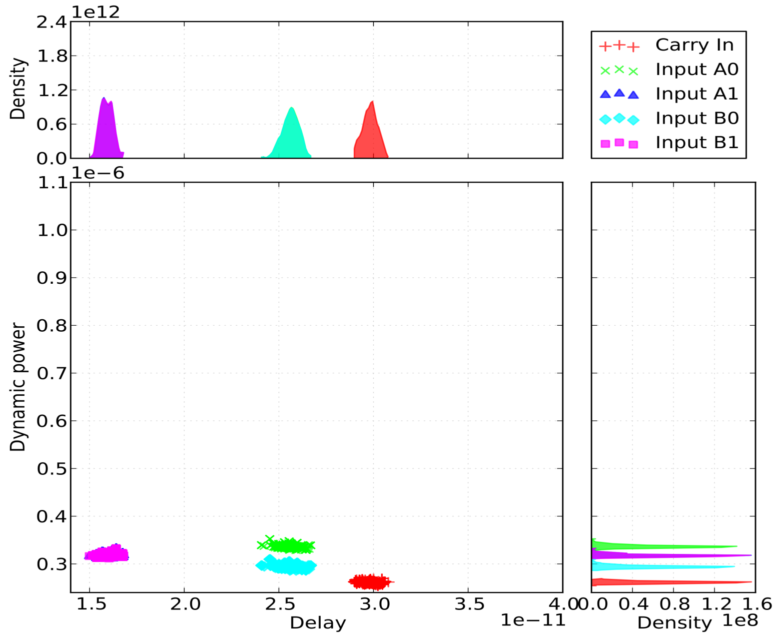
The variability results for the 2-Bit adder circuits are shown in Figure 6.10. In these plots, the measured propagation delay and dynamic power for each of the 1,000 RandomSPICE circuits is shown in a scatter plot. Each scatter plot contains the five different delays and powers that are observed when each of the circuit inputs are individually manipulated. This method of selectively manipulating each input clearly highlights which are the critical paths for timing and power in both the conventional and the evolved design. Similar comparative

Table 6.5: Comparison of the results for the conventional and MO-Boolean-CGP evolved 2-bit adder and 2-bit multiplier circuits, based on the CGP objective scores and the SPICE analysis

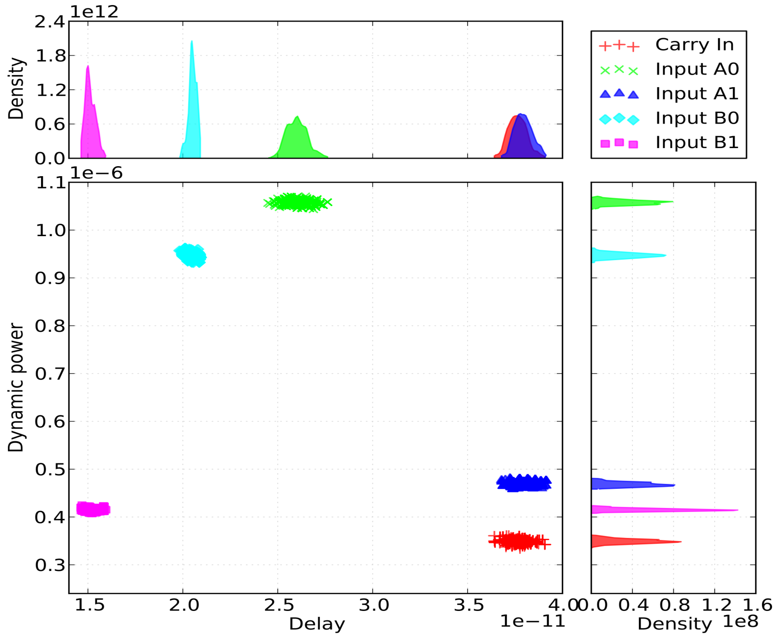
		2-bit Adder		2-bit Multiplier	
	Metric	Conventional	Evolved	Conventional	Evolved
CGP	Gate Count	10	10	8	7
	Transistor Count	64	60	54	35
	Longest Gate Path	4	4	3	2
	Longest Transistor Path	12	11	9	5
NGSPICE	Propagation Delay	$2.98e^{-11}$	$3.79e^{-11}$	$4.56e^{-11}$	$3.28e^{-11}$
	Dynamic Power	$3.37e^{-7}$	$1.05e^{-6}$	$3.73e^{-15}$	$4.39e^{-15}$

studies for the 2-Bit multiplier circuits are given in Figure 6.11. Table 6.5 compares the non-SPICE characteristics alongside the measured worse-case SPICE characteristics of all four test circuits.

In comparison between the evolved designs and the conventional designs, the results are mixed: for the evolved adder circuit, it is outperformed by the conventional design, particularly in power consumption with a 312% higher dynamic-power score observed, whilst operating 27% slower. In contrast, for the evolved multiplier circuit, an improvement of 28% in delay is observed over the conventional design and the expense of a 17% increase in power consumption. The improvement in the propagation delay for the evolved multiplier can be attributed to the reduced worst-case path length over the conventional design. However, both the conventional and evolved adder circuits are similar in path lengths, so it is perhaps surprising to see such an increase in delay in the evolved design, and the increase in power for both of the evolved designs is also surprising given the reductions in gate and transistor count. These observations highlight the limitations of the MO-Boolean-CGP algorithm which does not take into account power explicitly in the evolution objectives, and suggests that in future work the inclusion of a power metric may increase the performance of the algorithm; this however, would require that each circuit is analysed in SPICE or a similar circuit simulation package in the evolutionary process, significantly adding to the time taken to assess designs. When the variability-spreads for the circuits are considered, it is clear that the evolved 2-bit adder designs shows an increase in variability over the conventional design in both the spread of delay and power. However, the 2-bit multiplier exhibits a greater variability-tolerance over the conventional design in both power and delay statistics. The evolved 2-bit multiplier design has a reduction of 39% in the inter-quartile range, and a 27% reduction in the overall range, of the delay distribution across the 1,000 RandomSPICE circuits when compared to the conventional



(a) Conventional 2-Bit Adder



(b) Evolved 2-Bit Adder

Figure 6.10: Statistical intrinsic-variability analyses for the conventional and MO-CGP evolved 2-bit adder designs. The kernel density estimate plots above the main scatter plot shows the distribution in delay, with the plot to the right showing the distribution in power.

design. Similarly, a reduction of 25% IQR and 17% in overall range for the power distribution for the critical path is observed. Thus, whilst the evolved design consumes 17% more power than the conventional design, it more than compensates for this with the improvement in delay and variability-tolerance.

6.2.3 Observations

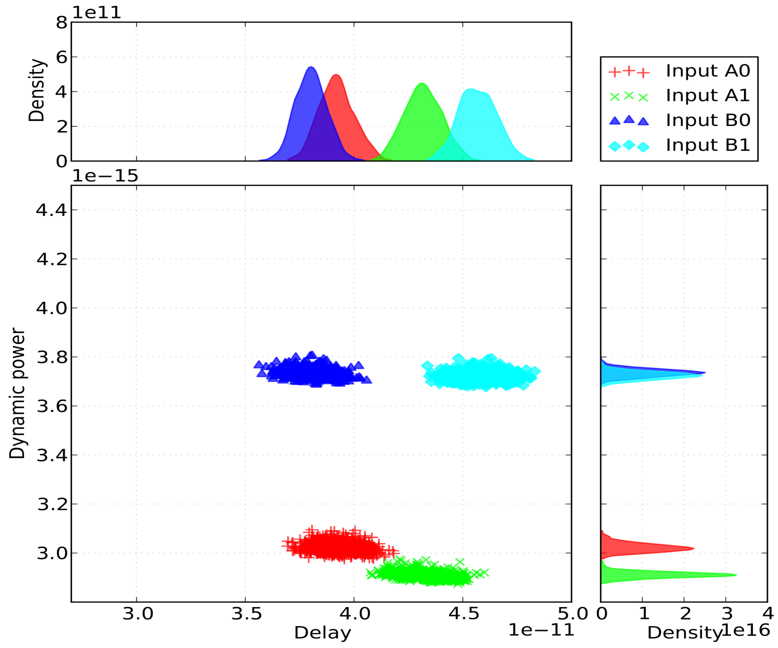
The results from the variability analysis demonstrate that the MO-Boolean-CGP system has the potential to evolve circuits which exhibit improved performance over conventional designs, both in the conventional metrics of propagation delay and dynamic power, and in terms of variability-tolerance, as seen in the 2-bit multiplier circuit. However, as is seen with the 2-bit adder circuit, this improvement is not guaranteed, even if the gate count and path length statistics suggest it is likely. One solution which would ensure that the circuits are demonstrating improvements during the evolution process would be to integrate the SPICE analysis into the evolutionary process, however this would drastically increase the computational resources and time needed to evaluate individuals.

6.3 Proposed System to Optimise Circuits at a Block Level

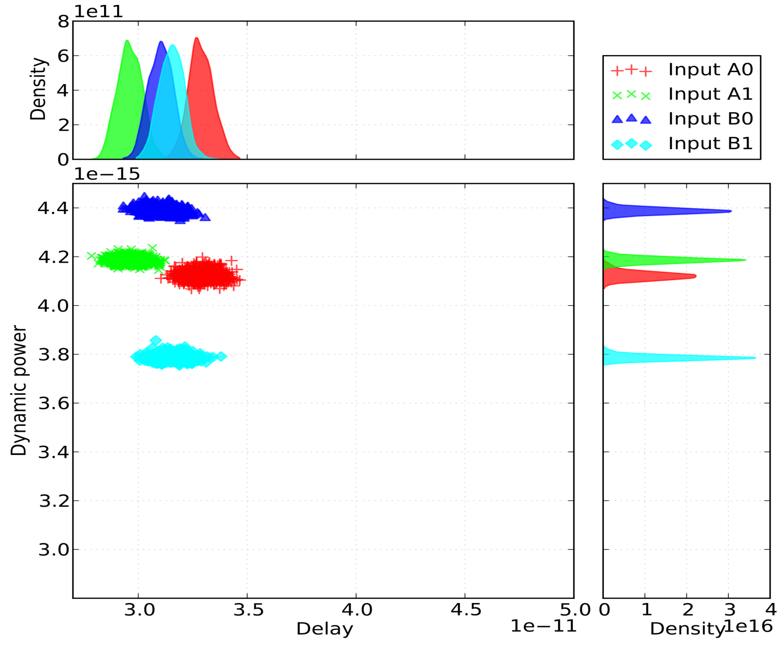
In this final section, as system to optimise evolved (or conventional) designs using an adaptation of the SGA system is briefly proposed. Whilst experiments to test the proposal have not been carried out, the aim of the proposal is to introduce a final stage in the evolution process by which CGP evolved boolean-topologies can be optimised using libraries of standard-cells, using the methods described in Chapter 5. Although the original goals of creating and evaluating a completely evolved logic-design process have not been realised, most of the integral steps necessary have already been created.

6.3.1 Methodology

The proposed methodology for evolving variability-tolerant is described in this section. This involves four main steps: the optional creation of novel standard-cell topologies, the creation of variability-tolerant standard cell libraries, the creation of larger logic topologies using the MO-Boolean-CGP system and finally the proposed optimisation of these circuits using an adapted form of the SGA system.



(a) Conventional 2-Bit Multiplier



(b) Evolved 2-Bit Multiplier

Figure 6.11: Statistical intrinsic-variability analyses for the conventional and MO-CGP evolved 2-bit multiplier designs. The kernel density estimate plots above the main scatter plot shows the distribution in delay, with the plot to the right showing the distribution in power.

6.3.2 Creation of Optimised Standard-Cell Libraries

The first steps in the process are to create SCLs which are optimised for both various traditional circuit characteristics alongside variability-tolerance. These structural topologies for the cells may be evolved using the CGP topology design system, or be based on conventional SCL designs. For a given set of variability models and cell-library, each cell would be optimised in turn, with a set of optimised results extracted from the final populations of the SGA system. As with most SCLs, it is likely that a high-performance (i.e. low-delay, high-power) circuit and a low-power circuit would be extracted, with possible additional balanced arrangements that fall between these on the power:delay front. However, in contrast to standard SCL optimisation methodologies, all extracted circuits would include variability-tolerance as objectives in the optimisation process, ensuring that for a given delay or power performance the observed variability would be minimised.

6.3.3 Creating of Larger Logic Topologies

The MO-Boolean-CGP system described in this chapter has demonstrated the ability to generate novel topologies for larger designs which are assembled through the combination of cells. The variability analysis done on these circuits has shown that the evolved designs can offer improved performance over conventional designs, however this is not guaranteed. By including a measure of power, and possibly variability, in the design process, it may be possible to steer the evolution process to producing more reliable circuits. As with conventional CGP (and other development processes such as GP) the algorithm exponentially increases in the time required when evolving larger circuits; as the MO does not begin until functionally correct circuits are created it cannot improve on the performance of conventional CGP in this regard. However, it is proposed that techniques such as ADFs, which have previously been used to accelerate the progress of CGP when evolving larger multiplier circuits, could be used here to accelerate the progress of evolution [170].

6.3.4 Optimising Larger Designs using SGA and Optimised Library

The final stage in the proposed system involves using the SGA system to optimise the larger cell-level designs. To achieve this, rather than optimisng transistor dimensions as is done in the standard-cell optimisation, the system will instead choose which of the available choices for each cell, that exist in the optimised SCL, to insert into the netlist. By using the `type 2` marker-strings in the template netlist, as described in Section 4.2.1, different potential cells can be enabled/disabled as required. For example, given a multiplier circuit evolved with the

MO-Boolean-CGP system, a template SPICE netlist for the circuit could be automatically generated in which each of the cells is shown in the netlist with a pair of subcircuits. The relevant gene in the genotype, when decoded, will determine whether the given cell should be replaced with the high-performance or the low-power variant of the cell.

The circuits will then all be evaluated in SPICE using the methodology described in 6.2.1, with variability analysis added as needed. Through the evolution process, the best performing combinations of circuits in terms of power, delay and variability-tolerance will be promoted to the final population. This system would allow the reuse of the MOTIVATED system with only minor modifications needed to the testing process. However, given the size of the circuits to be evolved in SPICE, the use of HPC to perform the circuit analysis will be essential in order to acheive results in an acceptable timeframe.

In this chapter, a system which enables all stages of the process of generating variability-tolerant logic circuits, such as adders and multipliers, has been proposed. To evolve the topologies of the larger circuits at a cell-level, an adaptation to the existing CGP algorithm has been discussed, in which the multi-objective algorithm used previously is employed to promote circuits with lower transistor and gate counts, and shorter path lengths. The results of this algorithm when compared to the tradition CGP algorithm, other EC algorithms and human-created designs, is discussed. Two of the extracted circuits: a 2-bit adder and a 2-bit multiplier, are assessed for variability tolerance, with the multiplier circuit demonstrating an improvement over the traditional design. At the end of the chapter the methodology in which all the algorithms discussed in this thesis could be combined to create a framework for evolving variability-tolerant logic is discussed. The following final chapter of this thesis contains concluding remarks about the experiments conducted and potential future work.

Chapter 7

Conclusion

In this thesis, a system has been created in which different evolutionary algorithms can be used to create circuits which offer an element of tolerance to the intrinsic variability that will become a major issue in future CMOS design. The majority of the work carried out has been on the use of a genetic algorithm, the SGA, to optimise the transistor widths within 2-input logic cell designs. In this algorithm, a multi-objective fitness function, based on the NSGA-II algorithm, has been used to rank a population of circuits based on their performance at a number of different circuit characteristics. The circuits are assessed using the SPICE simulation software, both with conventional BSIM models, and a special tool, RandomSPICE, which generates batches of netlist in which standard BSIM transistor models are replaced with randomly selected models from libraries of transistors. The transistors in these libraries have parameters tuned to match the output curves of IV data extracted from 3D-atomistic simulations of intrinsic variability in MOSFETs.

In this conclusion, the statements given in the hypothesis are addressed, with additional comments on the three questions that were given in the project proposal to be answered. A look at both the successes and the short-comings of the system that has been created is given, with a discussion of the future work that can be taken to achieve cell-library optimisation with real commercial value.

7.1 Can a genetic algorithm diminish the impact of intrinsic-variability?

This section addresses the first part of the hypothesis to this thesis, involving the use of the multiple-objective genetic algorithm to diminish the impact of intrinsic-variability in standard cell designs, through means of optimising device dimensions. The experiments which attempt to corroborate this part of the hypothesis, given in Chapter 5, have produced results in which the impact of variability can be seen to be reduced in the observed delay and power characteristics of the cell.

The majority of the time spent developing solutions and running experiments has been focussed on solving the problem of cell-library optimisation. The early work, discussed in Section 5.1, focussed on the creation of novel topologies for standard cells. This was perhaps over ambitious - the designs for the topologies of logic gates are well established, and, particularly for the smaller cells, it is highly unlikely that topologies that offer improved performance in terms of delay, power consumption and transistor count are achievable. Despite this, the studies proved invaluable in directing the future research of the project. The studies demonstrated the potential strength of the SGA system was in optimising dimensions within established topologies. The CGP system has created novel topologies for the XOR and XNOR gates, which, whilst offering poor variability-tolerance, did offer a significantly lower power requirement than the conventional SCL designs, thus may have some use in specialist applications or within parts larger circuits where delay is not critical.

The most clear lessons learnt from these early experiments was the need to establish an improved method for assessing the performance of circuits. This resulted in the choice of the NSGA-II multi-objective algorithm, with its strength of promoting those individuals which offered performance across a number of different objectives. By using such a algorithm, it was possible to generate populations in which Pareto-fronts of circuits which contained optimal balances of key characteristics including worst-case delay, standard-deviation of delays, and power. This technique meant that the user could run the algorithm knowing that any given time the circuits could be extracted which offered the best delay, power or balance of objectives at any time, without having to run several algorithms optimising for each characteristic individually.

The first sets of runs using the revised MO strategy resulted in promising results in terms of optimising for power and worst-case delay, but were hampered in their variability-tolerance through the fundamental flaw in the two-stage process used, in which the populations were initially optimised for traditional characteristics using uniform models, then reassessed for variability-tolerance using the variability-models. This strategy was necessary at the time in order to be able to extract results within a sensible time-frame, due to the limited computational resources at the time. Furthermore, whilst the results for optimisation of transistor widths in two-input logic circuits, given in Section 5.2, demonstrated significant improvements over the widths scaled from the SCL used, this is perhaps not a true fair comparison. The cell library used was designed for 130nm channel lengths, which is far enough removed from the 35nm lengths used that the scaling-down of transistor width may not give realistic results, due to the significant differences in the operational characteristics of the 130nm and 35nm devices. Furthermore, the limitation of using unit-widths added further inaccuracies in the scaling process, as many of the widths in the cell-library used sub-unit multiples of 130nm, which had to be rounded up or down.

A set of experiments into allowing multiple supply-voltage rails within cells, using the same libraries and models as in the previous experiments, produced results which suggested this may be a solution which can further reduce variability. It was intended to follow up this research at a later time using the improved HPC-based framework, allowing variability to be considered in results from the beginning, and to further study the effects of allowing similar dynamic body-biassing connections with the body connection of each device connected to independant supply rails. Unfortunately there was not time to carry out these analysis, but it is considered that this would be worthwhile future research in the field of minimising the impact of variability. Such techniques clearly add a significant complexity to the design of fabricated devices, but the use of multiple supply rails is undoubtedly a field which will increase in use and complexity in future microprocessor and other ULSI designs.

From both of these sets of experiments, three important lessons were learned for the future tests. Firstly, whilst the two-stage approach did produce circuits which demonstrated improved power, delay and variability-tolerance over the SCL designs, it was not an effective way of designing for variability-tolerance. To ensure that the populations are always focussed on variability, it is necessary to include the variability-analysis provided by RandomSPICE right from the start of evolutionary routine. To achieve this, it is virtually essential to use some form of high-performance computation in order to achieve results in a realistic time-frame. After extensive research of possible options, including the use of Grid technologies, it was decided that the most time-effective solution would be the use of a local cluster running Sun Grid Engine software, so the MAVEN cluster was purchased, and the MOTIVATED software adapted to allow parallel operation of the SPICE evaluation and data-processing sections on the cluster. The second key lesson learned was the need to carefully validate the test arrangement to ensure that it was correctly evaluating the circuit under test. The failure to do so in the previous experiments had resulted in the misleading results for the D-Type flip-flop, which had seemingly evolved very high-performing optimised designs due to a flaw in the test-setup. Finally, the need to migrate to a more modern cell library, designed for a similar process, would be necessary to increase the reliability of comparisons.

These issues were addressed in the final set of optimisation experiments carried out, which are found in Section 5.6. In these tests, two-input logic cells the Nangate Open-45nm cell library were used; at the time of the tests these were the closest available cells to the 35nm devices modelled in the variability library. The introduction of the MAVEN cluster allowed the variability-aware models to be introduced into the test process from the offset, with each individual being evaluated over 50 variability runs. From the final generation of the run, a single optimised cell was extracted for each test circuit, which demonstrated the best variability-tolerance whilst offering an improved power and delay score over the reference design. To improve the statistical accuracy of the results, each extracted circuit and its equivalent reference design were the re-evaluated for over 100,000 variability runs using the Scot-Grid cluster. This allowed a far greater confidence in the accuracy of results and a more detailed study of the tails of the distribution - the very worst performing circuits.

The results from these experiments were very positive, indicating a clear improvement in the variability-tolerance of the extracted designs for the buffer, XOR and XNOR gate over the reference design, with a particularly significant lessening of delay-variability in the extreme tails of the extracted designs over the reference designs. From the results of these tests, it is clear that the variability-tolerance of cells can be improved through optimisation of the widths of transistors, and that the genetic algorithm approach can automate the process of doing this optimisation.

7.1.1 Is a Genetic Algorithm the best approach?

The question of whether or not a genetic algorithm is actually the most suitable algorithm for the purpose of optimising cell libraries is very valid, and it is hard to draw a firm conclusion on based on this the studies carried out in this thesis. For the smaller circuits, the actual range of possible combinations - let alone sensible combinations - is small enough that other algorithms would make more sense. Indeed, with the models used, an exhaustive search is quite a possible solution for optimising the smaller 2-input logic gates. However, it must be stressed that the model libraries used are in their infancy, and as such are limited in the scope due to the restrictions on available device width and length options. At the time of writing, new libraries are being created which will vastly increase the range of valid width options, as well as introducing variable length options, thus vastly increasing the size of the search space for even the smallest circuits.

Analysis of the transistor dimensions within the publicly available cell libraries reveals that the range of values for device widths is far greater than those used in the experiments in this thesis: in the Nangate library, one can find transistors with channel widths that are specified at 1nm apart; this alone results in 35 times more possible values than discussed in this thesis, before channel lengths are taken into account. In the VSCLIB library, differences of 5nm are found between devices. Whether such fine grained specifications of transistors is actually achievable (or, at least, practical given the extra costs of the lithography processes) in fabrication is not so clear, but it is certain that a finer discretisation than is currently available in the models would be desirable for the optimisation process.

The hill-climbing approach demonstrated a decent performance improvement over the SGA on smaller circuits, arriving at the same solutions in less evaluations. In the larger circuits, with an increased search space and more local optima, the SGA generally found better solutions overall. As expected, both algorithms have their merits, and an algorithm which combines elements of both would potential offer increased performance over the SGA. A number of possible techniques for such an algorithm exist, such as the Parallel Genetic Algorithm, and have been demonstrated for other optimisation problems in the past, including methods which maintain split the population with half performing a hill-climbing or simulated annealing technique. One option which would be simple to implement is to maintain the SGA algorithm, but modify the mutation operator such that a number of the offspring feature a wide-ranging mutation as is presently used, whilst the others use a much smaller adjustment on a single parameter, as would be found in a hill-climbing algorithm.

7.2 Can a complete design process for variability tolerant circuits be created by combining evolutionary techniques?

Whilst the primary focus of the research in this thesis has focussed on the optimisation of standard-cell libraries, the goal of developing a system which would allow the creation of larger variability-tolerant circuits was partially addressed. This began with the construction of novel standard-cell designs using the CGP algorithm within MOTIVATED. Whilst novel designs were evolved, their performance was generally poor compared to the well established conventional designs, particularly when variability was considered. However, to completely rule out the system would be foolhardy, as there are not any well established design rules for creating variability-tolerant designs, and the ability of a system to potential automate this process must hold some merit. Further work in developing the system may yield improved results, particularly when larger standard-cell designs are used with more potential valid arrangements. The second step of the process is the generation of optimised libraries of cells; this has been partially achieved with the SGA system and is certainly achievable given enough development time.

The MO-Boolean-CGP algorithm developed to improve the performance of evolving boolean-logic designs using CGP demonstrated potential at reducing the gate and transistor counts, and corresponding path-lengths, of the evolved designs. Indeed some of the evolved designs were both more compact than traditional human-created designs, and corresponding designs that have been published using alternative EC algorithms. When analysed for using SPICE for performance and variability-tolerance, the circuits demonstrated a mixed response, with the 2-bit adder circuit performing worse than the conventional design, but a significant improvement in performance observed in the 2-bit multiplier, in all circuit characteristics except power consumption. These results suggested that further developments of the system, possibly including SPICE analysis periodically as part of the assessment process, may improve the extracted designs.

The final task necessary to positively address the statement from the hypothesis was unfortunately not addressed during the research, however it is proposed that the existing SGA\MOTIVATED system could be used, with minor modifications, to optimise the cells within a given evolved design by means of selecting the most appropriate design from an existing optimised cell-library. Without having accomplished this part of the process, it is hard to draw positive conclusions to the hypothesis in this regard. The ultimate goal is a system which integrates these algorithms into an automated package, in which the user presents the truth-table for a logic problem and a corresponding variability-aware set of BSIM models, and all the steps are followed, without user-intervention, through to creating an optimised, variability-tolerant design. Whilst such a design is a long way from realisation at this date, it does represent an interesting project for future research.

7.3 The Original Objectives of the nano-CMOS Project

Whilst the results discussed in this thesis set out to address the proposal of the hypothesis, there are also the three original questions that were created in the original proposal for the nano-CMOS project which the ISG were to answer. To varying degrees, all three questions have been address to some extent both in the work covered here and in other studies conducted by myself and Dr. James Walker. In this section, answers to how the three questions have been addressed and how they might be further enhanced in the future are given.

7.3.1 How can evolutionary techniques be used to limit the effects of parameter variations?

The research in this thesis has largely focussed on finding a solution to this question, and as such the optimisation of transistor-dimension within standard cells is the method which has demonstrated the most successful results to date. Attempts to generate novel topologies, whilst producing designs which required less power than conventional designs, did not produce results which demonstrated a greater tolerance to variations caused by intrinsic variability. However, the attempts to optimise dimensions within known designs, particularly using the HPC method developed towards the end of the project, demonstrated a significant improvement in variability tolerance over the reference designs.

7.3.2 How can evolutionary techniques be used within device models to alleviate parameter fluctuation problems?

It is probably this question which has been covered the least in the body of work carried out by the ISG. This is perhaps due to the arguably vague wording of the question itself, as it can be interpreted in different ways. If device models are assumed to mean the BSIM models used for simulating a single device, then at present these do not have the capability to account for parameter fluctuations; as such, it is hard to really consider the problems caused. It would be a reasonable assumption to predict that at some point in the foreseeable future, the BSIM model, or its successor, might be redesigned in such a way that intrinsic variability is directly accounted for within the parameters, in such a way that each instance of a device will have unique parameters based on the statistical sets of IV data extracted from simulations such as those carried out by the DMG; that is to say that SPICE and the models will themselves include a system not dissimilar to the RandomSPICE system discussed in this thesis.

It is perhaps a more fitting question to ask: ‘‘*How can evolutionary techniques be used within the creation of device models which include the effects of parameter fluctuations*”. In the process of 3D-atomistic modelling, huge volumes of data are extracted, which are reduced to sets of IV curves. The process of creating device models from this data involves the adjustment of a few of the BSIM model parameters to fit each of the curves. At the DMG, this work has been done almost entirely by one person, who can be considered an expert at the process of knowing which parameters to adjust to match the curves, a process which requires an in-depth knowledge of both the inner workings of the BSIM model and also the atomistic simulator. This is both painstakingly time-consuming and risky in the sense that loss of that person’s expertise will jeopardise the whole process of model creation. Furthermore, this method to date has been used to create only the unit-width models (models in which the channel width is a unit-multiple of the length).

At the time of writing, the DMG are close to releasing a revised version of RandomSPICE which allows for both variable lengths and widths of devices. To achieve this, the model library includes transistors of width 1-unit, 2-unit, 4-unit, 6-unit and 8-unit (where the unit is the minimum feature size). For devices which fall between these values, the RandomSPICE software adjusts four BSIM parameters based on curves which interpolate these values. Initially, the formula for this curve fitting was calculated by hand. In recent unpublished work, Dr James Walker has successfully investigated if the use of a genetic algorithm can improve the parameters used for this curve fitting process, with promising results. Whilst this research is at an early age, it has already produced results which significantly improve the accuracy of the problem, at the same time removing the need for detailed expertise in the problem area. It is proposed that a similar routine may also help with the problem of curve-fitting in creating the unit-width models, potentially both speeding up the model-extraction process and reducing the reliance on a single person's expertise in the field.

7.3.3 How can parameter variation datasets be best used by evolutionary techniques to improve system performance?

This question has not been solely answered by the work at York, but also significantly by the group at Glasgow, who developed the RandomSPICE software. It is the RandomSPICE tool, and the associated BSIM models which go with it, that have allowed variability to be brought into a conventional simulation system. The integration of RandomSPICE within the evolutionary cycle of MOTIVATED allows variability characteristics of each circuit to be taken into account in the fitness scoring process. To enable the system to be used effectively, it is necessary to exploit the inherent parallelism in the fitness evaluation process and distribute the RandomSPICE, SPICE and objective-score calculation methods across available HPC resources. This has focussed on the use of local-clusters, although the use of Grid-based systems has been exploited in performing subsequent detailed analysis of circuits.

7.4 Future Work

A major obstacle faced throughout the experiments that have been carried out in this project is that the effects of intrinsic variability are not included in any of the standard tools or models used in circuit simulation. As such, the processes to include variability have had to be developed alongside the tools to optimise designs using genetic algorithms, resulting in an ever-changing platform. As the four key components to the process: the MOTIVATED system, RandomSPICE, NGSPICE and the model-libraries themselves, have all undergone numerous updates throughout the course of the thesis, and are only now approaching a state of maturity in which reliable and accurate optimisation can be carried out.

For the RandomSPICE tool and the model-libraries, only now is a system being created in which both lengths and widths of transistors can be adjusted, and with it sub-unit multiples for both parameters. This will hugely increase the search space as the number of possible combinations for transistor dimensions grows. Furthermore it will increase the realism of the optimised designs when compared to reference designs, which already include sub-unit widths and in some cases lengths. The libraries themselves have grown in both the number of intrinsic-variability effects they include, with the earliest libraries only including the effects of RDD, and also the accuracy of the simulated effects. The NGSPICE software, which has been essential due it its licence-free operation, has itself recently undergone many updates, allowing for the updated BSIM models and significantly allowing the $.MEASURE$ statements. These statements increase the speed and accuracy of extracting relevant characteristic data for delay, slew and power from the simulated circuits, and also increase compatibility with the commercial SPICE variants, particularly HSPICE, which is essential for acceptance by industry.

The MOTIVATED system itself has adapted to cater for these developments. One of the most important ongoing areas of development is in adjusting the methods by which circuit characteristics are extracted, to ensure that results correlate directly with commercial tools. Whilst the delay and slew-rates are calculated using recognised methods, the power characteristics in all the experiments in this thesis have considered power-consumption from the supply as a single metric. In reality it is necessary to consider the dynamic and static components of power separately, and much recent work by Dr. James Walker has been done in separating these. The test-bench has been adjusted to reflect the test-bench used in commercial tools, which evaluate circuits under different load conditions through repeated tests with different capacitances at the output. At the time of writing, the latest version of MOTIVATED can produce objective scores which closely match the data analysis from commercial SYNOPSYS tools.

Whilst the focus of the optimisation problems discussed in this thesis has been concerned primarily with 2-input combinatorial logic circuits, there is most definately the scope to optimise other types of cells which are critical to CMOS design, most notably SRAM cells, as discussed in Section 2.7.5. SRAM consumes a large percentage of area in modern microprocessors and has been identified as an area in which intrinsic variability is likely to have an early impact. The reason why the tests in this thesis have steered towards standard-cell combinatorial logic has been based on the need to optimise such cells as part of the nano-CMOS project, with the aim of simulating variability in larger circuits [1]. However, with modest changes to the circuit test-bench and fitness calculation metrics, it would be possible to optimise various

[1]A target of nano-CMOS project as a whole has been to simulate variability in the communications-controller of an ARM CPU that forms part of the SpiNNaker system.

memory cells and flip-flops accurately using the MOTIVATED system.

There is certainly a position for automatic optimisation techniques similar to that used by MOTIVATED to be used in commercial environments for the designs of standard-cells. This is corroborated by letters of support and interest in using such tools submitted from staff at Arm, IBM and Xilinx, as part of a recent successful funding application for a proof-of-concept project into creating commercial tools based on the MOTIVATED system. The funded project, which goes under the name of CATS (Computer-Aided Transistor Scaling) will investigate the creation of a commercial optimisation system which integrates directly with the tools provided by Synopsys for cell-library optimisation, using models provided by Gold-Standard Simulations (GSS), a spin-off company based at Glasgow which is commercialising the RandomSPICE software and the creation of variability models [108].

Although there is still a way to go, the goal of creating a system which can automatically generate variability-tolerant optimised standard-cell libraries for a given set of variability models, with the minimum amount of human-intervention, is both realistic and acheivable. The prospect of a similar system which can go a step beyond this and generate complete optimised circuits to match a given truth-table is something that will take a lot longer, with many more developments in evolutionary technologies needed, to tackle all the scalability issues faced in such a problem. However, if the problems facing semiconductor growth can be overcome, the continual increase in computing-power and ever decreasing costs may provide the resources needed to make such a system feasible.

Appendix A

Additional Results

This appendix includes additional results information for the experiments described in Chapters 5 & 6, and additional material relevant to the experiments conducted.

A.1 Creation of Schematics

In the course of the MO-Boolean-CGP experiments described in Chapter 6, efforts were made to find a system suitable for creating schematic diagrams based on the output. However, there were no suitable free or open-source solutions that could be found. As a consequence, and given the significant time taken to convert even the smallest of circuits into a schematic diagram, it was decided to create a system to automate the process, and directly generate suitable schematic diagrams based on the output matrix received from the MO-Boolean-CGP program. To achieve this, an evolutionary algorithm was used, which is briefly described in this section.

The first step in the process is to parse the output matrix file from the CGP system, which details each of the standard-cells in the evolved design, with its corresponding input and output connection; the width is predetermined from the maximum length between input and output in this matrix. A population of random integer genotypes is then created, which is twice as long as the number of cells in the matrix. The decoding process arranges each column of cells in the output circuit based on the values within the genotype. At this point any empty columns are removed from the matrix, then the cells are wired as determined by the input matrix, with the total length of all the wires and the total number of crossed wires calculated. A fitness score based on the combination of the total wire length and an additional penalty for each occurrence of crossed wires is then calculated; the fittest circuits are those with the shortest combined fitness score.

The fittest member of the population is preserved to the next generation, which the remainder undergoing a tournament selection to select parents, with a predetermined percentage using 2-point crossover and the remainder using single-parent mutation. The algorithm is repeated for a predefined number of generations, at which point the best overall circuit: that with the shortest interconnect length and fewest crossed-interconnects, is extracted. In the extraction process, the circuit is automatically rendered to a scalable vector `.SVG` file, in which the required standard-cell symbols are added at the required locations, based on images originally rendered in the Inkscape vector graphics program. The interconnecting wires are then drawn as needed.

A.2 Results Tables

A.2.1 Results

Table A.1: Results for the reference and optimised NAND, AND, NOR and OR designs from the optimisation runs based on the VSCLIB bulk-130nm SCL, using RDD-aware Toshiba 35nm Models

			Uniform Models			RDF Models: Worst Case			RDF Models: Std-Dev		
Circuit	Design	Area	WC Delay	σDelay	Power	WC Delay	σDelay	Power	WC Delay	σDelay	Power
	Reference	1820	54	3.337	19.54	66	6.309	19.55	2.136	0.659	0.057
	High-Speed	2345	52	7.057	22.47	55	7.514	22.53	0.700	0.728	0.071
NAND	Balanced	1575	53	6.825	16.88	59	7.272	16.91	0.916	0.501	0.055
	Low-Power	420	97	13.11	8.468	147	31.90	8.580	9.300	3.712	0.041
	Variability Tolerant	2310	53	7.403	22.66	56	7.632	22.75	0.565	0.532	0.077
	Reference	1960	68	6.448	17.17	76	8.578	17.32	1.362	0.756	0.066
	High-Speed	2380	60	4.109	17.97	72	5.374	18.14	1.775	0.261	0.080
AND	Balanced	1365	68	4.749	12.79	83	6.798	12.99	2.233	0.367	0.067
	Low-Power	735	92	6.701	10.19	109	13.81	10.27	4.000	2.255	0.033
	Variability Tolerant	3570	65	5.983	13.90	72	8.013	14.02	1.095	0.620	0.049
	Reference	1820	72	18.50	19.99	78	20.17	20.24	1.019	0.737	0.065
	High-Speed	1925	69	12.60	22.22	74	14.55	22.32	0.591	0.583	0.058
NOR	Balanced	1435	72	15.92	18.46	79	18.47	18.58	0.975	0.767	0.050
	Low-Power	595	110	25.50	11.60	126	30.40	11.67	2.592	2.219	0.035
	Variability Tolerant	2100	71	19.01	22.37	76	20.39	22.55	0.658	0.397	0.054
	Reference	2240	68	4.548	22.26	77	6.204	22.63	1.569	0.389	0.086
	High-Speed	2310	62	3.479	21.64	72	5.072	21.94	1.166	0.365	0.075
OR	Balanced	1575	68	3.041	16.79	81	5.141	17.06	1.720	0.412	0.064
	Low-Power	630	96	3.774	11.72	130	12.19	11.96	5.556	1.981	0.056
	Variability Tolerant	2295	66	4.210	21.97	75	6.314	22.95	1.014	0.309	0.080

Table A.2: Results for the reference and optimised XNOR, XOR and D-Type flip-flop designs from the optimisation runs based on the VSCLIB bulk-130nm SCL, using RDD-aware Toshiba 35nm Models

			Uniform Models			RDF Models: Worst Case			RDF Models: Std-Dev		
Circuit	Design	Area	WC Delay	σDelay	Power	WC Delay	σDelay	Power	WC Delay	σDelay	Power
	Reference	3360	90	13.59	31.83	100	14.82	32.12	1.341	0.707	0.101
	High-Speed	4515	85	16.70	36.16	95	17.76	36.78	1.354	0.554	0.145
XNOR	Balanced	3045	91	12.48	30.08	102	13.93	30.67	1.616	0.573	0.155
	Low-Power	1925	104	14.26	22.10	118	17.14	22.22	2.610	1.619	0.078
	Variability Tolerant	4760	82	15.93	35.27	89	16.93	35.65	1.084	0.541	0.105
	Reference	3745	92	17.94	35.83	104	19.77	36.13	1.909	0.374	0.073
	High-Speed	3955	76	10.14	36.32	82	11.35	36.64	1.004	0.563	0.071
XOR	Balanced	2625	92	14.01	27.56	106	16.69	27.72	2.084	0.731	0.071
	Low-Power	1400	115	11.46	20.48	152	19.30	20.66	6.168	2.394	0.086
	Variability Tolerant	3815	79	14.84	35.15	89	17.14	35.47	0.911	0.574	0.082
	Reference	7630	141	15.5	39.81	168	21.5	63.42	4.407	1.841	4.254
	High-Speed	4865	90	0.471	29.35	106	4.027	30.35	2.150	0.931	0.826
D-Type	Balanced	3325	104	0.471	19.80	125	5.906	20.16	3.644	1.699	0.536
	Low-Power	2590	145	7.788	17.44	180	20.5	17.67	6.420	4.316	0.323
	Variability Tolerant	4795	108	3.091	25.99	122	5.5	26.32	1.886	1.265	0.641

Table A.3: The best observed results for the uniform optimisation of the NAND and XOR circuits comparing the SGA, SAHC and SAHC with random walk.

	NAND Gate				XOR Gate			
Algorithm	Area	Worst-Case Delay	σ Delay	Power	Area	Worst-Case Delay	σ Delay	Power
Simple Genetic Algorithm	8	52	3.216	8,468	18	76	5.812	20,484
Steepest-Ascent Hill Climbing	6	54	5.120	10,210	14	82	7.124	25,192
Steepest-Ascent Hill Climbing With Random Walk								
Best Overall	6	52	3.198	8,468	14	76	5.670	20,566
Mean Score across 5 runs	6	52	3.244	8,520	14	76	6.024	21,024

Table A.4: Results for the reference and optimised designs after the variability-stage of the multiple-voltage source optimisation tests on the VSCLIB cell library

Circuit	Models	Objective	Reference	MVS Only		Widths Only		MVS + Widths	
				Low Power	High Speed	Low Power	High Speed	Low Power	High Speed
		Area	3745	3745	3745	1400	3955	910	2380
	Uniform	WC Delay	92	81	70	115	76	93	60
X-OR	Uniform	Power	35.83	30.34	31.80	20.48	36.32	12.22	22.53
	Variability	WC Delay	104	90	77	152	82	114	71
	Variability	σ (WC Delay)	1.909	1.876	1.062	2.394	0.563	4.986	1.935
	Variability	σ (Power)	0.073	0.135	0.156	0.086	0.071	0.086	0.114
		Area	3360	3360	3360	1925	4515	980	2065
	Uniform	WC Delay	90	85	71	104	85	98	68
X-NOR	Uniform	Power	31.83	27.26	27.80	22.10	36.16	12.17	18.98
	Variability	WC Delay	100	95	81	118	95	127	87
	Variability	σ (WC Delay)	1.341	1.634	1.450	1.619	0.554	5.589	3.164
	Variability	σ (Power)	0.101	0.140	0.144	0.078	0.145	0.084	0.137
		Area	7630	7630	7630	2590	4865	2590	5390
	Uniform	WC Delay	141	126	101	145	90	101	90
D-TYPE	Uniform	Power	39.81	33.57	85.71	17.44	29.35	17.44	30.26
	Variability	WC Delay	168	147	116	180	106	180	107
	Variability	σ (WC Delay)	4.407	2.795	2.022	4.316	0.931	4.316	2.675
	Variability	σ (Power)	4.254	3.689	4.644	0.323	0.826	0.323	1.066

Table A.5: Comparison of the uniform-stage and variability-stage results between the optimised VSCLIB standard cell and CGP-Evolved XOR and XNOR designs selected for high-speed (HS), low-power(LP), and a balance between high-speed and low-power (SP)

		SCL XOR			SCL XNOR			Evolved XOR			Evolved XNOR		
	Property	HS	SP	LP	HS	SP	LP	HS	SP	LP	HS	SP	LP
Uniform	Area	3,955	2,625	1,400	4,515	3,045	1,925	2,590	1,190	630	2,555	665	490
	Power	36,320	27,565	20,482	36,164	30,088	22,117	12,959	6,942	5,642	16,047	6,402	5,316
	Delay[1]	76	92	115	85	91	104	116	160	230	122	200	248
	Slew Rate[1]	28	40	52	26	38	54	32	48	60	54	120	130
Variability	Best Delay[2]	80	94	124	88	94	108	124	154	190	130	198	248
	Worst Delay[2]	86	104	152	96	102	122	140	240	322	206	296	356
	σ Delay	2.71	4.52	11.16	3.12	4.49	7.10	8.59	18.79	50.26	13.62	38.13	49.83
	σ Slew Rate	4.97	6.12	12.32	5.12	5.97	14.91	8.92	13.97	18.05	15.67	39.20	38.95

[1] Worst-case scores across all transistions

[2] The best and worst cases for the worst-case delay scores across all transistions, across all Randomspice circuits

Appendix B

Pseudocode

The pseudo-code for some of the primary functional blocks of the SGA system are given in this appendix.

B.1 Mutation Operator

Algorithm 1 Mutation Operation. The genotype is replaced with a new genotype g_{new} in which n_m values are replaced with new random integer values, where n_m is determined by a random Poisson value based on `mutationRate`.

```
Copy old genotype to new genotype $g_{new}$
$n_m = 0$ {*Create new Poisson value*}
$p = 1$
$L = e^{-mutationRate}$
**repeat**
   $n_m + +$
   $u$ = new random double [0,1]
   $p = p \times u$
**until** $p < L$
**if** $n_m > 1$ **then**
   $n_m - -$ {*Make sure at least 1 value is adjusted*}
**end if**
**for** $i = 0$ **to** $n_m$ **do**
   $n_g$ = new random int [0,genotypeLength]
   $v$ = new random int
   $g_{new}[n_g] = v$
**end for**
**return** $g_{new}$
```

B.2 NSGA-II Implementation

Algorithm 2 Assign Crowding Distance. This calculates the rank of each individual within a NDFs, by assign a distance $c_{distance}$ to each one based on the density of solutions within the vicinity of p that are found within the fitness landscape. Based on [53].

```
for all $p$ do
    set $c_{distance} = 0$
end for
for all NDF$_i$ do
    for $obj = 1$ to no.objectives do
        sort $q$ of NDF$_i$ by $obj$
        normalise values
        $c_{distance}[q_0]$ =no.objectives
        $c_{distance}[q_{no.individuals}]$ =no.objectives {Preserve the two extremes}
        for $i = 1$ to no.individuals$-1$ do
            $c_{distance}[q_i] + = c_{distance}[q_{i-1}] + c_{distance}[q_{i+1}]$
        end for
    end for
end for
```

Algorithm 3 Non-Dominated Sorting. Classifies individuals into non-dominated fronts, dependant on how many individuals p dominates & how many individuals dominate p. When no individual dominates p, it is part of the first NDF: the Pareto-optimal front. Based on [53].

```
for $p = 1$ to populationSize do
  $n_d = 0$ {Set number of individual which dominate p to 0}
  $S_p = 0$ {Create empty list of individuals dominated by p}
  for $q = 1$ to populationSize do
    Assume $p$ dominates $q$ and $q$ dominates $p$
    for $ojb = 1$ to no.objectives do
      if $p_{\text{obj}} > q_{\text{obj}}$ then
        $p$ does not dominate $q$
      else if $p_{\text{obj}} < q_{\text{obj}}$ then
        $q$ does not dominate $p$
      end if
    end for
    if $p$ dominates $q$ and $q$ dominates $p$ then
      neither dominate
    end if
    if $p$ dominates $q$ then
      add $q$ to $S_p$
    else if $q$ dominates $p$ then
      $n_p + +$
    end if
  end for
  if $n_p$ = 0 then
    NDF$_0 + = p$
  end if
end for{Pareto-Optimal Front Found}
$i = 0$
while NDF$_i$ is not empty do
  for all $p$ in NDF$_i$ do
    for all $q$ in $S_p$ do
      $n_q - -$
      if $n_q = 0$ then
        NDF$_{i+1}$+=$q$
      end if
    end for
  end for
  $i$++
end while{All NDFs Found}
```

Appendix C

Netlists

C.1 Template Netlists

C.1.1 Template Netlist used in Early Experiments

The template netlist used for the early experiments, described in Section 5.1, is given below. It should be noted that the encoding scheme predates that described in Section 4.2.1, in that each marker string has only two values: the first corresponds to the type, and the second the gene index. The relevant ranges for all markers, and values for multiplication for type 1 markers, were hard-coded into the decoding scheme.

```
*Template netlist for AND\OR Gate Evolution
$2,0  MN1  $0,1  $0,2  $0,3  3 ATOMP L=3.5e-08 W=$1,4e-09
$2,5  MN2  $0,6  $0,7  $0,8  3 ATOMP L=3.5e-08 W=$1,9e-09
$2,10 MN3  $0,11 $0,12 $0,13 3 ATOMP L=3.5e-08 W=$1,14e-09
$2,15 MN4  $0,16 $0,17 $0,18 3 ATOMP L=3.5e-08 W=$1,19e-09
$2,20 MN5  $0,21 $0,22 $0,23 3 ATOMP L=3.5e-08 W=$1,24e-09
$2,25 MN6  $0,26 $0,27 $0,28 3 ATOMP L=3.5e-08 W=$1,29e-09
$2,30 MP1  $0,31 $0,32 $0,33 3 ATOMP L=3.5e-08 W=$1,34e-09
$2,35 MP2  $0,36 $0,37 $0,38 3 ATOMP L=3.5e-08 W=$1,39e-09
$2,40 MP3  $0,41 $0,42 $0,43 3 ATOMP L=3.5e-08 W=$1,44e-09
$2,45 MP4  $0,46 $0,47 $0,48 3 ATOMP L=3.5e-08 W=$1,49e-09
$2,50 MP5  $0,51 $0,52 $0,53 3 ATOMP L=3.5e-08 W=$1,54e-09
$2,55 MP6  $0,56 $0,57 $0,58 3 ATOMP L=3.5e-08 W=$1,59e-09
```

C.1.2 Template Netlists used in VSCLIB Optimisation

The seven template netlists used in the optimisation of transistor widths using two-input logic gates and memories. The netlists are taken from the **VSCLIB-013** standard cell library, which is available at `www.vlsitechnology.org` [135]. The node connections for these circuits are set by the defaults within MOTIVATED: ground is 0, V_{DD} is 10, the two inputs are 1 and 2 and the output is 6. All other internal nodes are here numbered from 20 upwards. These node choices are based on the implementation of the CGP-based topology evolution; they are preserved in the SGA system for compatibility purposes.

NAND Gate

```
* VSCLIB 2-INPUT NAND GATE
MP1 6 1 10 10 ATOMP L=35n W=&0_4_28_35\e-09
MP2 6 2 10 10 ATOMP L=35n W=&1_4_28_35\e-09
MN1 6 2 20 0 ATOMN L=35n W=&2_2_20_35\e-09
MN2 20 1 0 0 ATOMN L=35n W=&3_2_20_35\e-09
```

NOR Gate

```
* VSCLIB 2-INPUT NOR GATE
MP1 20 1 10 10 ATOMP L=35n W=&0_4_28_35\e-09
MP2 6 2 20 10 ATOMP L=35n W=&1_4_28_35\e-09
MN1 6 2 0 0 ATOMN L=35n W=&2_2_20_35\e-09
MN2 6 1 0 0 ATOMN L=35n W=&3_2_20_35\e-09
```

AND Gate

```
* VSCLIB 2-INPUT AND GATE
MP1 20 1 10 10 ATOMP L=35n W=&0_4_28_35\e-09
MP2 21 2 20 10 ATOMP L=35n W=&1_4_28_35\e-09
MP3 6 21 10 10 ATOMP L=35n W=&2_4_28_35\e-09
MN1 21 2 0 0 ATOMN L=35n W=&3_20_35\e-09
MN2 21 1 0 0 ATOMN L=35n W=&4_2_20_35\e-09
MN3 6 21 0 0 ATOMN L=35n W=&5_2_20_35\e-09
```

OR Gate

```
* VSCLIB 2-INPUT NOR GATE
MP1 20 1 10 10 ATOMP L=35n W=&0_4_28_35\e-09
MP2 6 2 20 10 ATOMP L=35n W=&1_4_28_35\e-09
MN1 6 2 0 0 ATOMN L=35n W=&2_2_20_35\e-09
MN2 6 1 0 0 ATOMN L=35n W=&3_2_20_35\e-09
```

XNOR Gate

```
* VSCLIB 2-INPUT XNOR GATE
MP1 20  1 10 10 ATOMP L=35n W=&0_4_28_35\e-09
MP2  6  2 20 10 ATOMP L=35n W=&1_4_28_35\e-09
MP3 21 20 10 10 ATOMP L=35n W=&2_4_28_35\e-09
MP4  6 22 21 10 ATOMP L=35n W=&3_4_28_35\e-09
MP5 22  2 10 10 ATOMP L=35n W=&4_4_28_35\e-09
MN1 20  1  0  0 ATOMN L=35n W=&5_2_20_35\e-09
MN2  6 22 20  0 ATOMN L=35n W=&6_2_20_35\e-09
MN3  6 20 22  0 ATOMN L=35n W=&7_2_20_35\e-09
MN4 22  2  0  0 ATOMN L=35n W=&8_2_20_35\e-09
```

XOR Gate

```
* VSCLIB 2-INPUT XOR GATE
MP1 20  1 10 10 ATOMP L=35n W=&0_4_28_35\e-09
MP2  6 21 20 10 ATOMP L=35n W=&1_4_28_35\e-09
MP3  6 20 21 10 ATOMP L=35n W=&2_4_28_35\e-09
MP4 21  2 10 10 ATOMP L=35n W=&3_4_28_35\e-09
MN1 20  1  0  0 ATOMN L=35n W=&4_2_20_35\e-09
MN2  6  2 20  0 ATOMN L=35n W=&5_2_20_35\e-09
MN3 22 20  0  0 ATOMN L=35n W=&6_2_20_35\e-09
MN4  6 21 22  0 ATOMN L=35n W=&7_2_20_35\e-09
MN5 21  2  0  0 ATOMN L=35n W=&8_2_20_35\e-09
```

D-TYPE Flip-flop

```
* VSCLIB DFNT1 D-type Rising-Edge Flip-flop
MP1 20 1 10 10 ATOMP L=35n W=&0_2_28_35\e-09
```

```
MP2 21 20 10 10 ATOMP L=35n W=&1_2_28_35\e-09
MP3 22 2 10 10 ATOMP L=35n W=&2_2_28_35\e-09
MP4 24 21 22 10 ATOMP L=35n W=&3_2_28_35\e-09
MP5 24 20 25 10 ATOMP L=35n W=&4_2_28_35\e-09
MP6 25 27 10 10 ATOMP L=35n W=&5_2_28_35\e-09
MP7 27 24 10 10 ATOMP L=35n W=&6_2_28_35\e-09
MP8 28 21 29 10 ATOMP L=35n W=&7_2_28_35\e-09
MP9 28 20 27 10 ATOMP L=35n W=&8_2_28_35\e-09
MP10 29 31 10 10 ATOMP L=35n W=&9_2_28_35\e-09
MP11 31 28 10 10 ATOMP L=35n W=&10_2_28_35\e-09
MP12 6 31 10 10 ATOMP L=35n W=&11_2_28_35\e-09
MN1 20 1 0 0 ATOMN L=35n W=&12_2_20_35\e-09
MN2 21 20 0 0 ATOMN L=35n W=&13_2_20_35\e-09
MN3 23 2 0 0 ATOMN L=35n W=&14_2_20_35\e-09
MN4 24 20 23 0 ATOMN L=35n W=&15_2_20_35\e-09
MN5 24 21 26 0 ATOMN L=35n W=&16_2_20_35\e-09
MN6 26 27 0 0 ATOMN L=35n W=&17_2_20_35\e-09
MN7 27 24 0 0 ATOMN L=35n W=&18_2_20_35\e-09
MN8 28 20 30 0 ATOMN L=35n W=&19_2_20_35\e-09
MN9 28 21 27 0 ATOMN L=35n W=&20_2_20_35\e-09
MN10 30 31 0 0 ATOMN L=35n W=&21_2_20_35\e-09
MN11 31 28 0 0 ATOMN L=35n W=&22_2_20_35\e-09
MN12 6 31 0 0 ATOMN L=35n W=&23_2_20_35\e-09
```

C.1.3 Template Netlists used in multiple-voltage source experiments

For conciseness, only the **MVS+W** circuits, in which both the width and voltage-source can be optimised, are shown here.

XOR Gate

```
* VSCLIB 2-INPUT XOR GATE
* Multi-Voltage Source (Fixed Bias) Skeleton Netlist
MP1 20  1 50 10 ATOMP L=35n W=&0_4_28_35\e-09
MP2  6 21 20 10 ATOMP L=35n W=&1_4_28_35\e-09
MP3  6 20 21 10 ATOMP L=35n W=&2_4_28_35\e-09
```

```
MP4 21  2 51 10 ATOMP L=35n W=&3_4_28_35\e-09
MN1 20  1  0  0 ATOMN L=35n W=&4_2_20_35\e-09
MN2  6  2 20  0 ATOMN L=35n W=&5_2_20_35\e-09
MN3 22 20  0  0 ATOMN L=35n W=&6_2_20_35\e-09
MN4  6 21 22  0 ATOMN L=35n W=&7_2_20_35\e-09
MN5 21  2  0  0 ATOMN L=35n W=&8_2_20_35\e-09
VSUB1 10 50 DC &9_0_50_10\m
VSUB2 10 51 DC &10_0_50_10\m
```

XNOR Gate

```
* VSCLIB 2-INPUT XNOR GATE
* Multi-Voltage Source (Fixed Bias) Skeleton Netlist
MP1 20  1 50 10 ATOMP L=35n W=&0_4_28_35\e-09
MP2  6  2 20 10 ATOMP L=35n W=&1_4_28_35\e-09
MP3 21 20 51 10 ATOMP L=35n W=&2_4_28_35\e-09
MP4  6 22 21 10 ATOMP L=35n W=&3_4_28_35\e-09
MP5 22  2 52 10 ATOMP L=35n W=&4_4_28_35\e-09
MN1 20  1  0  0 ATOMN L=35n W=&5_2_20_35\e-09
MN2  6 22 20  0 ATOMN L=35n W=&6_2_20_35\e-09
MN3  6 20 22  0 ATOMN L=35n W=&7_2_20_35\e-09
MN4 22  2  0  0 ATOMN L=35n W=&8_2_20_35\e-09
VSUB1 10 50 DC &9_0_50_10\m
VSUB2 10 51 DC &10_0_50_10\m
VSUB3 10 52 DC &11_0_50_10\m
```

D-Type Flip-flop

```
* VSCLIB DFNT1 D-type Rising-Edge Flip-flop
* Multi-Voltage Source (Fixed Bias) Skeleton Netlist
* Circuit has 8 sub-voltage sources beginning at node 50
MP1 20 1 50 10 ATOMP L=35n W=&0_2_14_70\n
MP2 21 20 51 10 ATOMP L=35n W=&1_2_14_70\n
MP3 22 2 52 10 ATOMP L=35n W=&2_2_14_70\n
MP4 24 21 22 10 ATOMP L=35n W=&3_2_14_70\n
MP5 24 20 25 10 ATOMP L=35n W=&4_2_14_70\n
MP6 25 27 53 10 ATOMP L=35n W=&5_2_14_70\n
```

```
MP7 27 24 54 10 ATOMP L=35n W=&6_2_14_70\n
MP8 28 21 29 10 ATOMP L=35n W=&7_2_14_70\n
MP9 28 20 27 10 ATOMP L=35n W=&8_2_14_70\n
MP10 29 31 55 10 ATOMP L=35n W=&9_2_14_70\n
MP11 31 28 56 10 ATOMP L=35n W=&10_2_14_70\n
MP12 6 31 57 10 ATOMP L=35n W=&11_2_14_70\n
MN1 20 1 0 0 ATOMN L=35n W=&12_2_14_35\n
MN2 21 20 0 0 ATOMN L=35n W=&13_2_14_35\n
MN3 23 2 0 0 ATOMN L=35n W=&14_2_14_35\n
MN4 24 20 23 0 ATOMN L=35n W=&15_2_14_35\n
MN5 24 21 26 0 ATOMN L=35n W=&16_2_14_35\n
MN6 26 27 0 0 ATOMN L=35n W=&17_2_14_35\n
MN7 27 24 0 0 ATOMN L=35n W=&18_2_14_35\n
MN8 28 20 30 0 ATOMN L=35n W=&19_2_14_35\n
MN9 28 21 27 0 ATOMN L=35n W=&20_2_14_35\n
MN10 30 31 0 0 ATOMN L=35n W=&21_2_14_35\n
MN11 31 28 0 0 ATOMN L=35n W=&22_2_14_35\n
MN12 6 31 0 0 ATOMN L=35n W=&23_2_14_35\n
VSUB1 10 50 DC &24_0_30_10\m
VSUB2 10 51 DC &25_0_30_10\m
VSUB3 10 52 DC &26_0_30_10\m
VSUB4 10 53 DC &27_0_30_10\m
VSUB5 10 54 DC &28_0_30_10\m
VSUB6 10 55 DC &29_0_30_10\m
VSUB7 10 56 DC &30_0_30_10\m
VSUB8 10 57 DC &31_0_30_10\m
```

C.1.4 Template Netlists for CGP evolved topologies

XOR Gate

```
* CGP Evolved 10-T XOR Design
M0 15 1 0 0 ATOMN L=35n W=&0_1_20_35\e-9
M1 16 17 15 0 ATOMN L=35n W=&1_1_20_35\e-9
M2 6 16 0 0 ATOMN L=35n W=&2_1_20_35\e-9
M3 17 2 0 0 ATOMN L=35n W=&3_1_20_35\e-9
```

```
M4 15 1 18 10 ATOMP L=35n W=&4_1_28_70\e-9
M5 18 17 10 10 ATOMP L=35n W=&5_1_28_70\e-9
M6 17 2 10 10 ATOMP L=35n W=&6_1_28_70\e-9
M7 6 16 10 10 ATOMP L=35n W=&7_1_28_70\e-9
M8 16 15 18 10 ATOMP L=35n W=&8_1_28_70\e-9
M9 16 1 17 10 ATOMP L=35n W=&9_1_28_70\e-9
```

XNOR Gate

```
* CGP Evolved 8-T XNOR Design
M0 15 1 0 0 ATOMN L=35n W=&0_2_14_35\e-9
M1 6 17 15 0 ATOMN L=35n W=&1_2_14_35\e-9
M2 17 2 0 0 ATOMN L=35n W=&2_2_14_35\e-9
M3 15 1 18 10 ATOMP L=35n W=&3_2_14_70\e-9
M4 18 17 10 10 ATOMP L=35n W=&4_2_14_70\e-9
M5 17 2 10 10 ATOMP L=35n W=&5_2_14_70\e-9
M6 6 15 18 10 ATOMP L=35n W=&6_2_14_70\e-9
M7 6 1 17 10 ATOMP L=35n W=&7_2_14_70\e-9
```

Appendix D

Glossary and List of Abbreviations

ADF	Automatically Defined Function
AFS	Andrew File System; a distributed file-system
ASIC	Application-Specific Integrated Circuit
BAC	Boltzmann Acceptance Criterion
BiCMOS	Bipolar-CMOS; circuits comprising both BJT and MOSFETs
BJT	Bipolar Junction Transistor
BSIM	Berkeley Short-channel IGFET Model; a SPICE transistor model series
BTE	Boltzmann Transport Equation
CGP	Cartesian Genetic Programming
CMOS	Complimentary-MOS; logic design based on NMOS:PMOS pairs
CPU	Central Processing Unit
CVD	Chemical Vapour Deposition
DMG	Device Modelling Group; research group at Glasgow University
DNA	Deoxyribonucleic Acid
DRAM	Dynamic Random Access Memory
EC	Evolutionary Computation
ECC	Error Correcting Code
EDA	Electronic Design Automation
EE	Evolutionary Electronics
EEPROM	Electronically Erasable Programmable Read-Only Memory
EMA	Evolutionary Module Acquisition
EP	Evolutionary Programming

EPSRC	Engineering and Physical Sciences Research Council
ES	Evolutionary Strategy
EUV	Extreme Ultra-Violet; proposed future lithography source
FET	Field Effect Transistor
FLASH	A non-volatile form of memory
FPAA	Field-Programmable Analogue Array
FPGA	Field-Programmable Gate Array
FPTA	Field-Programmable Transistor Array
GA	Genetic Algorithm
GPU	Graphics Processing Unit
GSI	Giga-Scale Integration; billions of transistors per chip
HPC	High-Performance Computation
IC	Integrated Circuit
IGFET	Insulated-Gate Field Effect Transistor
IPC	Instructions Per Clock; a CPU performance metric
IQR	Inter-Quartile Range
ISG	Intelligent Systems Group; research group at University of York
ITRS	International Technology Roadmap for Semiconductors
JFET	Junction Field Effect Transistor
LER	Line-Edge Roughness
LPCVD	Low-Pressure Chemical Vapour Deposition
MAVEN	A cluster computer using Sun Grid Engine
MMC	Metropolis Monte-Carlo Simulation
MO	Multi-Objective
MOEA	Multi-Objective Evolutionary Algorithm
MOS	Metal-Oxide Semiconductor
MOSFET	Metal-Oxide Semiconductor Field Effect Transistor
MOTIVATED	Multi-Objective Toolkit for Intrinsic Variability Aware Transistor-level Evolutionary Design
nano-CMOS	EPSRC funded research project into transistor variability
NBTI	Negative-Bias Temperature Instability
NEGF	Nonequilibrium Green's Function
NGSPICE	A free SPICE simulator
NMOS	N-Channel MOS Transistor
NSGA-II	Non-dominated Sorting Genetic Algorithm II

NTRS	National Technology Roadmap for Semiconductors
OPC	Optical Proximity Corrections
PCB	Printed Circuit Board
PECVD	Plasma Enhanced Chemical Vapour Deposition
PMOS	P-Channel MOS Transistor
PSGB	Poly-silicon Grain Boundaries
RAM	Random Access Memory
RANDOMSPICE	Software written at Glasgow University for creating variability-aware netlists
RDD	Random Discrete Dopants
RDF	Random Dopant Flunctuations; variability caused by RDD
RET	Reticle (or Resolution) Enhancement Technology; techniques to improve the accuracy of photolithography
RIE	Reactive-Ion Etching
RNA	Ribonucleic Acid
SA	Simulated Annealing
SAHC	Steepest Ascent Hill-Climbing
SCL	Standard-Cell Library; library of building-block logic circuits
SGA	Simple Genetic Algorithm
SGE	Sun Grid Engine; clustering software
SIA	Semiconductor Industry Association
SOI	Silicon-on-Insulator
SPICE	Simulation Program with Integrated Circuit Emphasis; a mixed-mode electronics simulator
SRAM	Static Random Access Memory
SSH	Secure Shell; a network protocol for remote administration of Unix computers
TTL	Transistor-Transistor Logic; integrated circuits based on BJTs
ULSI	Ultra-Large Scale Integration; millions of transistors per chip
UTB	Ultra-thin Body
VHDL	VHSIC Hardware Discription Language
VHSIC	Very-High Speed Integrated Circuit
VLSI	Very-Large Scale Integration; thousands of transistors per chip
V_T	Threshold Voltage

Bibliography

[1] Thermal oxidation. Online: http://www.siliconfareast.com/oxidation.htm.

[2] Industry agrees on first 450-mm wafer standard. EE Times. Online: http://www.eetimes.com/electronics-news/4079657/Industry-agrees-on-first-450mm-wafer-standard, October 2008.

[3] International technology roadmap for semiconductors executive summary 2009. Online: http://www.itrs.net/links/2009ITRS/Home2009.htm, 2009.

[4] Grid engine manual pages. Online: http://gridengine.sunsource.net/background.html, September 2010.

[5] Intel product information. Online: http://ark.intel.com/, 2010.

[6] July semiconductor sales up 37 percent year-on-year. Semiconductor Industry Association. Online: http://www.sia-online.org/cs/papers_publications/press_release_detail?pressrelease.id=1805, August 2010.

[7] Nangate 45nm open cell library. Online: http://www.nangate.com/openlibrary, September 2010.

[8] ScotGrid: Scotting grid service, equipment page. Online: http://www.scotgrid.ac.uk/equipment, September 2010.

[9] Varun Aggarwal, Varun Jain, Selcuk Kilinc, and Ugur Cam. Automatic design of trustworthy sine-wave oscillators by genetic algorithms, 2006.

[10] Varun Aggarwal, Wesley O. Jin, and Una-May O'Reilly. Filter approximation using explicit time and frequency domain specifications. In *GECCO*, 2006.

[11] Varun Aggarwal and Una-May O'Reilly. Design of posynomial models for mosfets: Symbolic regression using genetic algorithms. In *Genetic Programming Theory and Practice IV*. Springer, 2006.

[12] Bashir Al-Hashimi. *The Art of Simulation Using PSpice, Analog and Digital*. CRC Press, 1995.

[13] M. A. Alam. A critical examination of the mechanics of dynamic NBTI for PMOSFETs. In *Electron Devices Meeting, 2003 IEDM Technical Digest*, pages 14.4.1 – 14.4.4. IEEE International, December 2003.

[14] Bruce Alberts, Dennis Bray, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. *Essential Cell Biology: An Introduction to the Molecular Biology of the Cell*. Garland Publishing, 1997.

[15] Massimo Alioto and Gaetano Palumbo. *Model and Design of Bipolar and MOS Current-Mode Logic: CML, ECL and SCL Digital Circuits*. Kluwer Academic Publishers, 2005.

[16] Phillip E. Allen and Douglas R. Holberg. *CMOS Analog Circuit Design, Second Edition*. Oxford University Press, 2002.

[17] P. J. Angeline and J. B. Pollack. Evolutionary module acquisition. In D. Fogel and W. Atmar, editors, *Proceedings of the Second Annual Conference of Evolutionary Programming*, pages 154–163, La Jolla, CA, 25–26 1993.

[18] P. Antognetti and G. Massobrio. *Semiconductor Device Modelling with SPICE*. McGraw Hill, 1987.

[19] A. Asenov and S. Saini. Suppresion of random dopant-induced threshold voltage fluctuations in sub-0.1-μm mosfets with epitaxial and δ-doped channels. *IEEE Transactions on Electron Devices*, 46(8):1718–1724, 1999.

[20] Asen Asenov. Random dopant induced threshold voltage lowering and fluctuations in sub-50-nm MOSFETs: a statistical 3D 'atomistic' simulation study. *Nanotechnology*, 10:153–158, 1999.

[21] Asen Asenov. Meeting the design challenges of nano-CMOS electronics proposal. 2006.

[22] Asen Asenov. Variability in the next generation CMOS technologies and impact on design. In *International Conference of CMOS Variability*, 2007.

[23] Asen Asenov, Savas Kaya, and Andrew R. Brown. Intrinsic parameter fluctuations in decananometer MOSFETs introduced by gate line edge roughness. *IEEE Transactions on Electron Devices*, 50(5):1254–1260, 2003.

[24] A. C. Atkinson. The computer generation of poisson random variables. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):29–35, 1979.

[25] Thomas Bäck. Optimal mutation rates in genetic search. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 2–8, 1993.

[26] Thomas Bäck, David B Fogel, and Zbigniew Michalewicz, editors. *Evolutionary Computation 1: Basic Algorithms and Operators*. Institute of Physics, 2000.

[27] Thomas Bäck, Ulrich Hammel, and Hans-Paul Schwefel. Evolutionary computation: Comments on the history and current state. *IEEE Transactions on Evolutionary Computation*, 1(1):3–17, 1997.

[28] Walter Banzhaf. *Computer-Aided Circuit Analysis Using SPICE*. Prentice-Hall, 1989.

[29] K. Bernstein et al. High-performance CMOS variability in the 65-nm regime and beyond. *Advanced Silicon Technology*, 50, 2006.

[30] Yu Bi, D. Ioan, and N.P. van der Meijs. Process variation aware modeling of interconnect capacitance. In *Prorisc 2008 Proceedings*. Technologiestichting STW, 2008.

[31] C. F. Bohren and D. Huffmen. *Absorption and Scattering of Light by Small Particles*. John Wiley, 1983.

[32] Lashon B. Booker, David B. Fogel, Darrell Whitley, Peter J. Angeline, and A. Eiben. Recombination. In *Evolutionary Computation 1: Basic Algorithms and Operators*, pages 256–307. Institute of Physics Publishing, 2000.

[33] Shekhar Borkar. Design reliable systems from unreliable components: The challenges of transistor variability and degradation. *IEEE Micro*, 25(6):10–16, 2005.

[34] George E. P. Box. Evolutionary operation: A method for increasing industrial productivity. *Journal of the Royal Statistical Society, Series C (Applied Statistics)*, 6(2):81–101, 1957.

[35] S. D. Brotherton. Polycrystalline silicon thin film transistors. *Semiconductor Science and Technology*, 10:721–738, 1995.

[36] Andrew R. Brown, Gareth Roy, and Asen Asenov. Poly-Si-gate-related variability in decananometer MOSFETs with conventional architecture. *IEEE Transactions on Electron Devices*, 54(11):3056–3063, 2007.

[37] Cadence®. Cadence PSpice A/D & PSPICE advanced analysis: Advanced circuit simulation and analysis for analog and mixed-signal circuits. Technical report, Cadence®, 2010.

[38] Luigi Capodieci. From optical proximity correction to lithography-driven physical design (1996-2006): 10 years of resolution enhancement technology and the roadmap enablers for the next decade. In *Proceedings of SPIE, the International Society for Optical Engineering*, volume 6154(3), 2006.

[39] Anantha P. Chandrakasan, Samuel Sheng, and Robert W. Brodersen. Low-power CMOS digital design. *IEEE Journal of Solid-State Circuits*, 27(4):473–484, 1992.

[40] Demian D. Chapman, Mahmood S. Shivji, Ed Louis, Julie Sommer, Hugh Fletcher, and Paulo A. Prodohl. Virgin birth in a hammerhead shark. *Biology Letters*, 3(4):425–427, 2007.

[41] Pallab Chatterjee. Device scaling: The treadmill that fueled three decades of semiconductor industry growth. *IEEE Solid-State Circuits Newsletter*, 12(1):14–18, 2007.

[42] Yuhua Cheng and Chenming Hu. *MOSFET modeling & BSIM3 User's Guide*. Kluwer Academic Publishers.

[43] T. F. Ciszek. Techniques for the crystal growth of silicon ingots and ribbons. *Journal of Crystal Growth*, 66(3):655–672, 1984.

[44] J. Clegg, J. A. Walker, and J. F. Miller. A new crossover technique for cartesian genetic programming. In *Proc. of GECCO*, 2007.

[45] Mike Clinton. Variation tolerant SRAM design technologies. In Masayuki Mizuno and Vivek De, editors, *Proceedings of VLSI Circuits Short Circuits Short Course Program: "Design for Variability in Logic, Memory and Microprocessors"*, pages 150–186. VLSI Circuit Syposium, June 2007.

[46] Carlos A. Coello Coello. Twenty years of evolutionary multi-objective optimization: A historical view of the field. *IEEE Computational Intelligence Magazine*, 1(1):28–36, 2006.

[47] Carlos A. Coello Coello, Gary B. Lamont, and David A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems, 2^{nd} edition*. Springer, 2007.

[48] M. J. Cooke. *Semiconductor devices*. Prentice-Hall, 1990.

[49] Ben Coppin. *Artificial Intelligence Illuminated*. Jones and Bartlett, 2004.

[50] Nichael Lynn Cramer. A representation for the adaptive generation of simple sequential programs. In John J. Grefenstette, editor, *Proceedings of an International Conference on Genetic Algorithms and their Applications*, Carnegie-Mellon University, Pittsburgh, PA, July 1985.

[51] J. A. Croon, G. Storms, S. Winkelmeier, I. Pollentier, M. Ercken, S. Decoutere, W. Sansen, and H. E. Maes. Line edge roughness: characterization, modeling and impact on device behavior. In *International Electron Devices Meeting*, 2002.

[52] Charles Darwin. *On the origin of species by means of natural selection, or the preservation of favoured races in the struggle for life*. John Murray, 1859.

[53] K. Deb, Pratap A., S. Agarwal, and T. Meyarivan. A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6:181–197, 2002.

[54] Nitin Deo. Addressing manufacturing variation at advanced nodes with silicon-contour-based dfm. In *Cadence Designer Network, CDNLive EMEA*, Munich, Germany, April 2008.

[55] Sunil Dhingra, Rajnish Sharma, M. S. Yadav, and P. J. George. Chemical bath deposition of thin copper film for metallization on silicon. In R. Mukhopadhyay, editor, *Proceedings of the D.A.E. Solid State Physics Symposium*, pages 217–218, 1999.

[56] C. H. Diaz, Tao Hun-Jan, Ku Yao-Ching, A. Yen, and K. Young. An experimentally validated analytic model for gate line-edge roughness (LER) effects on technology scaling. *IEEE Electron Device Letters*, 22(6):287–289, 2001.

[57] W. Eccleston. The effect of polysilicon grain boundaries on MOS based devices. *Microelectronic Engineering*, 48:105–108, 1999.

[58] David Esseni. On the modeling of surface roughness limited mobility in SOI MOSFETs and its correlation to the transistor effective field. *IEEE Transactions on Electron Devices*, 51(3):394–401, 2004.

[59] Zhun Fan, Jianjun Hu, Kisung Seo, Erik D. Goodman, Ronald C. Rosenberg, and Baihai Zhang. Bond graph representation and gp for automated analog filter design. In Erik D. Goodman, editor, *Genetic and Evolutionary Computation Conference Late Breaking Papars GECCO - 2001*, pages 81–86, San Francisco, CA, July 2001.

[60] Stuart J. Flockton and Kevin Sheehan. Intrinsic circuit evolution using programmable analogue arrays. *Lecture Notes in Computer Science*, 1478:144–153, 1998.

[61] David B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE Press, 2000.

[62] L. J. Fogel, A. J. Owens, and M. J Walsh. *Artificial Intelligence through Simulated Evolution*. Wiley, 1966.

[63] Device Sizing for Minimum Energy Operation in Subthreshold Circuits. Benton h. calhoun and alice wang and anantha chandrakasan. In *IEEE Custom Integrated Circuits Conference*, 2004.

[64] D. P. Foty. *MOSFET Modeling with SPICE: Principles and Practice*. Prentice-Hall, 1997.

[65] David J. Frank. High performance CMOS variability in the 65nm regime and beyond. In *International Conference on CMOS Variability: "The Impact of Variability on Design"*, Royal College of Physicians, London, October 2007.

[66] Steve Furber. Dealing with variability in modern circuit design. In *International Conference on CMOS Variability*, 2007.

[67] Zbysek Gajda and Lukas Sekanina. Reducing the number of transistors in digital circuits using gate-level evolutionary design. In *Genetic And Evolutionary Computation Conference: Proceeding of the*9^{th} *annual conference on Genetic and evolutionary computation*, volume 1, pages 245–252, London, England, 2007.

[68] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[69] P. Gray and R. Meyer. *Analysis and Desing of Analog Integrated Circuits (3rd edition)*. John Wiley, 1993.

[70] James B. Grimbleby. Automatic analogue circuit synthesis using genetic algorithms. *Circuits, Devices and Systems, IEE Proceedings*, 147:319–323, December 2000.

[71] Fatima Z. Hadjam, Claudio Moraga, and Mohammed Benmohamed. Cluster-based evolutionary design of digital circuits using improved multi-expression programming. In *Genetic and Evolutionary Computation GECCO 2007, Late Breaking Papers*, 2007.

[72] P. Hazucha, T. Karnik, J. Maiz, S. Walstra, B. Bloechel, J. Tschanz, G. Dermer, S. Hareland, P. Armstrong, and S. Borkar. Neutron soft error rate measurements in a 90-nm CMOS process and scaling trends in SRAM from 0.25-μm to 90-nm generation. In *International Electron Devices Meeting (IEDM)*, 2003.

[73] B. Hoeneisen and C. A. Mead. Fundamental limitations in microelectronics – i. MOS techonology. *Solid-State Electronics*, 15(1):819–829, 1972.

[74] John H. Holland. *Adaptation in Natural and Artificial Systems*. Bradbord Books, 1992.

[75] Xuejue Huang, Wen-Chin Lee, Charles Kuo, Digh Hisamoto, Leland Chang, Jakub Kedzierski, Erik Anderson, Hideki Takeuchi, Yang-Kyu Choi, Kazuya Asano, Vivek Subramanian, Tsu-Jae King, Jeffrey Bokor, and Chenming Hu. Sub 50-nm FinFET: PMOS. In *Electron Devices Meeting, IEDM Technical Digest International*, 1999.

[76] Maturana Humberti R and Francisco J. Varela. *The Tree of Knowledge: The Biological Roots of Human Understanding*. Shambhala, 1998.

[77] Intel. Microprocessor quick reference guide, 2008. Online: `http://www.intel.com/pressroom/kits/quickreffam.htm`.

[78] Seonghoon Jin, Massimo V. Fischetti, and Ting-Wei Tang. Modeling of surface-roughness scattering in ultrathin-body SOI MOSFETs. *IEEE Transactions on Electron Devices*, 54(9):2191–2203, 2007.

[79] Martin Hartley Jones. *A Practical Introduction to Electronic Circuits*. Cambridge University Press, 2000.

[80] Michael Kanellos. Moore's law to roll on for another decade. CNET News. Online: `http://news.cnet.com/2100-1001-984051.html`.

[81] R. W. Keyes. Physical limits in digital electronics. *Proceeding of the IEEE*, 63(5):740–767, 1975.

[82] Yaser M. A. Khalifa, Badar K. Khan, and Faisal Taha. Multi-objective optimization tool for a free structure analog circuits design using genetic algorithms and incorporating parasitics. In *Genetic and Evolutionary Computation Conference, GECCO 2007 Late Breaking Papers*, pages 2527–2534, London, England, July 2007.

[83] J. S. Kilby. Miniaturized electronic circuits, 1964.

[84] Selçuk Kilinç, Varun Jain, Varun Aggarwal, and Ugur Cam. Catalogue of variable frequency and single-resistance controlled oscillators employing a single differential difference complementary current conveyor. *Frequenz: Journal of RF-Engineering and Telecommunications*, 04:142, 2006.

[85] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[86] Hans Kosina and Siegfried Selberherr. A hybrid device simulator that combines monte carlo and drift-diffusion analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and System*, 13(2):201–210, 1994.

[87] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[88] John R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, Cambridge, MA, 1994.

[89] John R. Koza. Routine human-competitive machine intelligence by means of genetic programming. *Applications and Science of Neural Networks, Fuzzy Systems, and Evolutionary Computation VI, proceedings of the SPIE*, 52:1–15, 2004.

[90] John R. Koza, David Andre, Forrest H. Bennett III, and Martin A. Keane. Use of automatically defined functions and architecture-altering operations in automated circuit synthesis with genetic programming. In *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 132–149, Stanford University, CA, 28–31 1996. MIT Press.

[91] John R. Koza, Forresst H. Bennett III, Jason Lohn, Frank Dunlap, Martin A. Keane, and David Andre. Automated synthesis of computational circuits using genetic pro-

gramming. In *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation*, pages 447–452. IEEE Press, 13-16 1997.

[92] John R. Koza, Forrest H. Bennett III, David Andre, and Martin A. Keane. Automatic WYWIWYG design of both the topology and component values of electrical circuits using genetic programming. In John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 123–131. MIT Press, 1996.

[93] John R. Koza, Forrest H. Bennett III, David Andre, and Martin A. Keane. Evolution using genetic programming of a low-distortion 96 decibel operational amplifier. In Barrett Bryant, Janice Carroll, Dave Oppenheim, Jim Hightower, and K. M. George, editors, *Proceedings of the 1997 ACM Symposium on Applied Computing*, pages 207–216, Hyatt Sainte Claire Hotel, San Jose, CA, 28 -2 1997.

[94] John R. Koza, Forrest H Bennett III, David Andre, and Martin A. Keane. *Genetic Programming III: Darwinian Invention and Problem Solving*. Morgan Kaufmann, 1999.

[95] John R. Koza, Martin A. Keane, Jessen Yu, and William Mydlowec. Automatic synthesis of electrical circuits containing a free variable using genetic programming. In Darrell Whitley, David Goldberg, Erick Cantu-Paz, Lee Spector, Ian Parmee, and Hans-Georg Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 477–484, Las Vegas, Nevada, USA, 10-12 2000. Morgan Kaufmann.

[96] Slawomir Koziel and Wladyslaw Szczesniak. Reducing average and peak temperatures of VLSI CMOS circuits by means of evolutionary algorithm applied to high level synthesis. *Microelectronics Journal*, 34:1167–1174, 2003.

[97] Wim Kruiskamp and Domine Leenaerts. DARWIN: CMOS opamp synthesis by means of a genetic algorithm. In *Proceeding of the 32nd ACM/IEEE conference on Design automation*, pages 433–438, San Francisco, CA, 1995. ACM Press.

[98] K. J. Kuhn. Reducing variation in advanced logic technologies: Approaches to process and design for manufacturability of nanoscale cmos. In *IEDM Digest of Technical Papers*, pages 471–474, 2007.

[99] V. Kursun and E. G. Friedman. *Multi-voltage CMOS Circuit Design*. Wiley, 2006.

[100] Jörg Langeheine, Joachim Becker, Simon Fölling, Karlheinz Meier, and Johannes Schemmel. A CMOS FPTA chip for intrinsic hardware evolution of analog electronic circuits. In *3rd NASA / DoD Workshop on Evolvable Hardware*, pages 172–175, Long Beach, CA, July 2001. IEEE Computer Society.

[101] Jörg Langeheine, Karlheinz Meier, and Johannes Schemmel. Intrinsic evoltion of quasi DC solutions for transistor level analog electronic circuits using a CMOS FPTA chip. 2002.

[102] Jörg Langeheine, Karlheinz Meier, and Johannes Schemmel. Intrinsic evolution of analog electronic circuits using a CMOS FPTA chip. In *International Congress on Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, 2003.

[103] Peng Li, Frank Liu, Xin Li, Lawrence T. Pileggi, and Sani R. Nassif. Modeling interconnect variability using efficient parametric model order reduction. *Design, Automation and Test in Europe*, 2005.

[104] Xiaojun Li, Jin Qin, Bing Huang, Xiaohu Zhang, and Joseph B. Bernstein. SRAM circuit-failure modeling and reliability simulation with SPICE. *IEEE Transactions on Device and Materials Reliability*, 6(2), June 2006.

[105] Yiming Li. An automatic parameter extraction technique for advanced cmos device modeling using genetic algorithm. *Microelectronic Engineering*, 84:260–272, February 2007.

[106] Jason D. Lohn and Silvano P. Colombano. Automated analog circuit synthesis using a linear representation. In *Proc of the Second Int'll Conf on Evolvable Systems: From Biology to Hardware*, volume 84, pages 125–133, Berlin, 1998. Springer-Verlag.

[107] Bo Lojek. *History of semiconductor engineering*. Springer, 2007.

[108] Gold Standard Simulations Ltd. Online: `http://www.goldstandardsimulations.com`.

[109] Mark Lundstrom. EE-612: Lecture 19: Device variability. Technical report, Purdue University, 2006.

[110] H. Karbasi S. Winters S. Minehane J. Babcock M. Ershov, S. Saxena. Dynamic recovery of negative bias temperature instability in p-type metal-oxide-semiconductor field-effect transistors. *Applied Physics Letters*, 83:1647–1649, 2003.

[111] A. Martinez, J. R. Barker, N. Seoane, A. R. Brown, and A. Asenov. Dopants and roughness induced resonances in thin Si nanowire transistors: A self-consistent NEGF-poisson study. *Journal of Physics Conference Series, Progress in Nonequilibrium Green's Functions IV*, 220:1–15, 2010.

[112] J. W. Mayer. Ion implantation in semiconductors. *International Electron Devices Meeting (IEDM)*, pages 3–5, 1973.

[113] James Meindl. Gigascale integration: Is the sky the limit? *IEEE Circuits and Devices Magazine*, 12(6):19–24, 1996.

[114] Zbigniew Michalewicz, Girish Nazhiyath, and Maciej Michalewicz. A note on usefulness of geometrical crossover for numerical optimization problems. In *Proceedings of the 5^{th} Annual Conference of Evolutionary Programming*, pages 305–312, 1996.

[115] J. F. Miller, D. Job, and V. K. Vassilev. Principles in the evolutionary design of digital circuits - part I. *Genetic Programming and Evolvable Machines*, 1:8–35, 2000.

[116] J. F. Miller and S. L. Smith. Redundancy and computational efficiency in cartesian genetic programming. *IEEE Transactions on Evolutionary Computing*, 10:167–174, 2006.

[117] J. F. Miller and P. Thomson. Cartesian genetic programming. In *Proc. of EuroGP*, 2000.

[118] Masayuki Mizuno and Vivek De, editors. *Proc. of VLSI Circuits Short Course Program,"Design for Variability in Logic, Memory and Microprocessor"*, Kyoto, Japan, 2007.

[119] T. Mizuno, J. Okamura, and A. Toriumi. Experimental study of threshold voltage fluctuation due to statistical variation of channel dopant number in MOSFETs. *IEEE Transactions on Electron Devices*, 41(11):2216–2221, 1994.

[120] Tomohisa Mizuno, Jun ichi Okamura, and Akira Toriumi. Experimental study of threshold voltage fluctuations using an 8k MOSFET array. In *Symposium on VLSI Technology, Digest of Technical Papers*, pages 41–42, 1993.

[121] Gordan E. Moore. Cramming more components onto integrated circuits. *Electronics*, 38, 1965.

[122] Gordon E. Moore. Progress in digital integrated electronics. In *Electron Devices Meeting*, 1975.

[123] Victor Moroz. Design for manufacturability: OPC and stress variations. In *International Conference on CMOS Variability*, 2007.

[124] Tanvir Hasan Morshed et al. *BSIM4.6.4 MOSFET Model - Users Manual*. University of California, Berkeley, CA94720, 2008.

[125] H. Mühlenbein, M. Schomisch, and J. Born. The parallel genetic algorithm as a function optimizer. *Parallel Computing*, 17:619–632, 1991.

[126] Heinz Mühlenbein. How genetic algorithms really work: I. Mutation and hillclimbing. In *Parallel Problem Solving from Nature 2*, pages 15–25. Elsevier, 1992.

[127] Laurence W. Nagel. The life of SPICE. Talk transcript, BCTM '96, September 1996.

[128] Laurence W. Nagel and D. O. Pederson. SPICE (simulation program with integrated circuit emphasis). Technical Report UCB/ERL M382, EECS Department, University of California, Berkeley, 1973.

[129] National Microelectronics Institute. *International Conferenceon CMOS Variability: "The Impact of Variability on Design"*, Royal College of Physicians, London, October 2007.

[130] Paolo Nenzi. *NgSPICE: Product History Page*. gEDA, 2010. Online: `http://ngspice.sourceforge.net/history.html`.

[131] Kenneth V. Noren and John E. Ross. Analog circuit design using genetic algorithms. In *Second Online Symposium for Electronics Engineers*, 2001.

[132] Phil Oldiges, Qinghuang Lin, Karen Petrillo, Martha Sanchez, Meikei Ieong, and Mike Hargrove. Modeling line edge roughness effects in sub 100 nanometer gate length devices. In *SISPAD Proceedings*, 2000.

[133] Vilfredo Pareto. Cours d'économie politique. *The Annals of the American Academy of Political and Social Sciences*, 9(128):128–131, 1897.

[134] D. Patterson. The top 10 innovations in the new NVIDIA fermi architecture, and the top 3 next challenges. Technical report, NVidia Corp., 2009.

[135] Graham Petley. VLSI and ASIC technology standard cell library design. Online: `www.vlsitechnology.org`.

[136] Jan M. Rabaey. *Digital Integrated Circuits: A Design Perspective*. Prentice-Hall, 1995.

[137] Mohsen Razavy. *Quantum Theory of tunnelling*. World Scientific, 2003.

[138] David T. Reid. *Large-scale simulations of intrinsic parameter fluctuations in nano-scale MOSFETs*. PhD thesis, University of Glasgow, 2010.

[139] Gareth Roy, A. R. Brown, Fikru Adamu-Lema, Scott Roy, and Asen Asenov. Simulation study of individual and combined sources of intrinsic parameter fluctuations in conventional nano-mosfets. *IEEE Transactions on Electron Devices*, 53(12):3063–3070, 2006.

[140] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach ($3^r d$ edition)*. Prentice-Hall, 2003.

[141] Ralf Salomon and Frank Sill. High-speed, low-leakage integrated circuits: An evolutionary algorithm perspective. *Journal of Systems Architecture*, 53:321–327, 2007.

[142] K. Samsudin, B. Cheng, A. R. Brown, S. Roy, and A. Asenov. Integrating intrinsic parameter fluctuation description into BSIMSOI to forecast sub-15nm UTB SOI based 6T SRAM operation. *Solid-State Electronics*, 50:86–93, 2006.

[143] K. Samsudin, B. Cheng, A. R. Brown, S. Roy, and A. Asenov. Sub-25nm UTB SOI SRAM cell under the influence of discrete random dopants. *Solid-State Electronics*, 50:660–667, 2006.

[144] Nobuyuki Sano and Masaaki Tomizawa. Random dopant model for three-dimensional drift-diffusion simulations in metal-oxide-semiconductor field-effect-transistors. *Applied Physics Letters*, 79(14):2267–2268, 2001.

[145] J. David Schaffer, Rich Caruana, Larr J. Eshelman, and Rajarshi Das. A study of control parameters affecting online performance of genetic algorithms for function optimization. In *Proceeding of the 3^{rd} International Conference on Genetic Algorithms*, pages 51–60, 1989.

[146] Dieter K. Schroder. Negative bias temperature instability: What do we understand? *Microelectronics Reliability*, 6:841–852, 2007.

[147] Hans-Paul Schwefel and Günter Rudolph. Contemporary evolution strategies. In F. Morana et al., editors, *Advances in Artificial Life*, pages 893–907. Berlin: Springer, 1995.

[148] Larry Seiler, Doug Carmean, Eric Spangle, Tom Forsyth, and Pradeep Dubey. Larrabee: a many-core x86 architecture for visual computing. *ACM Transactions of Graphics (TOG)*, 27(3), 2008.

[149] Harold Shichman and David A. Hodges. Modeling and simulation of insulated-gate field-effect transistor switching circuits. *IEEE Journal of Solid State Circuits*, SC-3:285–289, 1968.

[150] Silvaco. SmartSpice: Analog circuit simulator. Online. `http://www.silvaco.com/products/circuit_simulation/smartspice_35.pdf`, 2010.

[151] R. Sinnott, A. Asenov, D. Berry, D. Cumming, S. Furber, C. Millar, A. Murray, S. Pickles, S. Roy, A. Tyrrell, and M. Zwolinski. Meeting the design challenges of nano-CMOS electronics: An introduction to an upcoming epsrc pilot project. In *UK e-Science All Hands Meeting*, Nottingham, UK, 2006.

[152] Jim Smith and T. C. Fogarty. Self adaptation of mutation rates in a steady state genetic algorithm. In *Proceedings of the 3rd IEEE Conference on Evolutionary Computation*, pages 318–323, 1996.

[153] Matthew J Streeter, Martin A. Keane, and John R. Koza. Automatic synthesis using GP of both the topology and sizing for five post-2000 patented analog and mixed-analog digital circuits. In *Southwest Symposiumon Mixed-Signal Design*, 2003.

[154] Ben G. Streetman and Sanjay Banerjee. *Solid State Electronic Devices*. Prentice-Hall, 2000.

[155] C. R. M. Struck, R. Flauta, M. J. Neumann, K. N. Kim, R. Raju, R. L. Bristol, and D. N. Ruzic. Grazing incidence broad ion beams for reducing line-edge roughness. *Journal of Micromechanics and Microengineering*, 20:1–6, 2010.

[156] Ivan Sutherland, Bob Sproull, and David Harris. *Logical Effort - Designing Fast CMOS Circuits*. Morgan Kaufmann, 1999.

[157] Synopsys. *HSPICE®The Gold Standard for Accurate Circuit Simulation.* Online: `http://www.synopsys.com/Tools/Verification/`

`AMSVerification/CircuitSimulation/HSPICE/Documents/hspice_ds.pdf`.

[158] Michael A. Terry, Jonathan Marcus, Matthew Farrel, Varun Aggarwal, and Una-May O'Reilly. GRACE: generative robust analog circuit exploration. In Franz Rothlauf, Jurgen Branke, Stefano Cagnoni, Ernesto Costa, Carlos Cotta, Rolf Drechsler, Evelyne Lutton, Penousal Machado, Jason H. Moore, Juan Romero, George D. Smith, Giovanni Squillero, and Hideyuki Takagi, editors, *Applications of Evolutionary Computing, EvoWorkshops2006: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoInteraction, EvoMUSART, EvoSTOC*, volume 3907 of *LNCS*, pages 332–343, Budapest, 2006. Springer-Verlag.

[159] Adrian Thompson. On the automatic design of robust electronics through artificial evolution. 1999.

[160] Martin Albrecht Trefzer. *Evolution of Transistor Circuits*. PhD thesis, Ruperto-Carola-University of Heidelberg, Germany, 2006.

[161] H. P. Tuinhout. Impact of parametric mismatch and fluctuations on performance and yield of deep-submicron cmos technologies. In *ESSDERC Proceedings*, 2002.

[162] Yoshiyuki Uno, Akira Okada, Yasuhiro Okamoto, and Tameyoshi Hirano. High performance slicing method of monocrystalline silicon ingot by wire edm. *Initiatives of Precision Engineering at the Beginning of a Millenium*, 1:219–223, 2001.

[163] John P. Uyemura. *CMOS Logic Circuit Design*. Kluwer Academic Publishers, 2003.

[164] V. K. Vassilev and J. F. Miller. Embedding landscape neutrality to build a bridge from the conventional to a more efficient three-bit multiplier circuit. In *Gecco*, 2000.

[165] J. A. Walker and J. F. Miller. Evolution and acquisition of modules in cartesian genetic programming. In *Proc. of EuroGP*, 2004.

[166] J. A. Walker and J. F. Miller. The automatic acquisition, evolution and reuse of modules in cartesian genetic programming. *IEEE Transactions on Evolutionary Computation*, 12:397–417, 2008.

[167] James A. Walker, James A. Hilder, and Andy M. Tyrrell. Evolving variability-tolerant CMOS designs. In *Proc. of ICES*, pages 308–319. Springer-Verlag, 2008.

[168] James Alfred Walker. *The Automatic Acquisition, Evolution and Re-use of Modules in Cartesian Genetic Programming*. PhD thesis, University of York, December 2007.

[169] James Alfred Walker, James A. Hilder, and Andy M. Tyrrell. Evolving industry-feasible intrinsic variability tolerant cmos design. In *IEEE Conference of Evolutionary Computation (CEC)*, 2009.

[170] James Alfred Walker and Julian Francis Miller. Improving the evolvability of digital multipliers using embedded cartesian genetic programming and product reduction. In *International Conference of Evolvable Systems: From Biology to Hardware*, pages 131–142, 2005.

[171] James Alfred Walker, Richard Sinnott, Gordon Stewart, James A. Hilder, and Andy M. Tyrrell. Optimizing electronic standard cell libraries for variability tolerance through the nano-CMOS grid. *Philosophical Transactions of the Royal Society A*, 368:3967–3981, 2010.

[172] Feng Wang, Yuanxiang Li, Li Li, and Kangshun Li. Automated analog circuit design using two-layer genetic programming. *Applied Mathematics and Computation*, 185:1087–1097, 2007.

[173] Jin Wang and Chong Ho Lee. Evolutionary design of combinational logic circuits using VRA processor. *IEICE Electronics Express*, 6:141–147, 2009.

[174] Frank M. Wanlass. Low stand-by power complementary field effect circuitry, 1967.

[175] R. K. Watts. *Submicron Integrated Circuits*, chapter Lithography, pages 470–508. Wiley Interscience, 1989.

[176] S. Winkelmeier, M. Sarstedt, M. Ereken, M. Goethals, and K. Ronse. Metrology method for the correlation of line edge roughness for different resists before and after etch. *Microelectronic Engineering*, 57–58:665–672, 2001.

[177] Chao Wu. Evolutionary design of analogue circuits. Master's thesis, University of York, 2005.

[178] C. Wyon. Future technology for advanced MOS devices. *Nuclear Instruments and Methods in Physics Research B*, 186:380–391, 2002.

[179] T. Yamaguchi, H. Namatsu, M. Nagase, K. Yamakazi, and K. Kurihara. Nanometer-scale linewidth fluctuations caused by polymer aggregates in resist films. *Applied Physics Letters*, 71(16):2388–2390, 1997.

[180] Xin Yao and Yong Liu. Fast evolutionary programming. In *Proceedings of the Fifth Annual Conference on Evolutionary Programming*, pages 451–460. MIT Press, 1996.

[181] Xin Yao, Yong Liu, Ko-Hsin Liang, and Guangming Lin. Fast evolutionary algorithms. In *Natural Computing Series, Advances in evolutionary computation: theory and applications*, pages 45–94. Springer-Verlag, 2003.

[182] T. Yu and J. F. Miller. Neutrality and the evolvability of boolean function landscape. In *Proc. of EuroGP*, 2001.

[183] Ricardo S. Zebulum, Marco Aurélio Pacheco, and Marley Vellasco. Analog circuits evolution in extrinsic and intrinsic modes. In *ICES '98: Proceedings of the Second International Conference on Evolvable Systems*, pages 154–165, London, 1998. Springer-Verlag.

[184] Bart Van Zeghbroeck. *Principles of Semiconductor Devices*. Colorado University, 2001.

[185] Eckart Zitzler, Marco Laumanns, and Lother Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, ETH-Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, 2001.