Variability-Aware Circuit Performance Optimisation Through Digital Reconfiguration

Pedro Burmester Campos

Ph.D.

University of York
Electronics

November, 2015

Abstract

This thesis proposes optimisation methods for improving the performance of circuits implemented on a custom reconfigurable hardware platform with knowledge of intrinsic variations, through the use of digital reconfiguration.

With the continuing trend of transistor shrinking, stochastic variations become first order effects, posing a significant challenge for device reliability. Traditional device models tend to be too conservative, as the margins are greatly increased to account for these variations. Variation-aware optimisation methods are then required to reduce the performance spread caused by these substrate variations.

The Programmable Analogue and Digital Array (PAnDA) is a reconfigurable hardware platform which combines the traditional architecture of a Field Programmable Gate Array (FPGA) with the concept of configurable transistor widths, and is used in this thesis as a platform on which variability-aware circuits can be implemented.

A model of the PAnDA architecture is designed to allow for rapid prototyping of devices, making the study of the effects of intrinsic variability on circuit performance – which requires expensive statistical simulations – feasible. This is achieved by means of importing statistically-enhanced transistor performance data from RandomSPICE simulations into a model of the PAnDA architecture implemented in hardware. Digital reconfiguration is then used to explore the hardware resources available for performance optimisation. A bio-inspired optimisation algorithm is used to explore the large solution space more efficiently.

Results from test circuits suggest that variation-aware optimisation can provide a significant reduction in the spread of the distribution of performance across various instances of circuits, as well as an increase in performance for each. Even if transistor geometry flexibility is not available, as is the case of traditional architectures, it is still possible to make use of the substrate variations to reduce spread and increase performance by means of function relocation.

Contents

List of Figures

List of Tables

Acknowledgments

There are many people who have been essential in making it possible for me to finish this thesis.

I wish to express my gratitude to Prof. Andy Tyrrell, Dr. Martin Trefzer, Dr. James Alfred Walker and Dr. Simon Bale for their knowledge, understanding and encouragement throughout the duration of this work. Their exceptional advice, uncompromising patience, and research creativity have allowed me to pursue this work with a constant sense of curiosity. I realise I have been very fortunate for having such outstanding supervisors and colleagues.

I would also like to extend my gratitude to everyone involved in the PAnDA project, whose vision and work, even before I became part of it, allowed me to take my research in interesting directions.

Finally, I would like to thank my parents, my brother and my sister, for always being supportive of my work, during the good and the less good moments in the past four years. Without their understanding and encouragement, this work would not have been possible.

Declaration

This thesis is a presentation of my original research work. Wherever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature, and acknowledgement of collaborative research and discussions.

The work was done under the guidance of Prof. Andy Tyrrell and Dr. Martin Trefzer, at the Electronics Department of the University of York. This work has not previously been presented for an award at this, or any other, University.

The work presented in this thesis resulted in publication in the proceedings of the 2013 *IEEE Congress of Evolutionary Computation* [31], in the 2014 IEEE International Conference on Evolvable Systems [32], in the *Designing with Uncertainty* workshop at *DATE'15*, and finally a publication in the proceedings of the 2015 *International Conference on Field Programmable Logic and Applications (FPL)* [33].

Chapter 1

Introduction

Contents

1.1 Introduction

Technology has become a major part of society, playing a key role in almost any activity today. As such, electronic device performance has been relentlessly pushed forward to allow for more complex tasks to be completed with increasingly smaller amounts of energy.

This has motivated the constant shrinking of devices, pushing both their performance and their manufacturing to their physical limits. As transistors become only a few atoms wide, it becomes increasingly difficult to manufacture two devices that behave exactly alike [7]. Modern day devices can be made up of billions of transistors, and therefore these variations give rise to a dynamic fabric, with performance varying across the substrate depending on the magnitude of these variations affecting each transistor. This poses a significant threat for the manufacturing of reliable devices [29].

Reconfigurable hardware platforms provide flexible substrates on which to implement circuits, allowing for the same design to be implemented in different locations of the fabric, effectively making use of different hardware resources. Approaches as, for instance,

variability-aware mapping attempt to take these variations into account in the process of mapping designs to Field-Programmable Gate Arrays (FPGA), a family of reconfigurable hardware platforms. Combining this idea with the inevitable variability of the fabric, an opportunity arises to explore these variations for circuit performance optimisation. The Programmable Analogue and Digital Array (PAnDA) takes this reconfigurability one step further by allowing for a wide range of transistor geometries, effectively altering the electric properties of each transistor.

In this thesis, an approach is described which makes use of that variation to actually improve the performance of circuits, as well as to reduce the spread that it causes, on the custom reconfigurable hardware platform that is PAnDA. Investigating the variability mitigation of PAnDA across a large number of physical devices is a time-consuming process, and also potentially not economically viable. The more economically viable option is to run variability-aware simulations of the architecture, making use of tools such as RandomSPICE, but this tends to be even more time-consuming than real hardware. An embedded model is used to accelerate the performance characterisation process across a large number of virtual physical instances (VPI) of the PAnDA architecture, making use of the inherent parallelism of hardware to achieve the speed-up with respect to software-based simulations. Variability-aware data is included by means of a feature block which contains data extracted from low-level software-based simulations. The effects of variability then propagate to the circuit-level, causing a distribution of virtual devices in terms of performance.

With the developed variability-aware fast prototyping tool, evolutionary models inspired from Darwinian evolution are used along with digital reconfiguration to accelerate the exploration of the solution space, and to ultimately mitigate the effects of variability at the circuit-level.

1.2 The PAnDA Project

The PAnDA project was a four-year EPSRC (EP/I005838/1) funded project, started in October 2010, involving the Intelligent Systems Research Group at the University of York and the Device Modelling Group at the University of Glasgow, and it is also part of a special interest group including Imperial College London and the University of Southampton, and Gold Standard Simulations Ltd as industrial partners.

The project aims to tackle one of the main challenges in nano-scale electronic design: incorporating the effects of intrinsic variability – that become more pronounced as device shrinking continues to keep up with Moore's law – into the circuit design process, in order to achieve functional circuit designs. Both deterministic and stochastic variability have an impact on design, but whereas the former can be accounted for through specific design techniques, the latter requires statistical modelling, and is therefore more challenging to address.

This research aims to develop understanding of how stochastic variability will affect circuit design in deep sub-micron processes and to propose novel design methodologies to overcome these intrinsic variations. The project involved the design and fabrication of a novel reconfigurable variability-tolerant architecture, which allows for variability-aware design and rapid virtual physical instance creation by exploiting the configuration options of the architecture. These are vital steps towards the next generation of FPGA architectures [30].

1.3 Hypothesis

This work aims to test the following hypothesis:

It is possible to mitigate the effects of atomistic variability on the PAnDA architecture at the circuit-level through the use of digital reconfiguration, whilst making use of the substrate variations to allow for circuit performance optimisation.

This can be broken down into two sub-hypotheses:

It is possible to use digital reconfiguration of the PAnDA architecture to optimise the performance of a circuit.

It is possible to reduce the impact of variability on a circuit mapped to the PAnDA architecture making use of its digital reconfiguration resources.

In order to address this hypothesis, the following objectives are laid out:

- Evaluate the performance of a reconfigurable architecture under the effect of intrinsic variability (addressed in Chapter 4).
- Accelerate the circuit performance characterisation process (addressed in Chapter 5).
- Provide methodology to allow for the performance evaluation of large numbers of devices (addressed in Chapter 5).

- Provide an automated method for the performance optimisation process (addressed in Chapter 6).
- Make use of both substrate variations and digital reconfiguration resources to both improve circuit performance and to allow for standardisation across large numbers of devices (addressed in Chapter 6).

1.4 Structure of Thesis

The thesis is organised as follows: Chapter 2 provides a detailed overview of the problem of transistor variability in CMOS technology. First, a description of the structure of CMOS devices is provided, and process variations affecting the manufacture of such devices are divided into two categories: systematic (or deterministic) and intrinsic (or stochastic). The main sources of the former are described, along with the established methods to mitigate their effects. The main sources of the latter are then presented, along with the impact that these can have on device performance. The chapter concludes with the fact that random variations will become the leading cause for chip variability.

Chapter 3 then presents a survey of techniques applied at pre-fabrication, during the manufacturing process, and at the post-fabrication stage to mitigate the effects of random device variations. The potential of reconfigurable hardware platforms to improve yield and reliability as well as to provide methods which can be used to adapt local hardware resources to suit the performance requirements is given special emphasis, with the PAnDA architecture being identified as a flexible substrate on which to apply such methods.

In order to facilitate the study of the impact of variability on the PAnDA architecture across a large number of devices, Chapter 4 describes the development of the PAnDA Emulator, a model of the PAnDA architecture which incorporates statistically-enhanced transistor models. The model is then accelerated through the use of a hardware implementation, to allow for the characterisation of the impact of variability at the circuit-level, which requires prohibitive amounts of computation if done exclusively in software. The model includes custom-designed reconfiguration resources which are used in later chapters to allow for variability-aware circuit performance optimisation.

Chapter 5 presents the concept of using the PAnDA Emulator to generate *virtual physical instances* (VPI) of circuits, where each VPI represents a fabricated instance of that circuit, by using various combinations of transistor models with different electric properties in each.

The hardware-accelerated Emulator is able to predict the impact of variability on a design orders of magnitude faster than a software-based simulation. This acceleration comes at the expense of modelling accuracy, a matter which is also discussed in the chapter. Custom-designed frequency and propagation delay hardware modules are described and included in the design, to allow for the performance evaluation necessary for the optimisation methods.

In Chapter 6, the PAnDA Emulator is then revised to only allow for a single transistor size, and varying input-slew rates and output loads are included in the model in an effort to increase its accuracy with respect to a traditional software simulation. The concept of influencing the performance of a circuit by using the reconfiguration resources of the PAnDA architecture is then introduced, with a bio-inspired Genetic Algorithm being used to apply functionally-neutral changes to the implemented circuit through the use of the reconfiguration resources described in Chapter 4 as well as the performance evaluation hardware modules introduced in Chapter 5. The performance of a large number of VPIs are then improved through the use of the described methods, with the impact of variability being successfully reduced.

Chapter 7 presents the concluding remarks and observations, and provides some insight into future applications and extensions of the work described in this thesis.

Appendix A lists the file structure of the USB stick and CD-ROM provided, which contain the files required to replicate the work undertaken in this thesis.

1.5 Contributions and Novelties

The main novelties presented in this thesis can be summarised as follows:

- Development of an automated method for extracting statistically-enhanced propagation delay data for architecture-specific logic functions (Chapter 4);
- Development of hardware configuration resources to allow for online reconfiguration of PAnDA devices (Chapters 4 & 5);
- Incorporation of statistically-enhanced data into a hardware-accelerated Emulator of the PAnDA architecture, allowing for rapid instantiation of devices (Chapter 5);
- Development of an embedded frequency and delay measurement module to characterise circuits mapped to the Emulator (Chapter 5);

- Development of a bio-inspired optimisation algorithm to improve circuit performance in the presence of fabric variations, and incorporation into the hardware-accelerated Emulator to allow for overall performance improvement estimations across large numbers of VPIs (Chapter 6).

1.6 Publications

In 2013 a paper was published in the proceedings of the *IEEE Congress of Evolutionary Computation*, using the initial modelling work of the PAnDA architecture to develop fault-tolerant methods using evolution [31]. In 2014, the developed optimisation algorithm and the digital reconfiguration resources were incorporated to allow for operation-point matching of ring-oscillators, resulting in a paper published in proceedings of the *IEEE International Conference on Evolvable Systems* [32]. In 2015, the accelerated variability characterisation method developed in this work was presented at the *Designing with Uncertainty* workshop at the *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. Also in 2015, a paper was published in the proceedings of the *International Conference on Field-programmable Logic and Applications (FPL)*, analysing the effects of variability on a Zynq platform [33].

Chapter 2

Variability in CMOS Devices

Contents

2.1 Introduction

Variability has always existed in the circuit manufacturing process. Whether it is from wafer to wafer, from die to die, or between elements in the same die (intra-die) its presence and effects have always been recognised and integrated in the design process [34, 35, 36, 37, 38]. Most of these variations are deterministic in nature, and can therefore be modelled and are typically dealt with through the inclusion of appropriate margins, in a process known as Design for Manufacturability (DFM) [39].

As more and more complexity is demanded from devices, manufacturers have spent the last 50 years or so coming up with new ways to increase the transistor count in the same die

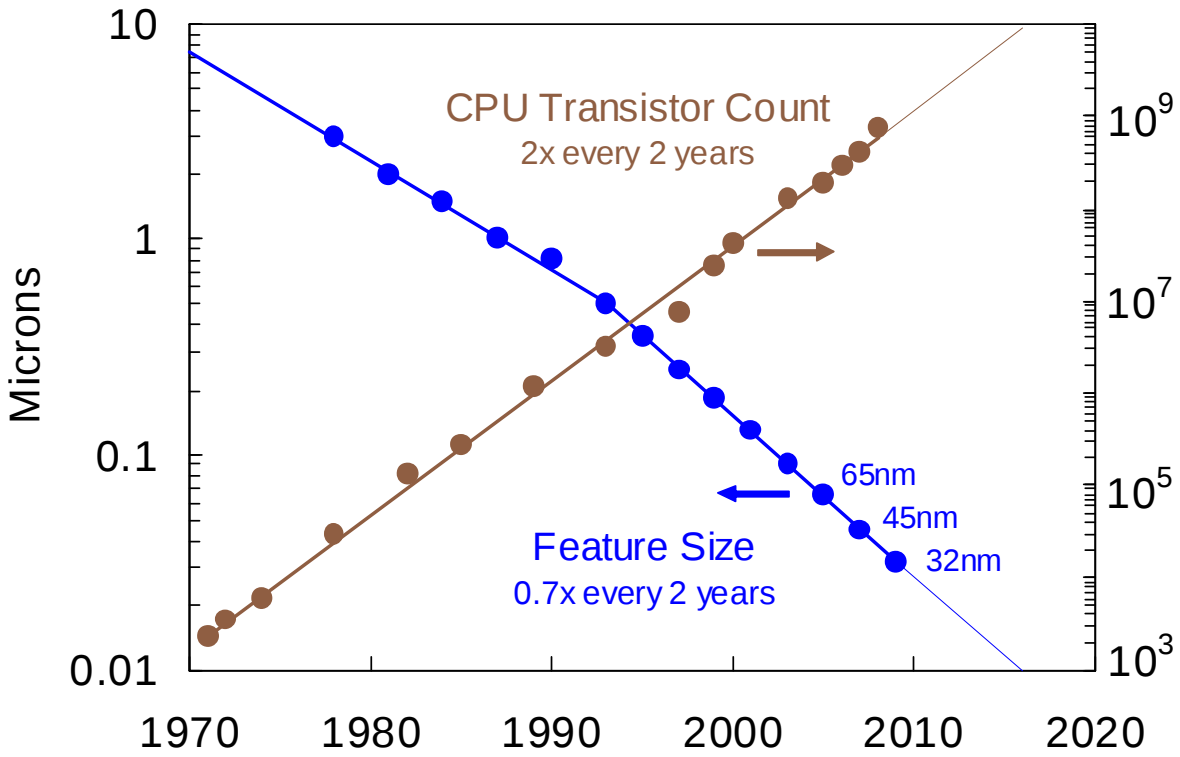


Figure 2.1: The evolution of the transistor count in a CPU (in brown, scale on the right) and the size of the transistors used (in blue, scale on the left) through the least 40 years. Figure sourced from [1].

area, mostly by reducing the size of individual transistors – also known as *feature size* – so as to maximise computational density. In what became known as *Moore's Law* [40], the rate at which the number of transistors increased per area unit was observed to be roughly 2x every two years, as Figure 2.1 illustrates. Up until the beginning of the 20th century, this transistor scaling translated to a reduction of oxide thickness, length and width of the transistors, as well as a reduction in the power consumption of each device, in what is known as the *Dennard scaling* [41]. Beyond the $130nm$ feature size, however, the performance of these transistors began to degrade, and manufacturers had to resort to material enhancers to compensate for this reduction in performance, such as *silicon strain* introduced to the $90nm$ and $65nm$ technology nodes [42].

As technology is scaled down in Complementary Metal-Oxide-Semiconductor (CMOS) designs, intra-die variability becomes the leading factor for physical parameter variations, overcoming the effects of inter-die, intra- and inter-wafer variations [43].

Although some of these are of deterministic nature, there is also a group of variations of *stochastic* nature. These physical variations create a large difference in performance from one transistor to another, within a die – which is characterised by a large standard deviation

in the statistical distribution of the manufacturing process. This in turn creates a problem for circuit manufacturers, since the same design could have a completely different performance when implemented on two different dies, and also a single manufacturing run could produce a large number of non-functional circuits, which represents a drastic yield reduction. This has a great economic impact, and therefore a minimisation of these effects is highly desirable.

Although this work does not describe an attempt to solve the issue of stochastic variability at the transistor level, it is still important to provide some background regarding the sources of stochastic variability in the device manufacturing process, which is the subject matter of this chapter. Given that CMOS is the most common technology present in electronic devices today, it is the focus of this background work.

2.2 CMOS Technology and Moore's Law

CMOS is a technology for constructing integrated circuits (IC). They currently form the most widely used technology across digital circuits, in part due to its exceptional power efficiency. CMOS circuits exhibit most of their power dissipation during switching, losing very little energy in a static state. Microprocessors, FPGAs, memory, ASICs and many other VLSI devices rely on CMOS technology. CMOS logic consists of symmetrical pairs of p- and n-type MOSFETs which are used to implement functions [44]. Figure 2.2 illustrates the architecture of an n-channel MOSFET. Figure 2.3 illustrates a cross-section of a CMOS pair of devices, fabricated with a p- substrate process.

To keep up with Moore's Law, MOSFET manufacturers consistently reduced the physical dimensions of the devices, typically shortening their channel length (L) and width (W), along with reducing the thickness of the oxide insulation layer (T_{ox}). Figure 2.3 shows a cross-section of a CMOS gate, with a PMOS and an NMOS transistors fabricated on a $p-$ type substrate. This approach proved to be successful up until the $130nm$ technology node, for which the scaling caused a significant reduction in performance [1]. For instance, the reduction in gate-oxide thickness was pushed so far that some electrons in the gate began to quantum-tunnel into the substrate, causing undesired behaviour in the transistors, and ultimately non-functional VLSI circuits.

Sub-$100nm$ transistor manufacturing then began including enhancers such as high-k dielectrics instead of the typical silicon dioxide for the oxide-insulation layer, and reincor-

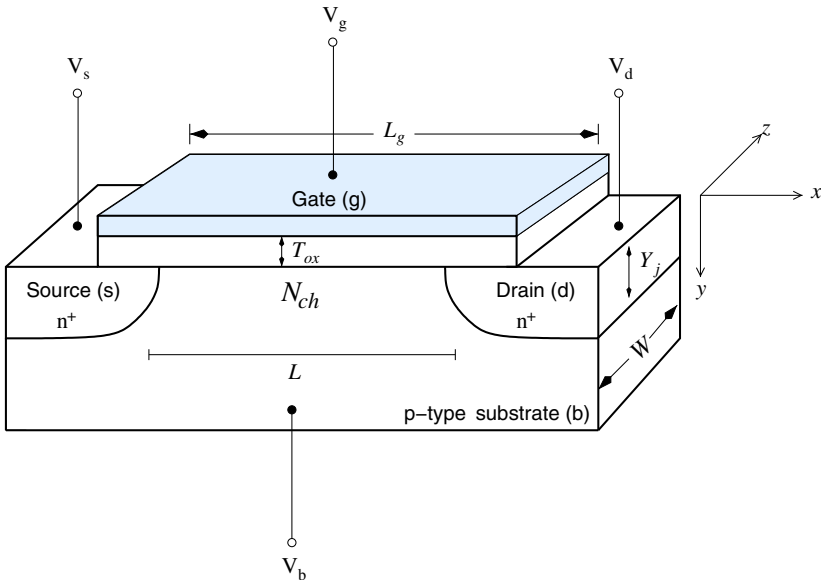


Figure 2.2: Physical structure of an n-channel MOSFET. Key physical features such as width (W), channel length (L), and oxide thickness (T_{ox}) are labelled in the figure. Figure sourced from [2].

porating metal gates [45, 42] in order to continue making reliable devices as the scaling progressed.

However, these new materials can only go so far. As device scaling takes us to feature sizes of only a few nanometers, the channels become only a few handfuls of atoms long, as Figure 2.4 illustrates, and any small atomic discrepancies between devices will translate to serious performance mismatches.

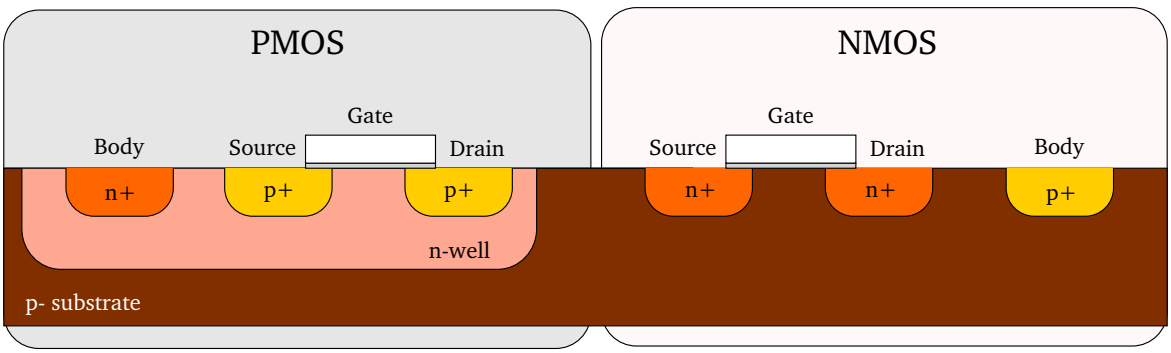
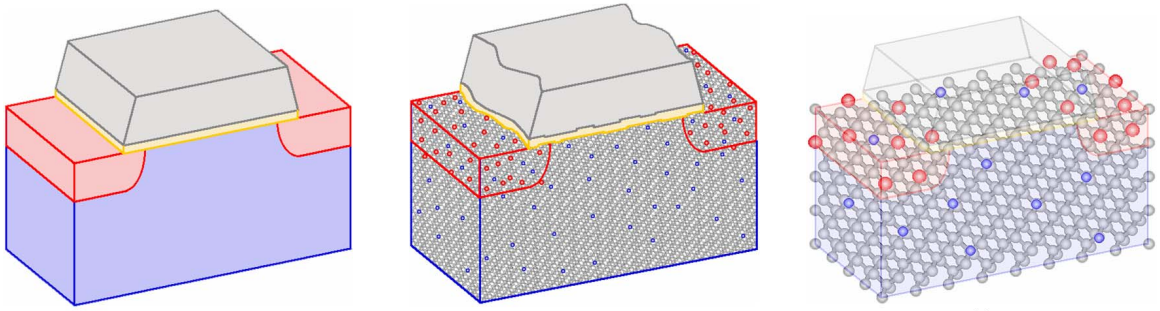


Figure 2.3: Cross-section of two transistors in a CMOS gate, fabricated in a $p-$ substrate process. The PMOS transistor, depicted on the left, includes an $n-$ type well. Typically, the higher the dopant concentration, the larger the conductivity of the material.



(a) (b) (c)

Figure 2.4: Illustration of the effect of feature scaling on a device. (a) The traditional device, featuring continuous ionised dopant charge and smooth boundaries and interfaces; (b) a 22nm MOSFET, featuring less than 50 silicon atoms along the length of the channel; (c) a 10nm MOSFET set for production in 2020, with only a handful of atoms along the channel's length. Figure sourced from [3].

2.3 Systematic Variability

As transistor scaling continued, increasingly smaller features had to be etched on a wafer, making it increasingly difficult to maintain precision in the manufacturing process. According to [29], systematic variations can be classified as *across-field* (position in photomask) and *layout dependent*.

2.3.1 Across-Field Variations

Circuits are manufactured using photolithography, or optical lithography. This is a process in many ways similar to photographic printing, where patterns that make up layers of a design are exposed on a silicon wafer, one layer at a time. *Across-field* variations are caused by precision limitations in the photolithographic etching process, such as dose, focus, exposure variations, lens aberrations, mask errors, and variations in etch loading [46, 47]. These variations are spatially correlated, however, and can be characterised through the use of test structures placed at strategic points across the reticle (photomask). Figure 2.5 illustrates how photolithography works: a photoresist layer, placed on top of the wafer, is exposed to UV light through a photomask; the areas of the mask which allow light to shine on the photomask will dissolve the photoresist, leaving the other areas untouched. Following from this process, etching can take place, and the design's features are chemically developed on the wafer. Disturbances across the photomask and during the several photolithography stages contribute toward across-field variations.

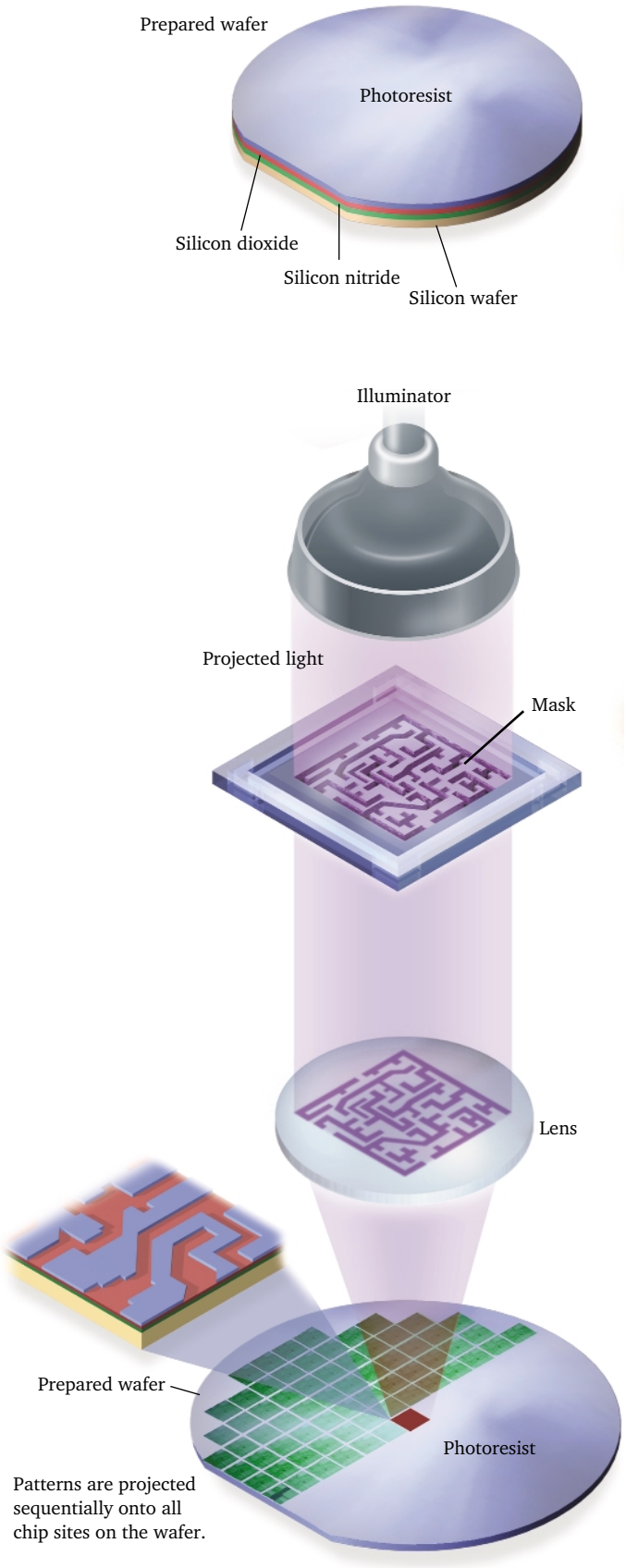


Figure 2.5: The process of photolithography, where light is passed through a photomask and a lens, and used to etch layout features on a photoresist. Figure sourced from [4].

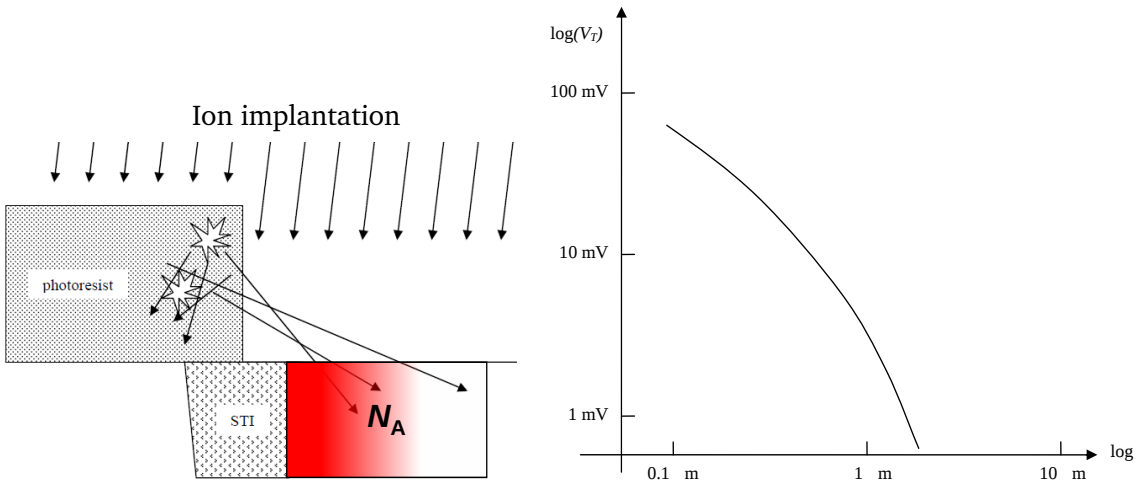


Figure 2.6: The Well-Proximity Effect, whereby implanted ions are non-uniformly distributed along the wells in which the nMOS transistors are formed. (a) the ion implantation and reflection mechanisms; (b) the V_T variation with respect to the distance between the well-edge and the gate-edge. Figure sourced from [5].

2.3.2 Layout-Dependent Variations

Layout dependent variations stem from different layout choices, such as having vertically- or horizontally-oriented gates, even when two different gates are placed next to each other [48]. As in the case of across-field variations, these can also be accounted for because they are deterministic in nature, and can be modelled as a function of layout structure and surrounding topological environment.

One very well known source of layout-dependent variations is the Well-Proximity Effect (WPE) [49]. In CMOS fabrication, areas where nMOS and pMOS transistors will be built receive separate ion implants, and each is covered with photoresist when the other is being implanted. When a transistor (either nMOS or pMOS) is located too close to the edge of the photoresist mask, some implanted ions can be reflected and get buried into its substrate, increasing the device's ion density along the edges of the photoresist, as Figure 2.6(a) illustrates. This ultimately causes undesired behaviour of the threshold voltage of the devices along the wafer.

Many other sources of variability, not just relating to lithography limitations but also material stress, whereby the compression or expansion of the silicon substrate – an effect that is influenced by the proximity of devices in the layout – causes changes in carrier mobility, ultimately resulting in drain current variations [50].

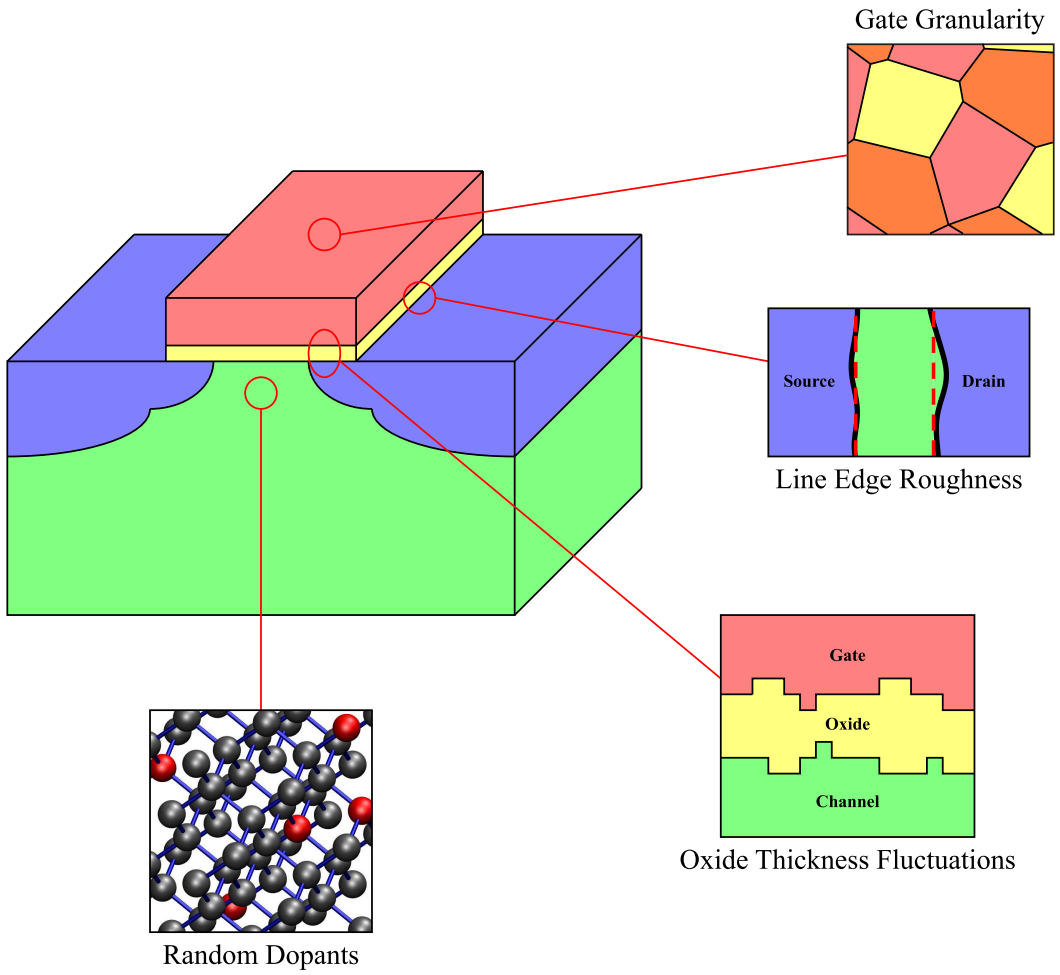


Figure 2.7: Illustration of some of the key sources of intrinsic variability on bulk MOSFETs. Figure sourced from [6].

2.4 Intrinsic Variability

Even though these manufacturing issues caused variations between devices, these would still fall within known boundaries and today are taken into account during the design process.

Another set of variations, however, is not as simple to deal with. These are typically referred to as *stochastic*, or random variations, and they focus on atomistic behaviours. As Figure 2.4 illustrated, the channel of a $10nm$ device will only be a few atoms long, and therefore considerably more sensitive to structural variations, such as the ones described in this section. Figure 2.7 provides an illustrated summary of the main sources of intrinsic variations.

2.4.1 Random Discrete Dopants

The creation of transistor models for performance estimation for a given technology size relied on continuous ionised dopant charge as well as smooth boundaries and interfaces, but as device scaling continues, the validity of these models starts to become less appropriate.

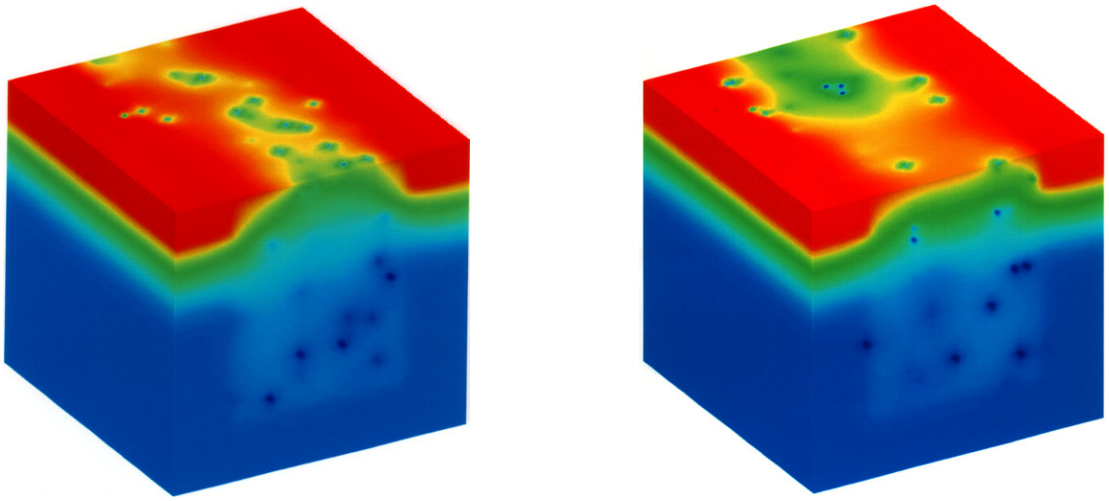
Performance models would use statistical averaging of dopant concentrations to calculate a device's electrical characteristics, since hundreds or thousands of dopant atoms would be present in the channel; the difference in dopant concentration between two devices would be negligible, and therefore the difference in their characteristics would also be negligible [43]. With only a handful of dopant atoms now present in the channel, as depicted in Figure 2.4(c), the behaviour of a device will depend upon the actual number of dopants present [7]. Not only is the *number* of dopant atoms a major contributor to electrical variations on the device, the *placement* of these atoms will also greatly affect its performance, as Figure 2.8 illustrates. In this case, two devices with the exact same number of dopant atoms, but different placement of these atoms, are compared. It is found that they exhibit a difference in threshold voltage above 20% [7], contributing toward potentially unstable and unreliable VLSI circuits.

Random Discrete Dopants (RDD) have been shown to contribute 60-65% of the total variability in measurements of $65nm$ and $45nm$ bulk silicon devices, although some other effects described in this section are likely to become the main sources of variability in future technology nodes [51].

2.4.2 Line-Edge Roughness

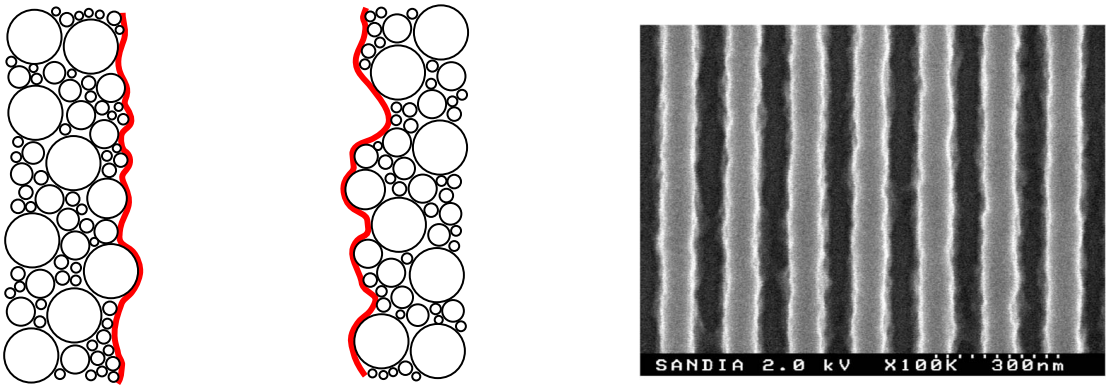
Line-edge roughness (LER) is defined as the deviation from the features outlined in the photomask. The discrete molecular nature of the photoresist causes imperfect lines to be etched on the wafer, resulting in a rough profile as depicted in Figure 2.9.

LER has always been a part of the manufacturing process, but has always been negligible when compared to the width of the feature being etched. Croon [52] has shown that LER does not significantly affect devices down to 80nm gate length, but is likely to become the dominant cause of intrinsic variability below this size [53]. This is largely due to the fact that almost all transistor parameters are a function of the gate length, and therefore LER variations will have a significant impact across the operation of the transistor, most notably in the form of threshold voltage fluctuations.



(a) (b)

Figure 2.8: Illustration of two different distributions of 130 dopant atoms in the channel depletion region, resulting from a 3D atomistic simulator. (a) MOSFET exhibiting a 0.97V threshold voltage; (b) MOSFET with 0.57V threshold voltage. Figure sourced from [7].



(a) (b)

Figure 2.9: Line-edge roughness (LER) caused by the discrete molecular nature of the photoresist. (a) shows an illustration of LER in a photoresist, with its molecules drawn as circles, as the developed edges drawn as red lines [6]; (b) shows an actual picture of LER in photoresist from Sandia Labs. Figure sourced from [3].

Additionally, other novel technologies such as silicon-on-insulator (SOI) [54] and FinFET [55] seem to also be susceptible to severe variations from LER.

2.4.3 Gate Granularity

The polycrystalline granular structure of both polysilicon and metal gates has also been identified as an important source of intrinsic parameter fluctuations.

Enhanced diffusion along the grain boundaries and localised penetration of dopants through the gate oxide into the channel from the high doping regions in the gate are potential sources

of variability. With the continuing scaling of the gate oxide thickness, it becomes easier for implanted ions to tunnel through the insulation layer, depositing in the channel substrate and forming traps which again cause fluctuations in the threshold voltage of the device [53].

2.4.4 Oxide-Thickness Fluctuations

Oxide-Thickness Fluctuations (OTF), sometimes referred to as Surface Roughness, is defined as the deviations from the intended thickness of layers. For instance, unevenly deposited oxide on the substrate can cause variations in the distance between the gate and the channel. This will translate to varying parasitic capacitances along the channel, resulting in variations in the threshold voltage of the device, not only in bulk MOSFETs but also in FinFET technology nodes [56].

2.4.5 Temporal Variations

The variations described so far have been directly related to device manufacturing, but they are not the only contributors to performance degradation. Another factor should be taken into account at design-time, and it is *temporal variability*. This is defined as any environmental or internal variation which can cause a device to suffer gradual performance degradation or even total breakdown, during its operational lifetime. The three main causes for device degradation over its lifetime are known as Negative Bias Temperature Instability (NBTI), hot carrier injection, and electromigration [57]. Recently, as high-κ metal gates were introduced, Positive Bias Temperature Instability (PBTI) started to have a non-negligible effect on voltage threshold variations in nMOS devices [58].

NBTI has a larger impact on pMOS devices, due to the negative voltage usually present at the gate, and it has the undesired effect of increasing V_T of a transistor, both temporarily and permanently [59]. This phenomenon occurs when pMOS devices are negatively gate-biased at high temperatures (stress phase), causing hydrogen-silicon bonds to break at the interface between the gate oxide and the substrate, as illustrated in Figure 2.10. Once the stress is eliminated, most of the floating hydrogen bonds with the dangling silicon once again. However, Some hydrogen bonds together to form H_2, permanently breaking the pre-existing Si-H bonds, resulting in permanent changes to the device's threshold voltage [8]. Conversely, **PBTI** affects positively-biased nMOS devices in a similar fashion, but their effect on threshold voltage variations seems to decrease as high-κ metal gate technology scales down [60].

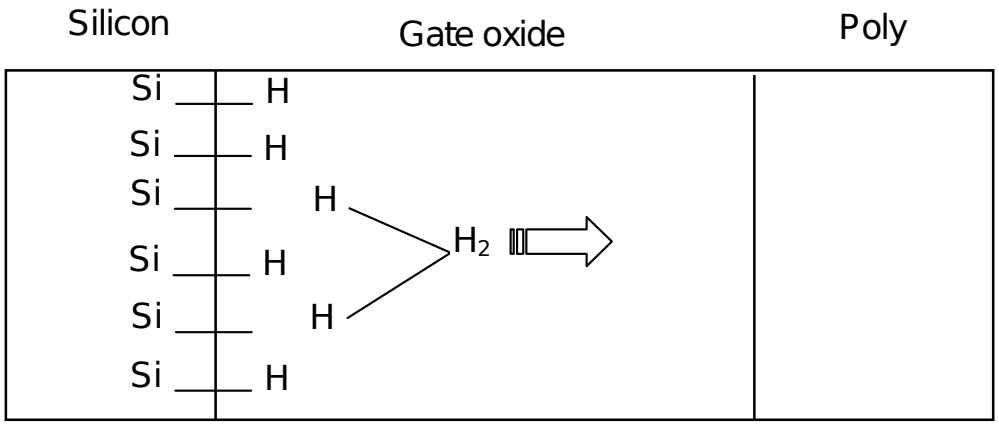


Figure 2.10: Broken Si-H bonds at the Si-SiO_2 interface, resulting from stress phases. Some H_2 is generated from the generated traps, resulting in permanent changes to the device's threshold voltage. Figure sourced from [8].

Hot carrier injection happens when a carrier (electrons in the case of an nMOS, holes in a pMOS) travels along the channel with a little more energy than usual, and escapes the channel into the insulating oxide [61]. It then becomes trapped in this layer, and builds up the charge in the gate, ultimately resulting in an increase in V_T. This results in a permanent change to the transistor's performance.

Electromigration occurs when electrons knock metal atoms loose and cause them to ride along with the electron flow, causing serious effects in unidirectional channels by means of metal build-up in the downstream, and metal depletion in the upstream. In some critical cases, the upstream metal connection might be severed altogether [61]. This effect occurs in high current density channels, and is also an irreversible change.

These phenomena are the leading causes of this temporal performance variation, which is commonly known as **ageing**.

2.4.6 Interconnect Variability

The complexity of the connections between elements also increases with device scaling. Many of the sources of variability in transistor manufacturing will also affect metal interconnect, such as LER and electromigration, introducing varying parasitic capacitances along tracks, and ultimately resulting in random variations in timing [62].

Modelling these effects, and taking them into account in the design process is a big challenge, due to the extreme connection density in VLSI systems and the computational cost that comes with trying to characterise these random variations [63].

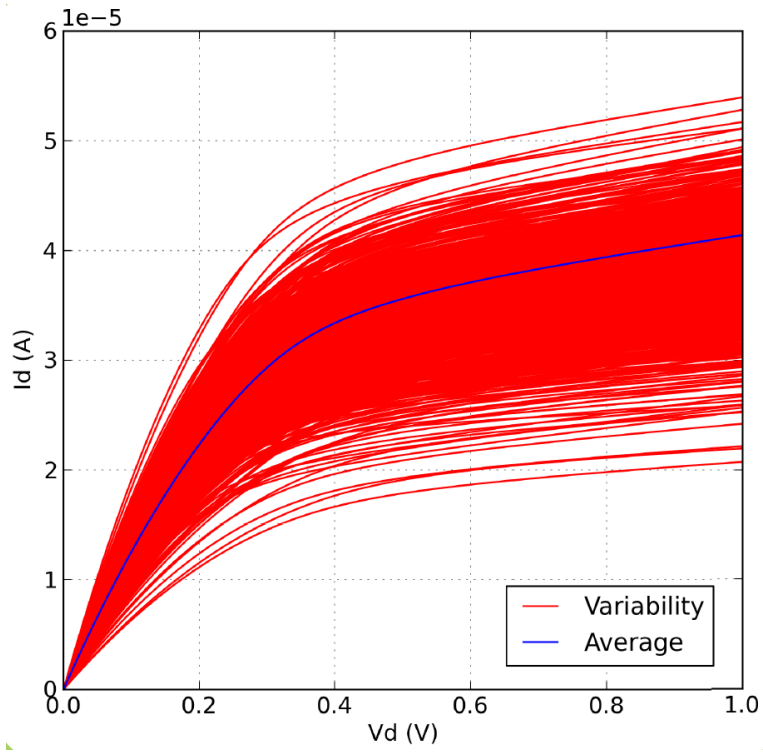
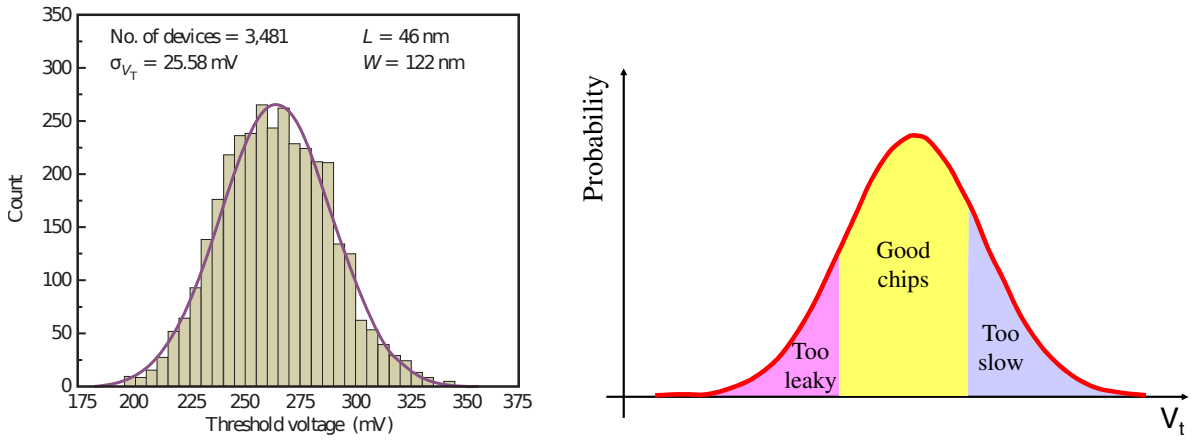


Figure 2.11: The characteristic $I - V$ curve of transistors plotted for a large number of modelled devices. Each of the 1000 red lines depicts the behaviour of one transistor, and the blue line represents the average. Figure taken from [9].

2.5 Impact of Variability

As the previous section has shown, intrinsic variability arises from many different sources, each providing their own contribution to the variation of the electrical characteristics of a transistor. Figure 2.11 illustrates this point, as one the ultimate consequences of variability is a large change in circuit behaviour.

One of the major contributors to performance variation is now the threshold voltage (V_T), mostly due to the reduction in power supply voltage brought on by the constant push to reduce power consumption [64]. A variation in V_T will translate to a variation in device delay, since the transistor only starts conducting current between the source and drain terminals at gate voltages larger than V_T. Figure 2.12(a) illustrates this variation measured with $90nm$ technology nodes. Variations in delay will then cause timing errors, and in extreme cases can cause complete device failure. Figure 2.12(b) highlights the consequence of V_T variation at the circuit level: some chips might become too slow, and some might consume too much static power.



(a) (b)

Figure 2.12: Illustration of V_T variation in CMOS technology. (a) variation in a $90nm$ process, with a sample of 3481 devices [10]; (b) the implications of a varying V_T for chip performance. Figure sourced from [11].

It will also translate to a considerable increase in leakage current [65]. For instance, a reduction in threshold voltage of about 85mV will translate to an increase in sub-threshold leakage current of about 10X [66]. This is undesirable behaviour, as CMOS technology is adopted by designers largely due to its energy efficiency, since most of its power consumption occurs at switching time.

In some cases, intrinsic variability can lead to timing skews which in turn cause circuit failures and yield loss at low voltages [36]. In addition to this, threshold voltage variations can also accelerate device ageing, resulting in a sizeable reduction of the mean time to failure (MTTF) of processors [67, 68]. Figure 2.13 illustrates the impact of ageing on the static noise

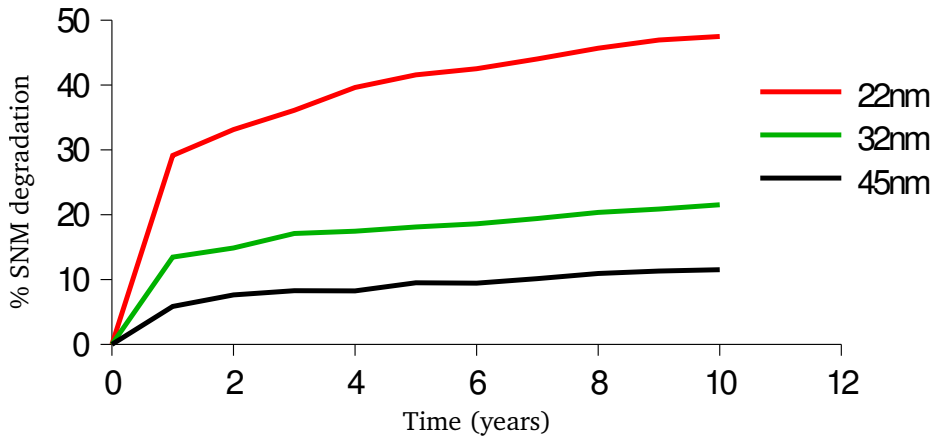


Figure 2.13: Percentual reduction in static noise margin (SNM) of an SRAM cell due to ageing, becoming more pronounced at more advanced technology nodes. Figure sourced from [12].

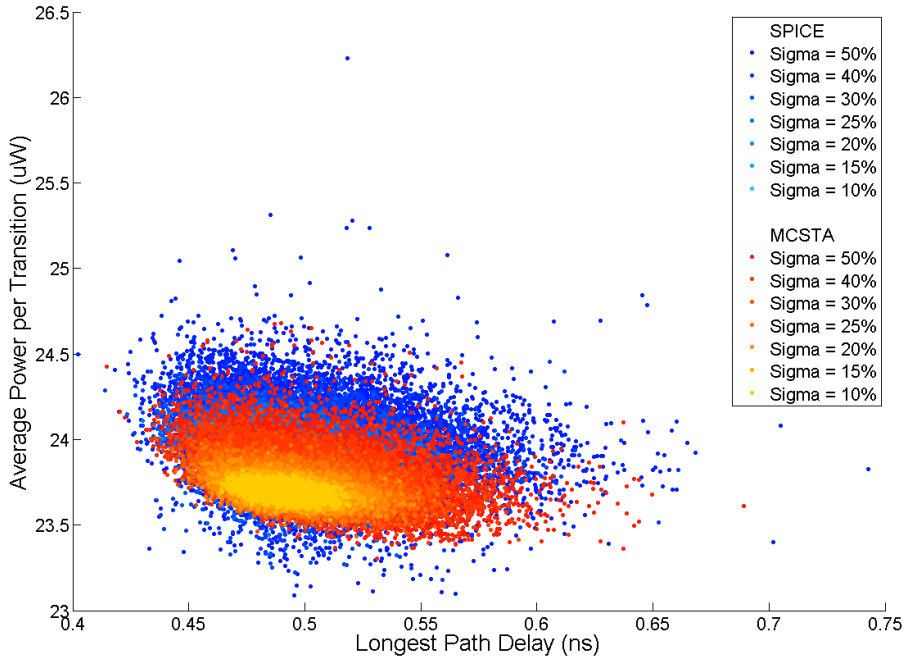


Figure 2.14: The impact of intrinsic variability on dynamic power consumption and delay, measured for various Monte-Carlo runs for different sigma values, using SPICE and Static Timing Analysis (STA). Figure sourced from [13].

margin (SNM) of an SRAM cell [13]. It is clear that the effects of circuit ageing become more pronounced as technology scaling continues.

The impact of variability on deep sub-micron designs has been thoroughly studied mainly in simulation, analysing its impact on both general circuit properties, such as on threshold voltage variation [7, 36], path delay [69] and yield [70], as well as particular cases of CMOS design, such as SRAM lifetime [71] and flip-flop timing [72]. Figure 2.14 illustrates the impact of variability on both power and delay of a one-bit full adder, measured through full simulations carried out for different sigma values, using Monte-Carlo SPICE simulations as well as Monte-Carlo static timing analysis [13].

Across all studies, one conclusion is always present: on deep sub-micron designs, intra-die physical variations become the leading cause for chip variability. Worst-case design procedures generate circuits that do not make the most of the fabric's features, considering that the worst- and best-case circuits are very far apart due to the effects of variability.

2.6 Summary

Random, or stochastic variability is an issue that has been become increasingly significant with the device scaling that the industry has been pushing for in the last 50 years.

Even though systematic variability has been well understood and incorporated into the design process, intrinsic variability still poses a challenge to both designers and manufacturers.

Solving this issue at the transistor level seems to be a task which requires a breakthrough in technology, since normal scaling of manufacturing process is almost certain to break down at the atomistic level. Other approaches have been suggested which attempt to accept random variations as a reality of device manufacturing, and choose to mitigate variability at the upper abstraction layers, all the way up to system-level, such as SSTA techniques applied to FPGAs. The next chapter will provide some examples of solutions that have been proposed for reducing the impact of device variability.

Chapter 3

Variability Mitigation in Circuit Design

Contents

3.1 Introduction

In order to continue progress as specified by Moore's Law, and with the relentless technology scaling taking manufacturing into the domain of atomistic devices, the uncertainty that is brought on by the limitations of the fabrication process can no longer be avoided or ignored, and make the life of a circuit designer more complex. Small atomistic variations between devices can cause asymmetries in performance, or even a full chip-level breakdown. The

implications of variability are not limited to designers any more; a considerable impact on yield now makes intrinsic variability a problem with a very real and serious economic impact [37, 64].

Dealing with this issue at the transistor level is proving to be increasingly difficult, and the quantum-effects that begin to surface with atomistic devices seem to be a barrier which cannot easily be surpassed from a manufacturing point-of-view [10].

Variability-aware design has therefore become an important avenue for future technologies. This design philosophy does away with the idea of homogeneous transistor behaviour across a circuit, and instead attempts to characterise the variations between its basic elements, and incorporate this variation into the design process, ultimately resulting in an increase in circuit reliability [73, 74].

Chapter 2 introduced the effects of variability at the transistor-level, whereas this chapter will focus mostly on how these effects can be mitigated at the circuit-level, since transistor-level approaches face the quantum-effects barrier.

This chapter provides some examples of variability-aware methodologies along the design process of a circuit, from pre-fabrication modelling to post-fabrication adjustments. Given their inherent reconfigurability, and therefore hardware redundancy useful for post-fabrication design modifications, reconfigurable hardware platforms are described in detail, along with some typical variability characterisation techniques that they allow for. The flow-chart in Figure 3.1 illustrates the stages during which these techniques can be applied.

3.2 Pre-Fabrication Approaches

The first opportunity to tackle variability exists before the a circuit is fabricated. Some techniques are available, as this section will show, which allow designers to evaluate their circuits and estimate how variations can affect their performance.

3.2.1 Variability-Aware Device Modelling

In order to estimate the performance of a particular design, a model is required which can solve the equations that describe its electrical behaviour. This section discusses circuit modelling, the inclusion of stochastic variability effects, and the application of these models for variability-aware design.

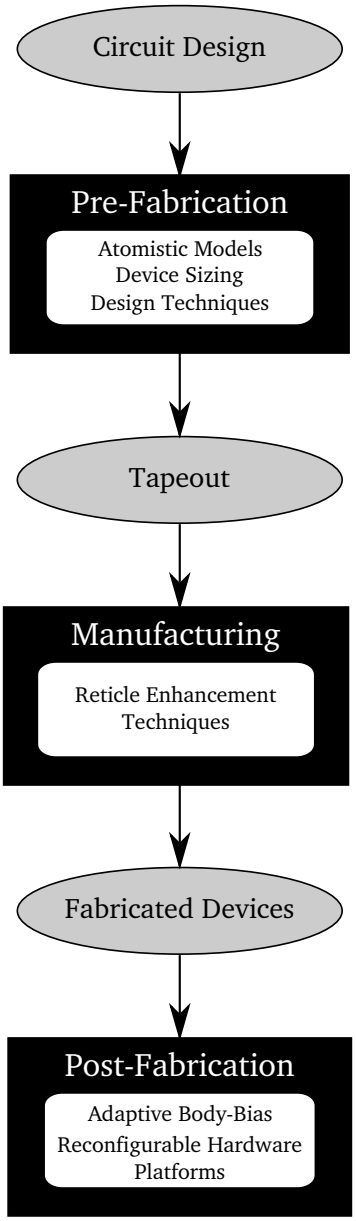


Figure 3.1: The various approaches to process variability mitigation, applied at different stages in the life-cycle of a design, from when it is elaborated and studied, through its fabrication, and down to the post-fabrication adjustments that its architecture might allow.

Since its release by Laurence Nagel in 1973, the Simulation Program with Integrated Circuit Emphasis, or **SPICE**, has been widely used by circuit designers all over the world [75]. In its current version, the tool includes small- and large-signal models of basic electronic components, including transistors. It takes in a text file – a *netlist* – which instantiates all the components that make up a circuit, as well as their connections. Using device analytical models, it finds the DC operating point of the circuit, and then performs a small-signal transient evaluation which characterises deviations from the calculated operating point.

Given its special standing as the most widely used device in commercial products, the field-effect transistor has been the subject of many modelling efforts. The first SPICE version included the Shichman & Hodges large-signal model [76], solved using the formula shown in Equation 3.1, where μ_n is the surface mobility of an n-channel device, W_{eff} and L_{eff} are the effective channel width and length, respectively, and C_{ox} is the capacitance per unit area of the gate oxide.

$$i_D = \frac{\mu_o C_{ox} W_{eff}}{L_{eff}}[(v_{GS} - V_T) - \frac{v_{DS}}{2}]v_{DS} \tag{3.1}$$

The small-signal model can then be derived from this equation, but it is only suitable for long channels, *i.e.* $L_{eff} > 10\mu m$ [73]. As technology scaled down, the accuracy of these models was significantly reduced, which led to the creation of another set of models by the BSIM (Berkeley Short-channel IGFET Model) Group, taking into account narrow- and short-channel effects, parasitic resistances, hot-electron effects, and many other physical phenomena that were beginning to play a major part in the behaviour of sub-micron devices. The BSIM3v3 became the first industry-wide standard of its kind, allowing for accurate transistor modelling down to $150nm$ [77]. The BSIM4 model built upon the BSIM3v3, by adding features such as more accurate intrinsic input resistance for both RF, high-frequency analog and high-speed digital applications, a more accurate channel thermal noise model and a noise partition model for the induced gate noise, among many others [78], making the models suitable for transistor sizes ranging from $130nm$ down to $20nm$. The BSIM4 models were used for the modelling efforts described in Chapter 4.

From the many available distributions of SPICE, *ngspice* [79] was chosen as one of the modelling tools used in this work due to its open-source nature and compatibility with the atomistic variability-enhanced tool, described next in this section.

The introduction of intrinsic variability modelling to SPICE-based circuit designs has been pioneered by the Device Modelling Group (DMG) at the University of Glasgow, who have developed a density-gradient-enhanced 3D drift-diffusion simulator capable of generating BSIM models which can be used by a SPICE tool to create variability-aware circuits. This simulator creates 3D transistor models, which incorporate the effects of RDD, LER, OTF and gate-granularity (described in Chapter 2) to generate sets of I-V curves, each with a shape variation which reflects these effects. The group then created a spin-off company

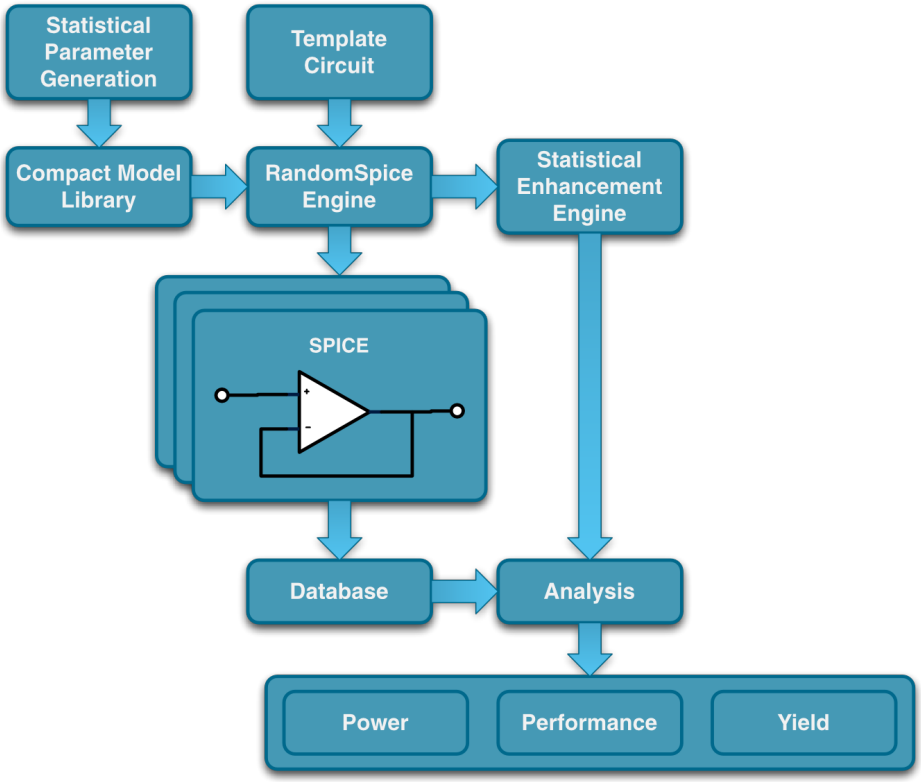


Figure 3.2: The operation of RandomSPICE: BSIM4 models are generated in with the atomistic simulator, which are then used on a template netlist, and run using a SPICE back-end. A database of virtually fabricated devices can then be analysed for variability-awareness purposes. Figure sourced from [14].

by the name of Gold Standard Simulations (GSS), which now is distributing the tool as a commercial product [80].

The operation of the simulator is described in further detail in [81]. A set of GSS tools have been used to generate the compact models used in this work. First of all, GARAND is used to perform the 3D atomistic simulations which are fed to Mystic, a compact model extraction tool, which generates a library of transistor models which include realistic effects of variability. Finally, RandomSPICE is used to replace standard SPICE transistor models with the GARAND+Mystic set, using Monte-Carlo sampling methods [14]. The creation of these models was calibrated through the use of a set of $35nm$ Toshiba devices [82].

In a nutshell, RandomSPICE takes in a normal SPICE netlist and replaces the standard BSIM4 models with the statistically-generated and variability-aware ones through Monte-Carlo sampling methods, each representing a single manufactured device. This process is repeated as many times as the user defines, generating a set of netlists, each representing a virtually manufactured instance of the circuit defined in the original netlist. Its statistically-

enhanced features allow the user to perform rare-event simulations, useful to study manufacturing yield. Figure 3.2 illustrates the operation of the RandomSPICE tool.

Although the atomistic simulator uses techniques to speed up the BSIM4 model extraction process, it still requires a large amount of computation to be performed. As an example, the BSIM4 variability-aware libraries used for this work required a total of 60 CPU years of computation to create [6].

This tool allows for the study of rare events that can occur from stochastic variations, and therefore allow the manufacturer to perform high-sigma power, performance and yield analysis on their technology. This ultimately led to GSS recently signing a contract with one of the worlds largest foundries, GlobalFoundries, for its complete variability-enhanced Electronic Design Automation (EDA) tool [83]. Variability-aware modelling is a useful tool, endorsed by the industry, which can help study the impact of variability at a pre-manufacturing stage.

Using these libraries, a study on variability-tolerant standard cell-design was reported in [73], and it was found that by allowing optimisation techniques such as Evolutionary Algorithms (described in detail in Section 6.3.2) to explore different transistor sizings, variability-tolerant libraries can be created.

As the number of components in a modern electronic device increases, so does the number of Monte-Carlo based simulations required to ensure device reliability, in what is called *high-sigma* design. As an example, achieving 6-sigma yield would mean that only 1 in 506,797,346 circuits would be allowed to fail. Determining whether or not a circuit achieves 6-sigma yield would mean that over 500 million simulations would need to be carried out. However, efforts have already been allocated to achieve high-sigma design with a reduction in the number of simulations required [84, 85]. Even so, as the complexity of devices increases, so will the required number of Monte-Carlo simulations to verify reliability, posing a considerable challenge for software-based solutions and for pre-fabrication modelling in general.

3.2.2 Configurable Analogue Transistors

Another approach to variability-aware circuit design has been carried out by Peter Wilson and colleagues at the University of Southampton, with the inclusion of Configurable Analogue Transistors (CATs) at strategic points in an analogue circuit [16] so as to maximise the reliability of the fabricated designs.

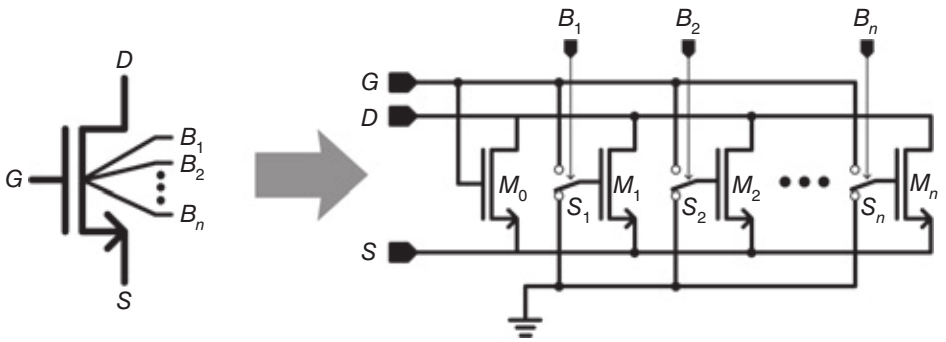


Figure 3.3: Schematic of a Configurable Analogue Transistor. Adding to the gate (G), drain (D) and source (S) common terminals, a configuration word of length n controls the number of adjustment devices connected in parallel with M_0, ultimately determining the width of the CAT. Figure sourced from [15].

The structure of a CAT is illustrated in Figure 3.3. A CAT consists of a main device, represented as M_0, and a series of n adjustment devices. The geometry of these adjustment devices depend on the context in which the CAT is being used, but typically they share the length of M_0 and their widths decrease incrementally by a factor of two.

Figure 3.4 illustrates the supporting methodology behind CAT-based design, and identifies the tools that it requires. A circuit schematic is the primary input to the design flow, and a Monte-Carlo analysis is carried out, which identifies the transistors which have a larger impact on yield, *i.e.* those on which process variations translate to a larger increase in the standard deviation of the performance of the device, and these transistors are replaced by CATs. There is a trade-off between yield improvement and the overhead introduced by CATs, and [16] has reported that only a small number of these need to be introduced to provide significant yield improvement.

An algorithm, described in detail in [17] is then used to calculate the optimal sizing for the transistors, along with the length n of the configuration word B, which controls the switches located at the gate of each adjustment transistor M_n. This sizing algorithm takes into account the expected process spread, and makes the number of adjustment transistors large enough to provide granularity to tackle it. In the work reported in [17], 3 adjustment transistors were considered.

Once the sizes of every CAT are determined, as well as the adjustment transistors for each, the design can be sent out for fabrication. The third and final stage of this approach involves configuring the CATs so that optimal performance is achieved. Although exhaustive search is typically used to find the optimal configuration, more refined and efficient optimisation algorithms can potentially be applied [16].

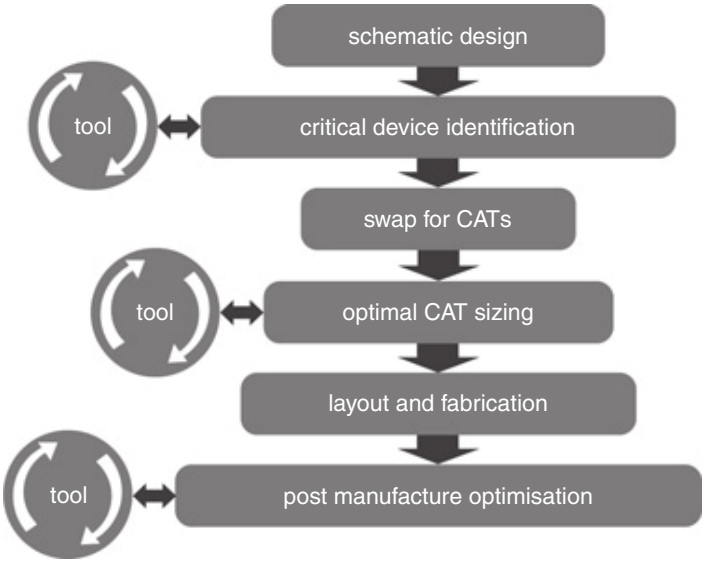


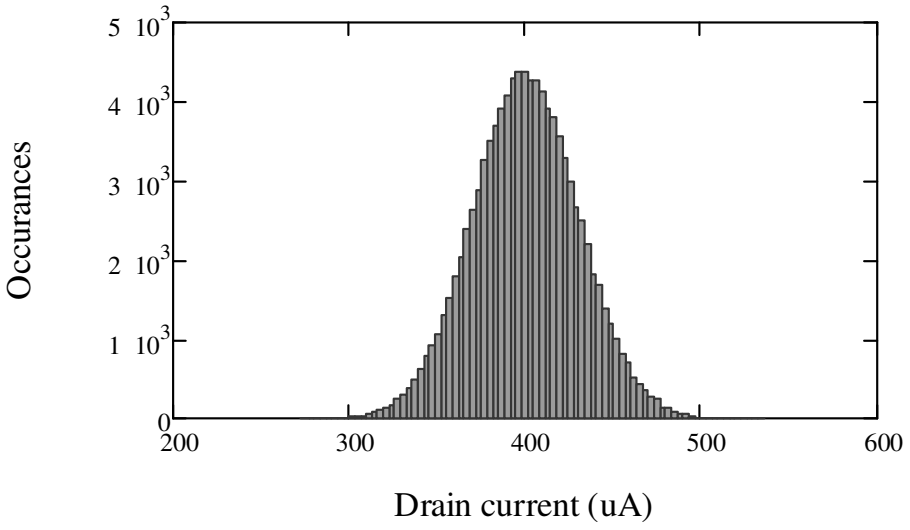
Figure 3.4: CAT methodology used in CAT-based design. Figure sourced from [16].

As an example, Figure 3.5(a) illustrates the distribution of 100,000 simulated $40\mu m$ wide nMOS devices, reporting an average drain current (I_D) of $400\mu A$ and a standard deviation of $30\mu A$. In order to reduce the impact of variations, as reported in [17], the standard transistor is replaced by a CAT, and three adjustment devices are added, with widths of 1, 2 and $4\mu m$. The flexibility introduced by these devices makes it possible to reshape the distribution as Figure 3.5(b) illustrates, reducing the standard deviation of I_D from $30\mu A$ to $10.11\mu A$, which can contribute toward a significant yield improvement.

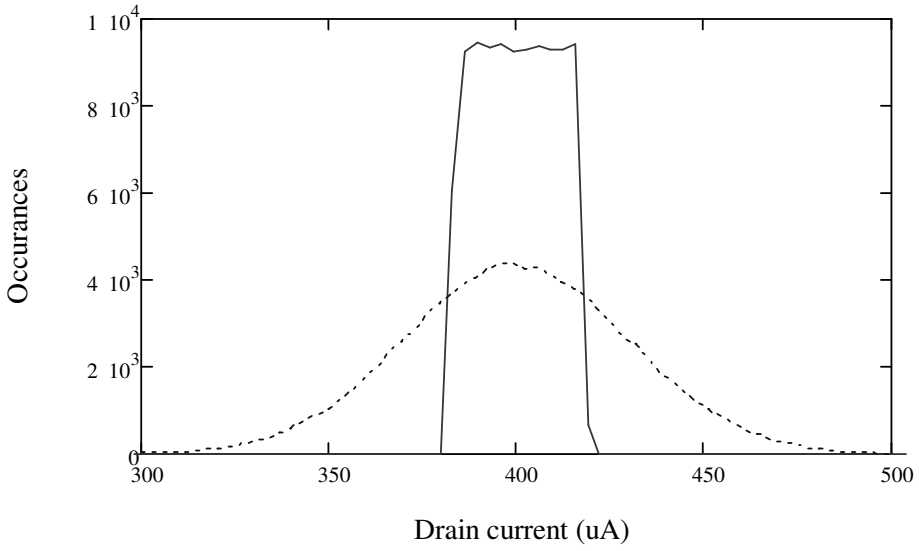
Although the final CAT configuration will be applied after the device is fabricated, a considerable amount of modelling effort is required at a pre-fabrication stage, hence their inclusion at this point in the chapter. Other similar approaches, such as FPTAs and PAnDA, described in Section 3.4, require less design-dependent modelling and are more general-purpose, and therefore are considered to be post-fabrication approaches.

3.2.3 Statistical Static Timing Analysis

Static timing analysis is a method which allows a designer to calculate the expected timing features of a digital circuit without requiring a costly complete simulation of the design [86, 87]. These tools allowed circuit designers to evaluate the performance of their design before manufacturing, helping to determine its maximum operating frequency, and potentially detect timing violations.



(a)



(b)

Figure 3.5: Distribution of the drain current of 100,000 simulated $40\mu m$ wide nMOS devices (a) before CAT introduction; (b) after CAT introduction with 3 adjustment devices of widths of 1, 2 and $4\mu m$. The dotted line in (b) represents the original distribution. Figure sourced from [17].

As technology scaled down, STA tools began to face the challenge of variability [88, 89], with gate and net delays being skewed from their nominal values.

In an effort to address the effects of variability on path delay, the concept of **Statistical Static Timing Analysis** (SSTA) was developed, which assumed a statistical distribution of delays for a particular gate or net, along with correlations between these [90, 91], as Figure 3.6 illustrates. Traditional STA analyses corner cases, in which the delay of each element in the design is assumed to be the worst-case possible. However, the likelihood of

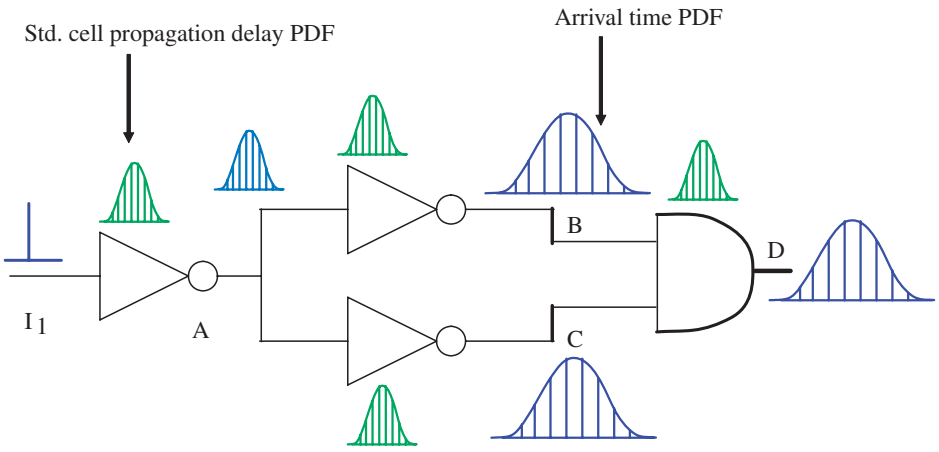


Figure 3.6: A probabilistic timing graph for a circuit, with gate (green) and interconnect (blue) delays represented as probability distribution functions (PDF). Figure sourced from [18].

this happening is very low, and therefore STA tools tend to generate overly conservative designs [92]. SSTA methods can be classified as either *path-based* or *block-based*.

A *path-based* approach involves analysing each path individually, taking into account the varying slew-rates along the path. Given the complexity of most modern VLSI designs, containing a very large number of distinct paths, only a small set can effectively be analysed, since each evaluation requires a large amount of computation. This makes path-based analysis a less popular approach for manufacturers and circuit designers [92]. After the delay distribution of each path is analysed, a statistical maximum is applied to all paths to determine the overall circuit delay distribution.

Block-based analysis assumes the worst case delay for multiple inputs of a given gate, and computes the delay between them using statistical *max* operations [18, 93]. With this method, only two delays are propagated from each gate (rise- and fall-times) making its runtime linearly dependent of circuit size. This approach lacks the accuracy provided by path-based analysis, but it generally speeds up the timing evaluation of a design. Figure 3.7 illustrates the operation of both of these approaches.

A full overview of the techniques used in SSTA can be found in [94].

3.3 Manufacturing Approaches

With the continuing shrinking of devices, a fundamental problem was faced by the manufacturing industry well before intrinsic variability became a first-order effect. As the required feature sizes decreased, so did the wavelength of the light required to etch them. The actual

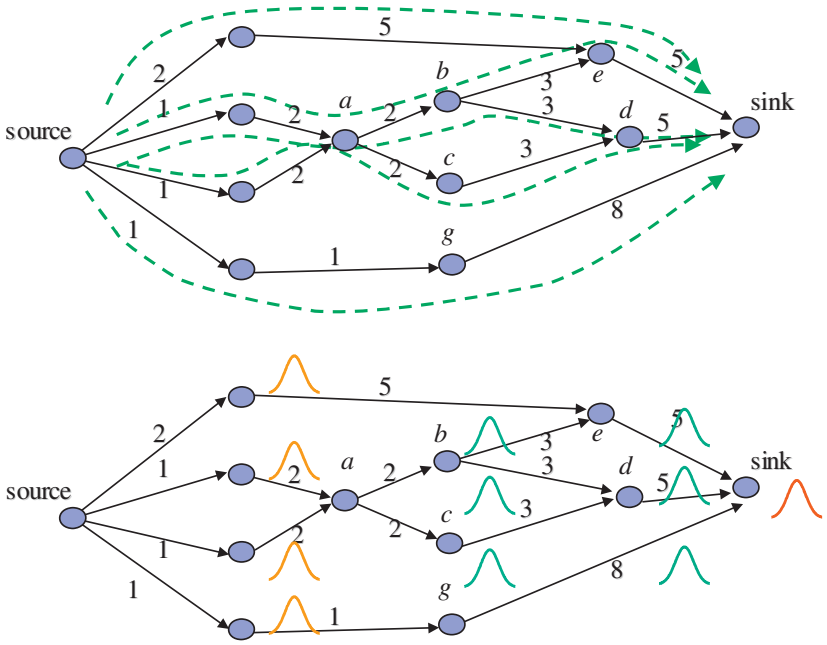


Figure 3.7: The path-based SSTA (top) and block-based (bottom). The former analyses the n most critical paths of the circuit, whereas the latter makes extensive use of the statistical *max* operation in the delay estimation to speed up computation. Figure sourced from [18]

wavelength has been reduced from $365nm$ in the 1980s down to the $193nm$ lithography process used today. The wavelength scaling could not continue beyond the $193nm$ mark without a drastic redesign of the photolithography process, since shorter wavelengths are absorbed by the quartz lenses used to direct the light which etched layout features on the photoresist, as described in Section 2.3.2.

In an effort to maximise the life-cycle of the $193nm$ photolithography equipment used by every manufacturer worldwide, **Resolution Enhancement Technologies** (RETs), were developed.

Some of these techniques are designed to allow for smaller features to be etched on the photoresist, even if the wavelength of the incident light is longer than the feature, such as Phase-Shift Mask (PSM) [95] and Multi-Patterning [96]. Since they are not directly related to variability mitigation, they are not covered here.

One RET, however, does have an effect on layout variation, and it is commonly incorporated into most EDA tools. It attempts to deal with the distortion introduced by some of the other RETs applied to feature sizes below the light wavelength. This is known as **Optical Proximity Correction** (OPC) [97]. Due to the inherent limitations of the photolithography process, the geometry of the features created in the layout don't always correspond to the etched features resulting in a manufactured device, as the left side of Figure 3.8 illustrates.

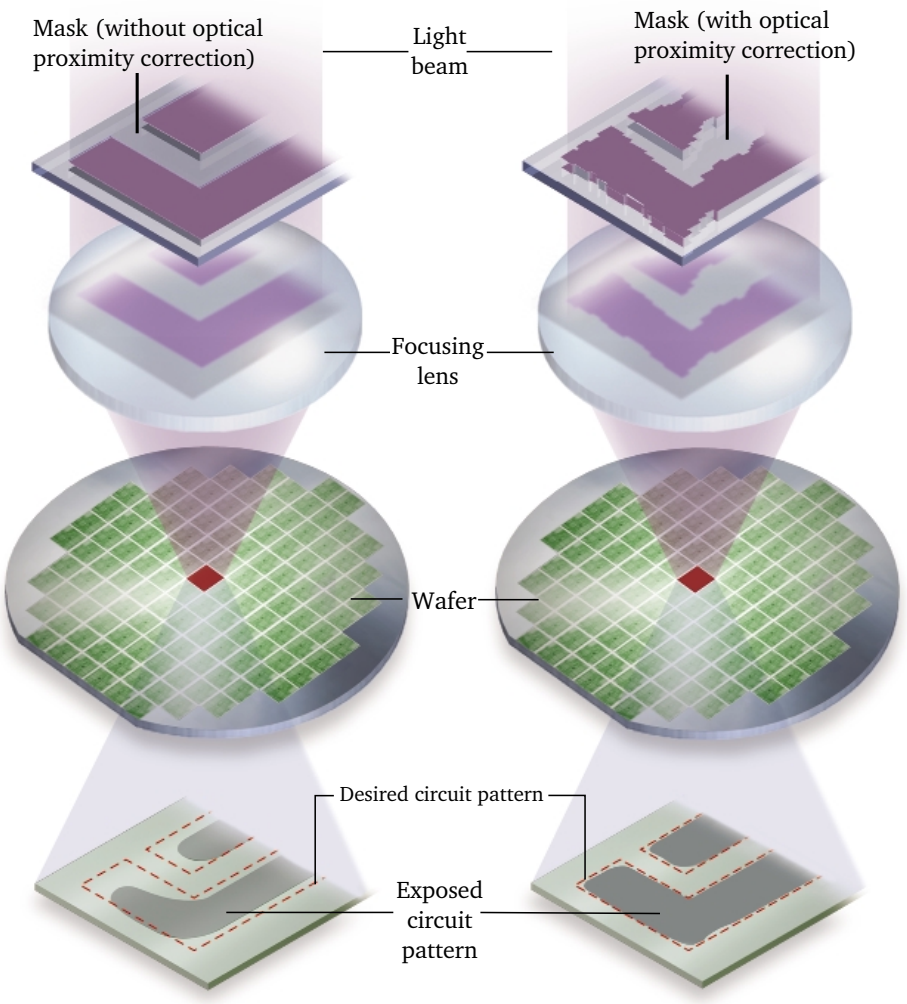


Figure 3.8: The variations in layout feature geometry introduced by photolithography limitations (on the left) and the same variations corrected with OPC (on the right). Figure sourced from [4].

Without OPC, the projected light ends up etching a feature with geometric irregularities with respect to the intended design, such as rounded corners or narrower trace widths [98]. This technique analyses the layout, anticipates potential light interferences, and alters the photomask in such a way that these inconsistencies can be avoided, as the right side of Figure 3.8 shows.

Using these techniques, systematic variability can be incorporated in the EDA tool flow, and kept within known boundaries [39]. This is typically known as Design for Manufacturability (DFM) [99].

3.4 Post-Fabrication Approaches

Up until this point, this chapter has described techniques which attempt to *avoid* manufacturing circuits with variability. This section introduces the concept of designs which embrace the inevitability of random parameter variation in their platforms. One of the most promising avenues in variability-tolerant design is created by *reconfigurable hardware platforms*, which allow for post-fabrication adjustments to the substrate and a potential compensation for these variations.

3.4.1 Adaptive Body-Bias

A technique which has received some attention in the past decade is called *Adaptive Body Bias* (ABB), it involves employing a non-zero body-to-source bias to modulate the threshold voltage of a transistor. There are essentially two ways of performing body-biasing: Forward Body-Biasing (FBB) and Reverse Body-Biasing (RBB) [100]. Typically, the body terminal of a FET device is connected to the same voltage level as the source terminal. By severing this connection and applying an independent voltage to the body terminal, the electrical behaviour of the transistor can be affected. Because of this property, the body terminal is sometimes referred to as the "back-gate" of the transistor.

To explain RBB, we can take the example of an nMOS device. If a negative body-to-source voltage is applied, then holes will start moving from the channel region toward the body terminal, leaving behind electrons which will increase its negative charge, and ultimately increase the threshold voltage V_T. FBB is achieved by applying a positive body-to-source voltage, which has the opposite effect of RBB, and will result in a decrease of V_T.

By changing V_T and keeping all other parameters equal, the performance of a circuit will be affected. An increased V_T will cause devices to take longer to switch on and to reduce leakage power, whereas a reduction of V_T will make the circuit quicker to react but with an increase in power leakage.

The *adaptive* part of this method is employed when performance measurements can be taken to decide whether a die should be Forward- or Reverse-Biased. By applying independent body-biasing to pMOS and nMOS devices in a CMOS circuit, it becomes possible to perform multi-objective optimisation for performance and power leakage [101]. This technique has been demonstrated to have significant results for inter-die and even intra-die variability

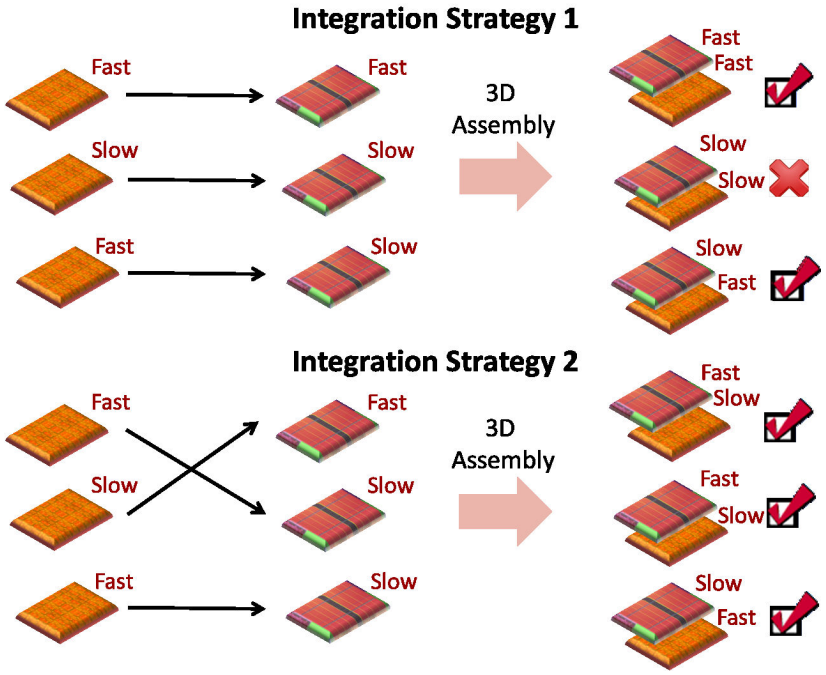


Figure 3.9: 3D integration using Variation-aware Die Matching, combining slow layers with fast ones, resulting in each 3D chip meeting performance requirements. Figure sourced from [19].

[100, 102]. Body-bias islands can also be created to provide fine-grained control of body-bias voltage across different areas of the chip, as reported in [103].

3.4.2 3D Stacking

3D designs have become increasingly popular over the last few years, being found in some of products such as processor stacks, phone memories, and even flash cards [104]. This design philosophy moves away from a simple planar 2D IC to allow several of these to be stacked in an additional manufacturing process.

What this means for the purposes of variability mitigation is that each fabrication 2D layer can be individually evaluated, and techniques such as *Variation-aware Die Matching* [105] can be applied, whereby the dies that make up a 3D stack can be combined so as to average out the performance across 3D chips, guaranteeing that at least one of the layers operates at the nominal frequency – assuming that each layer operates in a different clock domain –, as Figure 3.9 illustrates.

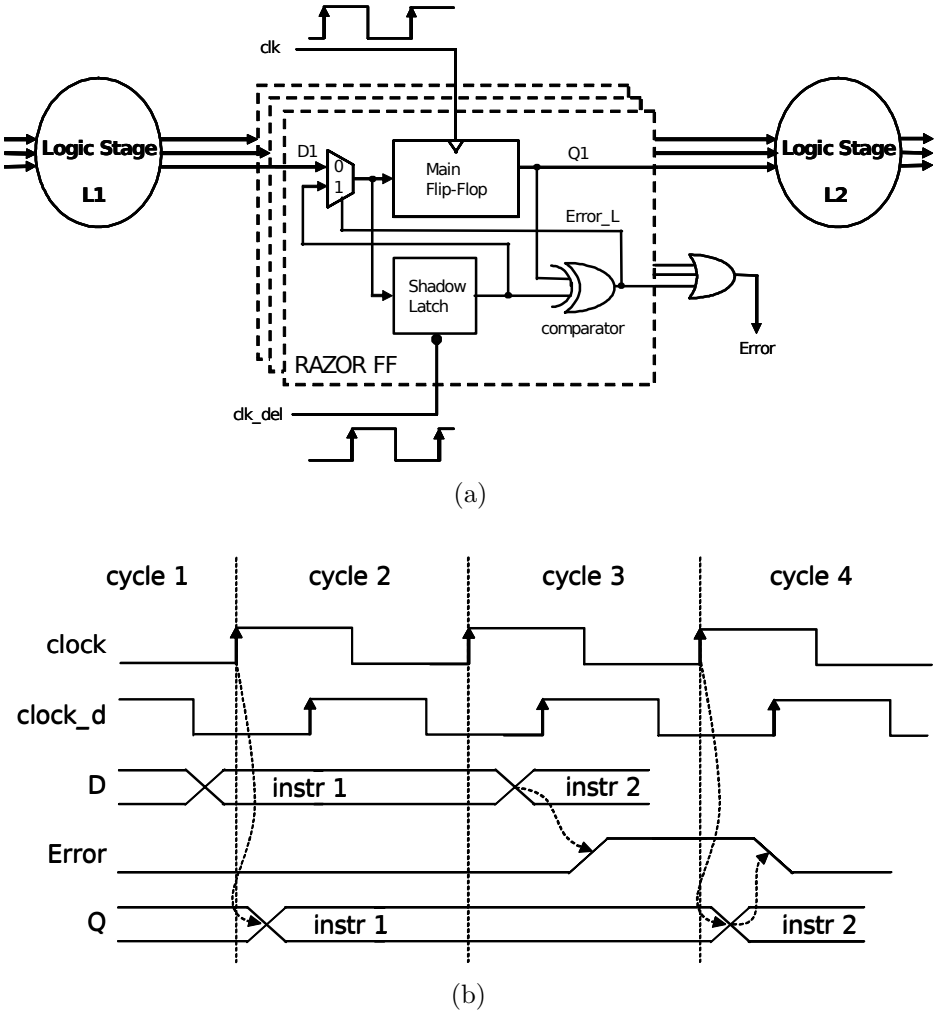


Figure 3.10: The Razor pipeline operation. (a) the internal structure of a Razor flip-flop; (b) the error detection mechanism being activated due to a mismatch between the outputs of the main and shadow flip-flops. Figure sourced from [20].

3.4.3 Razor

One of the most successful approaches to variability-aware design has been developed at the University of Michigan, is referred to as **Razor**, and it involves the inclusion of a custom-designed flip-flop in a series of logic paths in a pipelined design, which are used to detect timing errors. This information is fed back to voltage scaling mechanisms which adjust the power supply in an attempt to suppress these errors [20].

As mentioned in Section 2.5, one of the major effects of process variations is a shift in a device's threshold voltage, which in turn translates to timing variations. The Razor approach attempts to deal with this by controlling the supply voltage as necessary to compensate for these potentially damaging variations.

Figure 3.10(a) illustrates the structure of a Razor flip-flop which is included in the design as an additional pipeline stage, and Figure 3.10(b) shows a timing diagram of the signals involved in this pipeline stage. The shadow latch illustrated in the structural diagram is controlled by a clock signal which is slightly delayed with respect to the main flip-flop, and therefore they both sample the same data with a slight timing offset. If any changes occur during the period defined by the difference in the edges of both clocks, a comparator will generate a flag which represents the occurrence of an error. Feeding this information back to a voltage controller, the supply voltage can be increased until the error disappears. This also allows for efficient power use, as the supply voltage being applied to the circuit will provide only the power strictly necessary for the operation of that particular design.

3.4.4 Reconfigurable Hardware Platforms

Reconfigurable hardware platforms, although finding their origins in *reconfigurable computing*, provide another promising avenue in variability-tolerance due to their inherent flexibility.

The concept of reconfigurable computing has its origins around 1960, when Gerald Estrin first started working on a new computer architecture which would include a standard processor and an additional module described as *reconfigurable hardware* [106]. This novel architecture aimed to make the most of the flexibility of a processor and enhance its operation speed by offloading a particular task to a local dedicated hardware array, which can be faster than a processor. If the task changes, the hardware array can be *reconfigured* to accommodate the new task.

As Estrin put it in [107]:

"We are firmly convinced that when a special purpose configuration may be accomplished using available facilities, a new level of inventiveness will be exercisable".

Over the next few decades, and following from his work, architectures were designed with the goal of providing the idealised hardware reconfigurability.

As sources of intrinsic variability take great strides toward becoming first-order effects on the behaviour of transistors, the inherent reconfiguration abilities of these platforms could provide designers with a powerful tool not just to fight against the variations, but to work with them to develop better circuits. The aim of this work is to make use of reconfigurability to mitigate device variations, making reconfigurable hardware platforms the most relevant research avenue to explore.

FPGAs

Field Programmable Gate Arrays (FPGA) are general purpose, user-programmable reconfigurable devices which implement logic functionality through the use of lookup-tables (LUTs), flip-flops, multiplexers and additional control, as illustrated in Figure 3.11.

Historically, FPGAs have been known to provide slower, more power and resource consuming solutions than their application-specific integrated circuit (ASIC) counterparts which are designed for a specific use, but recent developments by FPGA manufacturers, such as the inclusion of "hard" multipliers and accumulators in the reconfigurable fabric, have contributed toward the reduction of the performance gap between the two devices [108].

Although this performance gap is being addressed by the manufacturers, FPGAs are not yet ready to completely replace ASICs in day-to-day tasks. Still, these devices are used today in a broad range of applications, such as accelerators for video processing, ultra-sounds, data-mining, networking and much more.

Basic logic functionality is implemented with LUTs, flip-flops and multiplexers which make up what is typically referred to as a *slice*, or *logic cell*. Some Xilinx FPGA families also include different types of slices.

Implementing a design on an FPGA follows the process illustrated in Figure 3.12. A Hardware Description Language (HDL) such as Verilog or VHDL is used to describe the logic behaviour of the circuit to implement. A logic synthesis tool takes this information and translates the design into an equivalent one constructed with basic logic gates. The mapping stage takes this set of logic gates and groups them in the best possible way to fit the hardware resources available on the FPGA. During the placement stage, these groups are assigned to actual hardware resources on the device, and the routing stage takes care of establishing the connections between the gates. A Static Timing Analysis (STA) tool performs delay calculations for every path in the design, and detects any setup- or hold-time violations, calculating the design's maximum operating frequency. Finally, a bitstream is generated which contains all this information which is read by the configuration resources – typically static-RAM (SRAM) cells – of the FPGA, implementing the design.

A Configurable Logic Block (CLB) typically contains a few slices, that have been introduced as the main logic resource for implementing both sequential and combinatorial circuits.

At the topmost abstraction layer, an FPGA can be seen as a sea-of-CLBs connected by routing resources (switch boxes) and I/O pads, as depicted in Figure 3.13.

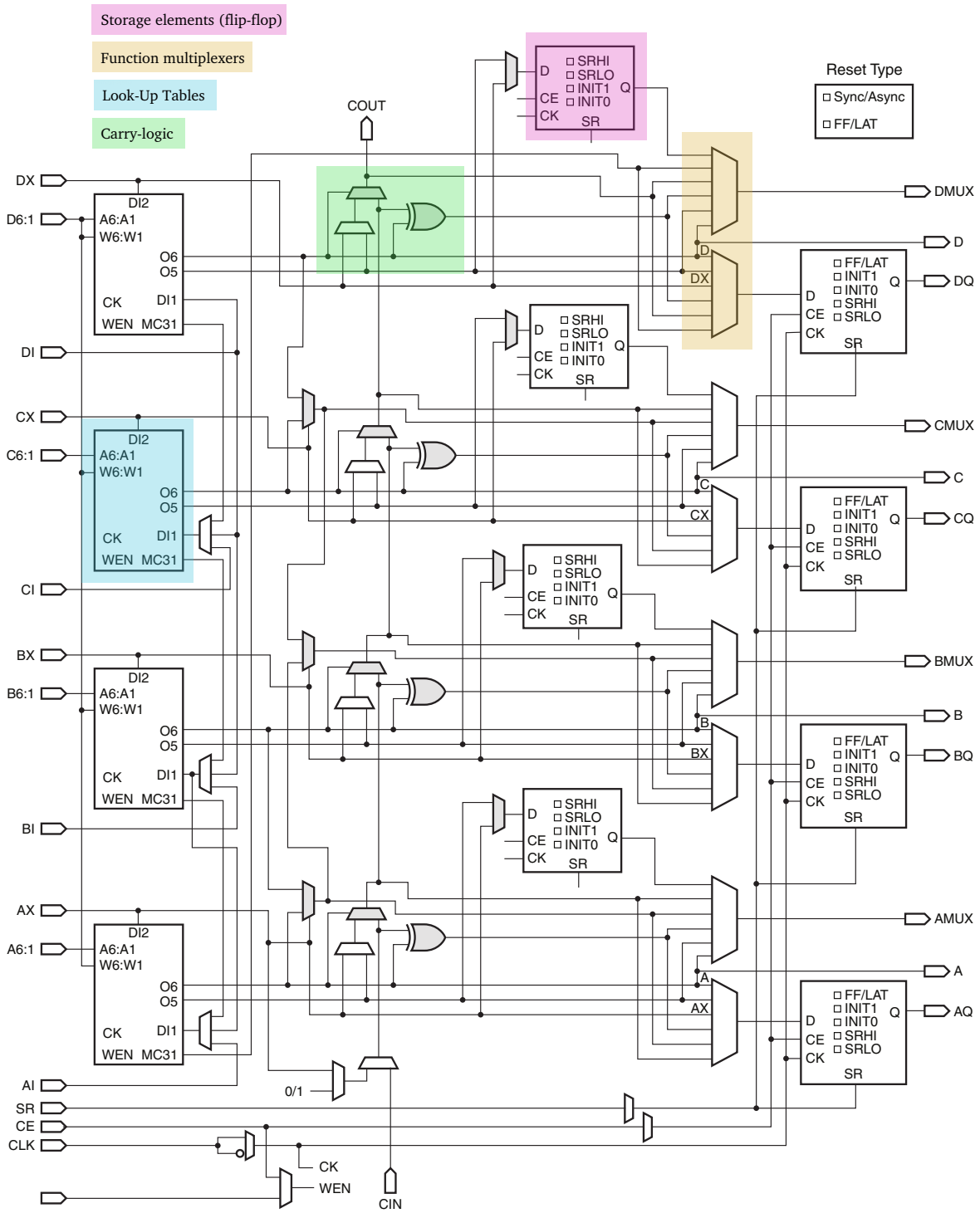


Figure 3.11: A logic cell (or slice) from the Xilinx 6-series family of FPGAs, containing look-up tables, storage elements, multiplexers, and carry-logic. Figure sourced from [21].

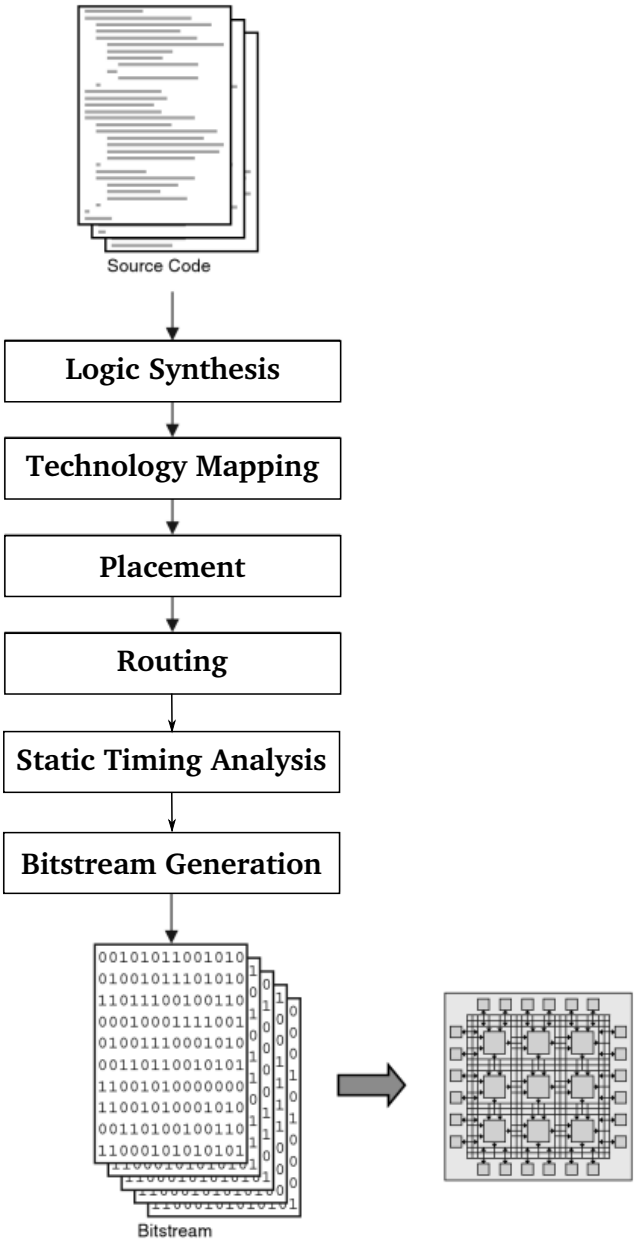


Figure 3.12: The process of translating a circuit specification to a bitstream which configures the FPGA. Figure sourced from [22].

A very common approach in the implementation of a design on an FPGA is to include a soft-microprocessor – i.e. a microprocessor that can be fully implemented using logic synthesis – such as the MicroBlaze [109], to simplify some of the computation. Most FPGA manufacturers also provide a software development toolkit to make full use of the soft-processor. Combining this soft-processor with the memory and other hardware resources of the FPGA, a standard computer architecture can be implemented.

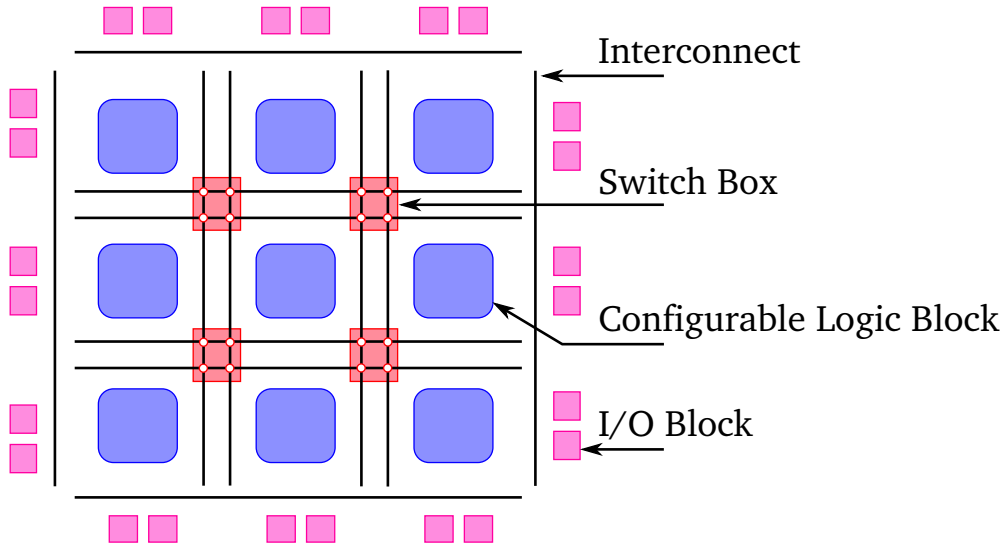


Figure 3.13: Top view of a standard FPGA architecture

Some modern FPGAs such as the Xilinx Virtex-6 contain more than 100K slices which, depending on the architecture of the slice, means that the device can implement circuits with several hundred thousand gates, making it possible for very complex designs to be created, bringing the FPGA closer to the complexity of a System-on-a-Chip (SoC).

These devices provide a large amount of flexibility regarding the logic implementation of a particular design, but their nature is digital. If one is concerned with power efficiency or speed of a design, there is little than can be done to impact it significantly by using the FPGA design flow, other than circuit re-mapping.

The intrinsic variability issue introduced in Chapter 2 poses a significant threat to FPGAs, since they can cause both timing and functionality issues [110, 111, 112]. The design flow presented in Figure 3.12 assumes a deterministic set of variations characterised at a pre-fabrication stage.

Non-deterministic variations have to be dealt with at a post-fabrication stage through a process called "speed-binning", whereby devices are grouped (binned) into separate clusters depending on the maximum speed they can achieve, which are known as *speed grades* in the case of Xilinx devices. The price that the devices are sold for depends on the "bin" they've been assigned to. From an economic perspective, this approach is costing the manufacturer a great deal of money, since ideally all devices would be grouped in the top category (plus or minus a certain tolerance in performance).

On the designers' side, they cope with these variations by picking subsets of worst-case conditions – *i.e.* variables which contribute to process variations, such as temperature,

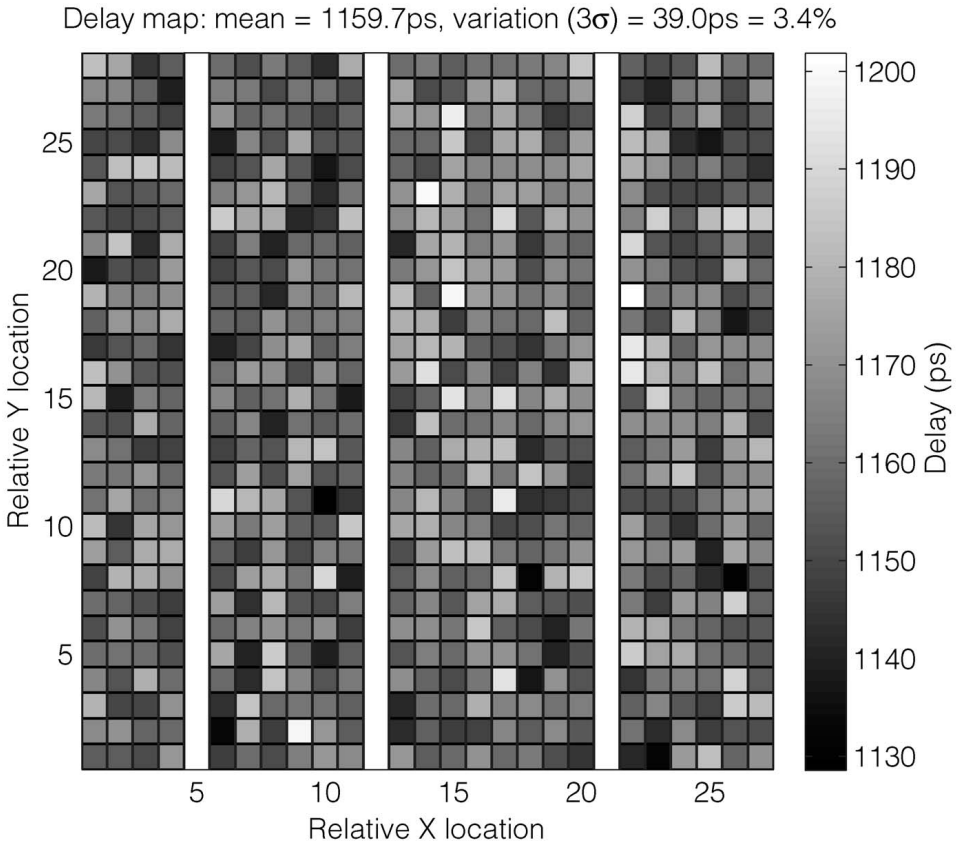


Figure 3.14: Example of a delay map extracted from an Altera Cyclone III FPGA. A variation of 3.4% is observed between the measured logic cells. This information can be fed back to place and route algorithms for variation-aware implementations. Figure sourced from [23].

voltage, transistor width, etc – and analysing their design under such conditions. This can be seen as too conservative, as these worst-case scenarios may have a very small probability of ever occurring, but it may also not be conservative enough, as more severe outliers in these conditions can exist [113].

An extensive array of techniques have been developed for not only characterising variability in programmable devices, but also to perform variation-aware mapping, making the most of reconfiguration abilities of these devices.

A particularly popular technique is Razor. Although not exclusively designed for use in FPGAs, the reconfigurable nature of these devices allow for the inclusion of these modules at any point in a circuit, making them a useful technique in the FPGA device reliability field [114, 115].

Making use of these timing-error detection capabilities of the Razor flip-flop on reconfigurable hardware, research undertaken at Imperial College has focused on using this information to allow for **variation-aware circuit mapping** [23]. These mechanisms make it possible to

capture data such as Figure 3.14 illustrates where the impact of variability can be represented by a figure of merit, for example circuit delay. This information can be redirected to the place & route tools, which are then able to place timing-critical portions of the design to faster areas of the fabric, and assign lower priority tasks to slower areas [110]. This methodology makes it possible to push the mapped circuit beyond the operating conditions established by the manufacturer, ultimately resulting in power-savings and performance improvement. By performing online measurements, finding the optimal mapping for a particular design does not require any over-conservative margins or guard-bands. Additionally, this approach provides a suitable mechanism to detect ageing-related *degradation*, contributing toward the reliability of devices [23].

Performing online measurements requires a hardware overhead, which some manufacturers may not consider economically viable, and therefore resort to modelling. This is what Static Timing Analysis tools have been used for in the last three decades [86, 87], and **Statistical Static Timing Analysis** has included the effects of variability in its models [116, 117], allowing for variation-aware design implementation. In the case of FPGAs, they have been proven useful since the delay of a particular logic gate, or net of a given length, could be modelled and would take the worst-case value extracted from the model. For a particular circuit, these delays would be added and after a design has been placed and routed on the FPGA, its maximum operating frequency could be calculated. For FPGAs, this information is fed to the implementation process, which qualifies it as a post-manufacturing approach rather as opposed to its use in traditional IC design, where it is used at a pre-manufacturing stage.

FPTAs

In traditional circuit design, the designer creates a schematic and corresponding layout, chooses a feature size, calculates the component values to meet the operating specifications under the specified conditions, and performs an analogue simulation of the design for validation. As covered in Chapter 2, the circuit manufacturing process suffers from both deterministic and stochastic variations. The former have been around for many decades, and are fed back into the design process in the form of *guard-bands*.

The circuit designer then knows how much of an effect these variations can have in the performance of their design, and usually adjusts it to accommodate these variations in order to provide a guard-band which makes it more difficult for this to impact the functionality

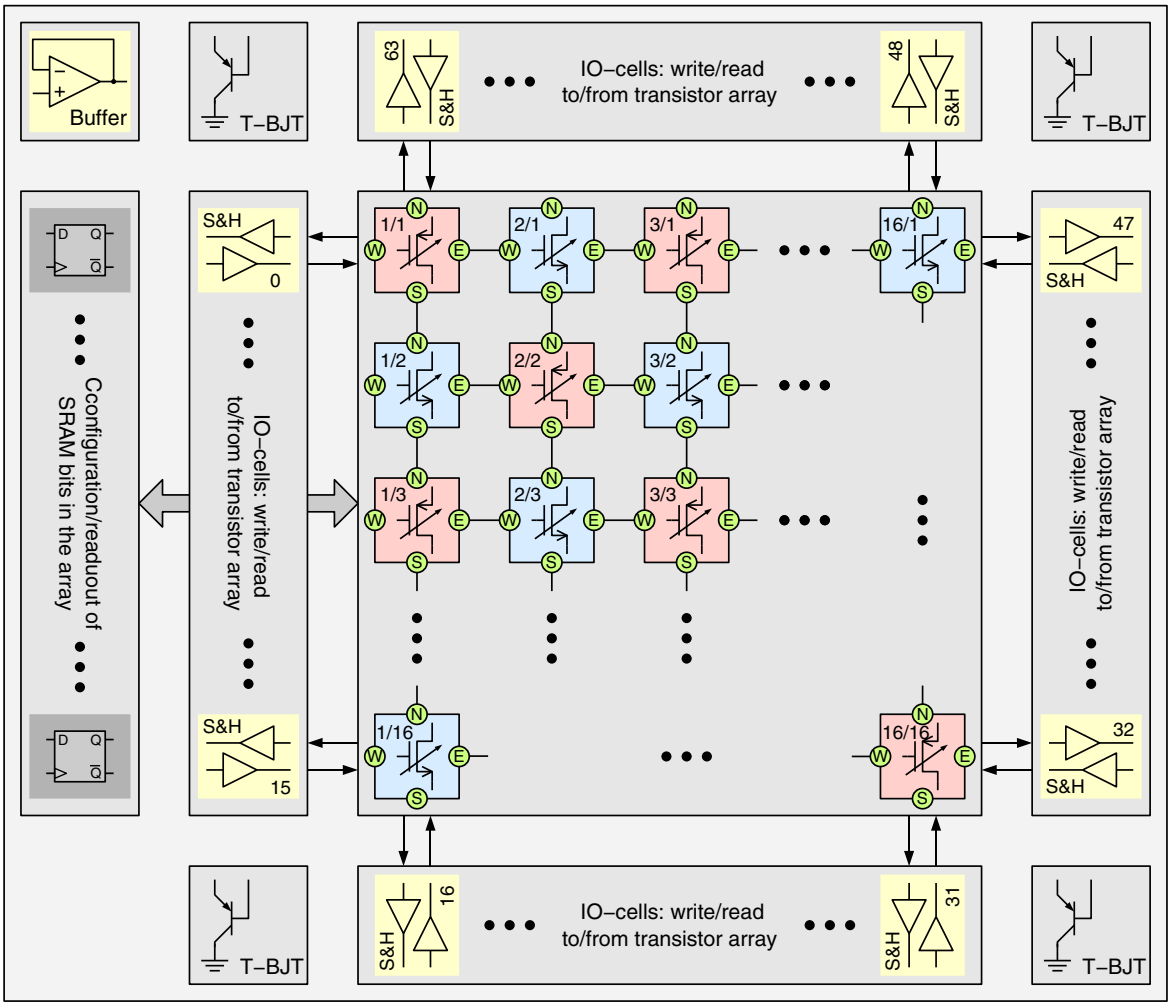


Figure 3.15: A simplified architecture of the Heidelberg FPTA. nMOS cells are depicted in blue, pMOS cells in light red. Connections and transistor widths are configured through SRAM. The array is 16x16 cells large. Input voltage patterns as well as measured voltages are buffered in the represented IO cells and subsequently applied to the transistor array. The figure is taken from [24].

of the circuit. Intrinsic variability, however, greatly increases the range of variations that a circuit can experience, therefore potentially forcing the designer to increase their guard-bands to a point where many resources are wasted to protect the design against variations which have a very low – but non-negligible, due to the amount of devices comprising a modern design – probability of occurring.

The Field-Programmable Transistor Array (FPTA) provides a post-fabrication solution for circuits which exhibit extreme variations in performance, by allowing the user to *reconfigure* the geometry of the transistors which make up a particular design. Much like the CATs introduced in Section 3.2.2, configurable transistors are introduced in the place of normal de-

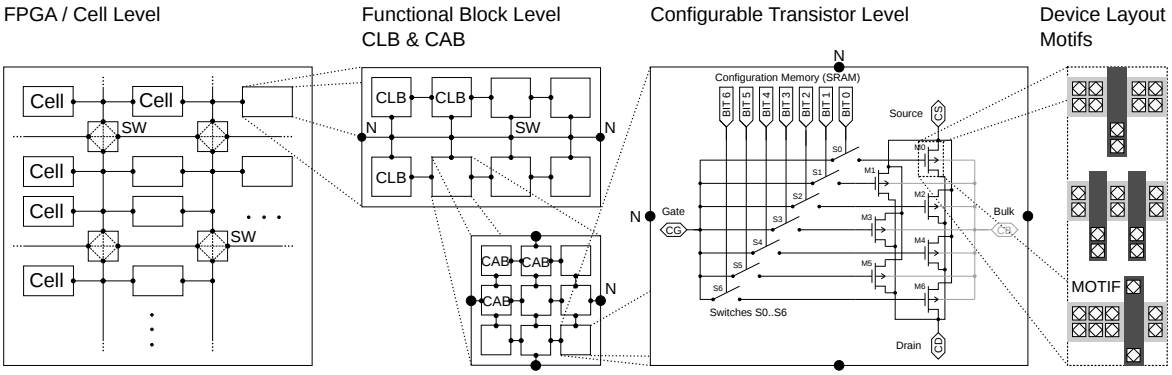


Figure 3.16: The hierarchical architecture of PAnDA, with the hierarchy being shown from top-layer (left) to bottom-later (right). The topmost layers host the logic functionality of a design, and the bottom layers provide the analogue flexibility through the use of Configurable Transistors. Figure sourced from [25].

vices, but instead of inserting them in strategic locations – which depend on the design being implemented – in FPTAs every transistor is replaced with their configurable counterpart.

The Heidelberg FPTA [118], illustrated in Figure 3.15, is a device specifically designed with an evolutionary mindset – a topic further discussed in Chapter 6 – which aims to exploit the device's reconfigurable connectivity in order to construct analogue circuits which meet a set of performance requirements, whether they are digital in nature (e.g. logic functionality) or analogue (e.g. slew rate, output drive strength). A similar, yet smaller device was designed at the Jet Propulsion Labs [119], which reported the design of unconventional circuits such as combinatorial designs for fuzzy logics using aspects of evolvable hardware, a concept which is introduced later in this thesis. A comparison of the two architectures can be found in [120].

PAnDA

With the aim of combining the flexibility of FPGAs with the analogue access of FPTAs, the **P**rogrammable **A**nalogue a**n**d **D**igital **A**rray (PAnDA) provides a scalable fabric which allows the designer to map a circuit and fine-tune it to comply with a set of performance specifications [25].

Its main advantage over FPGAs is the added analogue flexibility, and over FPTAs it is the scalability.

PAnDA is a hierarchical architecture which resembles a traditional FPGA at its topmost layers, and provides additional analogue flexibility through configurable transistor sizing at its bottom layer, as illustrated in Figure 3.16.

Given that it combines two architectures known for their use in the field of evolvable hardware, it is also a suitable platform to evolve circuits, as well as to use the concepts of evolution as fault-tolerance mechanisms [31].

Additionally, as this work sets out to demonstrate, the PAnDA architecture has the potential to mitigate some of the effects of intrinsic variability.

For these reasons, PAnDA qualifies as both a pre- and post-fabrication approach, since it includes structures specifically designed (at pre-fabrication) to mitigate variability, and it allows for post-fabrication adjustments to be made through reconfiguration (to both analogue and digital layers).

In a standard FPGA, the basic hardware is fixed. Although the user can configure the logic functionality and connections inside the fabric, the performance characteristics of the designed circuit (e.g. power consumption, speed) will remain unchanged unless the design is re-mapped in a different way, and therefore using a different set of transistors present at a different location on the fabric.

The **Configurable Transistor** aims to introduce a new degree of configuration, allowing the user to essentially change lower-level properties of their design without the need to find a new location on the fabric which provides the desired performance. This flexibility has been proven useful for the purposes of reliability, as reported in [16].

Taking inspiration from the FPTA design of [26, 24], an array is constructed with 7 basic transistors of the same channel length (L) but different widths ($W_0 ... W_6$) connected in parallel, with a common gate, source, bulk and drain. These 7 basic transistors can either all be pMOS or all nMOS.

A set of switches is then positioned between the common gate and the gate of each individual transistor, shown in Figure 3.17 as $S_0 ... S6$. If all switches are closed, then all signals will be common to all transistors. The resulting circuit will be equivalent to a single transistor with channel length L, but with a width that is defined by the sum of all Ws.

Additionally, a configurable clamp is attached to the common gate of each CT. This clamp connects the gate to either Vdd or Gnd, or the input of the CT. The CT will consequently be permanently open (off), permanently closed (on) or input-dependent, respectively. In networks of CTs implementing a range of logic functions, these clamps are essential to ensure the appropriate CTs are turned on or off, according to the implementation of the function.

The CTs can then be in one of the following states:

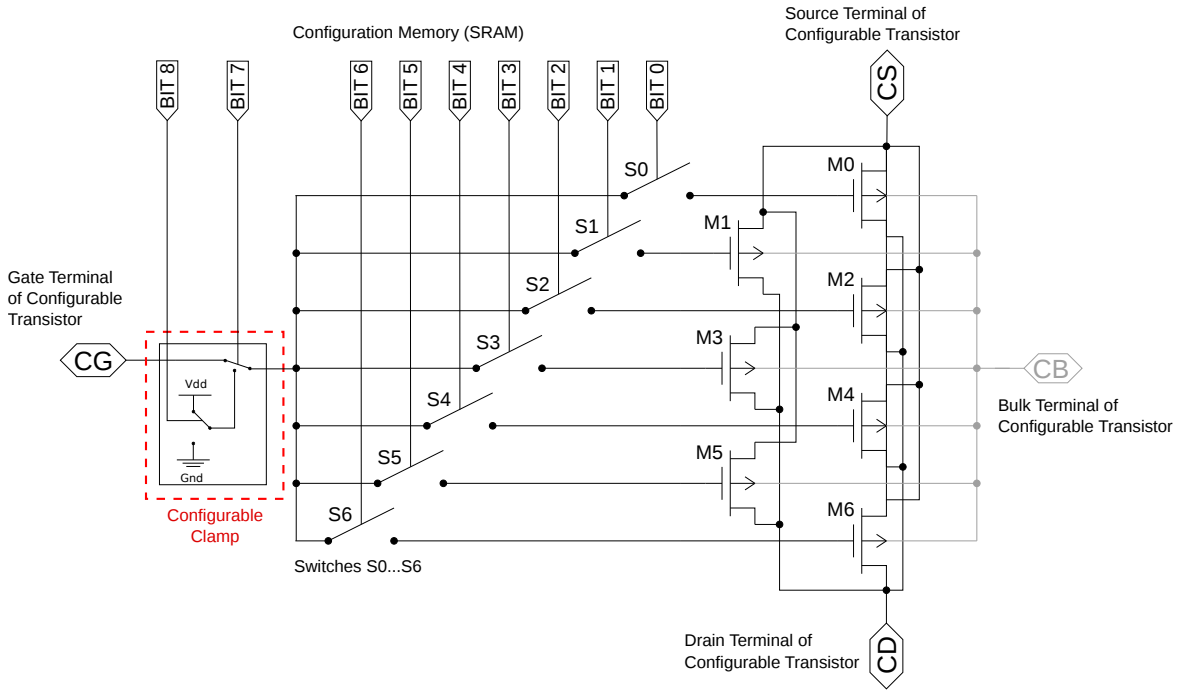


Figure 3.17: Schematic of a pMOS PAnDA Configurable Transistor. The transistor sizes used are $L_{0...6} = 40nm$ and $W_{0...6} = [120, 120, 140, 160, 180, 200, 220]nm$, allowing for CT widths between 120 and $1140nm$. Modified image from [26].

Enabled when the CT behaves as a single transistor, with its width configured by switching individual transistors on and off using the first 7 bits of the configuration word. This is achieved by using bit 7 of the configuration word to establish a connection between the common gate of the CT and the gates of the individual transistors.

Disabled when the CT's state is not input-dependent because the common gate has been disconnected from the individual transistor gates through configuration bit 7. Bit 8 of the configuration word controls whether the disabled CT is *insulating* or *conducting*. For the former case, the CT is seen as an open connection in the CAB structure, and in the latter it is seen as a wire.

For example, if two of these transistors, one with $\frac{W}{L} = \frac{120nm}{40nm}$ and another with $\frac{W}{L} = \frac{200nm}{40nm}$ are connected in parallel, the resulting circuit will be equivalent to a transistor with $\frac{W}{L} = \frac{320nm}{40nm}$. This resulting circuit is called a *Configurable Transistor* (CT).

By opening and closing the gate switches, one can effectively change the resulting width of the CT, which – as presented in the previous chapter – has an effect on the drain current, consequently making the transistor faster or slower.

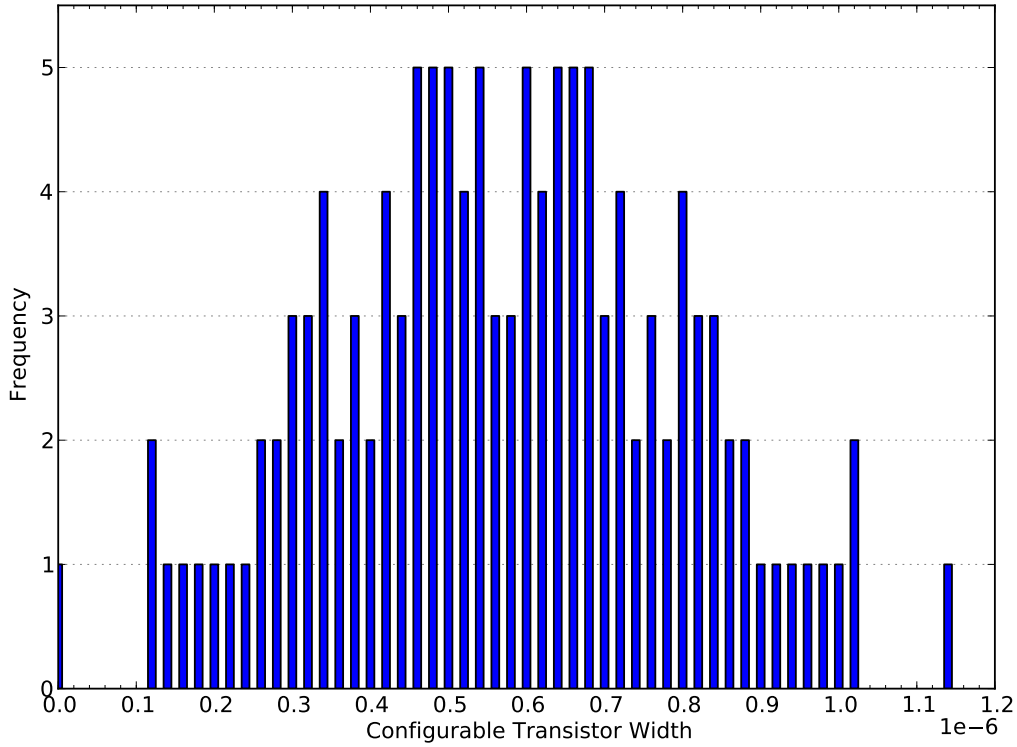


Figure 3.18: All CT width configurations achievable with set of individual transistor width set $W_{0...6}$=[120, 120, 140, 160, 180, 200, 220]nm for transistors $M_{0...6}$ which make up a Configurable Transistor. Figure sourced from [27].

The channel widths of the 7 transistors that make up a CT are as follows: 120 (x2), 140, 160, 180, 200 and 220nm. All 7 transistors share the same channel length of 40nm. This set was chosen to make it possible to change the channel width of a CT in 20nm increments (half of the channel length) between 120nm and 1140nm, allowing for a total of 128 unique widths for a particular CT. This set of sizes provides a range of operating points (in terms of speed and power consumption) which allow the designer to deal with variations by altering the geometry of a given transistor to better fit the performance requirements for a particular design. Figure 3.18 illustrates the CT channel widths that can be achieved using different combinations of the 7 transistors. This graph also illustrates how the same geometry can be achieved in different ways – especially for the mid-range channel widths – which makes it possible to conserve the analogue characteristics of a particular design in the event that one of the 7 transistors inside the CT experiences a fault. It also provides redundancy, which can be very useful when dealing with varying characteristics between devices brought on by stochastic variability.

Figure 3.19 shows the $I-V$ curves associated with all the 128 CT width configurations of one nMOS CT, and how intrinsic variability causes the curves to overlap. 100 RandomSPICE

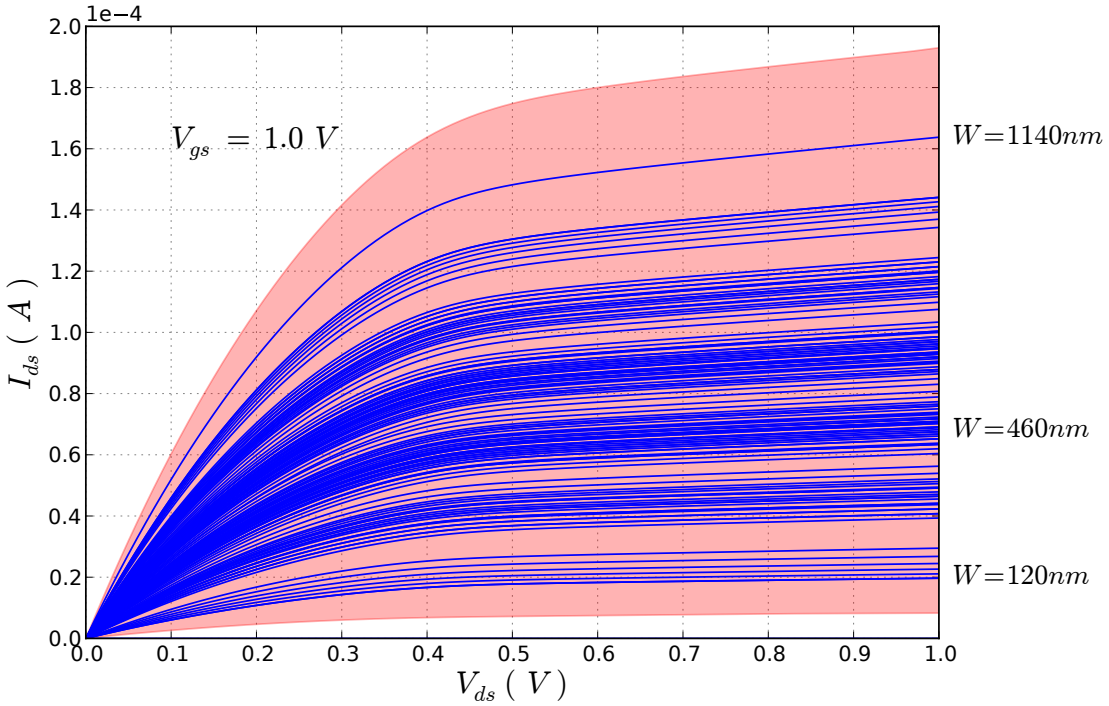


Figure 3.19: $I-V$ characteristics for all 128 possible width configurations of a nMOS CT (blue lines) and the corresponding effect of variability (salmon-coloured area). The drain-source voltage (V_{ds}) is plotted on the x-axis, the drain-source current (I_{ds}) is plotted on the y-axis, and the gate-source voltage (V_{gs}) is 1V. The effective width of the CT corresponding to certain $I-V$ curves are shown on the right. Figure sourced from [25].

netlists were created for each CT width configuration, so in total 12,800 SPICE simulations were run [25]. The point illustrated by Figure 3.19 is that intrinsic variability can blur the relationship between transistor width and $I-V$ curve shape. With variability, it will not necessarily be true that a wider transistor will be providing a larger amount of current for the same gate voltage when compared to a narrower transistor. In fact, in extreme cases it may even be the other way around. The salmon coloured areas between the variability-free blue lines represent this blurring of performance.

Another experiment was carried out in [25] to investigate how variability affects a CT of a particular channel width depending on which transistors are used to achieve it. Figure 3.20 illustrates the results of this experiment carried out for an nMOS CT channel width of 460 nm achieved through the use of five different transistor combinations inside of a CT. For each configuration, 100 RandomSPICE simulations are performed, and the performance variations are added to the $I-V$ curves in the form of the shadowed area. The plots suggest that the different size configurations respond differently to variability, further validating the advantages of the hardware redundancy provided by PAnDA.

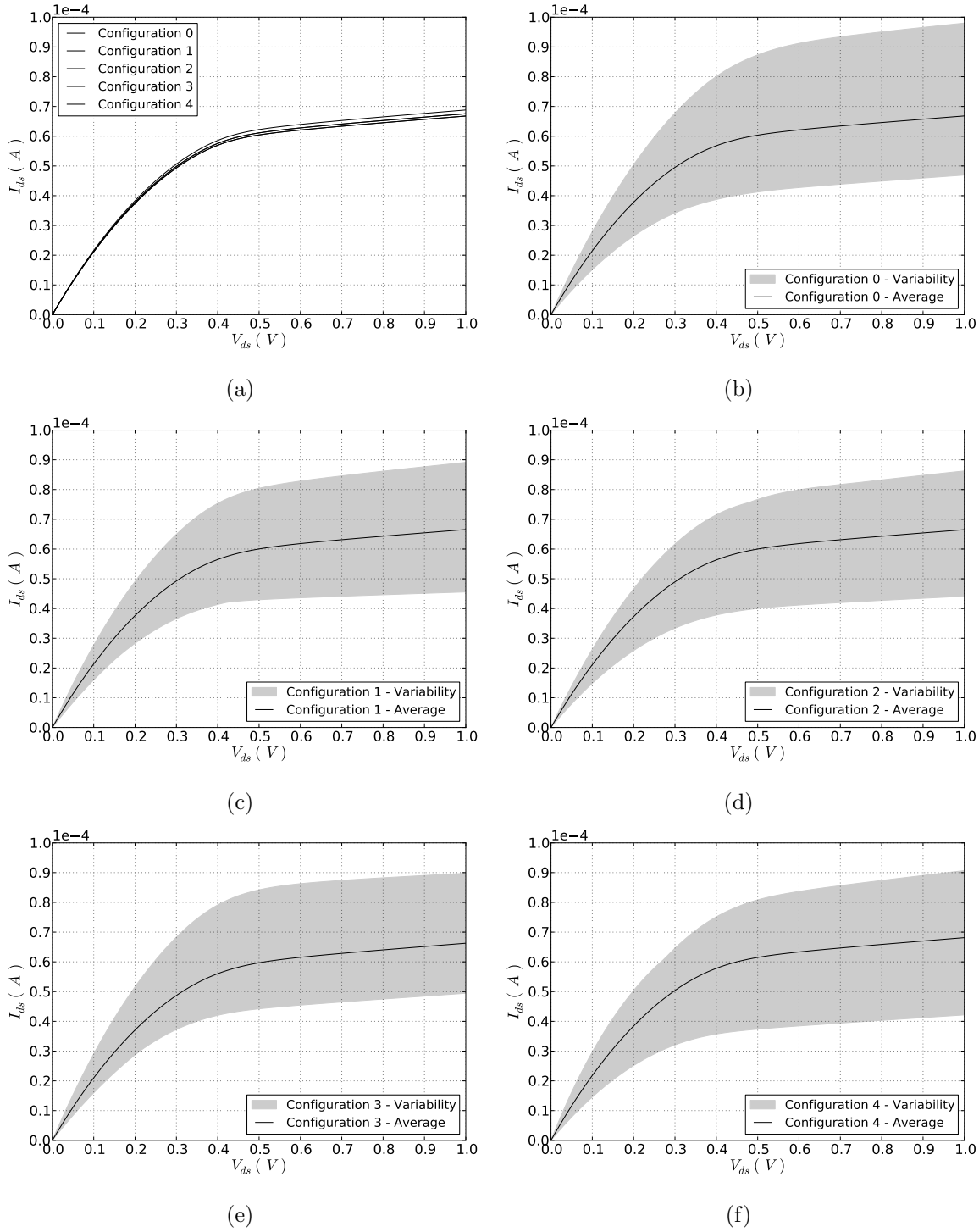


Figure 3.20: $I - V$ characteristics of the five different configurations for width 460 nm of a nMOS CT (a) and the effect of stochastic variability upon each of the five width configurations (b-f). The drain-source voltage (V_{ds}) is plotted on the x-axis, the drain-source current (I_{ds}) is plotted on the y-axis, and the gate-source voltage (V_{gs}) is $1V$. Figure taken from [25].

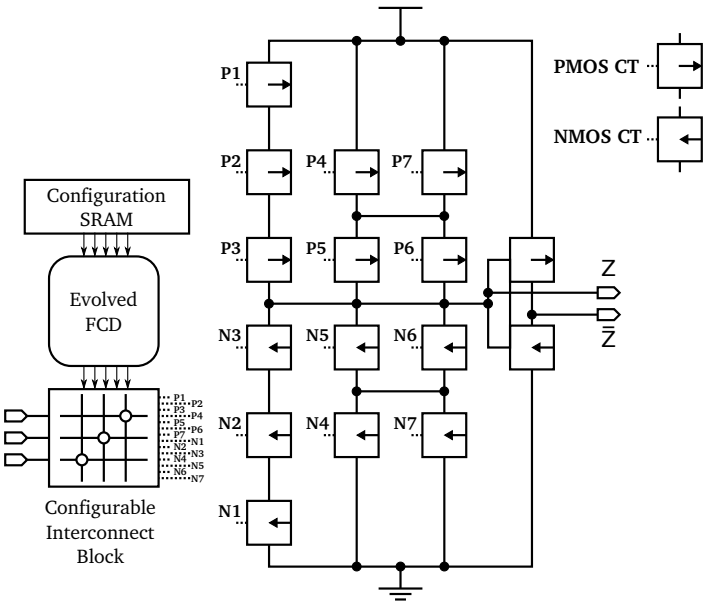


Figure 3.21: Schematic of a Combinational CAB. The Function Configuration Decoder is configured through SRAM, in turn bringing the configurable interconnect block to the appropriate configuration, routing the correct signals to the inputs of the CTs. This is a modified version of a figure used in [28].

Changing the geometry of a given CT in a design will effectively change its analogue characteristics, causing potential alterations to the drive current which can either increase or decrease the output slew-rate, and will therefore have a positive or negative effect on the overall circuit performance.

Moving up one level in the hierarchy, a number of pMOS and nMOS CTs are put together in the form of an array to create a **Configurable Analogue Block** (CAB).

For the design iteration used as the subject for this modelling exercise, two types of CAB were available: *Combinational* (CCAB) and *Sequential* (SCAB), each with their own set of implementable functions. The former comprised 8 nMOS and 8 pMOS CTs, and the latter 10 nMOS and 10 pMOS CTs. The structures of a CCAB and of an SCAB are illustrated in Figures 3.21 and 3.22, respectively.

Whereas the purpose of the CCAB is to implement basic combinational logic blocks which can be implemented on an FPGA fabric, the SCAB is designed to represent pass transistor logic blocks and basic tri-state logic blocks that can be combined to create sequential logic.

CCABs take in three inputs (A, B and C) and SCABs take in four (A, B, C and D) and both generate two outputs, Z and $\bar{Z}$. Each individual CT in the array takes in only one of the inputs to the CAB, controlled by a Configurable Interconnect Block.

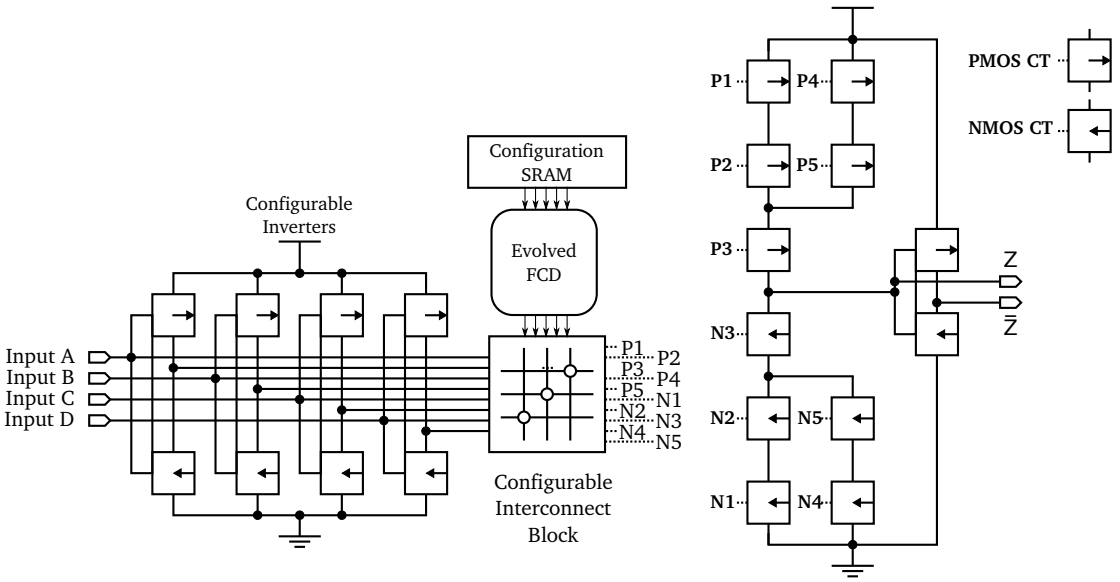


Figure 3.22: Schematic of a Sequential CAB. A set of configurable inverters The Function Configuration Decoder is configured through SRAM, in turn bringing the configurable interconnect block to the appropriate configuration, routing the correct signals to the inputs of the CTs. This is a modified version of a figure used in [28].

In order to reduce the overhead introduced by the flexibility of the PAnDA architecture, a Function Configuration Decoder (FCD) has been evolved using Multi-Objective Cartesian Genetic Programming [25] which controls the configurable interconnect block, effectively controlling the function performed by the CAB.

The Configurable Interconnect Block generates the signals which control the gates of the CTs inside a CAB. Besides establishing a path between one of those signals and a CAB input, it can also disable or make a CT transparent, by connecting its gate to V_{ss} or *Gnd*.

In [25], the FCD was designed to control up to 8 different functions for CCABs and another 8 for SCABs. Since the inverted output is also available, the number of functions they can actually implement is 16. These function sets are described in further detail in Tables 3.1 and 3.2.

Whereas CTs can be seen as the foundations of the PAnDA architecture, CABs can be regarded as the building blocks for the digital logic implementation.

The **Configurable Logic Block** (CLB) sits at the next level in the PAnDA hierarchy, and it comprises a cluster of 4 CCABs and 4 SCABs. On a conventional FPGA, these are typically comprised of Lookup Tables (LUT) and D-type flip-flops. The CCABs can be seen as equivalent to a standard FPGA's LUTs, and the sequential nature of SCABs is more closely related to flip-flops.

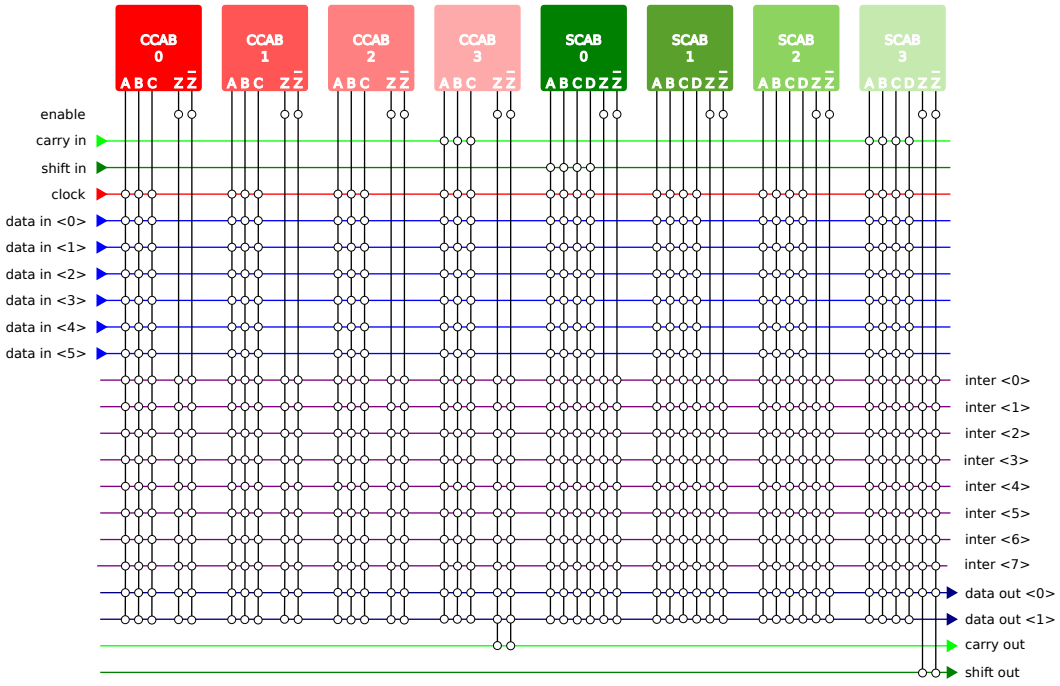


Figure 3.23: Schematic of the switch matrix associated with one CLB, establishing the required connections between the CABs, as well as routing the signals which will be propagated to other CLBs.

A CLB exposes all of the input and output buses of its CABs to the switch matrix associated with it, which handles all the internal (between CABs) and external (to the rest of the fabric) connections.

Associated with each CLB is a **switch matrix**. It is structured as illustrated in Figure 3.23, taking in a 6-bit wide bus as input along with carry- and shift-chain inputs and a clock signal, and the output signals from every CAB inside of the CLB. It is essentially a crossbar switch.

Table 3.1: Configurable CCAB Functions

Configuration	Function (Standard Output)	Function (Inverted Output)
0	AND-OR-Invert	AND-OR
1	Inverter	Buffer
2	2-input NAND	2-input AND
3	3-input NAND	3-input AND
4	2-input NOR	2-input OR
5	3-input NOR	2-input OR
6	OR-AND-Invert	OR-AND
7	Inverted Majority	Majority

Table 3.2: Configurable SCAB Functions

Configuration	Function (Standard Output)	Function (Inverted Output)
0	Inverter	Buffer
1	2-input XOR	2-input XNOR
2	2-input XNOR	2-input XOR
3	2-input multiplexer	2-input multiplexer with inverted output
4	Tri-state inverter	Tri-state inverter with inverted output
5	Tri-state inverter with $\overline{\text{enable}}$	Tri-state inverter with $\overline{\text{enable}}$ and inverted output
6	Clocked multiplexer	Clocked multiplexer with inverted output
7	Two inverters with common output	Two inverters with common inverted output

It outputs a two-bit wide data bus and carry- and shift-chain outputs, as well as all the input signals to every CAB inside of the CLB.

The connections between buses are made through a series of multiplexers and demultiplexers, the select signals which are configured from memory. Each CAB also includes an output enable signal.

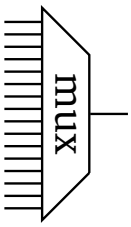
The 8-bit wide interconnect bus allows the switch matrix to connect the output of one CAB to the input of another, essential for propagating signals internally in a CLB.

Figure 3.24 lists the connections that are established for each particular value of the select line for both the input multiplexers and the output demultiplexers that make up the switch matrix.

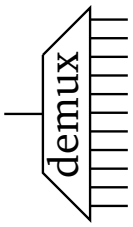
Figure 3.25 presents an additional illustration of the connections between a CLB and its associated switch matrix. Together, these two elements form the building block of the PAnDA architecture at the topmost layer.

At the topmost abstraction layer, the PAnDA architecture can be regarded as a **sea of CLBs** (with their associated switch matrices). At this level, a conventional FPGA and a PAnDA device look very similar from a functional point-of-view.

PAnDA-EiNS, the first design iteration of PAnDA, is an 8-row by 4-column array of CLB and switch-matrix pairs, with a fixed row-based connectivity.



SEL	CCAB	SCAB
0000	clk	clk
0001	dataIn[0]	dataIn[0]
0010	dataIn[1]	dataIn[1]
0011	dataIn[2]	dataIn[2]
0100	dataIn[3]	dataIn[3]
0101	dataIn[4]	dataIn[4]
0110	dataIn[5]	dataIn[5]
0111	inter[0]	inter[0]
1000	inter[1]	inter[1]
1001	inter[2]	inter[2]
1010	inter[3]	inter[3]
1011	inter[4]	inter[4]
1100	inter[5]	inter[5]
1101	inter[6]	inter[6]
1110	inter[7]	inter[7]
1111	carryIn	shiftIn



SEL	CCAB	SCAB
0000	inter[0]	inter[0]
0001	inter[1]	inter[1]
0010	inter[2]	inter[2]
0011	inter[3]	inter[3]
0100	inter[4]	inter[4]
0101	inter[5]	inter[5]
0110	inter[6]	inter[6]
0111	inter[7]	inter[7]
1000	dataOut[0]	dataOut[0]
1001	dataOut[1]	dataOut[1]
1010	carryOut	shiftOut
1011	Z	Z
1100	Z	Z
1101	Z	Z
1110	Z	Z
1111	Z	Z

Figure 3.24: The input multiplexer and output demultiplexer connections established with different select signals. A Z is shown when the output is in a high-impedance state.

The device takes in two 8-bit inputs, represented by $A0..A7$ and $B0..B7$, and produces two 16-bit outputs, one being the concatenated outputs of the first column of CLBs and the other the concatenated outputs of the last column of the array.

Each switch matrix propagates two output data bits to the CLBs in the two columns that follow it, making the architecture routing row-based.

The PAnDA structure includes both **carry-** and **shift-chains**, allowing for the implementation of more efficient arithmetic functions and sequential shifters.

Each row on PAnDA-EiNS contains its own separate carry-chain, whereas a single shift-chain spans across the entire array in a snake-like pattern, to allow for the implementation of large shift-registers.

The **configuration** of routing and functionality is achieved by means of a flip-flop chain, which shifts in a static random access memory (SRAM) address, along with 14 bits of data. Once a write signal is issued, the data word is written to the memory block at the specified address.

As Figure 3.26 shows, each address is directly connected to the configuration to either CT sizes, CAB functionality, or switch matrix connections.

An initial set of experiments was carried out on a fabricated PAnDA-DREI device, the third design iteration of the PAnDA architecture. Although not directly comparable to the results described in this work due to the architectural differences, they still provide some insight into how intrinsic variability manifests itself in actual fabricated hardware. A series of 32

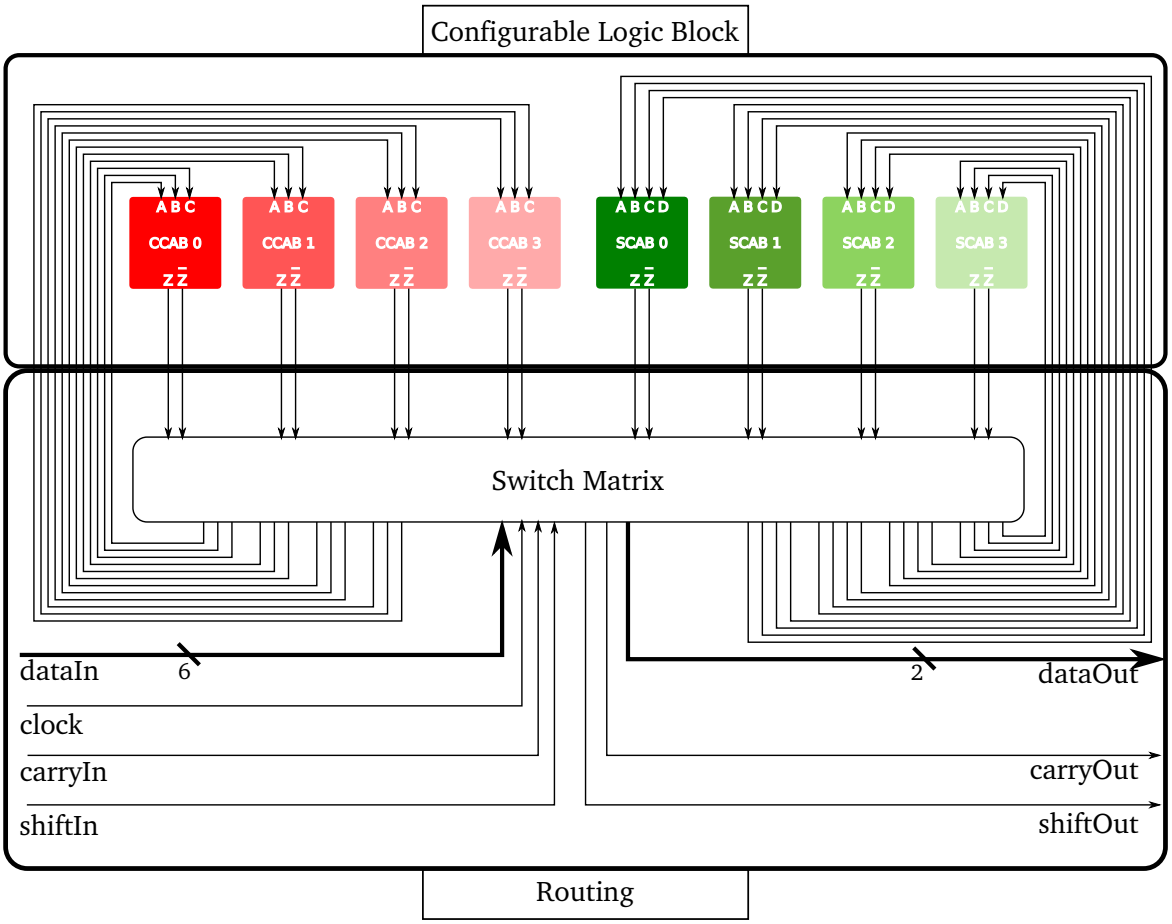


Figure 3.25: A CLB-switch-matrix pair, depicting all internal and external connections.

seven-stage ring oscillators were mapped to a PAnDA-DREI chip and the frequencies were measured, with the results being illustrated in Figure 3.27. Although the plot only depicts absolute error, some of the frequencies were below the average, and some were above.

3.5 Summary

This chapter presented a summary of some of the most popular variability mitigation techniques at pre-fabrication, during manufacturing, and also at the post-fabrication stage.

The effects of atomistic variability are still on the rise, and likely to become first-order effects in the operation of transistors, and manufacturing techniques face considerable physical limitations on the impact they can have on the reduction of these effects.

Pre-fabrication variability mitigation techniques require a large amount of computing power to perform, and this requirements is only likely to increase as technology continues to scale down. For this reason, post-fabrication techniques are presented as the most promising

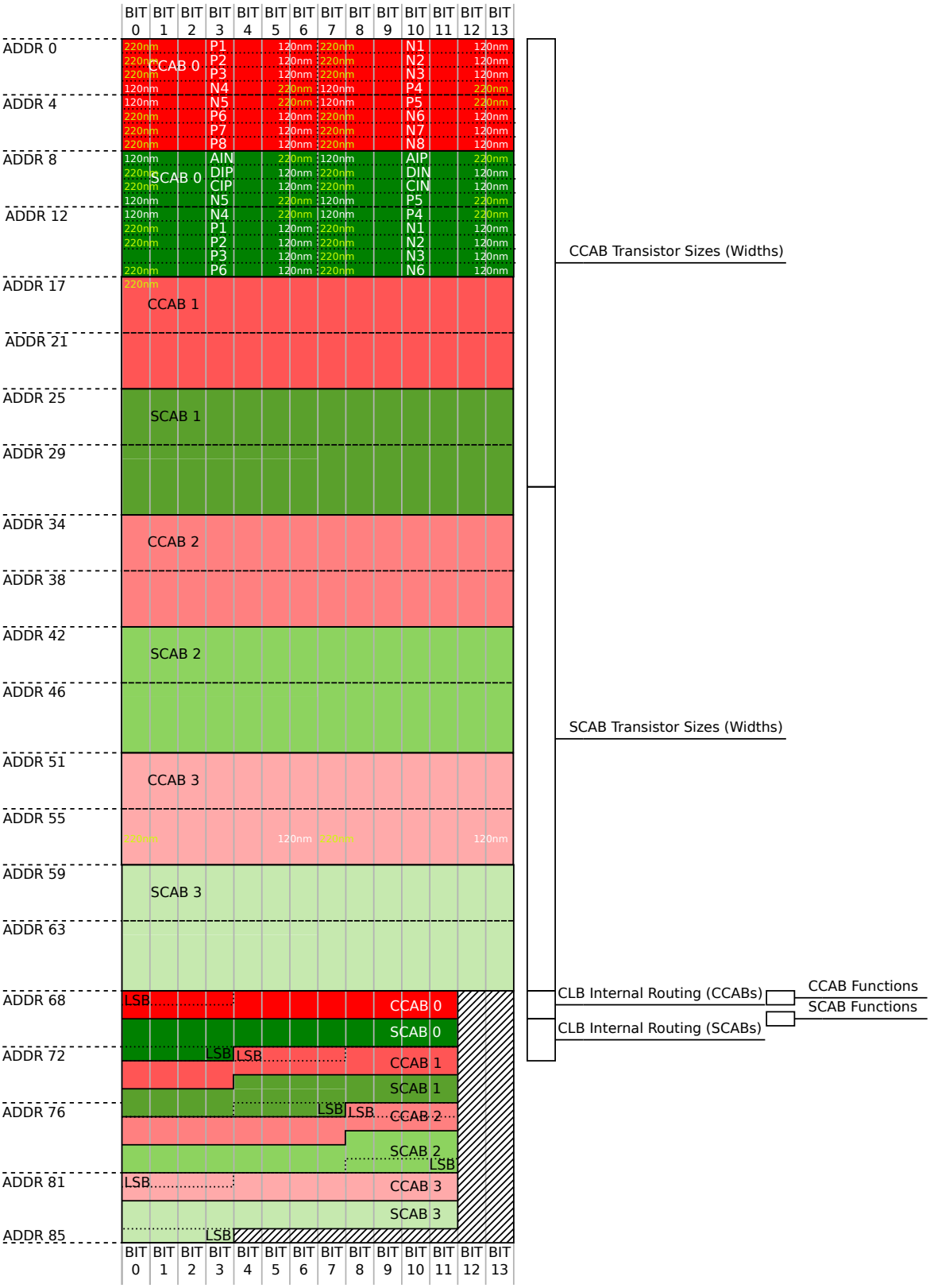


Figure 3.26: Breakdown of the SRAM mapping for the configuration of one CLB on PAnDA. 85 14-bit words are used to fully configure a CLB, including connectivity, functionality, and CT geometry.

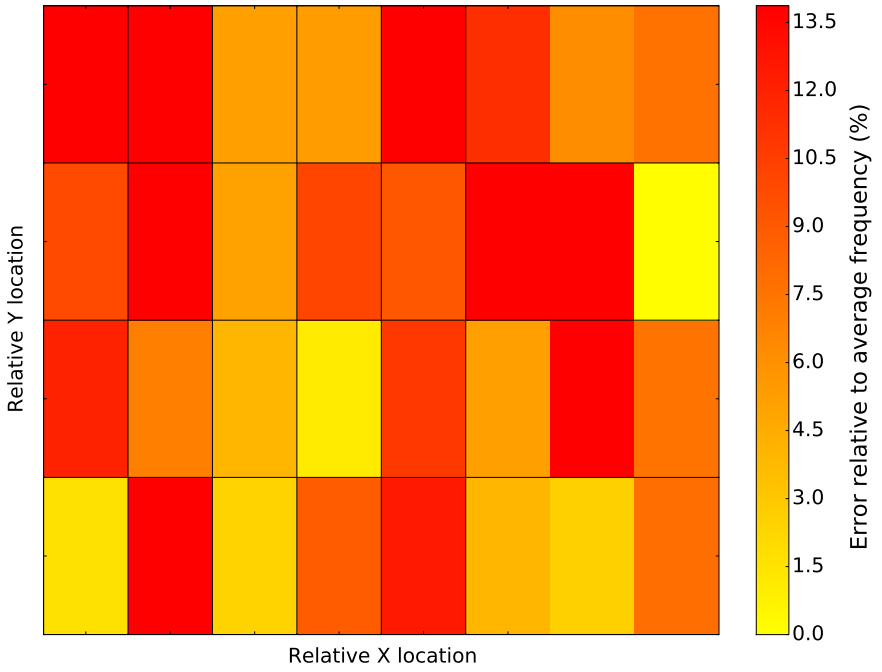


Figure 3.27: A series of 32 7-stage ring oscillators implemented on a PAnDA-DREi chip, fabricated at $65nm$. Each illustrated square represents one ring oscillator. The ring oscillators consist of 7 inverters connected in series, with CT widths set to $275nm$. The colours illustrate the relative error of the measured frequencies with respect to the calculated average across the 32 oscillators.

avenue for variability mitigation, since they can be used to perform adjustments to otherwise failing devices, and also have the potential to allow for circuit optimisation and reliability enhancement.

This chapter introduced the concept of reconfigurable hardware as one of the potential platforms on which to develop variability mitigation methods, along with notable examples such as the FPGA and the FPTA. The PAnDA architecture is then introduced as the hybrid platform which combines features from both FPGAs and FPTAs, and is presented in full detail. This architecture allows for the implementation of digital circuits in much the same way as FPGAs, since at the logic block level they look similar, whilst also allowing for some of the analogue reconfigurability introduced by FPTAs, by providing reconfiguration resources at the transistor-level, enabling the same kind of post-fabrication adjustments made possible with CATs.

The next chapter describes the modelling efforts that went into replicating PAnDA in a software-hardware hybrid model, which makes use of the parallelism of hardware for simulation acceleration and the software models of variability included with the GSS tools.

Chapter 4

PAnDA Emulator: A Tool for Accelerated Variability Characterisation

Contents

4.1 Introduction

The PAnDA architecture is the one reconfigurable hardware platform out of the candidates presented in Chapter 3 which provides a suitable fabric for the implementation of large circuits along with the ability to control individual transistors to alter the performance of a particular design. In addition, all the details regarding its structure are fully available, and for these reasons it was chosen as the platform for this work.

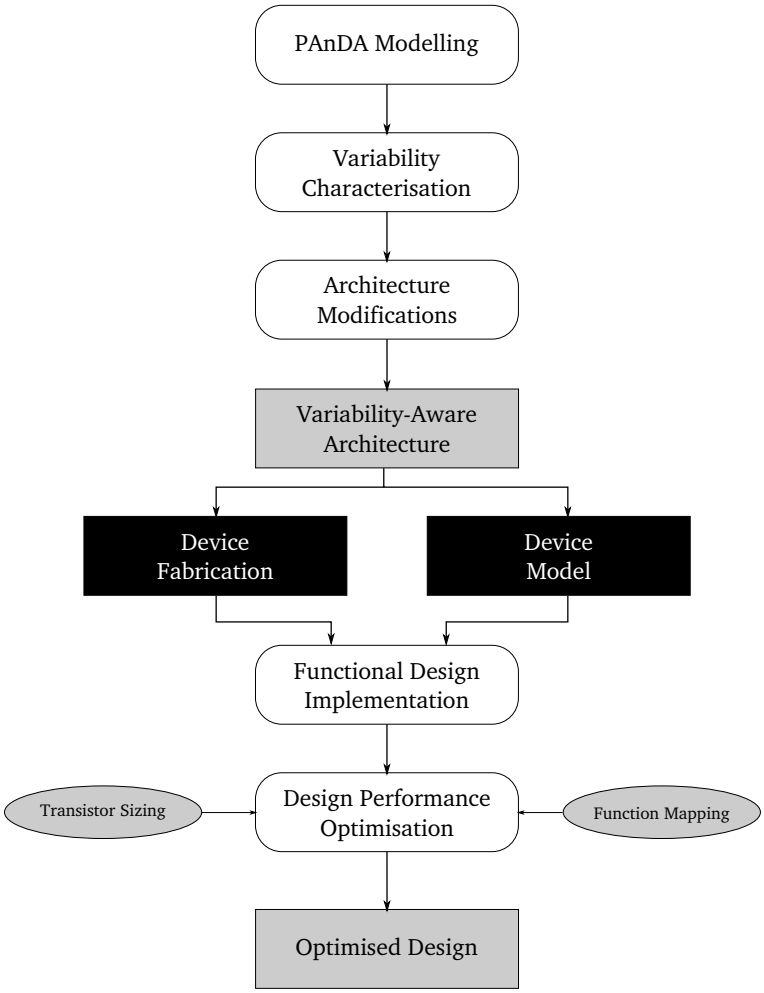


Figure 4.1: Flow-chart depicting the ultimate goal of the modelling of PAnDA for variability-tolerance, along with design optimisation at a post-fabrication stage performed on both the fabricated device and the model, exploiting the reconfigurable nature of the architecture.

As previously introduced, this hierarchical architecture carries a strong resemblance to a standard FPGA at the logic block level, and to an FPTA at the transistor level.

In order to properly assess how variability affects this architecture (and other architectures in general) one would require a very large number of devices – at least in the order of hundreds of millions, for large-scale manufacturing – to give confidence in any variability measurement. An alternative is to construct appropriate models and use these for assessing the architecture's performance under the effect of variability.

A model of the PAnDA architecture which incorporates the effects of variability serves as a pre-fabrication method of mitigating against variability, as optimal design configurations which minimise the impact of variability can be investigated. In this sense, such a model would tackle some of the effects of variability at **pre-fabrication**, and the inherent reconfigurability of the PAnDA architecture would deal with any additional issues not resolved by

the model, at a **post-fabrication** stage. Figure 4.1 presents a flow-chart which illustrates how the variability-aware model can enhance the variability-tolerance of the architecture, and also how the reconfigurability of PAnDA enables further adjustments when a device is fabricated and operating in the field.

In summary, two main investigative strands can be identified with this flow-chart. Firstly, using variability-aware transistor libraries in SPICE to create a model of a specific design iteration of PAnDA provides an estimate of how variability is likely to affect these devices once they are put through fabrication. This data can be fed back into the architecture design process, further strengthening the variability-tolerance of the design. The second strand deals with identifying optimisation strategies on an already fabricated device, taking advantage of the reconfiguration resources of PAnDA. Investigating these strategies on an actual device will take longer than using the model, so these strategies can be studied on the model and then deployed on a device.

This work puts more emphasis on the second strand, due to issues concerning the accuracy of the model (detailed in later sections). Circuits are mapped to a model of a PAnDA device, and reconfiguration is used to minimise the circuit's propagation delay in the presence of variability.

In order to accelerate the study of the impact of variability on PAnDA, a model was created which would be used for pre-fabrication evaluation. A hierarchical SPICE netlist was built for the Configurable Analogue Blocks, encompassing the CAB and CT layers. Intrinsic variation was then added through the use of the RandomSPICE tool [14], which has the feature of increasing the SPICE simulation run-time.

Chapter 2 described how one of the effects of variability involves the threshold voltage of a transistors, which in turn can affect the time it takes for a device to be turned on or off, ultimately resulting in variation of the device's propagation delay. For this reason, *propagation delay* was used as the figure of merit for variability.

To address the issue of the large simulation time required by a full RandomSPICE run, a VHDL-based model of the PAnDA architecture was designed which incorporates some of the measurements taken from the variability-enhanced SPICE simulations, providing a faster alternative for the study of the impact of variability at the circuit-level.

This chapter provides further details regarding the hierarchical design of PAnDA, and it also deals with the translation of the PAnDA hierarchy into both SPICE and VHDL implemen-

tations, which are consequently combined to form the PAnDA Emulator, with the ultimate goal of accelerating the characterisation of variability for the PAnDA architecture.

Given that intrinsic variability comes from the bottom layer, the modelling efforts for each of the abstraction layers of the PAnDA architecture are introduced in a bottom-up fashion.

4.2 Configurable Transistors

The bottom layer of a hierarchical model of PAnDA will begin at the Configurable Transistor level, and therefore a netlist describing a CT was created, comprising standard transistors and a custom netlist (also using only standard transistors) for the configurable clamps.

With the complete model in place, the RandomSPICE tool was used to introduce variability into the SPICE transistor models. Each time a RandomSPICE netlist is created, the BSIM model describing each transistor is replaced by another model taken from the variability-enhanced libraries, where some of the BSIM curve-fitting parameters have been changed to reflect the minor variations in the $I - V$ curve caused by variability.

The netlists designed for the work reported in [25], which resulted in Figures 3.19 and 3.20, described in Chapter 3, were used as the building blocks for this work.

4.3 Configurable Analogue Blocks

The SPICE netlists created to model a CAB did not include the Function Decoder; the configuration was hard-coded into the netlists by fixing the clamps and routing the appropriate inputs based on the function selected, as the function-switching transients are not the focus of this work. This resulted in a separate netlist for each function performed by the CAB. The transistor libraries used to model the PAnDA architecture targeted devices with a channel length of 25nm. The actual PAnDA device was designed with a 65nm process, so the created model will most likely feature a stronger effect from variability, and the variability-tolerant methodologies devised with the model will still be applicable to the device.

Since the CCAB and SCAB structures follow the same principle, this work focuses exclusively on the former, under the assumption that the modelling techniques and findings will also apply to the latter.

RandomSPICE was again used to introduce variability into the architecture through the variability-enhanced transistor models, this time looking at its effect at the CAB-level. Eight

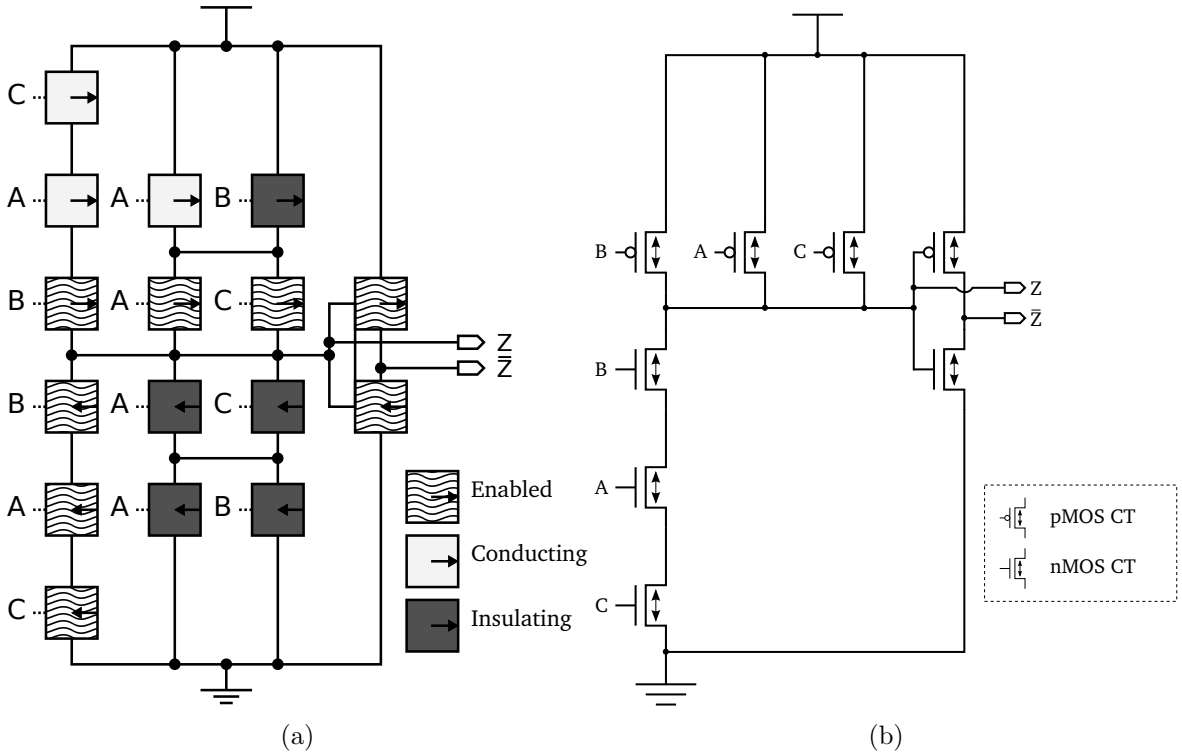


Figure 4.2: A CAB structure configured as a 3-input NAND gate (a). The state of each transistor is represented by a different block illustration, and it is this configuration which confers upon the CAB the desired functionality. (b) shows the simplified equivalent circuit.

pMOS and eight nMOS CTs are instantiated, in accordance with the schematic pictured in Figure 3.21. Simulations were run for combinations of different CT sizes and functions. As an example, a CAB was configured as a 3-input NAND gate, with every nMOS transistor sized at 240nm and every pMOS at 480nm, achieving a CMOS ratio of 1:2 (nMOS:pMOS), an arbitrary approximation of the typical 2:3 ratio used in CMOS designs. Figure 4.2 depicts the CT configuration and hard-wiring of inputs which configures the CAB structure as a 3-input NAND gate – note that the Function Decoder has not been included in the modelled netlist.

As previously mentioned, the *propagation delay* of the Circuit Under Test (CUT) has been chosen as the figure of merit which illustrates the effects of variability. The propagation delay for a rising-edge is defined to be the length of time measured from when a transitioning input signal reaches 50% of its final value to when the output signal reaches 50% of its final value. In order to fully characterise the delay of a gate, for instance, an input pattern must be constructed which stimulates the gate in such a way that all possible output transitions are covered. To achieve this goal, two groups were created: input combinations which result in a logic zero at the output of the gate; input combinations which result in a logic one

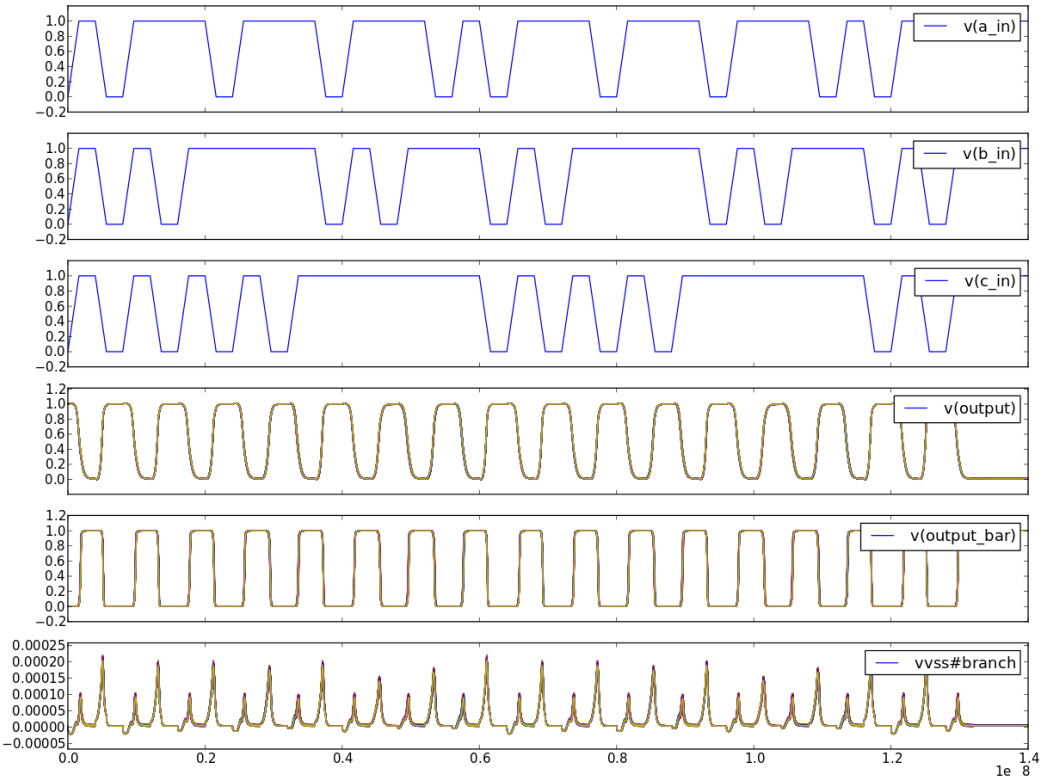


Figure 4.3: Waveforms for 300 RandomSPICE runs of a 3-input NAND gate implemented using the SPICE model for a PAnDA CCAB, with nMOS transistors 240nm wide and pMOS 480nm. The bottom waveform depicts the current behaviour at the output of the CAB. The input pattern depicted by the top three waveforms was generated so as to cover every possible output transition for a 3-input NAND gate.

at the output. The resulting sequence must include every possible combination of items from the two groups, and will differ according to the function implemented. Every possible combination of inputs which causes an output transition is covered so that all dynamic transient behaviour for each gate is extracted in the CAB characterisation process.

This input sequence computation has been applied to every function, meaning that each full characterisation of a CAB-based function has its own separate input pattern. More complex functions will have longer input sequences, whereas simple ones such as the inverter function will require short input patterns.

Variability was then introduced through the use of the RandomSPICE tool, and 300 circuit instances were created. This number of created instances is a trade-off between the simulations required for a Monte-Carlo process estimation and the limited amount of computing resources. 300 instances provides a 3-sigma process coverage, although this could potentially be extended to a larger data set in the future. The resulting output waveforms can be seen

on Figure 4.3. The CAB output voltage waveforms are oscillating because of the nature of the input pattern, which alternates between elements of the two groups detailed previously.

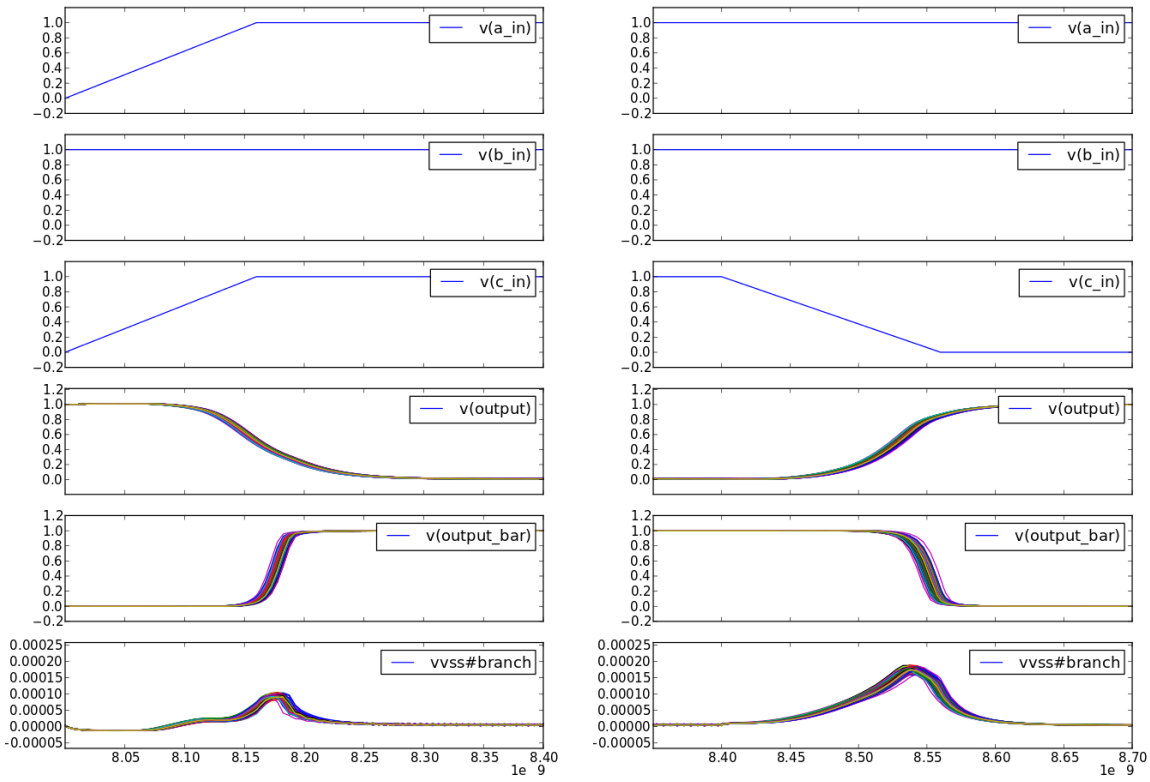
To calculate an appropriate slew rate for the input stimulus, a set of 3 CABs were connected in series – the first one providing a buffer stage for the input stimulus, and the last serving as an output load for the middle stage. This approach is undertaken because the input signals are assumed to originate from another CAB, and therefore their slew rate should be as realistic as possible. An input stimulus with a slew rate of $1V/ps$ was applied to the first CAB, and the worst case slew rate of the second CAB, measured at $6.25mV/ps$, was taken as the reference. This was done so that the slew-rate of the stimulus applied to each gate in the delay extraction process is approximately equivalent to the output slew-rate of another CAB, which would be the case on a PAnDA design.

Zooming in on some of the transitions of the full transient plot, it is possible to see more clearly the effects that variability has on the shape of the output waveform, and consequently on the gate's propagation delay, as Figure 4.4 illustrates.

Due to the structure of a PAnDA CAB, different input combinations will result in different output drive strengths. For instance, in the case of the 3-input NAND gate, when the inputs change from [111] to [000], all three pMOS CTs (labelled as *Enabled* on Figure 4.2) will drive the output from 0 to 1 in parallel. In contrast, when inputs change from [111] to [001], only one of the pMOS CTs will be driving the output to 1, resulting in a weaker drive strength and consequently a larger propagation delay.

This dependency between number of CTs involved in the transition and the gate's propagation delay is clearly demonstrated in Figure 4.5, as different levels of propagation delay are visible, according to the number of inputs that change. Transitions where only one input changes are slower, followed by those where two inputs change, and the fastest transition occurs when all inputs toggle. This can be explained by the increased drive strength resulting from more than one CT driving the output.

In an effort to fully characterise the performance of the building blocks of the PAnDA architecture, this methodology was applied to every CAB function, extracting propagation delays associated with input transitions which also result in output transitions. Across the 8 functions that each CAB can be configured to perform, the total number of unique transitions required to cover every possible transition amounts to 400. As for the CT sizes, it would not be feasible to run simulations for all of the 128 possible size configurations. For this reason, a smaller set of 5 sizes spanning the possible range from the minimum to the maximum



(a) (b)

Figure 4.4: A set of 300 RandomSPICE runs of a falling (a) and rising (b) transition of a 3-input NAND gate implemented using the SPICE model of a PAnDA CAB, with nMOS Configurable Transistors 240nm wide and pMOS 480nm. The bottom waveform depicts the current behaviour at the output of the CAB.

widths was chosen, with nMOS increments equal to the minimum width achieved by a CT of 120nm and pMOS increments designed to maintain a stable CMOS ratio, as detailed in Table 4.1. This information is extracted in this characterisation process so as to inform the digital reconfiguration resources of PAnDA to potentially allow for performance optimisation.

Table 4.1: CT sizes selected for the characterisation of variability on the PAnDA architecture.

ID	pMOS Channel Width (nm)	nMOS Channel Width (nm)	Channel Length (nm)
120n240p	120	240	25
240n480p	240	480	25
360n720p	360	720	25
480n960p	480	960	25
580n1140p	580	1140	25

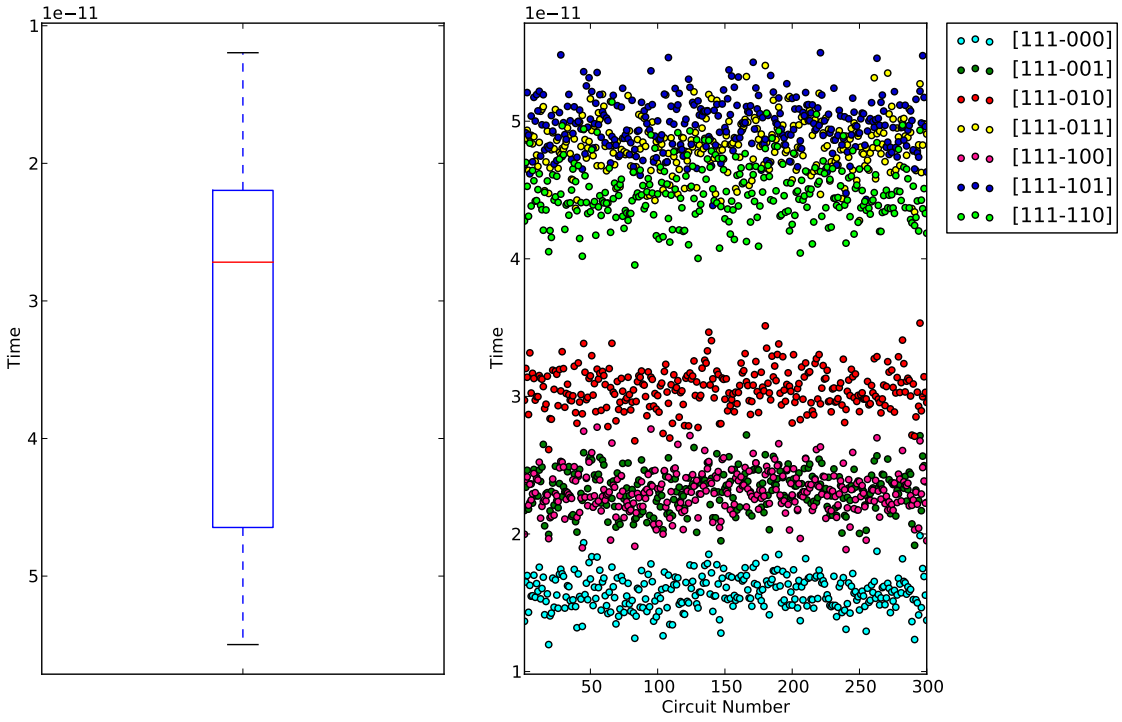


Figure 4.5: Rising edge propagation delay characterisation of a 3-input NAND gate implemented using 300 RandomSPICE simulations of a PAnDA CAB, with nMOS Configurable Transistors 240nm wide and pMOS 480nm. The scatter plot on the right expands on the left boxplot by making it possible to see the transitions associated with every propagation delay measurement.

Getting a sense for how variability affects the PAnDA architecture at the CAB-level can then be done through the extraction of propagation delays. For a given function, CAB models are created with the CTs sized according to the specified set. Using the same RandomSPICE transistor models, delay measurements are taken for every CT size of the set, and for every function specified in Table 3.1. This generates a matrix of measurements which fully characterises an instance of a CAB in terms of delays. This is called a CAB Model Card (CM Card). Using a different set of RandomSPICE transistor models will create another CM Card. With the 300 RandomSPICE transistor model combinations that have been carried out, a CM Card library is generated, populated with these 300 elements. This approach is summarised in Figure 4.6.

With this data, it becomes possible to directly compare the performance of different CT sizes for the same function implemented on a CAB. Taking the same example of a 3-input NAND gate, Figure 4.7 illustrates how the propagation delay distribution is shaped across different CT sizes. The peaks present in Figure 4.7 represent the levels first seen in Figure

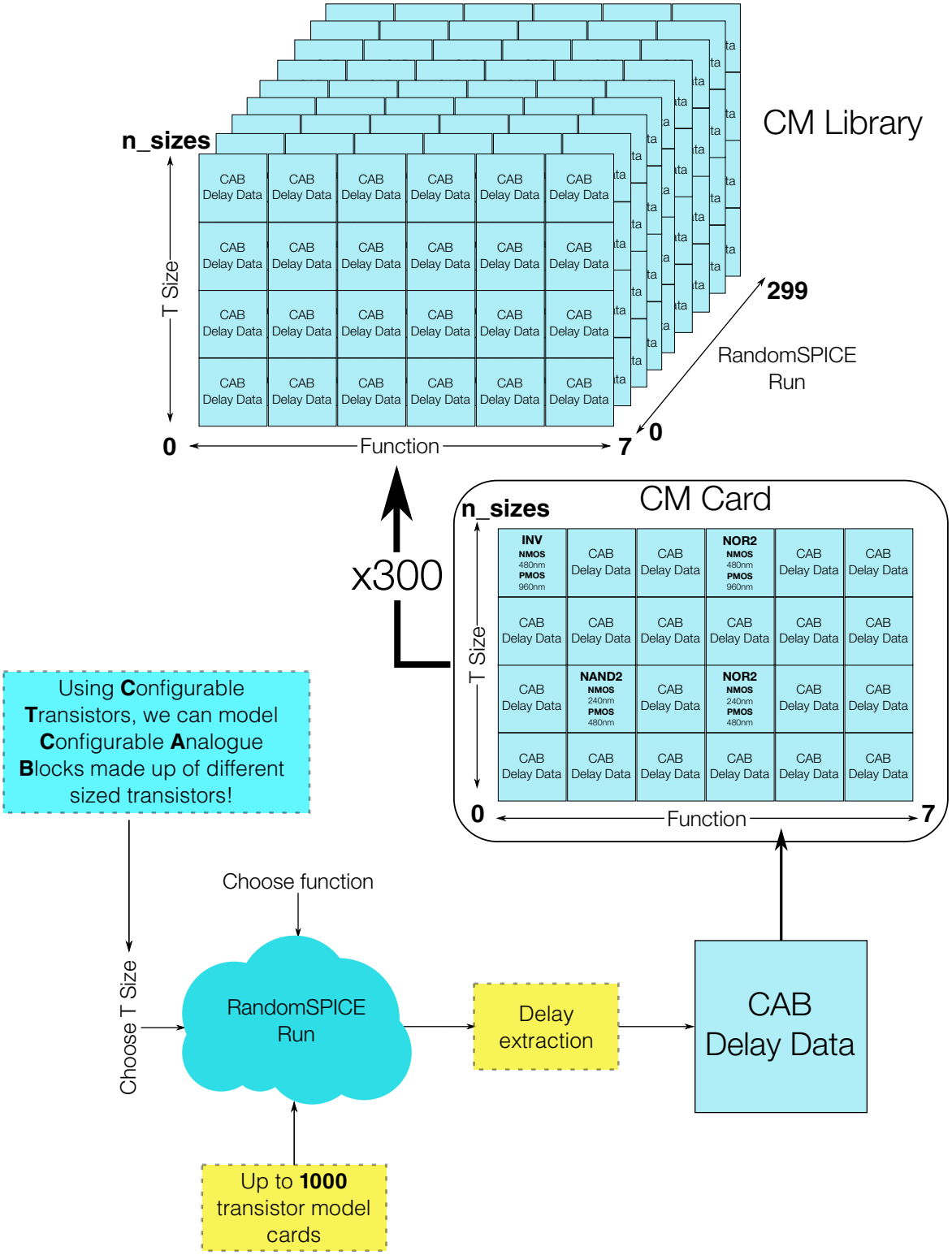


Figure 4.6: The process of extracting propagation delays from a RandomSPICE model of a PAnDA CAB, depending on the CT sizes and mapped function, and repeating it for different combinations of RandomSPICE transistor models. The end result is a library of CAB Model Cards (CM Cards).

4.5, and the variations around those values are caused by variability. Without variability, the plot would consist solely of overlapping points at the peaks.

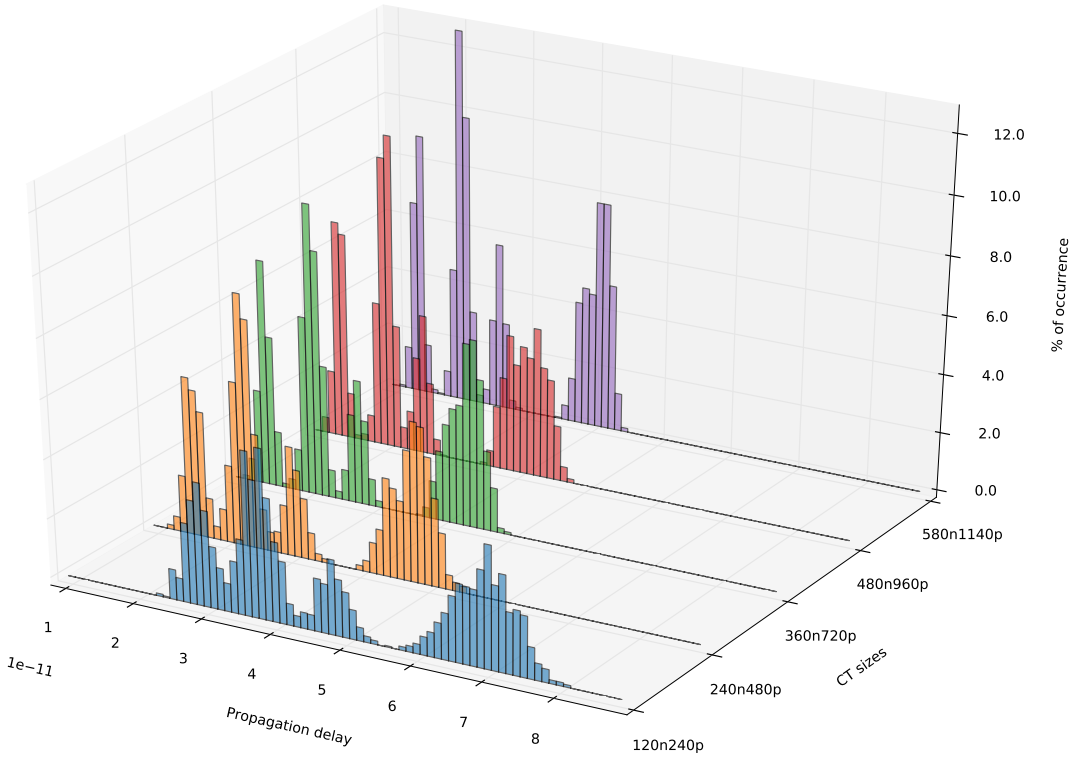


Figure 4.7: Rising edge propagation delay characterisation of a 3-input NAND gate implemented using 300 RandomSPICE simulations of a PAnDA CAB, with nMOS and pMOS CTs sized according to the specified set.

The case of the inverter, depicted in Figure 4.8, shows this more clearly, where only one peak is visible for the full characterisation. This is due to the fact that the implemented inverter only uses one pMOS and one nMOS transistor on the CAB structure (this excludes the CT-based CMOS inverter which generates the complementary output). As the CT sizes are increased, there is a clear narrowing of the distributions, as well as a decrease in the median due to the higher current drawn by the CTs by reason of their increased width.

Another interesting case is that of the AOI21 function, where the peaks only become visible at larger sizes, where the effect of variability has a smaller impact. Figure 4.9, which depicts the falling-edge propagation delay distribution for each CT size, shows that four different peaks are visible at the larger CT sizes, corresponding to different transistors being activated. However, as the CT size is reduced these peaks begin to blend with each other. This suggests that the propagation delay for a transition involving only one fast CT might be similar to that of another transition which involves two slower CTs, despite their sizes being the same.

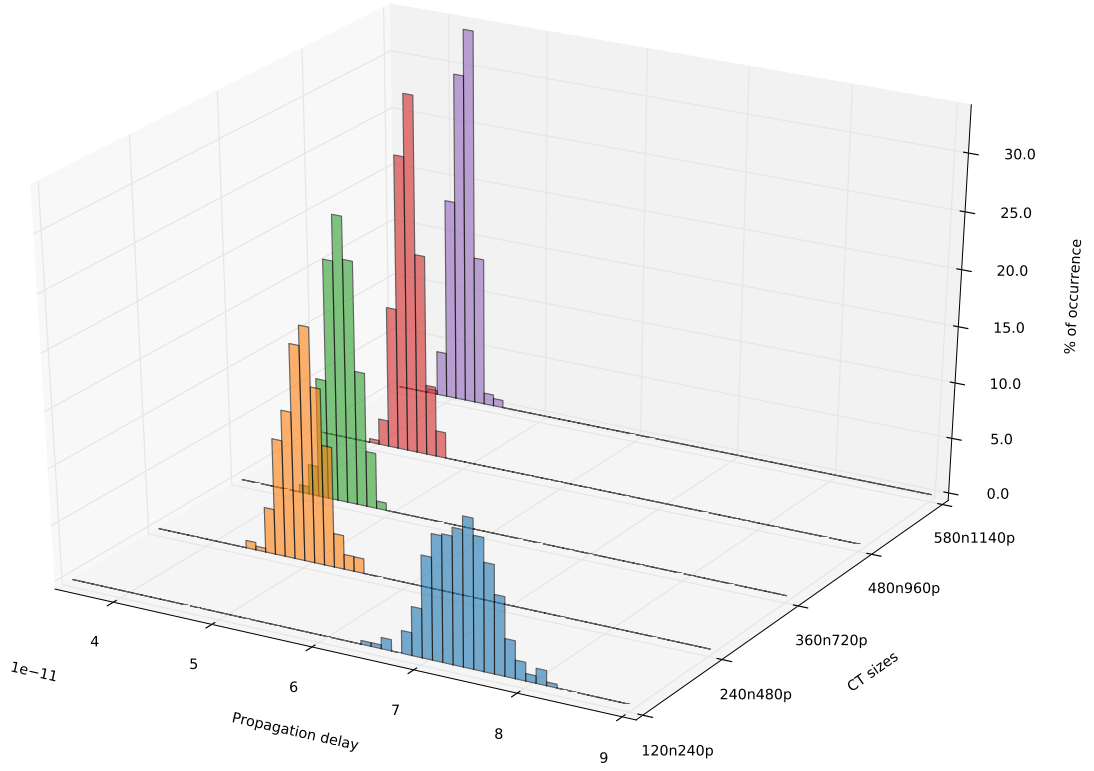


Figure 4.8: Rising edge propagation delay characterisation of an inverter implemented using 300 RandomSPICE simulations of a PAnDA CAB, with nMOS and pMOS CTs sized according to the specified set.

An additional experiment has been carried out with the aim of investigating how different combinations of transistors inside of a CT – used to achieve the same size – will behave in the presence of variability. A set of 300 RandomSPICE runs were carried out for the inverter

Table 4.2: A breakdown of the transistors used in the standard and alternative CT configurations to achieve the set of sizes specified for the experiment. An *X* denotes a used transistor, whereas an *o* represents a not-used transistor.

	nMOS Transistors used in CT (nm)							pMOS Transistors used in CT (nm)						
CT size	120	120	140	160	180	200	220	120	120	140	160	180	200	220
Size 0 std	X	o	o	o	o	o	o	X	X	o	o	o	o	o
Size 0 alt	o	X	o	o	o	o	o	X	X	o	o	o	o	o
Size 1 std	X	X	o	o	o	o	o	o	o	X	X	X	o	o
Size 1 alt	X	X	o	o	o	o	o	X	o	X	o	o	o	X
Size 2 std	o	o	X	o	o	o	X	X	X	X	X	X	o	o
Size 2 alt	o	o	o	X	o	X	o	o	o	X	X	o	X	X
Size 3 std	X	o	o	X	o	o	X	X	X	X	X	o	X	X
Size 3 alt	o	o	X	X	X	o	o	X	X	X	X	o	X	X
Size 4 std	o	o	o	X	o	X	X	X	X	X	X	X	X	X
Size 4 alt	X	X	o	X	X	o	o	X	X	X	X	X	X	X

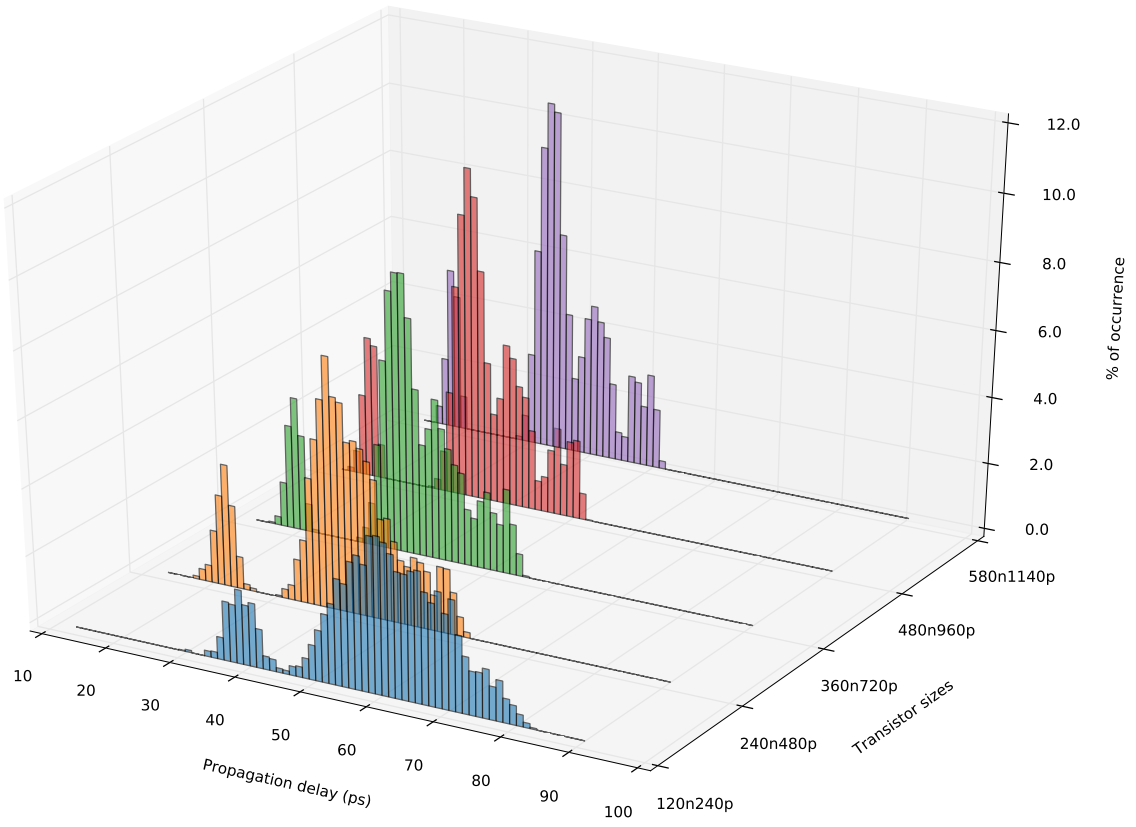


Figure 4.9: Rising edge propagation delay characterisation of a AOI21 function implemented using 300 RandomSPICE simulations of a PAnDA CAB, with nMOS and pMOS CTs sized according to the specified set.

function implemented on a CAB across each of the five CT sizes specified in Table 4.1, this time achieved through a different combination of transistors, as detailed in Table 4.2.

In some cases, as Figure 4.10 illustrates, the alternative size configuration outperforms the original in average value and spread, but it underperforms in others. These results give further substance to the argument that the redundancy provided by the individual transistors inside of a CT can be beneficial for the mitigation of variability, but only at post-fabrication. This means that at a post-fabrication stage, an alternative CT configuration can be attempted to optimise the performance of the design, but as these results suggest, there isn't a significant difference between the spread in performance obtained by each of the size configurations. A non-parametric Vargha-Delaney significance test was carried out on this data set, resulting in A-values of around 0.5 with very low p-values, suggesting that the two distributions are not significantly different from each other, and therefore that the different combination of transistors in the CT would not contribute toward a significant increase in variability-tolerance.

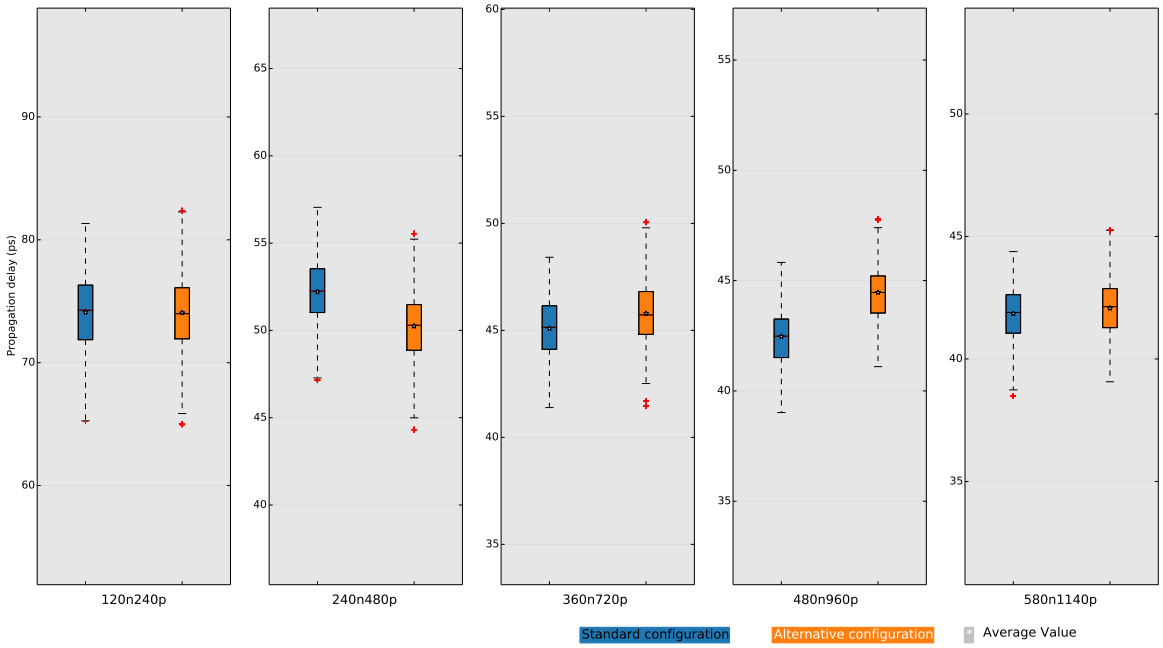


Figure 4.10: Comparison between the propagation delays of 300 RandomSPICE runs of a CAB-based inverter using the standard and alternative configurations to achieve the specified CT. The blue boxplots represent the standard configuration, and orange represent the alternative.

4.4 SPICE: A Scalability Issue

As each CAB contains a total of 16 Configurable Transistors, each comprising 7 standard transistors and a clamp to enable each, as well as a configurable clamp for the common gate, the SPICE simulation time quickly grows as more CABs are added to the model.

Figure 4.11 depicts the increase in simulation time according to the number of CABs (configured as a majority function) included in the netlist. The experiment that this plot relates to consisted in simulating a varying number of CABs connected in a tree fashion, with a simulation time-step of $5ps$ and a total run-time of $5ns$.

As Figure 4.11 indicates, the simulation time increases linearly with the number of CABs instantiated in the netlist, with each additional CAB contributing around $358s$ of simulation time overhead.

However, if a netlist includes more CABs, it is also very likely that the complexity of the circuit it implements also increases, and therefore more transitions will have to be evaluated to get a full characterisation of the behaviour of the implemented circuit.

Investigating the effects of variability on the PAnDA architecture at the circuit level rapidly becomes a task which requires an infeasible amount of time, and therefore a new method is

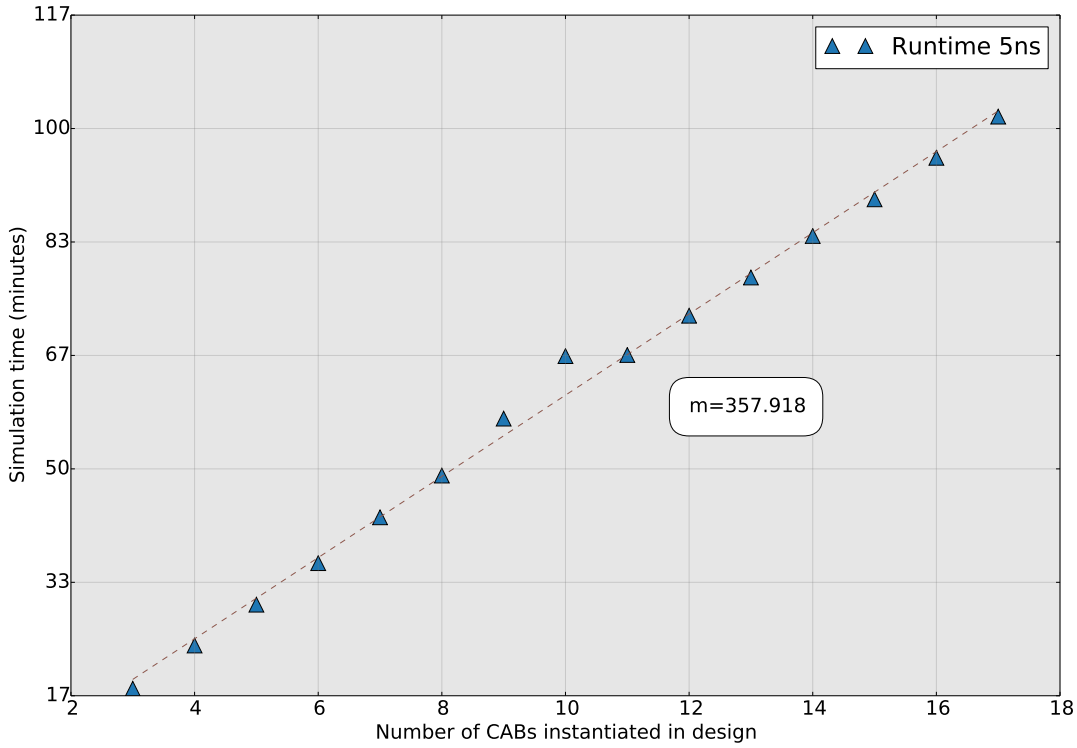


Figure 4.11: Time required to simulate a design in SPICE with varying numbers of CABs, with a $1ps$ time-step and a duration of $5ns$. The slope, labelled as m, suggests that each additional CAB represents an overhead of $358s$ in simulation time.

required which can reduce this simulation overhead. As an example, a 5 ns simulation of a device with 1,000 CABs (a number very shy of the typical logic cell count of a traditional FPGA) would take just under 100 hours to run on a single processor. Simulating 300 of these to study the impact of variability would then drive the simulation time up to 4 years.

4.5 Accelerating SPICE in Hardware

In order to address the scalability issue that a full SPICE simulation for variability characterisation presents, a different approach was required which could process the simulation data in a parallel fashion, since the root of the scalability problem lies in the sequential evaluation of the SPICE model equations by a processor.

Hardware is inherently parallel, making it a strong candidate for the platform on which the model can be accelerated. In addition, it can be beneficial to use a flexible type of hardware which can be used to investigate potential architecture changes that provide benefits to PAnDA.

For these reasons, an FPGA-based implementation of PAnDA was the chosen approach to accelerate the characterisation of the impact of variability on the architecture. On its own, such a model can be used to emulate the logical behaviour of a PAnDA device, but it cannot emulate its more analogue properties such as propagation delay and power consumption. Moreover, it does not take into account the variations imposed by intrinsic variability, like the variability-aware SPICE models provided by RandomSPICE. A more established approach such as SSTA does not take into account the specialised structure of the PAnDA architecture, which as Figure 4.7 shows, results in different delay distributions based on the transition of inputs that is used to stimulate the different configured CABs. For the purposes of *digital reconfiguration*, the properties of the transistors inside of each CAB must be retained when switching from one function to another. Although some methods have been reported which allow for the circuit-level estimation of the impact of variability [121, 13], they were not designed to accommodate the reconfigurable logic features of an architecture such as PAnDA, and the correlation between functions implemented on the same CAB. This is desirable since in the real hardware this correlation will be present, and will therefore be key for local optimisation.

4.5.1 Feature Block

To address these issues, the concept illustrated in Figure 4.12 was applied to the FPGA-based model written in VHDL. The logic set of combinational functions that a PAnDA CAB can be configured to perform, previously described in Table 3.1, are implemented natively in VHDL, using simple logic. This provides the functionality (logic) backdrop which will then accommodate the data extracted from the RandomSPICE simulations.

The outputs of a CAB are connected to a **feature block** which detects any changes in its inputs and outputs the SPICE-extracted performance measurements relating to that particular transition, stored in a block of memory. For this work, the features provided by the feature block are limited to *delay* insertion, but this could easily be expanded to include other performance metrics such as power consumption. The operation of the feature block for delay insertion consists of delaying the updating of the values of the outputs by an amount specified in memory, previously measured in simulation and relating to the transition observed by the CAB. This delay is achieved by means of digital counters, which are loaded with the value extracted from a RandomSPICE simulation.

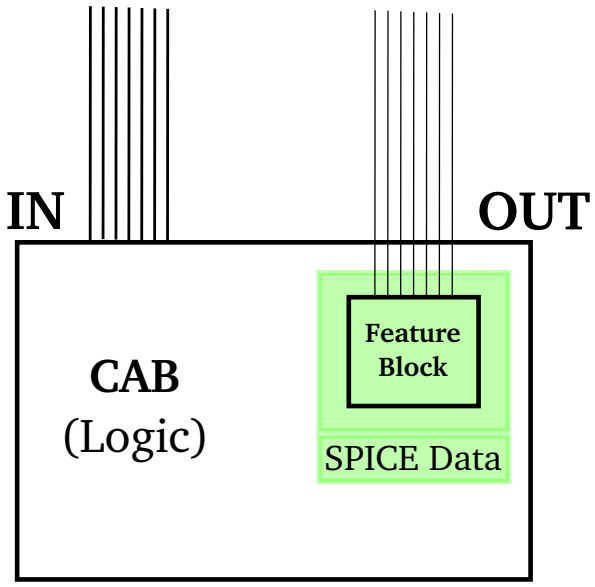


Figure 4.12: The basic concept behind the incorporation of RandomSPICE simulation data into a hardware-based model of PAnDA. The outputs of a CAB are connected to a feature block, which incorporates SPICE data stored in a block of memory. In the case of the delay characterisation used in this work, the feature block detects any changes in its inputs and delays the process of updating the outputs by an amount specified in memory, previously measured in simulation.

The process of characterising the propagation delay of a CAB based on the geometry of its CTs, the function it is configured to perform, and the occurring transition, illustrated in Figure 4.6, will generate a matrix of values which the counters can read. All of these measurements are represented in picoseconds, since the delays extracted from the RandomSPICE simulations ranged from approximately 20 to a few hundred picoseconds. The propagation delay exhibited by a particular CAB on the FPGA-based model will be proportional to that measured in SPICE, according to the ratio between one picosecond and the inverse of the frequency of the clock used to operate the digital counters. For instance, if the counters are clocked at 1MHz, then a SPICE-extracted delay of 100 picoseconds (written simply as 100 in memory, since the order of magnitude is implicitly assumed as 10^{-12}) will be measured as $t = \frac{100 \times 10^{-12}}{\frac{1}{10^6}} = 10ms$ in the hardware model.

The feature blocks comprise a finite state-machine with a digital timer/counter. This FSM detects a change at the inputs and fetches the delay corresponding to that particular transition from a block of memory, which contains the data extracted from RandomSPICE simulations. This value gets loaded to a timer, which blocks the output of the CAB from being updated until the timer elapses. This behaviour seeks to emulate the propagation

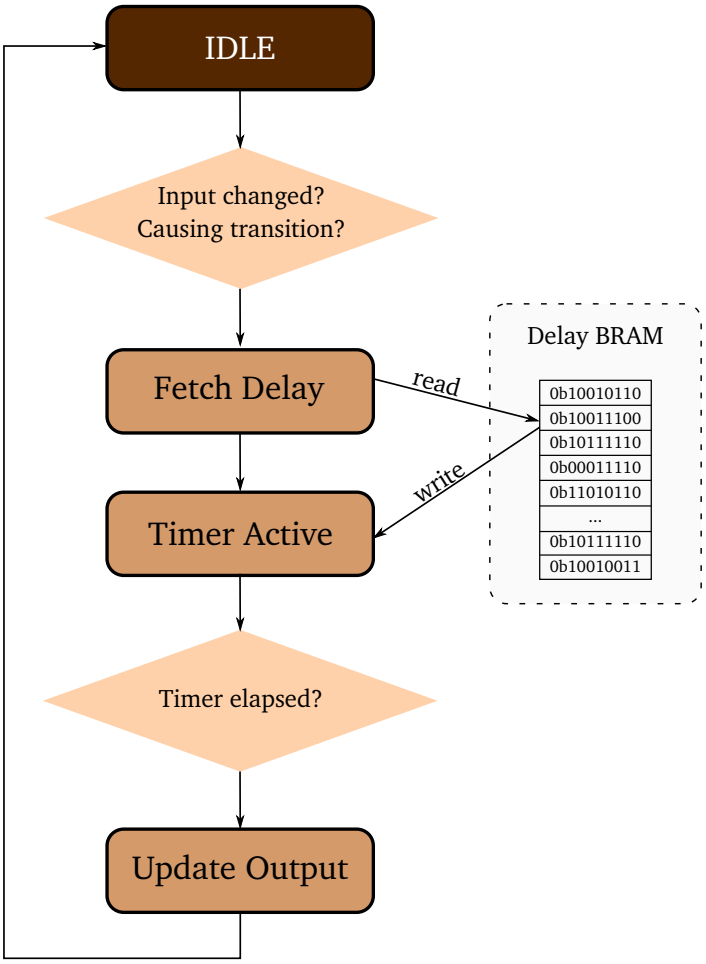


Figure 4.13: The finite state-machine which controls the operation of the feature block attached to each CAB on the PAnDA Emulator. Based on input transitions and CAB function, the value loaded to the timer will determine when the output gets updated following from change in inputs.

delays at the circuit-level which are also observed in SPICE. This process is summarised in Figure 4.13.

In a similar fashion to the way the RandomSPICE tool replaces standard transistor models in a netlist with variability-aware ones, CM Cards are chosen at random from the CM Card library and are assigned to any CAB instantiated on the FPGA-based model through the use of the digital counters. This is illustrated in further detail in Figure 4.14. Because the seed used for the RandomSPICE netlist generation are the same for every function, this ensures that the same "virtual" transistors are instantiated every time, maintaining the correlation of performance across different functions.

This approach makes it possible to *emulate* the different geometries that can be achieved with a CT, simply by loading the appropriate value to the digital counters associated with the CABs. The outcome of this methodology is a hardware-accelerated computation of the

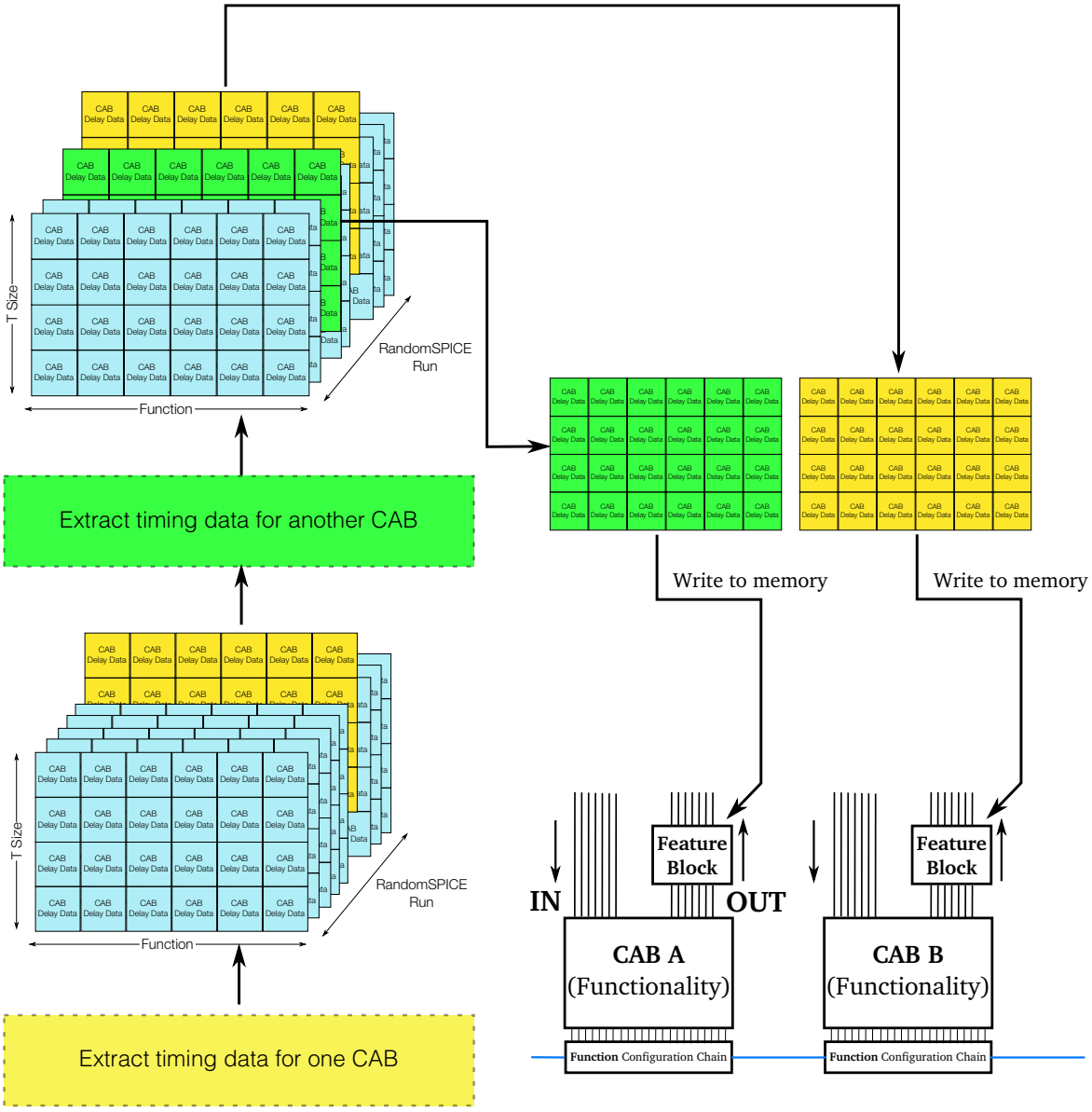


Figure 4.14: The process of randomly choosing a CM Card from the library to be written to the memory which is read by the digital counters in the feature blocks associated with CABs (A and B).

logic functionality accompanied by a set of data which provides a measure of variability at the CAB-level.

4.6 Configurable Logic Block

For the hardware model, 4 CCABs and 4 SCABs are instantiated per CLB, but since only the former have been fully characterised in SPICE, they are also the only ones which incorporate feature blocks at their outputs, and consequently only they are used in experiments. In the hardware description, these consist of 4 CCAB and 4 SCAB hierarchical modules, written in VHDL.

The switch matrix is instantiated in VHDL in the form of combinatorial logic.

4.7 PAnDA Emulator v1: A Sea of CLBs

Since the hardware model written in VHDL is hierarchical and therefore modular, any $N \times M$ array of CLB-switch-matrix pairs can be implemented on an FPGA, provided that there is enough space available on the device.

Complying with the PAnDA-EiNS topology, an 8×4 array of CLB-switch-matrix pairs is instantiated in VHDL, and the **PAnDA Emulator** is created. It is a modular and fully-parametrised model of the PAnDA architecture which incorporates variability-aware timings and which can be used to study the effects that variability has on the architecture. Different CT sizes can be *emulated* by loading appropriate SPICE timings to the feature blocks.

This approach aims to model the propagation delays of PAnDA logic gates in a circuit-level design, but it does not include delay associated with the interconnect. This is, however, something that can be included in the model at a later stage.

This was implemented on a Xilinx Virtex-6 (XC6VLX760-FF1760) FPGA.

A model of this size takes up less than 20% of the device's logic resources, which means that the model can be expanded to include more CLBs. However, if the size increases, the frequency at which it can operate will need to be reduced in order to meet design timing requirements. This is because of interconnect delays between the different modules used to implement the CABs, and the need for synchronisation between them. The larger the model, the longer the distance between two CABs at the extremes of the array, and consequently the longer it takes for a signal to be propagated between them.

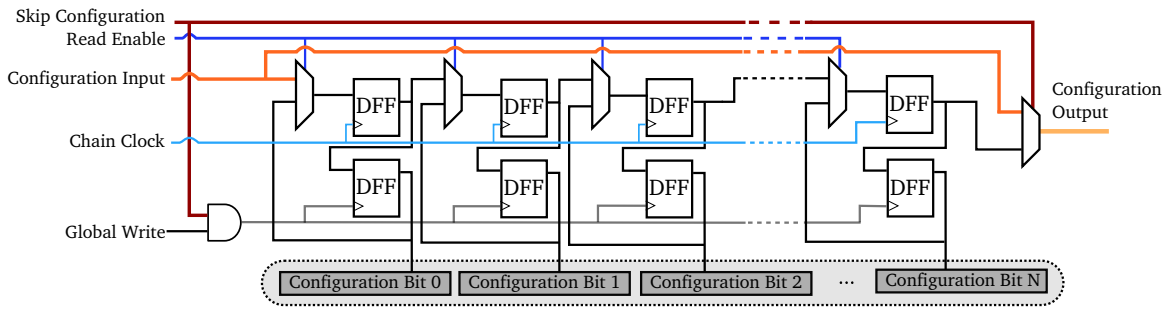


Figure 4.15: The flip-flop based configuration-chain of the PAnDA Emulator. Multiplexers control whether the bitstream is routed into the CLB or if it gets passed along without configuring it.

The shift- and carry-chains are also instantiated in the design as a series of wires.

4.8 Configuring PAnDA

In order to allow for additional features such as partial reconfiguration, desirable for optimisation tasks, the configuration of the PAnDA Emulator differs slightly from the original architecture. Although FPGA manufacturers such as Xilinx and Altera offer partial reconfiguration facilities, these usually come in the form of reconfigurable modules which need to be identified at a pre-synthesis stage, and their size is limited [122]. With the aim of allowing for the maximum flexibility for reconfiguration, a flip-flop based configuration chain is instantiated in VHDL along with the rest of the model.

The configuration chain is built in such a way that one individual CLB can be reconfigured without affecting the rest of the design, paving the way for online partial reconfiguration. This is achieved by means of multiplexers which can be configured to either route a configuration bitstream through the CLB they control or to pass it along transparently, as Figure 4.15 illustrates. A shadow-chain of flip-flops is used to shift in the configuration bits without affecting the current configuration, and the output of each register in the shadow-chain connects to the input of another register on the main chain. The bits of the configuration bitstream which control the routing and functionality are connected to the output of these main-chain registers. When a *global write* signal is active, the outputs of the shadow-chain are latched to the output of the main-chain, effectively overwriting the configuration of the CLB.

To further increase the flexibility of the configuration of the PAnDA Emulator, two separate chains have been created for functionality and routing. It becomes possible to change the

functions performed by the CABs inside a CLB without disturbing the connections between them, and vice-versa, leaving the possibility of evolvable hardware open.

To fully configure a CLB, comprising 4 CCABs and 4 SCABs, including the signals required to route the inputs and outputs along with output enables, the routing bitstream is 192 bits long. For the functionality configuration, 3 select bits are required for each of the 4 CCABs and 4 SCABs, giving a total of 24 bits for this bitstream. In total, 216 bits fully configure a CLB. For the case of the PAnDA Emulator, with a total of 8×4 CLBs, the full chip requires 6912 bits for a full configuration.

As previously mentioned, the CT geometry is not configured through the bitstream as in the original PAnDA architecture, but it is instead configured through software which populates the memory associated with the feature blocks.

A full illustration of the bitstream which configures a CLB on the PAnDA Emulator is pictured in Figure 4.16.

4.9 Summary

This chapter introduced the constructed model of the PAnDA architecture in detail from the bottom-level Configurable Transistors which provide the analogue flexibility, to the top-level structure which is similar to that of a standard FPGA. The model was created in this fashion because one of its main aims was to include intrinsic variability, which comes from the device level. The resulting features of the model at the device level should then propagate to upper layers. The modelling efforts for each abstraction layer are presented, along with the challenges that each entails.

A set of variability-aware transistor models are created and included in SPICE netlists which model the Configurable Transistors, introducing intrinsic variations. The effects of these variations include a warping of the $I - V$ curve for each instantiated CT, resulting in a broad range of $I - V$ profiles and consequently affecting their propagation delays.

Models for Configurable Analogue Blocks – which sit at the next level in the PAnDA hierarchy – are created also in SPICE, using the CT models. These are characterised for different CT configurations, providing various geometries and functions, and the effect of variability at this level is evaluated. These effects are in fact visible at the CAB level, and affect CABs in different ways, depending on the function they are configured to perform.

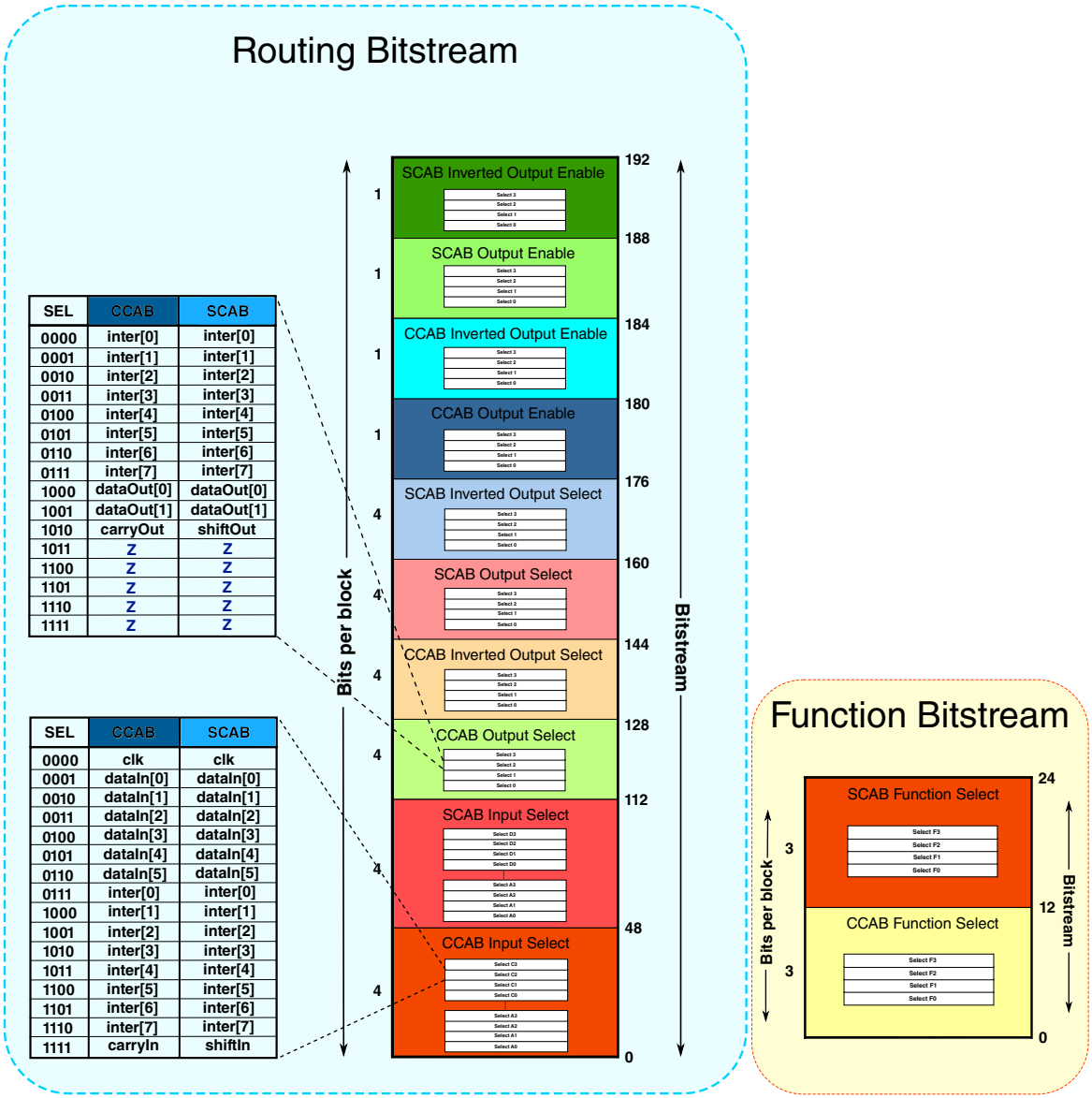


Figure 4.16: Breakdown of the bitstream required to configure one CLB on the PAnDA Emulator. 4-bit input select and 3-bit output signals are stacked, along with output enables, for the routing bitstream. The function bitstream includes one 3-bit select word for each CAB in the CLB. The numbers of the left represent the number of bits of each white box included in the bitstream, and the numbers on the right show the length increase in the bitstream as each set of blocks is added.

Motivated by these results, CAB-variability-enhanced netlists are instantiated as models, and moved up to the CLB level, where traditional logic circuit design takes place. At this CLB (or multi-CAB) level, SPICE simulation run-time begins to play a significant role, due to the high complexity of the dynamic nature of the circuits. This is tackled by exploiting the parallelism of hardware through the use of an FPGA-based emulator. SPICE data, taken

from previous modelling steps, is imported to the emulator through feature blocks which mimic the varying CAB output delays, resulting in a dynamic model.

Additional details are provided regarding the process of writing the values extracted from the SPICE simulations to the Emulator, through the use of counters associated with each CAB instantiated on the model. This methodology creates a dynamic and variability-enhanced behaviour on digital circuits, grounded on real SPICE measurements.

Finally, the configuration of PAnDA (and of the hardware model) is explained in detail. The next chapter will focus on using the Emulator for the purposes of accelerating the characterisation of variability in a number of implemented test-cases.

Chapter 5

Virtual Physical Instances and Model Accuracy

Contents

5.1 Introduction

Having created a model which allows for the characterisation of the effects of variability on mapped designs on the PAnDA architecture at the circuit level, efforts were focused on quantifying the error which arises from the simplifications that were included in the model to allow for the hardware speed-up.

With the inclusion of the variability-aware feature blocks, the Emulator shows a dynamic behaviour based on the measurements taken from RandomSPICE simulation runs. It is shown that this approach greatly reduces the time it takes to get a sense of how a particular

design will respond to the presence of variations in the CABs it comprises when compared to a full SPICE simulation, and this difference is quantified.

The methodology described in this chapter can be classified as a **pre-fabrication** approach to variability mitigation, as it enables the study of the effect of variability across the architecture for a particular technology size. Since the transistor models utilised in the RandomSPICE runs – undertaken to create the CM Card library described in the previous chapter – were for devices with a channel length of $25nm$, the results obtained from this study will also apply only for this technology size. Methods devised to cope with the effects of variability will also be applicable to larger channel-length devices, although these effects are likely to be less serious.

Two circuits are used for the experimental set-up: a 3-stage ring oscillator and a 2-bit multiplier. The former due to its typical use in hardware variability studies, and the latter to provide a more complex combinational design which can be instantiated several times in a single PAnDA-EiNS model, allowing for multiple measurements in parallel.

An in-built propagation delay and frequency measurement mechanism is designed such that these pertinent measurements can be taken for each instantiated circuit, with the purpose of assessing the variability in performance. An input pattern is generated for the Circuit Under Test (CUT) which stimulates it in such a way that every combination of inputs which causes an output transition is tested. A finite state-machine (FSM) then reads the outputs of the Emulator and determines the delay for every output. An additional mechanism is included in the FSM which measures frequency, for the case of the ring oscillator. A number of *virtual physical instances* (VPIs) of a PAnDA device are created using the Emulator, and variations in performance are measured.

In the case of the ring oscillator, the full experiment is replicated using RandomSPICE, and the results are compared to those obtained using the Emulator. For the 2-bit multiplier, this is not feasible, as each individual RandomSPICE simulation takes more than 4 hours to run, and therefore a few of the VPIs are replicated with RandomSPICE, and subsequently a comparison is made with the Emulator delay measurements.

This comparison highlights some inaccuracies in the model which are dealt with by adjusting the output load for the CABs during the RandomSPICE-based modelling process of extracting the propagation delays. This approach provides a better match with the RandomSPICE measurements without affecting the hardware acceleration of the Emulator.

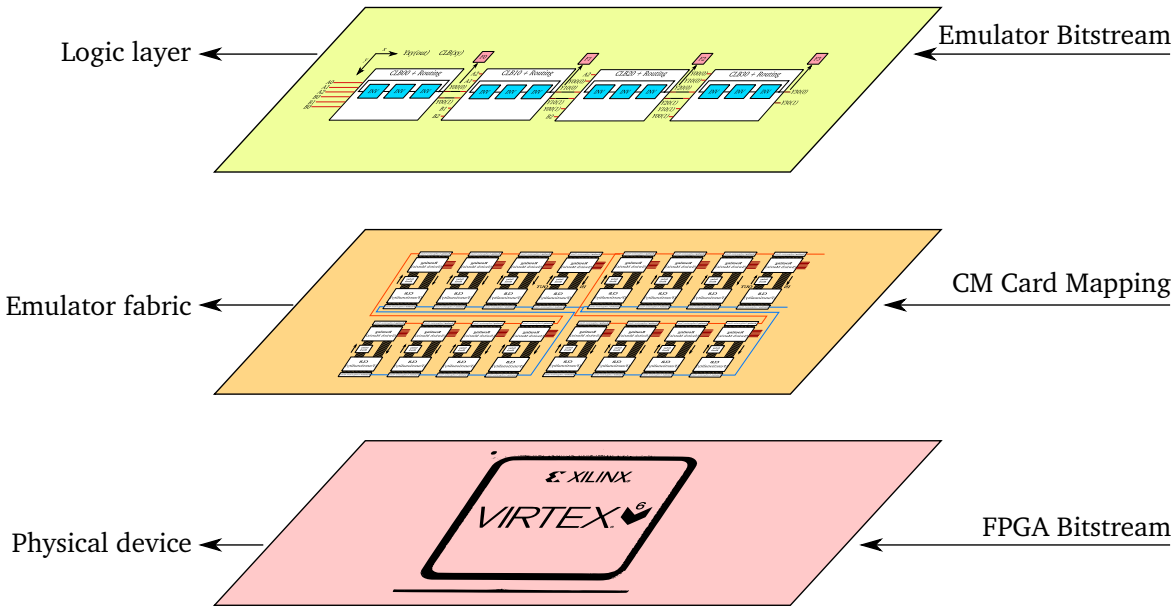


Figure 5.1: The three different layers that make up the PAnDA Emulator. The creation of PAnDA VPIs is done by configuring the top layer for a particular design, and then iterating through different configurations of the middle layer, by assigning sets of CM Cards to the feature blocks.

This chapter introduces the concept of creating *virtual physical instances* and using these for design evaluation, providing further information about how this is done on the Emulator, along with additional implementation details of the hardware. The mapping of the test circuits is shown, along with the operation of the FSM. Finally, the inaccuracies between the Emulator and the RandomSPICE models are pointed out, and the adjustments performed at the modelling stage are described.

5.2 Virtual Physical Instances

In the same way that RandomSPICE populates a netlist with transistor models taken from a variability-aware library, the Emulator populates the CABs in the hardware model with CM Cards taken from the library created for this purpose. In a RandomSPICE run, a set of netlists is created containing different transistor models; in an Emulator "run", the feature blocks are populated with different CM Cards, as Figure 4.14 illustrated in Chapter 4. Each Emulator "run" can be regarded as a **virtual physical instance** of a PAnDA chip, since it represents a potential combination of variation profiles for the CABs inside a particular PAnDA chip, and therefore a *virtual* device.

Figure 5.1 illustrates the different layers that make up the Emulator. The actual FPGA used to implement the Emulator is shown at the bottom, and it takes in a Xilinx bitstream for configuration. Above it, the feature blocks accept a CM Card which confers upon them the delays extracted from RandomSPICE. At the top, the logic resources on the Emulator establish the functionality of each CAB and routing of signals, based upon a configuration bitstream downloaded through the Emulator's configuration port.

The total number of separate propagation delays for a CAB of a particular CT size, taking into account every possible transition for each of the 8 functions it can be configured to perform is 400. As previously explained, a set of 5 different CT sizes was modelled instead of the 128, due to time constraints. This then results in a total of 2000 values for a single CM Card.

The full CM library, consisting of 300 CM Cards, will then hold a total of 600,000 different propagation delays. These values range from a few dozen picoseconds to just over 200, and therefore can be encoded using just 8 bits.

Making full use of the embedded system features of an FPGA-based implementation, a microBlaze soft-processor is included in the design for the purpose of holding the entire CM library in the form of an array of 8-bit values.

The XC6VLX760 board used to implement the Emulator does not include sufficient BRAM blocks or external RAM to hold the 4.8Mb required by the CM library (since most of the BRAM resources were already being used by the feature blocks), and so an external FPGA is added to the experimental set-up in order to store this information and also to provide some additional features, described in the following section.

5.3 Control Module

In order to provide the external memory required by the CM library, a XUPV5-LX110T board, hosting a Virtex-5 LX110T Xilinx FPGA is connected to the Emulator implemented on the Virtex-6 on the XC6VLX760 board by means of a 40-pin ribbon cable, as depicted by Figure 5.2.

This additional device stores the entire CM library in SDRAM, and communicates it to the Emulator through two 32-bit wide channels (one for writing operations and another for reading) connected to GPIO pins on both boards. It performs all top-level tasks such as

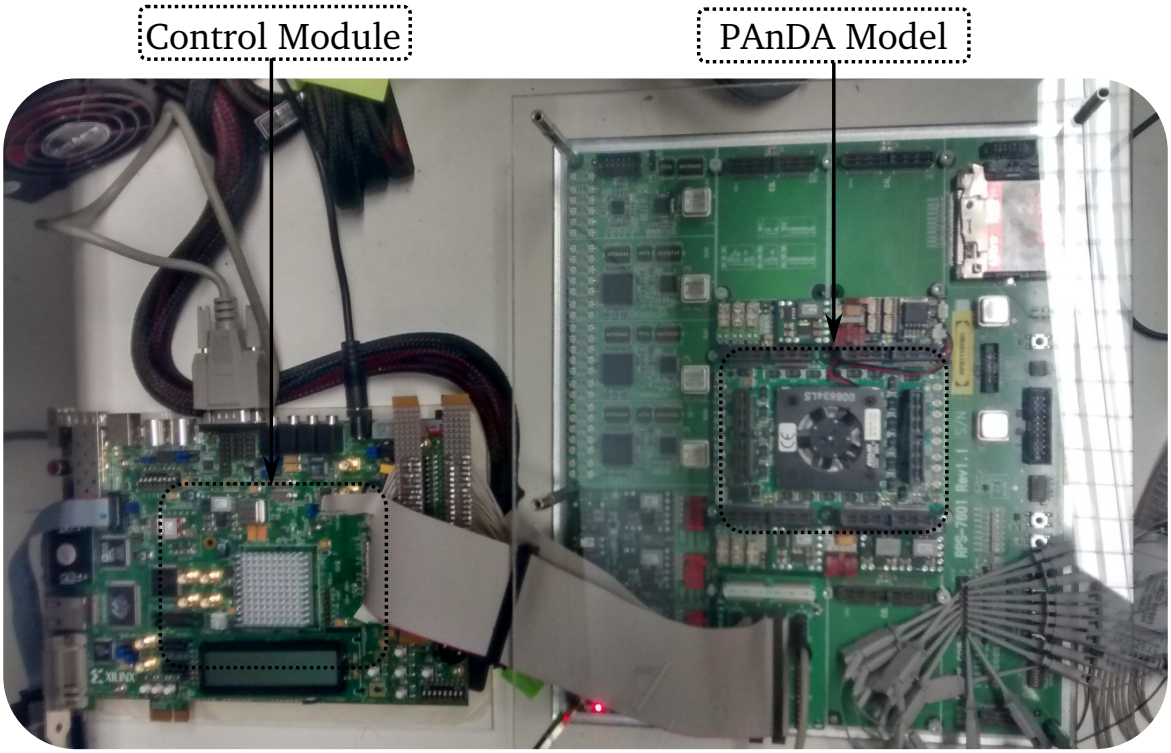


Figure 5.2: The hardware set-up for the PAnDA Emulator, with the XC6VLX760 board displayed on the right, housing the PAnDA model, and the control module implemented on the XUPV5 board, displayed on the left. Both are connected through a 40-pin ribbon cable.

shifting in the configuration bitstream and writing CM Card values to the feature blocks on the Emulator. Due to its top-level nature, it is referred to as the Control Module.

Both devices are running programs written in C through the use of an embedded microBlaze processor, and communicate based on an established message protocol, setting fields for the task, data and for a communication flag. This flag is simply a bit which gets toggled when a device has a message ready to be read by the other. When either of them registers a change in this bit, the message gets read and an acknowledgement message is sent back.

The devices operate at different frequencies, but this does not affect the communication between them, the frequency of which is limited by the Emulator, which is run at 100MHz as opposed to the Control Module, running at 125MHz.

As previously presented, the implemented hardware model for PAnDA is based on the PAnDA-EiNS design iteration, consisting of an array of CLBs organised in 8 rows and 4 columns, each with 4 CCABs and 4 SCABs. Given that only the former have been modelled, every time the Emulator is populated with CM Cards, 128 of these must be chosen from the CM library containing 300 Cards. For the purposes of creating PAnDA VPIs,

the Control Module will randomly select 128 CM Cards and pass the values along to the Emulator, each time creating a device with a unique performance.

This means that for the same configuration bitstream, a very large number of different PAnDA VPIs can be instantiated, and the differences in propagation delays will serve as a quantitative measure of how much that design will be affected by variability.

5.4 Monitoring and Measuring Variability

As previously pointed out, the variability-aware feature blocks on the Emulator cause a dynamic behaviour of the outputs, which change state based on the values loaded to the counters. This can be verified on the oscilloscope, but an internal measurement method provides further possibilities for closed-loop operation, paving the way for on-line circuit performance optimisation.

With this goal in mind, a finite state-machine based delay measurement module, written in VHDL, was synthesised along with the Emulator. This module is synchronised with the clock that is fed to the feature blocks, so it measures the Emulator's relative time-steps rather than an absolute delay.

For the 2-bit multiplier, one is interested in measuring propagation delay from the inputs to the outputs of the design, whereas for the ring oscillator the pertinent measurement to take is output frequency. For this reason, the FSM was designed to operate in two separate modes of standard delay and frequency measurement. The two modes of operation can be described as follows:

Delay Mode – the FSM detects a change at the input of the device, and begins writing the values of the outputs to a RAM block, at the same rate as the feature blocks operate at. This block is referred to as the *Output Sample* RAM. This block is as wide as the number of outputs of the Emulator, and has a depth of 1024, meaning that all outputs are recorded for a duration of 1024 time-steps. This value was chosen to be larger than the maximum delay for a 2-bit multiplier design, ensuring no transition goes unrecorded. The values of the outputs at the instant when the input transition occurred are then recorded as a reference. The FSM then reads bit i of RAM containing the sampled outputs, representing a single output, and iterates through the 1024 addresses until it finds a change in the state of that output. The actual value of the address at which this change is found is written to another RAM block at address i, and it

represents the actual propagation delay for output i. If no change is detected, the value written is the maximum of 1023, which is interpreted later as a non-changing output. This second RAM block is referred to as the *Measured Delays* RAM. The process is repeated for every output that has been recorded, and the *Output Sample* RAM is fully populated. Once this is completed, the delay values can be read by an external source, such as the microBlaze processor. The FSM then returns to its initial state, awaiting another input transition or a change in its mode of operation. Figure 5.3 provides an illustration of this mode of operation, pointing out a few examples for further clarification.

Frequency Mode – in this mode, a counter is connected to every output of the Emulator. When a rising-edge is detected, the counters are enabled, and run until the following rising-edge. The values of the counters are then stored in registers, and they represent the period of the output signals. The FSM then reads the registers one by one and stores the values on the *Measured Delays* RAM. Once again, as in the case of the Delay Mode, these values can be read by an external source. This process is repeated until the mode is changed by the user. Figure 5.4 presents a flow-chart which summarises the operation of both modes.

The FSM is written in VHDL and instantiated alongside the design of the Emulator. Many of the steps taken by the FSM require a considerable number of clock cycles, and in order to allow for a large enough margin so that the measurement is not affected by timing violations, the clock which controls both the feature blocks and the output sampling is designed to work at a frequency $100x$ slower than the Emulator. Since the Emulator operates at 100MHz, this clock is set to 1MHz.

What this means is that, as explained in Section 4.5, a propagation delay on the Emulator of 100 microseconds will represent a delay of 100 picoseconds in the device being modelled.

5.5 Test-Circuits

Two test-circuits were designed to serve as the object of the investigation, and this section describes not only their operation but also how they were implemented on the Emulator.

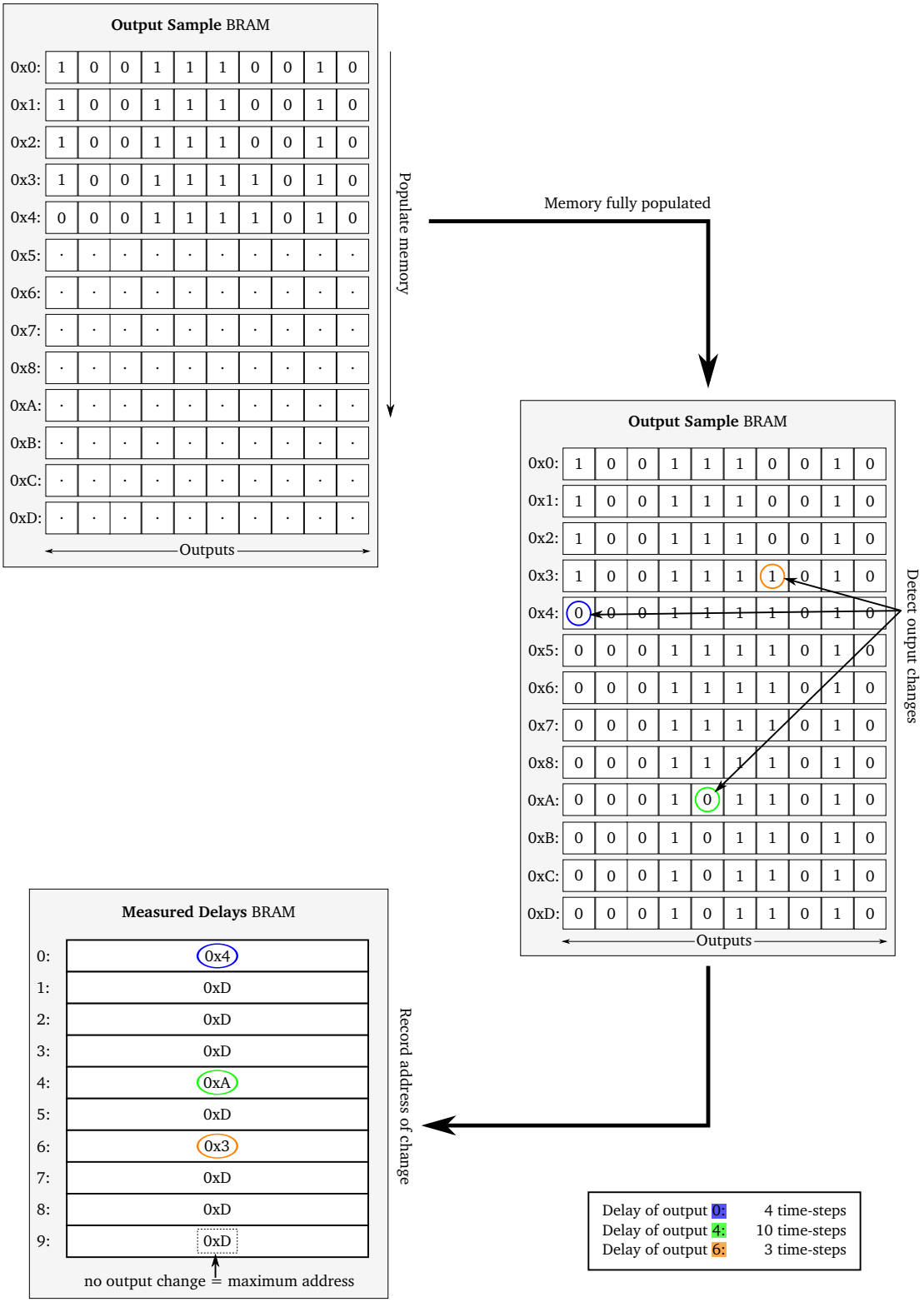


Figure 5.3: Detailed operation of Delay Mode of the implemented finite state-machine, illustrating the output sampling, transition detection and delay storage stages for each output on the Emulator. A RAM depth of 14 addresses is represented instead of the actual 1024 for simplicity of representation. The outputs at which no transition is detected are assigned a delay equivalent to the maximum address, which is later interpreted as a non-transitioning output by the processor.

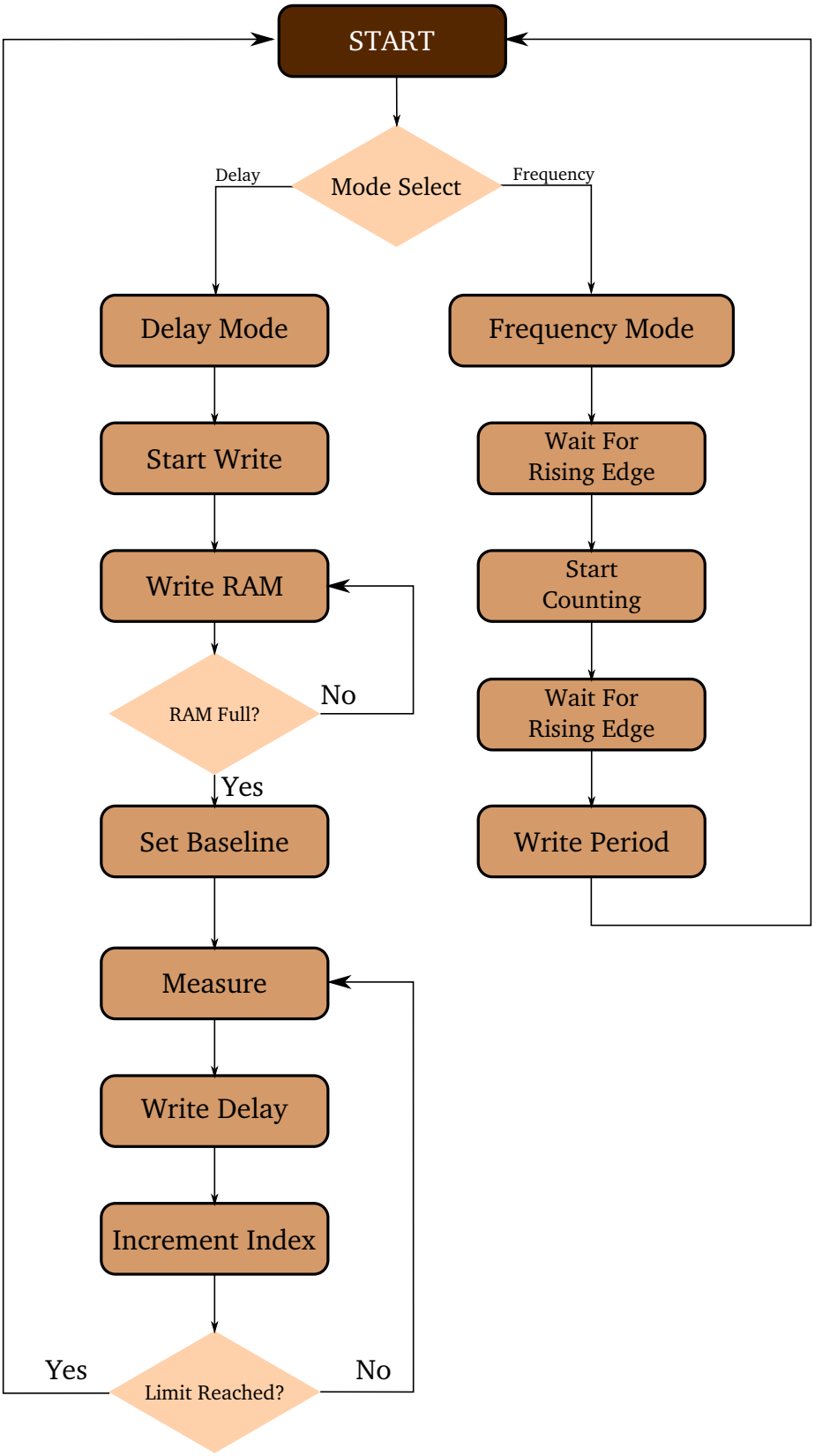


Figure 5.4: Operation of the measurement finite state-machine, which either measures the propagation of each output on the Emulator, or the frequency of the outputs of ring oscillators, depending on which mode is configured by the user.

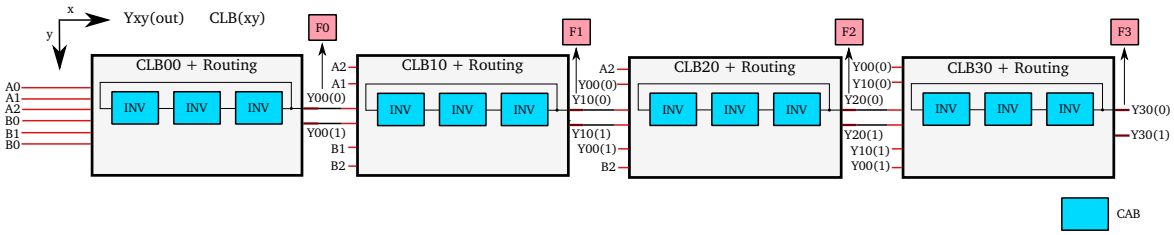


Figure 5.5: A series of four 3-stage ring oscillators implemented in a row on the Emulator, using the inverter function of a CAB. Each CAB block is represented in light blue. The outputs on which the individual frequencies are measured are illustrated as red boxes.

5.5.1 3-Stage Ring Oscillator

The first implemented design is a 3-stage ring oscillator based on inverters connected in series, with the input of the first inverter being connected to the output of the last, forming a feedback loop. This circuit (with varying numbers of stages) is widely applied for post-fabrication variability characterisation on programmable devices, since the output frequency they generate is a straightforward figure of merit for variations [58, 123, 124, 112]. This frequency depends on the delay in each stage, which in turn depends on the physical characteristics of the hardware implementing the function of the inverter (the CAB in the case of PAnDA).

Each oscillator can be implemented with one CLB, and therefore a total of 32 can be implemented on the Emulator. Figure 5.5 depicts a row on the Emulator with a 3-stage ring oscillator mapped to each CLB.

An additional advantage of using this circuit is that it is just small enough that the time required to run a series of RandomSPICE simulations becomes feasible (around 110 seconds for each oscillator), allowing for a direct comparison between the Emulator and the full RandomSPICE simulation.

5.5.2 2-bit Multiplier

As an example of a more traditional combinational design, a 2-bit multiplier is also used as a test circuit in the experiments carried out for this work. It is mapped to the Emulator using the CAB configurations and routing illustrated in Figure 5.6.

In order to provide a full characterisation of the propagation delays of a 2-bit multiplier circuit, an input pattern was designed such that every possible input combination which causes a change in at least one of the four outputs is applied to the circuit.

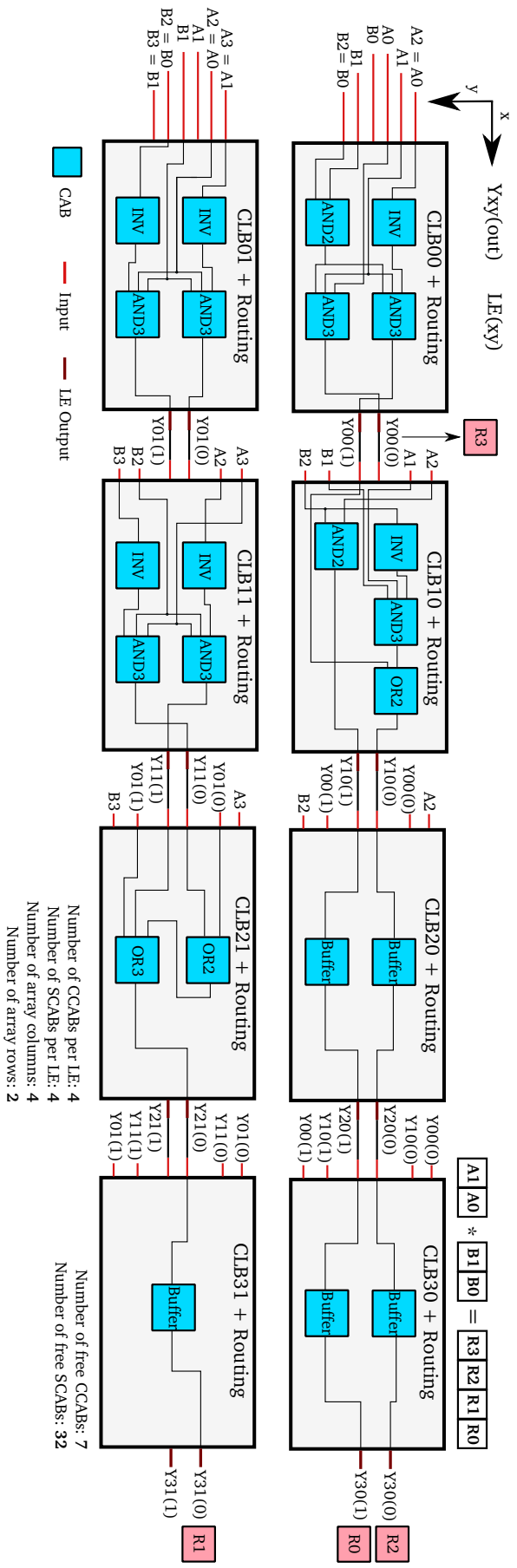


Figure 5.6: The mapping of a 2-bit multiplier function on the Emulator, using two of its rows of CLBs. It takes inputs $A2$, $A1$, $B1$ and $B0$ and outputs a four-bit number R. The outputs of the multiplier are represented by the red boxes.

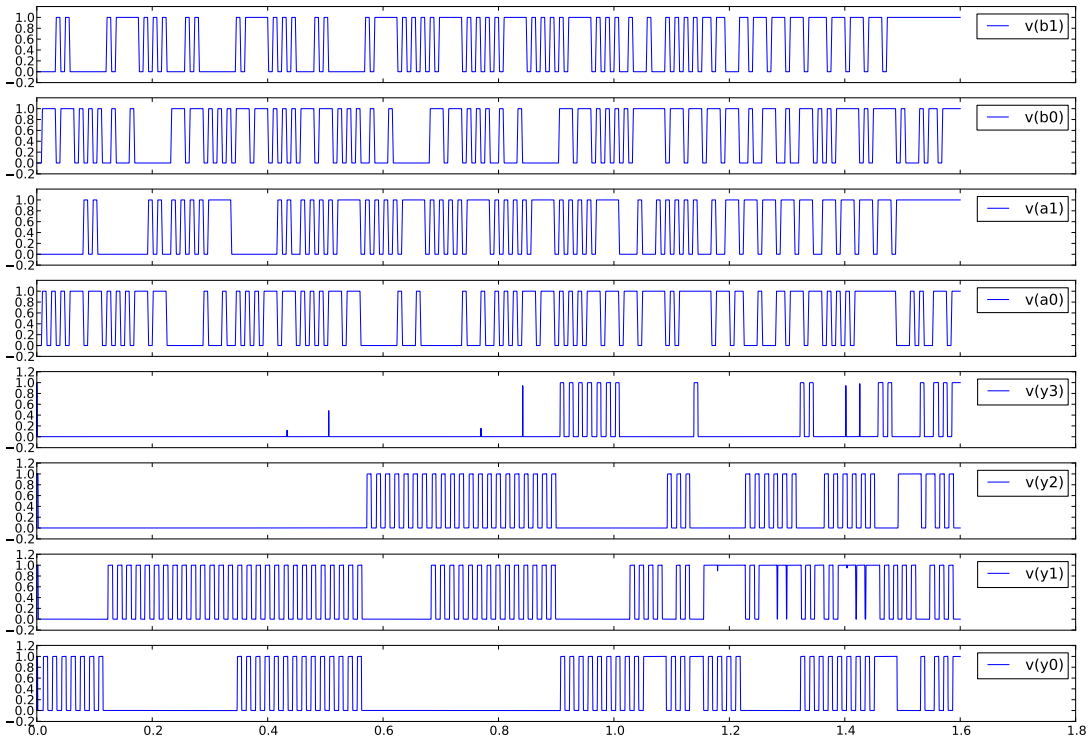


Figure 5.7: The input pattern generated for the full propagation delay extraction of a 2-bit multiplier circuit. The top four waveforms represent the 2-bit inputs A and B, and the bottom four represent the four bits of result R. All y-axes represent voltage expressed in volts (v), and the common x-axis represents time expressed in hundreds of nanoseconds.

Given that some input combinations cause more than one output to change its state, some measurements can be taken in parallel. For instance, with inputs $[A1, A0, B1, B0]$ transitioning from [0,0,0,0] to [1,1,1,1], the outputs $[R3, R2, R1, R0]$ will change from [0,0,0,0] to [1,0,0,1]. In this case, propagation delays for outputs $R3$ and $R0$ can be measured in parallel. For this reason, the total number of input combinations applied to extract each of the 324 propagation delays required to fully characterise a 2-bit multiplier circuit can be reduced down to 198. Figure 5.7 shows the SPICE voltage waveforms corresponding to the input pattern for signals A and B, along with the outputs $R3...R0$. Where two transitions overlap, the measurements can be taken in parallel.

5.6 Correlation with SPICE

With the test circuits in place, as well as the embedded variability measurement resources, the initial set of experiments can be carried out. First, a program written in C, running on the Control Module, randomly selects a set of 128 CM Cards and passes them along to another program running on the microBlaze associated with the Emulator, which then writes

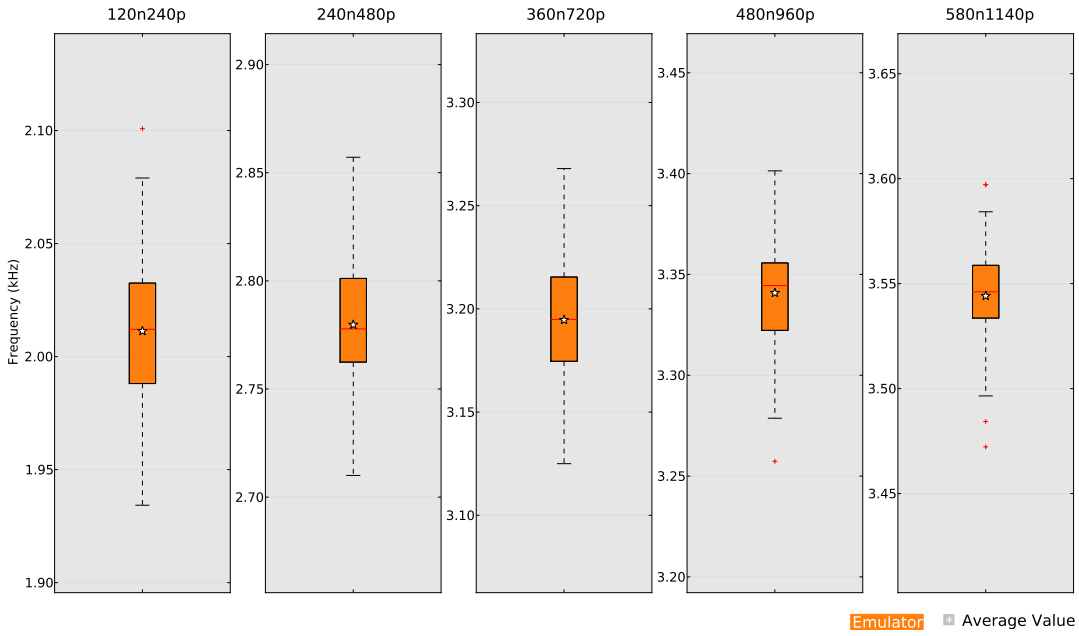


Figure 5.8: Boxplots of the distributions of frequencies generated for each CT size. Each boxplot contains 300 frequencies measured on the Emulator, with a sampling clock of 1MHz. Taking into account the scaling factor of the model, these frequencies would be multiplied by a factor of 10^6, moving them to the GHz range.

the values to the memory attached to each feature block. Once this process is completed, a PAnDA VPI has been created. At this point, the Emulator is a virtual device awaiting functionality and routing configuration.

Following from this, a series of 32 ring oscillators are mapped to the Emulator, already implemented in hardware. The Control Module sets the configuration bitstream for the Emulator, which then proceeds to shift it through the configuration-chain, as described in Chapter 4. The measurement FSM is consequently enabled, and the frequencies generated are stored in the registers. As mentioned in Chapter 4, only the first and last columns of CLBs have externally-readable outputs, and therefore only 16 of the 32 oscillators are actually sampled by the FSM.

This procedure is repeated 19 times, enough to extract a total of 300 unique oscillator frequencies, each generated by a random combination of CM Cards assigned to each inverter in the ring oscillator design. Each of the repetitions takes about 4 seconds to run on the Emulator.

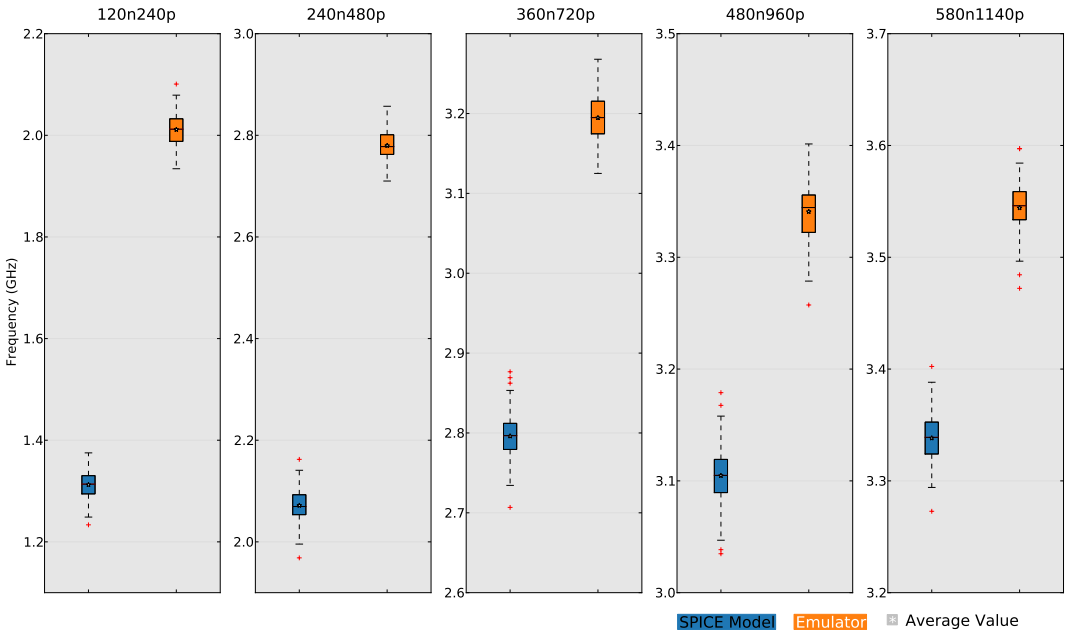


Figure 5.9: Correlation between the frequencies generated by 300 ring oscillators implemented on the Emulator and simulated in SPICE.

This experiment is carried out once for each CT size of the modelled set specified in Table 4.1, and Figure 5.8 illustrates the frequency distributions measured for each size. The total run-time required for these experiments amounted to roughly 7 minutes.

Due to the small size of the 3-stage ring oscillator design, a set of similar experiments were carried out using RandomSPICE. A hierarchical netlist comprising three CABs configured as inverters connected in series with a feedback loop at the end was created, resulting in a 3 stage-ring oscillator. RandomSPICE was then used to create 300 netlists for this circuit, each with a different set of transistor models, to introduce variability. Once again, this procedure was carried out once for each CT size specified in the set. This resulted in 300 RandomSPICE runs for each of the 5 sizes, each taking roughly 1 minute and 50 seconds to simulate, slower than the Emulator by a factor of **27x**.

The CM Cards randomly selected for the Emulator experiment are not the same used for the RandomSPICE runs, and for this reason a perfect match between the two is not expected. The means of the distributions should, however, be similar, as well as the variances. Figure 5.9 shows the boxplots for both experiments across the set of modelled CT sizes. It is clear from the comparison that the Emulator consistently generates frequencies which are considerably larger than those observed in SPICE, which suggests an inaccuracy in the

modelling of CABs for the Emulator. This inconsistency is addressed in detail in Section 5.7.

The same set-up was created for the 2-bit multiplier, shifting in the appropriate bitstream, resulting from the mapping illustrated by Figure 5.6, and setting the FSM to *Delay Mode*. Because each multiplier requires two rows on the Emulator, 4 separate instances can be mapped in parallel on the 8 rows of the Emulator, allowing for parallel evaluation and consequently reducing the time required for variability characterisation. The set of 198 input combinations described in Section 5.5.2 is applied to the design, and 324 individual propagation delays are extracted for each instantiated multiplier.

The evaluation of a 2-bit multiplier design takes around 8 seconds (with the sampling clock running at 1MHz), but when using the parallel features of the Emulator this run-time allows for the characterisation of 4 individual instances. Because this is a considerably larger design, the comparison with RandomSPICE simulations cannot be done in the same way as for the ring oscillators, since a single multiplier created in SPICE using the hierarchical CAB models takes roughly 4 hours and 40 minutes to run. Therefore, the CM Cards used for the Emulator experiments were noted down, and a separate SPICE netlist was created for each of the four instantiated 2-bit multipliers, using the RandomSPICE transistor models associated with the CM Cards they incorporated.

The total run-time required for the four 2-bit multipliers in SPICE amounted to about 18 hours and 30 minutes. When comparing with the 8 seconds required by the Emulator implementation, the full SPICE simulation is almost **four orders of magnitude** slower.

Having extracted the propagation delays for the same input combinations, a direct comparison can be made between the two implementations. Figure 5.10 illustrates a correlation plot between the extracted propagation delays for the Emulator implementation and for a full SPICE simulation of the same circuit. Each point shown on each graph represents a particular propagation delay, measured on both implementations. As in the case of the ring oscillator experiments, the measurements have been adjusted to fit the same scale. The propagation delays were measured in microseconds on the Emulator, but they in fact represent picoseconds, as Section 4.5 explained. A point representing a perfect match should sit on the line labelled as *ideal correlation* where, for instance, a 200 picosecond delay on the Emulator would correspond to a 200 picosecond delay on the full SPICE simulation.

On every correlation plot, the points are located mainly **below** the ideal correlation curve, suggesting that the design on the Emulator tends to be faster (lower propagation delays)

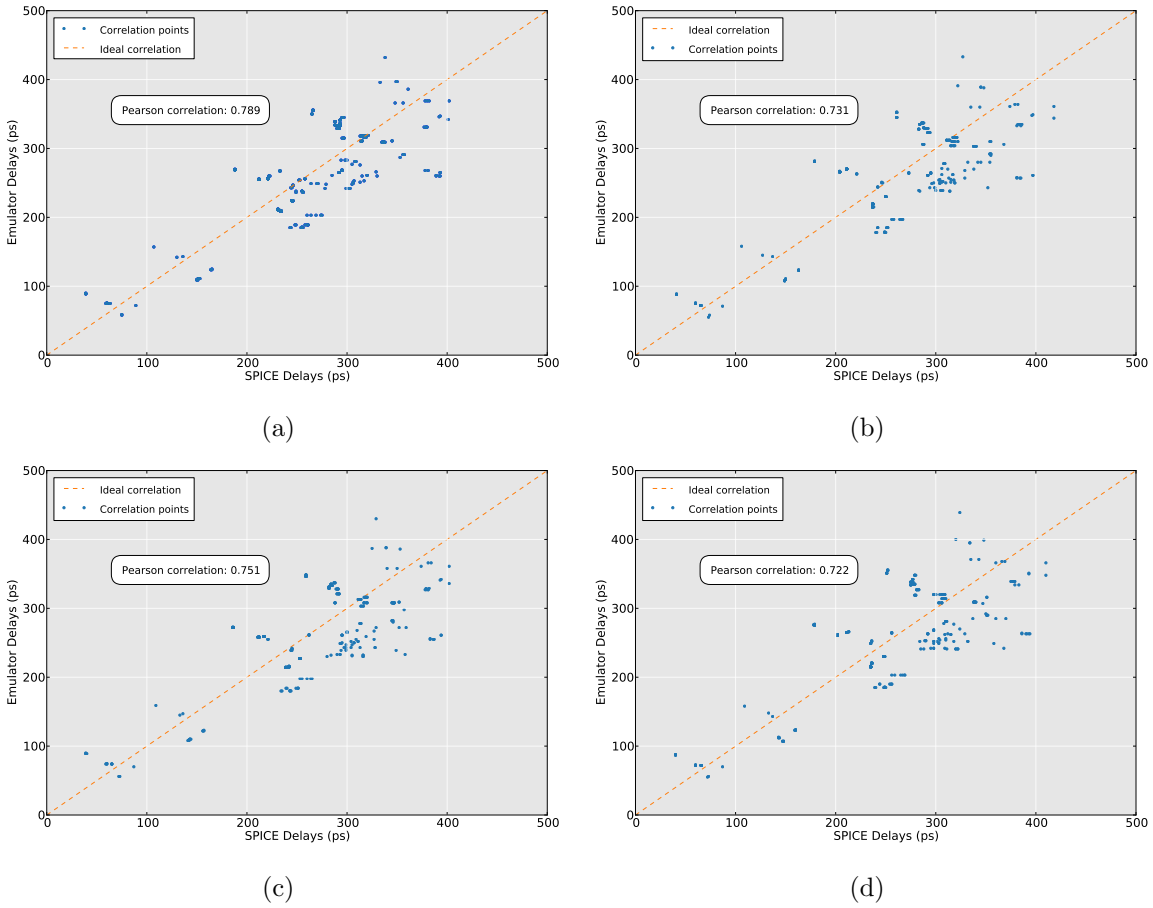


Figure 5.10: Correlation between the first (a), second(b), third (c) and fourth (d) 2-bit multipliers instantiated on the Emulator, and their respective simulations in RandomSPICE. The Pearson correlation is calculated for each multiplier.

than the SPICE simulation. As this was also the case in the ring oscillator experiments, the evidence points towards a modelling inaccuracy which causes the Emulator to be overly optimistic. Even with these limitations, the Pearson correlation calculated for the multipliers is approximately 0.75 on average.

The next Section explores the causes of these mismatches, and provides details about how these have been addressed.

5.7 Inaccuracies in FPGA-Based Model

As the previous section demonstrated, although the Emulator provides a hardware-accelerated platform on which the study of variability can be performed without prohibitively long simulation run-times, the accuracy of the obtained results is not high enough for it to be considered a viable alternative to a full SPICE simulation of a particular design.

With respect to the ring oscillator experiment, Table 5.1 shows the difference in means between the distributions obtained on the Emulator and SPICE, which in turn are illustrated in Figure 5.9.

Part of this mismatch between the results has been found to exist due to the different output load incorporation in the CAB modelling stage. In the full SPICE simulation, each CAB configured as an inverter sees another inverter (comprising the same size CTs) at its output, since they are instantiated side-by-side on the netlist describing the 3-stage ring oscillator. In the CM Card extraction stage, where only one CAB is modelled at a time, a 1 femto Farad capacitor was connected to the output, regardless of the size configured on the CTs, which does not reflect the actual varying load that the full SPICE simulation naturally models. The mismatch has more of an impact on the frequency generated by the oscillators using the smaller sized CTs, suggesting that the 1 fF load was smaller than the real load seen by each of the inverter stages in a ring oscillator configuration.

Additionally, the extraction of delays for a CAB configured as an inverter is done by applying a set of different combinations of inputs in order to stimulate the output by using every possible combination of transistors. This input signal had a slew rate of 6.25mV/picosecond, which actually corresponded to the output slew rate of a 3-input NAND gate with CT sizes set to $360nm$ for nMOS and $720nm$ for pMOS. The slew rate of the input signal to a CAB should depend on both CT size and function of the CAB directly upstream, i.e. connecting its output to the CAB under test. The CT size of the CAB under test should also influence the slew rate of its inputs, as it is in fact the output load seen from the upstream CAB.

The next Section presents some of the modelling adjustments carried out in order to improve the accuracy of the Emulator.

Table 5.1: Comparison between the means of the distributions of frequencies generated by the 3-stage ring oscillators implemented in SPICE and on the Emulator, for each modelled CT size.

nMOS (nm)	pMOS (nm)	SPICE mean (GHz)	Emulator mean (GHz)	difference (%)
120	240	1.31	2.01	34.76
240	480	2.07	2.78	25.47
360	720	2.8	3.19	12.48
480	960	3.10	3.34	7.07
580	1140	3.34	3.54	5.81

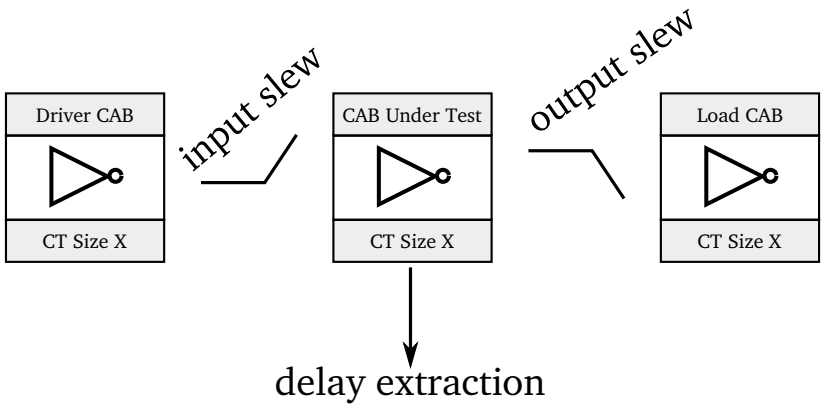


Figure 5.11: The revised delay extraction set-up, with the slew-rate of the input stimulus and the output load both being provided by CABs with CTs of the same size.

5.8 Adjustments to the Model

Having observed a significant mismatch between the delays and frequencies generated in the Emulator and in SPICE, a set of adjustments and subsequent experimental verification was carried out in order to increase the accuracy of the Emulator.

The first consisted of quantifying exactly how much the mismatch between the Emulator and SPICE is affected by varying output loads and input slew rates. To achieve this, 5 different slew rates were extracted from SPICE, each measured by instantiating two CABs configured as inverters connected in series, comprising CTs of the same size, in a netlist. The input of the first inverter had its inputs simulated, and the slew rates output of the second inverter was measured. As the inverter configuration only uses two CTs in a CAB, this set-up generates two different slew rates: one for a rising-edge and another for a falling-edge. The resulting average of these two measurements was used as the reference slew rate for the input stimulus of a CAB configured as an inverter.

The CAB netlist used to extract the CM Cards for each CT size was then modified in two ways: the slew rate of the input signals was approximated to be that resulting from another CAB configured as an inverter, using the same CT size, connected upstream; a CAB comprising CTs of the same size was added to the output, serving as a load. The CM Card extraction process was then repeated for these adjusted netlists. Figure 5.11 illustrates the delay extraction process with the adjusted input slew-rate and the added CAB to serve as a load for the CAB being characterised.

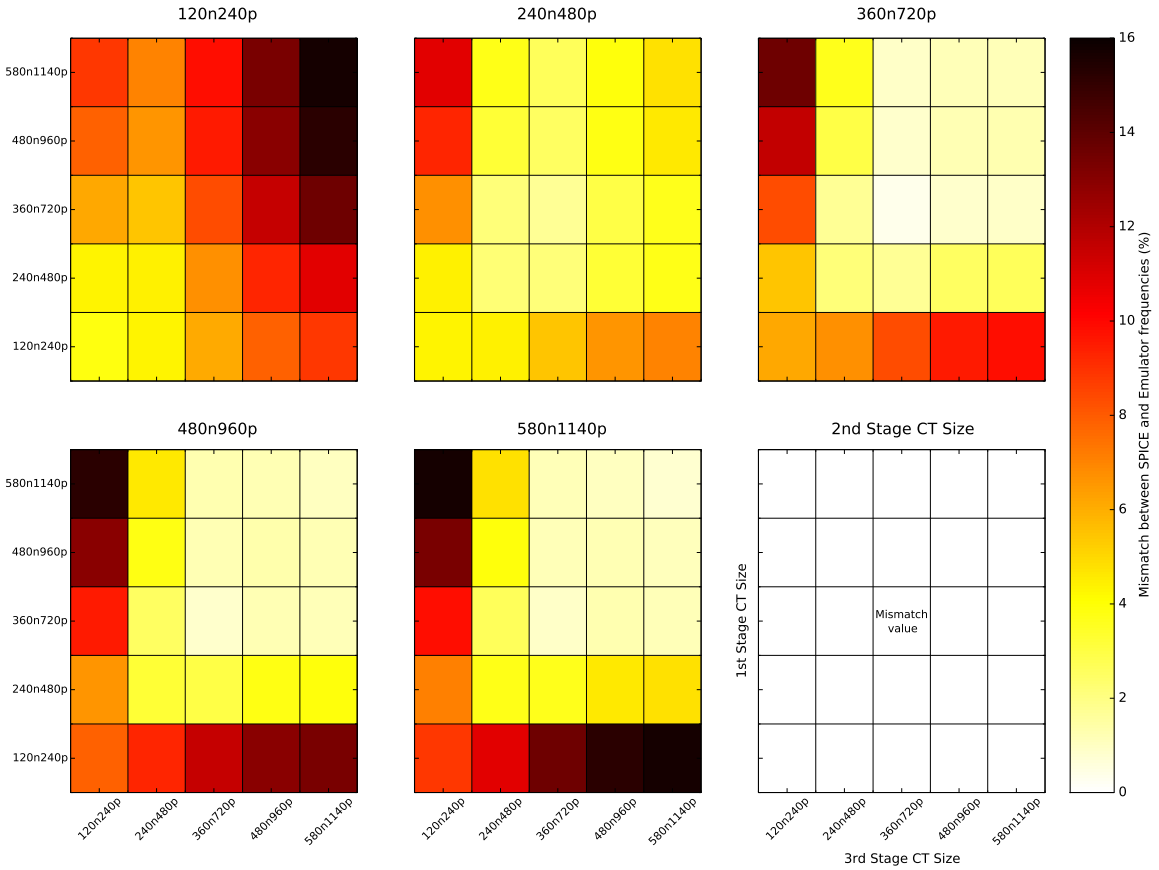


Figure 5.12: Characterisation of the error emerging from different combinations of CT sizes for each of the 3-stages of the ring oscillators. The bottom right figure shows how the graphs should be read: the top legend above each graph shows the CT size of the second inverter stage; the x-axis on each graph shows the CT size of the 3rd inverter stage, and the y-axis depicts the CT size of the 1st inverter on the oscillator. The error is plotted through the use of a heat-map, with lighter areas representing a higher error.

Subsequently, a set of 3-stage ring oscillators were simulated both on the Emulator and in SPICE, using every possible combination of CT sizes for each stage, using the CM Cards updated to include appropriate output loads and input slew rates.

Figure 5.12 shows a heat-map of the mismatch between the generated frequencies measured on the Emulator and in SPICE. It suggests that a very accurate match occurs when the CT sizes of the CAB making up each stage in the oscillator are configured to be of the same size. This is not unexpected, as the approximations used for the modelling (*i.e.* using the input slew rate corresponding to an inverter of the same size at the previous stage, and an output load corresponding to an inverter of the same size at the subsequent stage) actually correspond to the ring oscillator configuration being modelled.

The largest mismatch occurs when there are inverters of the minimum size (120n240p) and of the maximum size (580n1140p) included in the same ring oscillator. This is expected, as there will be a large difference between the assumed output load used for a particular CM Card in the propagation delay extraction stage, and the actual output load downstream. For instance, the CM Card of a 120n240p inverter at the first stage connected to a 580n1140p inverter at the second stage will assume a 120n240p load. Additionally, the CM Card for this 580n1140p inverter at the second stage will assume an input driver of another 580n1140p inverter, where in fact it is minimum sized.

Table 5.2 shows the results of the experiment described in Section 5.7, where 300 instances of 3-stage ring oscillators were simulated in SPICE and on the Emulator, repeated after the adjustments were included. Figure 5.13 presents the distributions of the 300 ring oscillators simulated for each of the 5 CT sizes. Although the mismatch is still present, these results show how the modelling accuracy of ring oscillators where all stages use the same CT size is greatly increased when comparing against the results obtained before the model adjustments. Whereas for same-size ring oscillators the maximum mismatch observed was of about 35%, the slew rate and load adjustments reduce the maximum mismatch to about 16% (maximum value observed in Figure 5.12), even when mixed-size oscillators are taken into account. The persisting error can be derived from the fact that the 1:2 CMOS ratio chosen for the CT sizes results in a different value for rising- and falling-edges. This issue is addressed in Chapter 6.

When attempting to generalise this approach for circuits which contain CABs configured as many different functions, approach has a few limitations. As previously mentioned, the input slew rate used for the input signals of a given CAB X depends on several factors: the function and CT sizes used on the CAB connected to the input of CAB X, and the function and CT sizes of CAB X itself. The output load will depend on the function and CT sizes on the CAB which CAB X connects to.

Table 5.2: Mismatch between the frequencies generated by the 3-stage ring oscillators implemented in SPICE and on the Emulator, for each modelled CT size, before and after the CM Cards were updated to include appropriate output loads and input slew rates.

nMOS Size (nm)	pMOS Size (nm)	no adjustment (%)	with adjustment (%)
120	240	34.76	3.17
240	480	25.47	4.53
360	720	12.48	0.15
480	960	7.07	0.28
580	1140	5.81	1.41

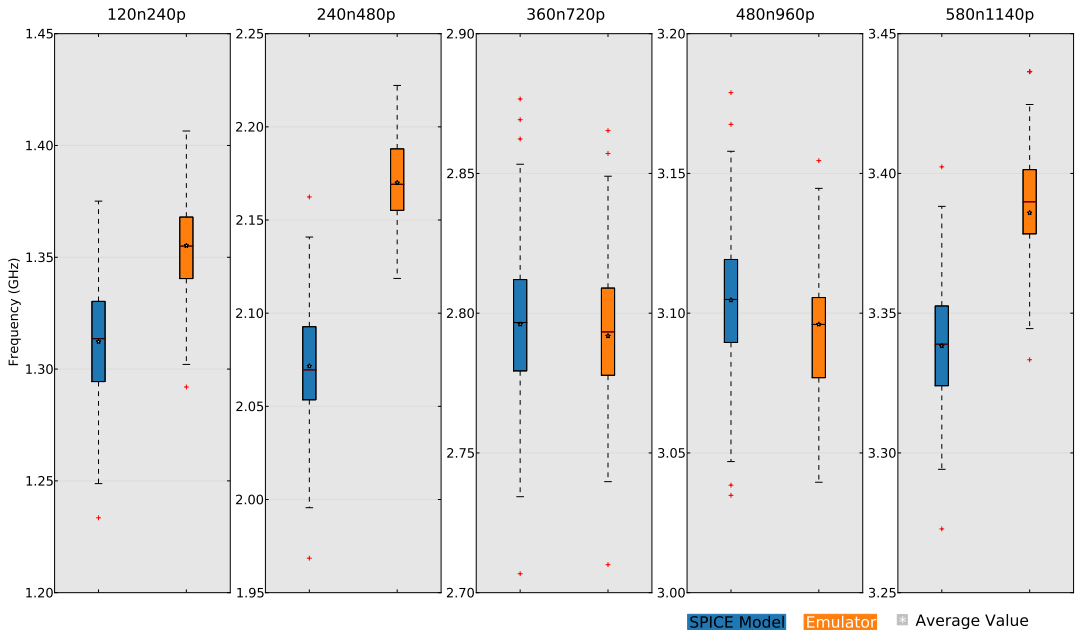


Figure 5.13: Correlation between 300 ring oscillators implemented on the Emulator and simulated in SPICE, after corrections applied to the load and input slew rate during the modelling stage.

In the case of the ring oscillator, the inverter function is common to all CABs, and therefore the mismatch in modelling will depend on the varying output loads and slew rates resulting from the CT sizes in each CAB. It would then be possible to increase the accuracy of the model by expanding each CM Card to include propagation delays measured for each combination of output load and input slew rate. Adding this extra dimension would transform the CM Card from the square represented in Figure 4.6 to a cube, and the CM Library to a hyper-cube. The Emulator could then select the appropriate section of the CM Card for a particular CAB, based on the connections established by the configuration bitstream, and the CT sizes of each CAB involved in the design. This approach carries with it a significant limitation, as when the set of CT sizes is doubled, the number of different combinations of input slew rate and output load will quadruple, greatly expanding the size of the CM Card, and consequently the memory resources on the FPGA which store the CM Cards will soon be fully exhausted.

Taking into account varying input slew rates (which depend on function implemented, CT size and fan-out of the CAB upstream) for each input, and varying output loads (which depend on fan-out of the output line, and also function implemented and CT size of the CAB downstream) for each output, the number of CAB Delay Data blocks (as depicted in

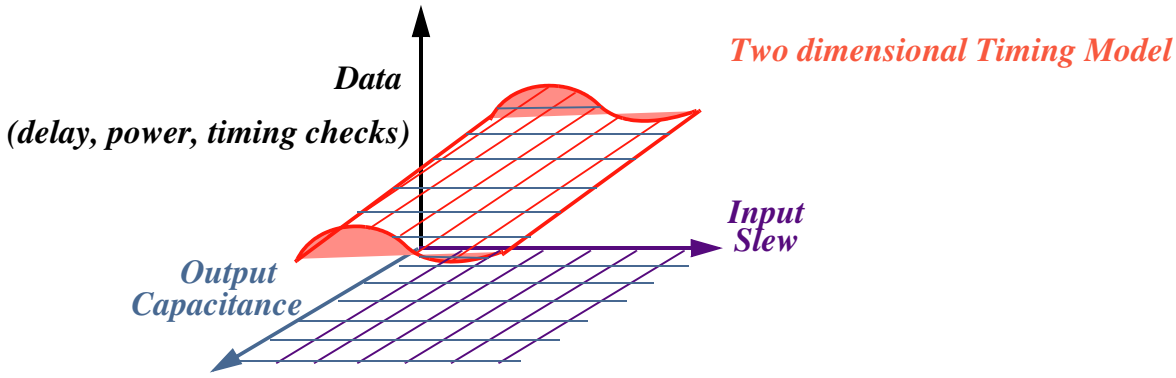


Figure 5.14: Extraction of delays and other features of VLSI standard cells, based on input slew rate and output capacitance, to create Liberty files which are imported to an ECAD tool to enable the identification of timing violations.

Figure 4.6) required to fully characterise a PAnDA CAB with this new set of requirements would very quickly grow to a combinatorial explosion.

The simplifications allowed for in the experiments presented in this chapter, such as assuming a constant fan-out of 1 and an equal input slew rate for all inputs, keep the combinatorial explosion contained, but the price is paid for in accuracy.

Some of these limitations are only related to PAnDA, as in a standard architecture which cannot change the sizes the transistors which make up its logic gates, N would take the value of 1. Further simplifications can be made if the functions are performed by hardware Look-Up Tables, as is the case of standard FPGAs, making this approach even more viable.

In fact, techniques such as Static Timing Analysis (STA) and Statistical Static Timing Analysis (SSTA) make such simplifications to allow for a quick identification of timing violations on FPGA designs, as described in Section 3.4.4.

In VLSI design, Liberty files are used for the characterisation of standard cells, consisting of matrices which estimate the delay of a particular cell of a given geometry based on its input slew rate and output capacitance, as shown in Figure 5.14.

Liberty files are incorporated into most Electronic Computer-Aided Design (ECAD) tools to allow a designer to check for timing violations on a given circuit. Creating Liberty files for the PAnDA architecture would be considerably more complicated, as varying transistor geometry necessitates the addition of at least one other dimension to the two-dimensional Timing Model illustrated in Figure 5.14. By including variability, yet another dimension would need to be added to the model.

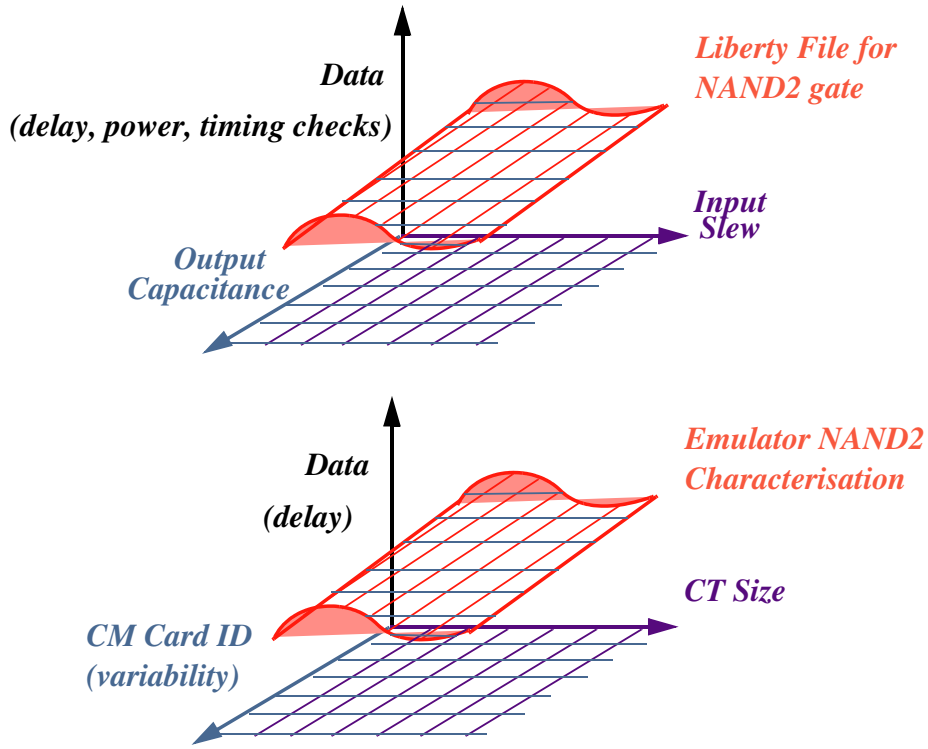


Figure 5.15: Comparison between the data included in a Liberty file associated with a NAND2 logic cell and a CAB configured as a NAND2 on the Emulator.

Figure 5.15 illustrates how the data generated for a particular function in the version of the Emulator designed for the experiments described in this chapter (a NAND2 is used as an example) is similar to a standard Liberty file, where output capacitance and input slew rate are replaced by CM Card ID (or variability run) and CT size used.

Enhancing the accuracy of the model would necessitate the inclusion of varying input slew rates and output loads, as the ring oscillator experiments suggest, which would present a new set of technical challenges regarding the memory utilisation on the hardware model.

This technical limitation of memory justify the creation of a second version of the Emulator which replaces the CT size variation with fan-out modelling, allowing for increased fidelity to the full SPICE simulation. This second version of the Emulator is described in detail in the next chapter, along with the experiments carried out with it.

5.9 Summary

After the modelling stage which resulted in the generation of the CM library, the model of the PAnDA architecture is implemented in hardware, named PAnDA Emulator, along with in-built measuring methods and processor-based control and configuration.

A 3-stage ring oscillator and 2-bit multiplier designs are used as testbenches on which the characterisation of the effects of variability on circuit propagation delay is carried out. A measurement finite state-machine capable of extracting both delay and frequency measurements is included in the design.

A considerable speed-up is achieved for these two designs, of about **one** order of magnitude for the 3-stage ring oscillator and **three** orders of magnitude for the 2-bit multiplier. This speed-up is directly proportional to the complexity of the design, providing further justification for the inclusion of hardware acceleration in the model.

A number of PAnDA VPIs are created, utilising different combinations of CM Cards on CABs instantiated on the mapped designs, and feature extraction is carried out. Equivalent experiments are run in RandomSPICE, and the results are compared. Mismatches between the Emulator and the equivalent RandomSPICE simulation are identified and investigated, and subsequently a set of adjustments are made to the model in order to incorporate more appropriate input slew rates and output loads at the CM Card generation stage, only concerning the ring oscillator experiment due to its reduced size. Although the results suggest an increased fidelity to the full SPICE simulation when the adjustments are in place, these require an infeasible amount of memory resources to be added to the hardware currently available.

The next chapter deals with the further enhancement of the accuracy of the model with respect to SPICE by adjusting the model to include a varying load. Furthermore, it describes the methodology used for circuit performance optimisation, a **post-fabrication technique**, in this variability-aware model.

Chapter 6

Mitigating Variability With The PAnDA Emulator

Contents

6.1 Introduction

This chapter deals with the application of a bio-inspired algorithm to optimise circuit performance using the Emulator. This is done in terms of *operation-point matching*, where 2-bit

multipliers and ring-oscillators are used as the designs of choice, and more general minimisation of propagation delays of d-type latches and C17 benchmark circuits. The former is a simplification of the scenario in which multiple chips are fabricated on the same die, and their performance is expected to be standardised. The latter is a classic optimisation case, where each "virtually fabricated" instance of a circuit is optimised so as to deliver the best possible performance.

The first set of experiments is concerned only with a single instance of a PAnDA VPI, where the performance of all circuits on it are standardised. The second set deals with large number of VPIs, and performance optimisation across the entire distribution of virtual devices.

For each of the test-circuits on the second set of experiments, 1000 PAnDA VPIs are created and their performance is evaluated. Due to the reduced size of the d-type latch circuit, the same number of netlists were created with RandomSPICE and subsequently simulated in full2, with the intention of verifying the accuracy of the results obtained with the PAnDA Emulator v2. Although the correlation with SPICE is considerably increased with respect to the results described in Chapter 5, modelling inaccuracies are still present, due to different input combinations resulting in the activation of a different number of CTs in a CAB, causing a rise- and fall-time variations which is not taken into account during the modelling stage. A discussion is presented in this Chapter regarding this topic, and future adjustments are suggested which can further bridge the gap between the Emulator and SPICE.

In an effort to demonstrate the **post-fabrication** circuit performance optimisation abilities of the PAnDA architecture, a Genetic Algorithm (GA) is used to minimise the worst-case propagation delay of both test-circuits, by finding an optimal CAB placement for the cells used in the designs. Despite the modelling inaccuracies, the GA will improve the performance of a design implemented in the PAnDA architecture, given that variations are present and measurable.

6.2 Bio-inspired Circuit Design

A conventional engineering task consists in receiving the specifications and designing a device or method which provides a satisfactory solution to a problem. But what if the specifications aren't completely detailed? What if the environment changes abruptly, resulting in a new set of specifications? And even if the environment doesn't change, how optimal is the devised solution?

These questions become increasingly relevant as the effects of intrinsic variability begin to have a significant impact on the performance of electronic designs, as Chapter 2 demonstrated. Nature appears to have solved this issues over time, ensuring the survival of organisms through the process of adaptation [125].

Bio-inspired hardware [126, 127, 128] provides a promising solution for these emerging problems. It takes inspiration from Evolutionary Algorithms and applies those principles to create electronic circuits. In a purely analytic problem tackled by an EA, each solution is a set of parameters; with this approach, each potential solution is an actual circuit.

To evaluate the fitness of each solution, two methods can be used:

Extrinsic evolution – the fitness is extracted from a simulation of the candidate;

Intrinsic evolution – candidate solutions are actually implemented in hardware, and their fitness extracted through actual physical measurements.

Bio-inspired algorithms have even influenced the design of antennas, leading to high-performing devices being sent to space [129, 130, 131]. Taking inspiration from the swarm behaviour of organisms such as bees and ants, the heuristics which govern these swarms have been interpreted and applied to antenna design, resulting in high-performance devices which would not likely be explored by a human designer [132, 133].

Bio-inspired hardware presents a promising approach to dealing with the issue of getting electronics to work in a dynamic environment. It necessitates two key elements: a reconfigurable substrate where solutions can be tested, and an Evolutionary Algorithm to provide the test solutions. Not only does it aim to provide a configuration which provides optimal performance for a particular environment, but it is also capable of evolving circuits which fit a particular task. Field-programmable devices are usually the preferred substrate candidates, and an Evolutionary Algorithm can be used as the solution generator.

This concept gained a lot of attention in 1996 when Adrian Thompson evolved a tone discriminator on an FPGA using fewer than 40 logic gates, which is a very low number for such a circuit [134]. Using a Xilinx XC6216, Thompson allowed for "unconstrained intrinsic hardware evolution", whereby a Genetic Algorithm was used to create a circuit which determined whether the frequency of an input signal is either 1kHz or 10kHz. The evolved design was very unconventional but compact, with a particular set of cells not connected to the output. It was found that removing this set would actually impair the functionality of the

circuit, suggesting that although not directly connected to the output, this set of cells was actually playing a part in the physical dynamics inside the device. Once the same bitstream was loaded to another FPGA, the circuit did not work. These results made it clear that human design expertise alone does not cover the entire solution-space of a circuit design task, potentially leaving out local solutions as the one uncovered by this experiment.

In summary, circuits can be evolved in two different ways:

Circuit Topology – a task is specified, the logic building blocks (e.g. logic gates) are listed, and different combinations of these elements are evolved to find the solution which satisfies the termination condition.

Performance Optimisation – the topology of the circuit is known, but the object of evolution is the circuit parameters such as transistor size and drive strength, in an attempt to find the optimal configuration of these parameters which results in the circuit with the best performance.

Regarding circuit topology evolution, one of the most popular algorithms used is a graph-based extension of GP, Cartesian Genetic Programming (CGP), developed by Julian Miller [135]. This approach takes a digital circuit and translates functions, inputs, outputs and internal connections into integers. Each building block of a given circuit will have an integer for the function it performs, one for each input and another for each output. The full design can then be described by a genotype made up of these integers. With this representation, it becomes straightforward to evolve circuits, simply by manipulating the genotype. As an example, this technique managed to come up with a design for a 3-bit multiplier which uses 20% fewer gates than the best known human design [136].

Lukáš Sekanina has also reported the use of genetic programming to globally optimise the number of gates in circuits which have already been synthesised using common methods [137]. Similar bio-inspired approaches have been undertaken to evolve IP cores on FPGAs [138], with some of these targeting the recent Zynq System-on-Chip platform [139, 140], developed by Xilinx. Improvements to CGP runtime have also been reported in [141].

A more analogue approach has been undertaken by Martin Trefzer [120], who used the Field-Programmable Transistor Array (FPTA) – described in detail in Chapter 3 – as well as Adrian Stoica who worked on the JPL FPTA at the same time [142], to evolve logic gates and comparators with a finer-grained control over a solution's analogue properties, paving the way for hybrid architectures such as PAnDA. This approach evolves not only the circuit

topology, i.e. the connections between transistors to match a particular logic behaviour, but also figures of merit such as drive strength, speed, and power consumption.

James Walker [143, 144, 145, 9] and James Hilder [73, 146] have worked on evolving digital circuit topologies with standard cell optimisation for variability. The Multi-Objective Toolkit for Intrinsic Variability Aware Transistor-Level Evolutionary Design (MOTIVATED) was created, which aimed to evolve circuit topologies through the use of CGP, and then optimise the design for variability tolerance by testing out combinations of BSIM transistor models for each of the gates that are part of the evolved design. The variability in each gate is modelled with RandomSPICE, described in detail in Chapter 2. This approach allowed them to have a system which can potentially be incorporated with the standard EDA flow to find the design implementation which matches the given specification with the highest tolerance to substrate variations.

Overall, the concept of bio-inspired design methods can provide two major contributions to the hardware design community:

Design beyond human limitations – as Thompson's tone-discriminator and Miller's 3-bit multiplier examples show, design in bio-inspired hardware can sometimes outperform a trained specialist.

Adaptation and fault-tolerance – when Evolution is allowed to perform online monitoring and parameter optimisation, it can explore the hardware to find an alternative implementation of a particular function, once events such as faults or changes in the environment cause obstructions to its operations [31].

6.3 Mitigating Variability with Digital Reconfiguration

The work described in this chapter aims to apply the concepts of bio-inspired design to variability-aware performance optimisation, through the use of a hardware model which includes statistically-aware performance data of the lower-layer of the modelled architecture. One of the novelties of the Emulator consists in allowing analogue performance data to be included in a hardware-based model, and this chapter focuses on making use of that substrate, the modelled architecture's reconfiguration resources, and optimisation algorithms to reduce the spread in performance caused by intrinsic variability.

Although there are still differences in the results obtained from the Emulator when compared to a full SPICE simulation, as discussed in Chapter 5, these are issues that could be addressed in the future. When the model is refined, performance variations of some magnitude will most likely still be present. The mismatch between the Emulator and SPICE will make it more difficult to investigate alterations that could be made to the architecture since the model does not fully represent it. This has an impact on the initial ambition of building a model which can perform pre-fabrication optimisation, by methods such as identifying crucial CTs in the architecture, or even a more optimised set of transistor sizes. With the inaccuracies present in the current version of the model, this task becomes less feasible.

However, variations between virtual instances are still present in the model. These variations will depend, on a first instance, on the CM Cards used to create PAnDA VPI. Secondly, they will depend on the mapping of the design to the Emulator's logic resources, *i.e.* which logic functions are used. Thirdly, the size of the CTs used in the CABs that take part in the active logic of the design will also cause variations in the performance of the circuit. And finally, empty CABs inside a CLB can be configured to replace another active CAB, most likely utilising a different CM Card and consequently causing further variation in the circuit's performance. This section deals with finding an optimised mapping and sizing for a particular virtual instance of a circuit. That is to say, once a PAnDA VPI has been created, how can the reconfiguration resources of PAnDA contribute toward finding an alternative implementation of a given circuit which improves its performance on the device?

This section provides details on how these reconfiguration resources can be exploited particularly to improve the performance of a given circuit, and also generally to shape the performance curves created by variability.

6.3.1 Functionally-Neutral Operations

It should be made clear that this work does not focus on logic synthesis efforts, which would explore alternative implementations of a given circuit through the use of different gate combinations. Instead, efforts are concentrated on the series of changes that can be made to a synthesised logic circuit on the Emulator, without affecting the functionality. It is assumed that an already mapped circuit is available, as illustrated in Figures 5.6, 5.5, 6.15 and 6.18, and any optimisation which takes place on the Emulator does not affect this mapping. The Emulator can make use of two major operations for its circuit performance optimisation efforts:

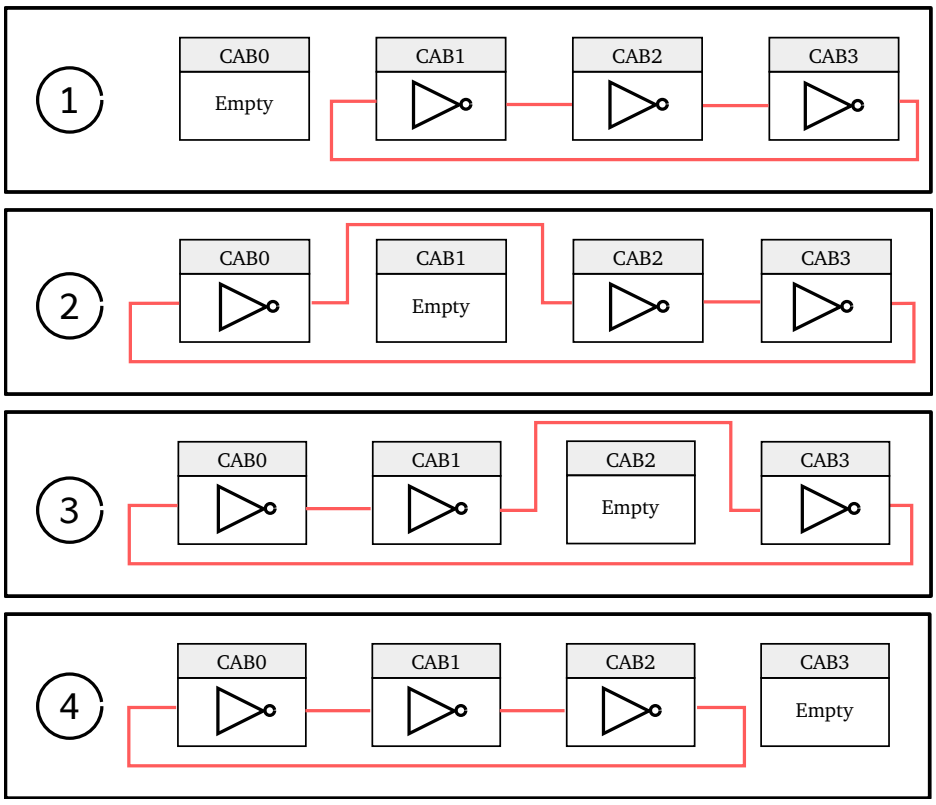


Figure 6.1: Different rotations of the CABs inside a CLB, configured as a 3-stage ring-oscillator.

- **Configurable CT Size** – select any one of the modelled CT sizes for the geometry of every CT inside a given CAB;

- **CAB Permutations** – inside each given CLB, up to four CABs can be configured to perform any particular function for the PAnDA function set. These functions are transferable between the CLB's CABs, preserving the CLB's functionality but changing the CABs that perform a given function. Figure 6.1 illustrates a particular type of permutation, where the empty CAB in a CLB configured as the familiar 3-stage ring-oscillator circuit is rotated among the four CABs. Each permutation will result in a different oscillating frequency, since a different combination of CABs is used each time. Since each active CAB is configured as an inverter, swapping functions between them has no effect on the performance of the circuit. Once all functions are different, more options would be available for meaningful permutations. With four different functions available to implement on the four CABs in a CLB, there are a total of $4! = 24$ different function-to-CAB assignments.

These two operations are considered **functionally-neutral**, as they do not affect the logic (or functionality) of the circuit. As will be made clearer in this chapter, exploring these two operations is key for the purposes of circuit performance optimisation.

6.3.2 Genetic Algorithms

The search space for the optimal CT sizing and CAB permutation for a particular circuit is large, and it grows with both the size of the Emulator and the complexity of the circuit. For instance, take a generic circuit which fully occupies each CLB in a row on the Emulator. For each CAB being used, there are five different CT sizes it can take (as specified in Table 4.1). For a single CLB, this represents a total of 625 different combinations of CAB CT sizes. In the best possible scenario, every function in the CLB's CABs is the same, and therefore CAB permutations have no effect on the performance of the design. With the number of CLBs (n_r) in a row equal to 4, there are 625^{n_r}, (roughly over 100 billion) candidate solutions. Once different functions in a CLB are allowed, the search space becomes even larger. Any feasible optimisation algorithm applied to this problem needs to be efficient in its exploration of the search-space.

Evolutionary Algorithms (EAs) are part of bio-inspired computing, a broader field of study which aims to both study living phenomena with the aid of computers, and to capture some of the aspects of nature and apply them to computer architectures and algorithms. A particular type of EAs, mostly applied to optimisation problems, is called a **Genetic Algorithm** (GA). GAs are adaptive heuristic search algorithms which take inspiration from the process of biological evolution, finding an optimal solution though the course of "generations" of candidate solutions, and essentially mimicking the Darwinian principle of *survival of the fittest* [125, 147]. Their exploration of the solution-space is random in nature, but they are also efficient at identifying peaks and troughs, making them an appropriate choice for the optimisation problem presented by the Emulator.

To understand Genetic Algorithms (and EAs in general), one must first understand that most of an individual's physical traits are specified at the gene level, i.e. in their DNA. This information is passed on from an individual to their offspring, with some modifications. As such, an individual's ability to thrive in its environment will be in part determined by its genetic heritage. Those individuals who possess traits which increase their ability to survive in their environment will have a larger probability of passing their genetic heritage to their off-spring, i.e. to reproduce. This is the core idea of Darwin's theory of evolution.

An individual's genetic information is defined as its *genotype*. The physical expression of this genetic information is called the *phenotype*, which is the individual itself. In most GA applications, there is a *one-to-one* genotype-to-phenotype mapping, where the two representations might even be interchangeable. The biological process of translating an organism's genetic information to the physical and morphological features of the organism is called *ontogenesis*. This process actually experiences random variations, which results in a one-to-many genotype to phenotype mapping. In Evolutionary Algorithms, ontogenesis is usually assumed to be deterministic, so that a particular genotype will always generate the same phenotype.

The genotype of an individual organism is defined as its genetic constitution, i.e. its entire set of genes. These genes comprise segments of DNA, which in turn are made up of combinations of nitrogenous bases – adenine, thymine, guanine and cytosine – sugar and phosphate. The study of molecular genetics – or how these genetic building blocks are combined to generate strands of DNA – is outside of the scope of typical Evolutionary Algorithms.

To solve a problem with a GA, the parameters which describe it must first be expressed in terms of genes, typically in numeric form, such as real numbers, binary bit-strings, or floating point numbers. A candidate solution for the problem will then be a combination of values for these parameters. This is the definition of a genotype for Evolutionary Computation.

To pass on its genetic information, an individual must engage in the process of *reproduction*, sexual or asexual. In the latter case, the individual creates copies of itself, subject to errors in the copying process, usually referred to as *mutations*. In the former case, a new individual is created which combines the genotypes of its two parents, in a process called *crossover*. The process of sexual reproduction is also open to the occurrence of mutation.

These errors can give rise to novel features in the off-spring, which can potentially increase or decrease its adaptability to its environment. For instance, take the example of a population of brown bears living in the Arctic. A baby bear is born from two brown parents, but due to a mutation in its genotype, it is born with white fur. This feature will provide the new individual with an advantage when it goes on a hunt, since its fur will blend it with the predominantly white landscape. This individual will be more likely to be successful in its hunting endeavours, increasing its likelihood of survival, and therefore, reproduction. This individual and its off-spring have an advantage over the rest of the brown bear population, which have a hard time competing for resources and may perish. Several generations later,

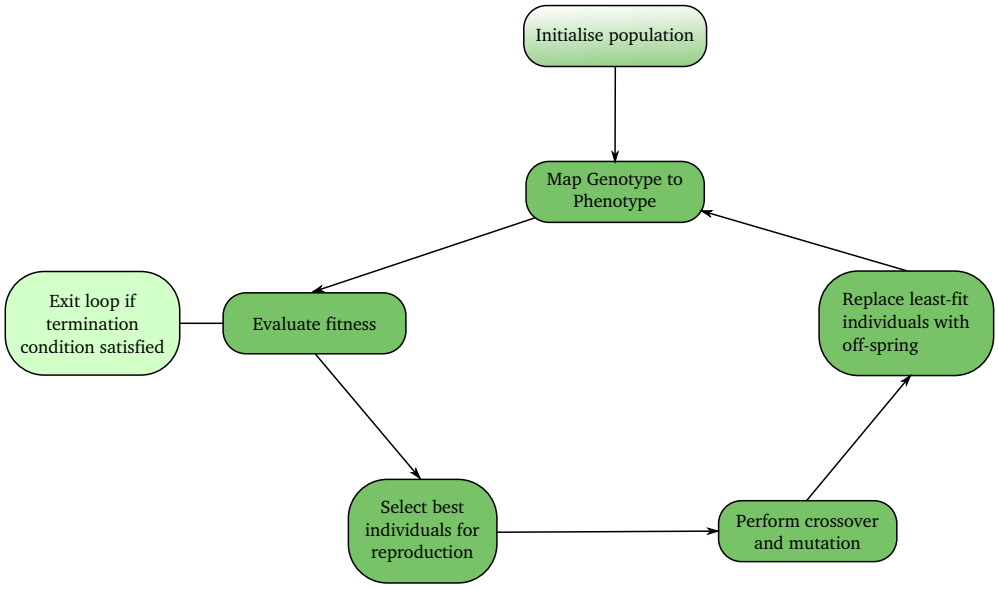


Figure 6.2: A flowchart depicting the behaviour of a basic Genetic Algorithm.

one might find that white-fur bears dominate the environment completely, all stemming from a single mutation.

For GAs, *mutation* represents a change to one or more of these parameters. In the case of a binary representation of the genotype, this would be a bit-flip. *Crossover* is usually represented as an exchange of a given set of parameters between two parents.

A GA will take advantage of these features of evolution, by exploring the search-space in a creative way, allowing for mutations which can an off-spring to a point that is far from its parent. These constant mutations, along with the different combinations of genetic material from two separate individuals, decrease the likelihood of the algorithm getting stuck at a local optimum.

The operation of a basic Genetic Algorithm is depicted in Figure 6.2, and can be summarised as follows:

Initialise population – A set of initial candidate solutions – or individuals – is created, with gene values usually assigned at random.

Genotype-to-phenotype mapping – The parameters are applied to the problem, i.e. the solution described by the encoding of an individual (genotype) is realised to a phenotype.

Evaluate fitness – For each generated phenotype, a metric of how well they solve the problem must be available. If the problem is balancing a pendulum, for instance, this metric might be how long the pendulum is balanced for.

Select best individuals – The candidate solutions are ranked with respect to their evaluation metric, or fitness.

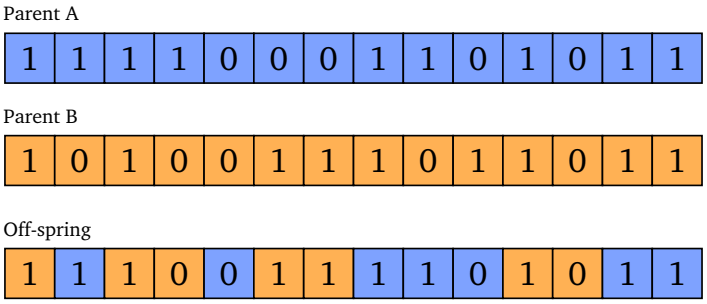
Perform crossover and mutation – The new population is created by picking two parents from the pool of the best individuals of the previous generation. Their genetic information is combined, mutations are carried out, and an off-spring is created. Figure 6.3 illustrates the behaviour of the crossover operator.

Replace population – A new population is created by replacing the least-fit individuals by the newly generated off-spring. The genotype-to-phenotype stage is carried out, and the cycle repeats until the termination condition is satisfied.

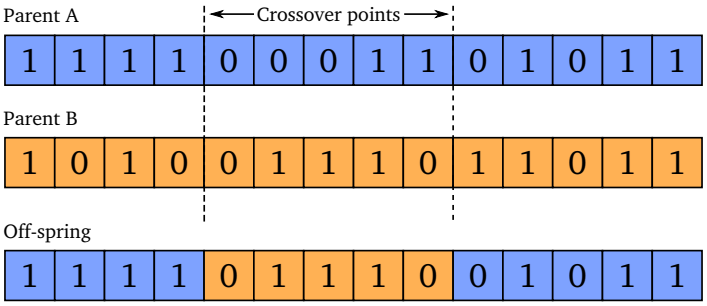
Termination condition – The algorithm terminates when at least one of two conditions is satisfied: a solution is found which solves the problem, or the algorithm reached its maximum number of generations, specified at the start.

This brief introduction to Genetic Algorithms presents all the principles that were used for this work. There is a plethora of other applications of GAs, as well as a very broad selection of other Evolutionary Algorithms, which were not addressed in this section since this work does not focus on the optimal/more efficient optimisation algorithm that can be applied to the Emulator. The purpose of this chapter is to demonstrate that *at least one* optimisation algorithm can provide benefits to PAnDA in the presence of variability.

Genetic Algorithms do not require any knowledge of internal mechanisms of the architecture, and in that sense they are *generic*. Any changes to the CT size set, for instance, or even changes to the number of rows and columns of the Emulator, would require *no changes* to the algorithm. Additionally, if more accurate CM Cards are created as part of future work, an implemented GA will still find an optimal configuration for performance. In other words, a GA will optimise a circuit on the Emulator regardless of the values used on the CM Cards, provided that there is variation in them.



(a) For uniform crossover, each gene is treated separately, with a given probability of taking genetic information from Parents A or B for each particular gene.



(b) For n=2, two crossover points are chosen, and the off-spring will inherit Parent B's genetic information located between the points. The rest of the genotype is copied from Parent A.

Figure 6.3: The generation of an off-spring from two different parents, using the crossover operator in (a) uniform mode and (b) n-point mode.

6.3.3 Bio-Inspired Performance Optimisation on PAnDA Emulator v1

A batch of optimisation experiments were carried out for the first version of the PAnDA Emulator, using the 3-stage ring-oscillator and the 2-bit multiplier as the subjects. These experiments were based upon the concept of *operation-point matching*, whereby multiple instances of the same circuit mapped to the same PAnDA VPI were pushed to have the same propagation delays across every transition. One of the instances is defined as the *target*, and the others are modified so as to minimise the difference between propagation delays to those of the target. This experiment is an attempt at homogenising the behaviour of circuits which, due to intra-device variability, behave heterogeneously, as the studies on the effects of variability on worst-case propagation delay from Chapter 5 have suggested.

Figure 6.4 summarises the operation of the optimisation loop being run for the experiments described in this section. As a first step, the digital circuit is implemented on the Emulator through the use of its Logic layer, by shifting in a configuration bitstream. As the optimisation experiments carried out in this section involve *operation-point matching*, several instances of the circuit under evaluation (2-bit multipliers for one set of experiments, ring-oscillators for another) are created on the Emulator. Once this process is completed, the

Control Module selects CM Cards at random and passes them along to the Emulator, which writes the values to the feature blocks associated with each CAB through the use of the CM Card layer. This creates a virtual instance of a PAnDA device, or **PAnDA VPI**. At this point, multiple instances of the circuit under evaluation are implemented on the Emulator, each with a different set of performance characteristics, stemming from the use of different CM Cards, since they are located on different parts of the virtual hardware. Next, one of the instances is chosen as the "target", and the optimisation goal is to minimise the difference in performance between every instance of the circuit. The GA is responsible for manipulating the bitstream in a way which causes the difference in performance of the multiple circuit instances on the Emulator to be minimised.

Individual Representation

First of all, the problem must be encoded into a format which can be used by the Genetic Algorithm. As defined in Section 6.3, two functionally-neutral operations can be applied on the Emulator, to change the performance of a given circuit in terms of propagation delay: CAB permutations and CT sizing. These are applied at the CLB level, and so each CLB encoding is called a *slice*, as depicted in Figure 6.5. The GA will then attempt to find the combination of these operators which provides the best optimal performance characteristics as defined for the experiment.

Genotype-Phenotype Mapping

Each **genotype** will have two effects on the phenotype: the *permutations* will change the PAnDA configuration bitstream that is shifted-in by the Emulator, and the CT sizing will load the corresponding values from the CM Card associated with each CAB. The resulting bitstream and CM Card size selection will represent the **phenotype**.

In the experiments carried out for this work, the genotype always defines the permutations and CT sizes of the entire PAnDA VPI, *i.e.* 32 slices, 1 for each CLB instantiated in the 8-row by 4-column Emulator. In that sense, each genotype corresponds to the configuration of an entire VPI. Since each slice contains one permutation gene and four CT sizing genes, the full genotype consists of 160 genes.

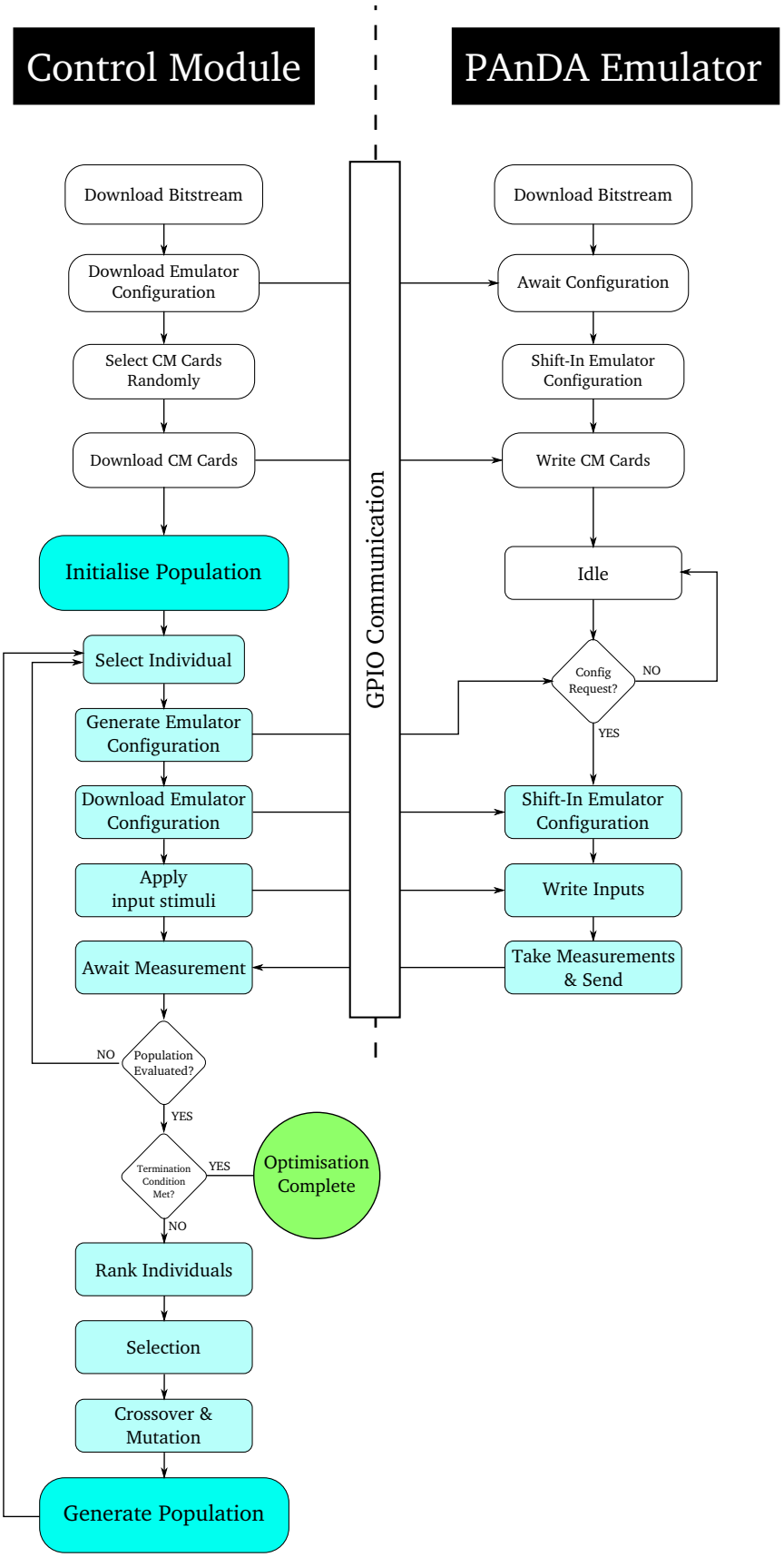


Figure 6.4: Integrating a Genetic Algorithm with the PAnDA Emulator set-up. The Control Module runs the GA and communicates the necessary data and actions to the Emulator, which sends results back through the GPIO communication channel.

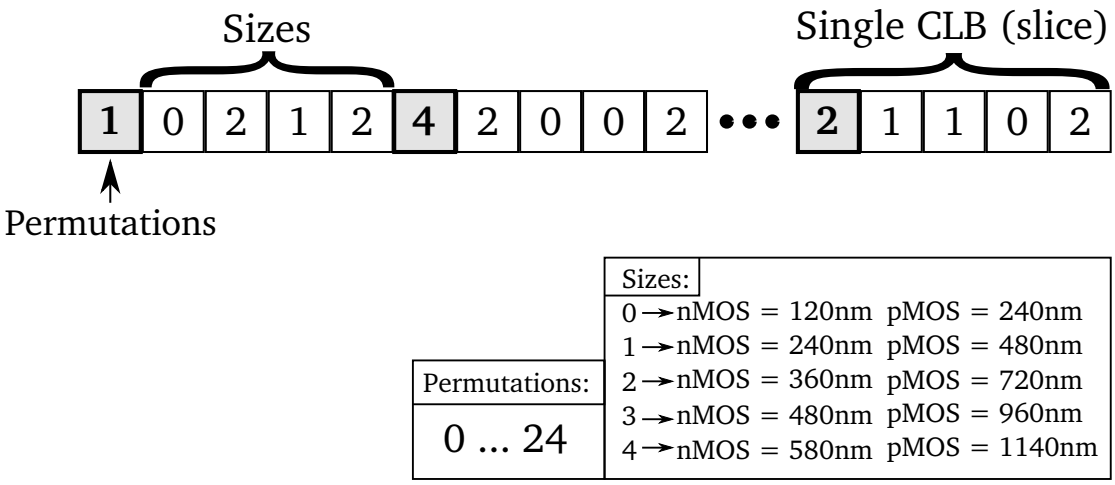


Figure 6.5: Example encoding of an individual for the Genetic Algorithm running on the PAnDA Emulator.

The initial population is created by initialising the genotypes with random values, kept within the scale for each gene type. The **phenotype** is then created by applying the permutations and CT sizes specified in the genotype to the Emulator configuration bitstream.

Fitness Evaluation

Each individual is then **evaluated** separately, by first shifting-in the Emulator bitstream through the Emulator's configuration-chain, and then loading the correct CM Card values to the memory blocks associated with each CAB. The input stimulus required to fully evaluate the performance of a design is applied to the Emulator's inputs, and the PAnDA FSM extracts the associated propagation delays, communicating them to the Control Module.

The **fitness** of each individual is calculated according to the specifications of each experiment, and will be quantified later in this section. After every individual in the populated is evaluated, they are ranked according to their fitness value.

Selection & Reproduction

Selection is done by picking out the T individuals in the population with the best fitness. Again, the value of T is specified for each experiment. Having selected this group of individuals, reproduction can take place to generate the subsequent population.

For the process of **reproduction**, the implemented GA takes two individuals at random and performs **two-point crossover**, selecting two random points in the genotype of these two individuals, swapping the genes within those two points between them. This process

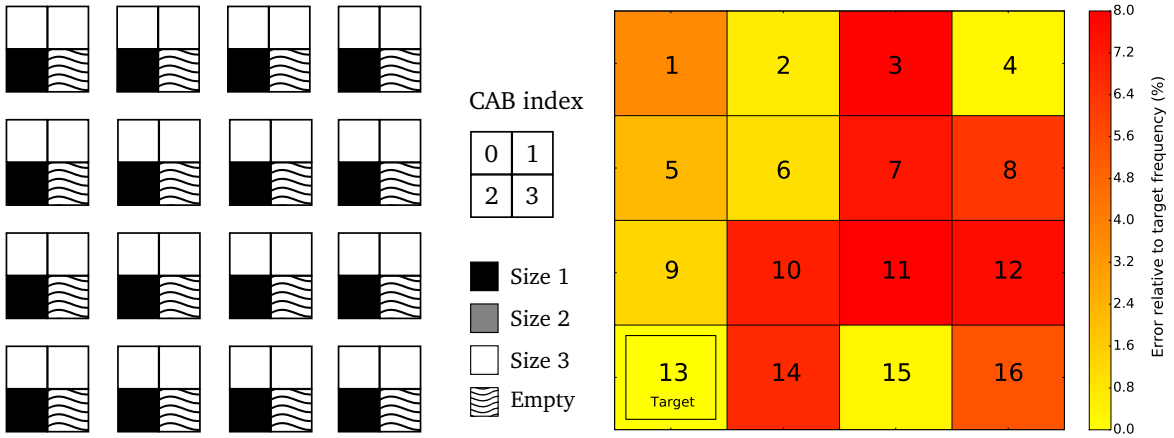
generates two new individuals. The **mutation** operator will then change any of the genes in the genotypes of the two new individuals to a random number (keeping within the scale for that particular gene). The mutation probability is defined as follows: a random number generator is used to output a value between 0 and 4; this value determines the number of mutations that will be applied to the entire genotype. The genes that will be the target of this genetic operator are then chosen at random. This means that there is a 20% chance that no mutations will take place anywhere in the genotype. Also, depending on the size of the circuit mapped to the Emulator, there exists the possibility that the mutated genes have no effect on it. As an example, take the C17 circuit, which actively uses 3 CLBs on the Emulator. The likelihood of selecting one of those slices for mutation is $\frac{3}{32}$, and therefore the probability of *at least one mutation* targeting one of those slices is $0.8 \times \frac{3}{32} = 7.5\%$. For the transparent latch, actively using only two CLBs, this probability drops to 5%. This dynamic mutation rate provides some of the benefits of having a high mutation rate at the initial stage of the experiments, without sacrificing solution convergence in later generations.

After this process is completed, the two new individuals are added to the population. This process is repeated until the population is the same size as the previous generation. Implementing what is known as *elitism*, the GA transfers the T fittest individuals from the previous generation to the subsequent one. This helps to push the optimisation to further "climb up" good areas in the search-space.

Experiments & Results

The first experiment consisted on mapping a series of 16 ring-oscillators to the Emulator fabric, using the first and last columns of CLBs. One of the instances was chosen as the target, and the GA was deployed to minimise the difference in the resulting oscillating frequencies. For this experiment, an individual consisted of one slice for each of the CLBs on the Emulator, and therefore its genotype is 32 slices long.

Figure 6.6 illustrates the difference in frequency of all other 15 ROs with respect to the target, chosen as oscillator 13. It should be noted that each CLB, configured as an oscillator, was using the exact same CT size and CAB permutation as the target. That is to say, each *slice* in the initial genotype was exactly the same, as illustrated by Figure 6.7(a). Although the genotype is 32 slices long, only the active 16 are shown in the image, as the remaining 16 did not actively take effect the oscillators under measurement. As this experiment was carried out in the early stages of this work, only three of the CT sizes had been modelled



(a) (b)

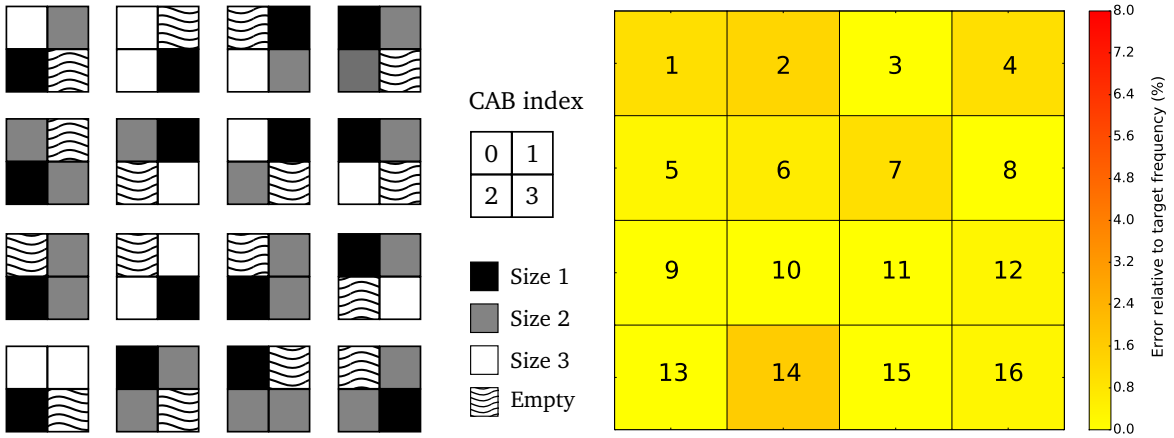
Figure 6.6: The initial circuit for the 16 ring-oscillators mapped to the Emulator (a), and the corresponding variation in the resulting frequencies for each oscillator (b), showing a maximum variation of around 8%.

($Size1 = 240n480p, Size2 = 360n720p, Size3 = 480n960p$), and therefore only those are used.

The GA was allowed to run for a maximum of 50 generations, with a population of 100 individuals, with the top 20 of each generation being used for selection ($T = 20$) and carried on to the subsequent generation. The evolved genotype and the resulting frequencies for the oscillators are illustrated in Figure 6.7 and detailed in Table 6.1. It is clear to see that almost every slice on the evolved genotype is different, suggesting that the algorithm found solutions which dealt with the variations locally, without any explicit knowledge of how fitness can be increased or decreased. The algorithm managed to reduce the difference between frequencies from 8% to 1.3%. These results were published in [32].

If this experiment were to be carried out using RandomSPICE, each individual would require 675 seconds to simulate. With the Emulator, each individual is evaluated in around 8 seconds. The entire optimisation took just over 11 hours on the Emulator, but it would have taken more than 930 hours to run with RandomSPICE.

A second experiment was carried out, this time using a 2-bit multiplier as the target for optimisation. This time, the full set of modelled CT sizes was used, as well as the alternative CT configuration mentioned in Section 4.3. This alternative configuration, although having been proven not to provide any global benefit over the standard one, as Figure 4.10, might provide a benefit to the solution on that particular CT. This meant that a new gene needed to be added to the genotype for each CAB, resulting in the individual encoding illustrated in Figure 6.8.



(a) (b)

Figure 6.7: The evolved solution for the 16 ring-oscillators mapped to the Emulator (a), and the corresponding variation in the resulting frequencies for each oscillator (b), having succesfully reduced the maximum variation with respect to the target frequency to 1.3%.

Given that each multiplier takes two Emulator rows, as illustrated in Figure 5.6, four instances were mapped to a single PAnDA VPI. The CT sizes for each CAB on the device were then initialised with random numbers. One of the instances was then defined as the target and evaluated, extracting all 324 propagation delays – necessary to fully characterise the multiplier design – setting the performance benchmark for the other 3 instances. It then became the GA's goal to minimise the difference between each of the 324 propagation delays

Table 6.1: Table showing the frequencies and respective relative errors of the 16 oscillators, of both the initial and evolved solutions. The (-) and (+) signs indicate if the frequency is below or above the target, respectively.

Oscillator #	**Initial**		**Evolved**	
	f(GHz)	Rel. Error(%)	f(GHz)	Rel. Error(%)
1	3.40	6.7(+)	3.14	1(-)
2	3.15	0.6(-)	3.13	1.3(-)
3	2.94	7.9(-)	3.17	0
4	3.16	0.3(-)	3.20	1(+)
5	3.10	2.2(-)	3.18	0.3(+)
6	3.20	1(+)	3.19	0.6(+)
7	2.96	7.3(-)	3.14	1(-)
8	2.98	6.3(-)	3.17	0
9	3.13	1.3(-)	3.17	0
10	3.41	7(+)	3.17	0
11	3.46	8.2(+)	3.17	0
12	3.44	7.6(+)	3.18	0.3(+)
13	3.17	0	3.17	0
14	3.40	6.7(+)	3.15	0.6(-)
15	3.18	0.3(+)	3.17	0
16	3.01	5.4(-)	3.16	0.3(-)

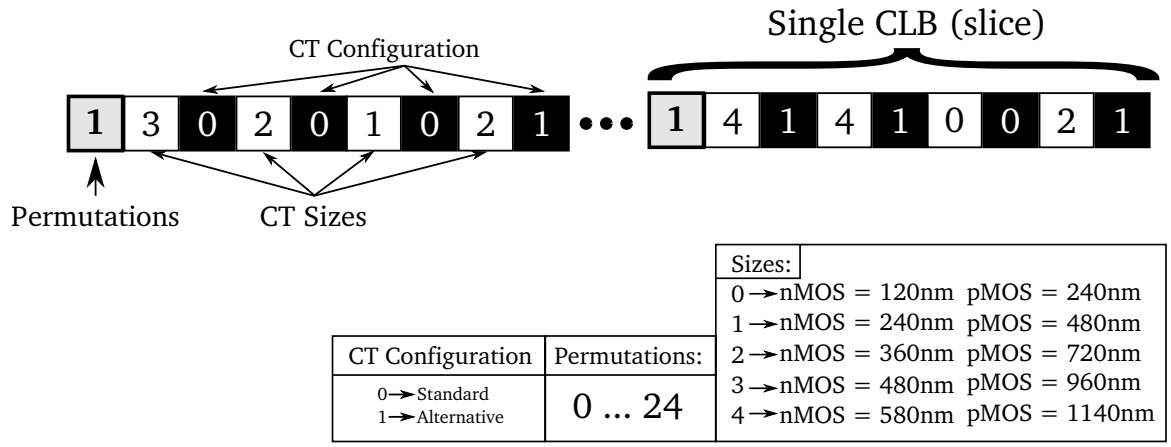


Figure 6.8: Encoding of individual for the optimisation running on the PAnDA Emulator for a series of 2-bit multipliers, including the alternative CT configuration.

of the 3 multipliers with respect to the target, in another *operation-point matching* exercise. As the topmost instance of the multiplier represented the target, it would not be included in the optimisation efforts. With 9 genes per slice, and each multiplier actively using 8 slices, the genotype was 216 genes long.

The population was then initialised at random, and the GA was allowed to run for 600 generations, a larger number than the previous experiment due to the increased complexity of the multiplier circuit. The fitness for this experiment was set to be the *maximum difference between any two equivalent propagation delays* (equivalent here meaning that they correspond to the same transition) *of any of the multiplier instances with respect to the target.* This way, the GA will force each of the circuit instances to "mimic" the target's behaviour.

Figure 6.9 illustrates the evolved circuit of the 3 multipliers along with the topmost "target" one, and Table 6.2 details the maximum differences in propagation delay with respect to the target before and after the optimisation, for each multiplier. It is clear to see that although the end result is similar in terms of fitness, the CT sizes and permutations used in each multiplier are considerably different, suggesting that the GA is exploiting local variations to

Table 6.2: The differences in propagation delay between the target multiplier and the other three instances, before and after running the GA.

	Maximum delay difference to the target			
	Initial		Evolved	
	Absolute (ps)	Relative (%)	Absolute (ps)	Relative (%)
Multiplier 1	65	19.5	8	3.2
Multiplier 2	61	18.5	8	3.2
Multiplier 3	66	19.7	8	3.2

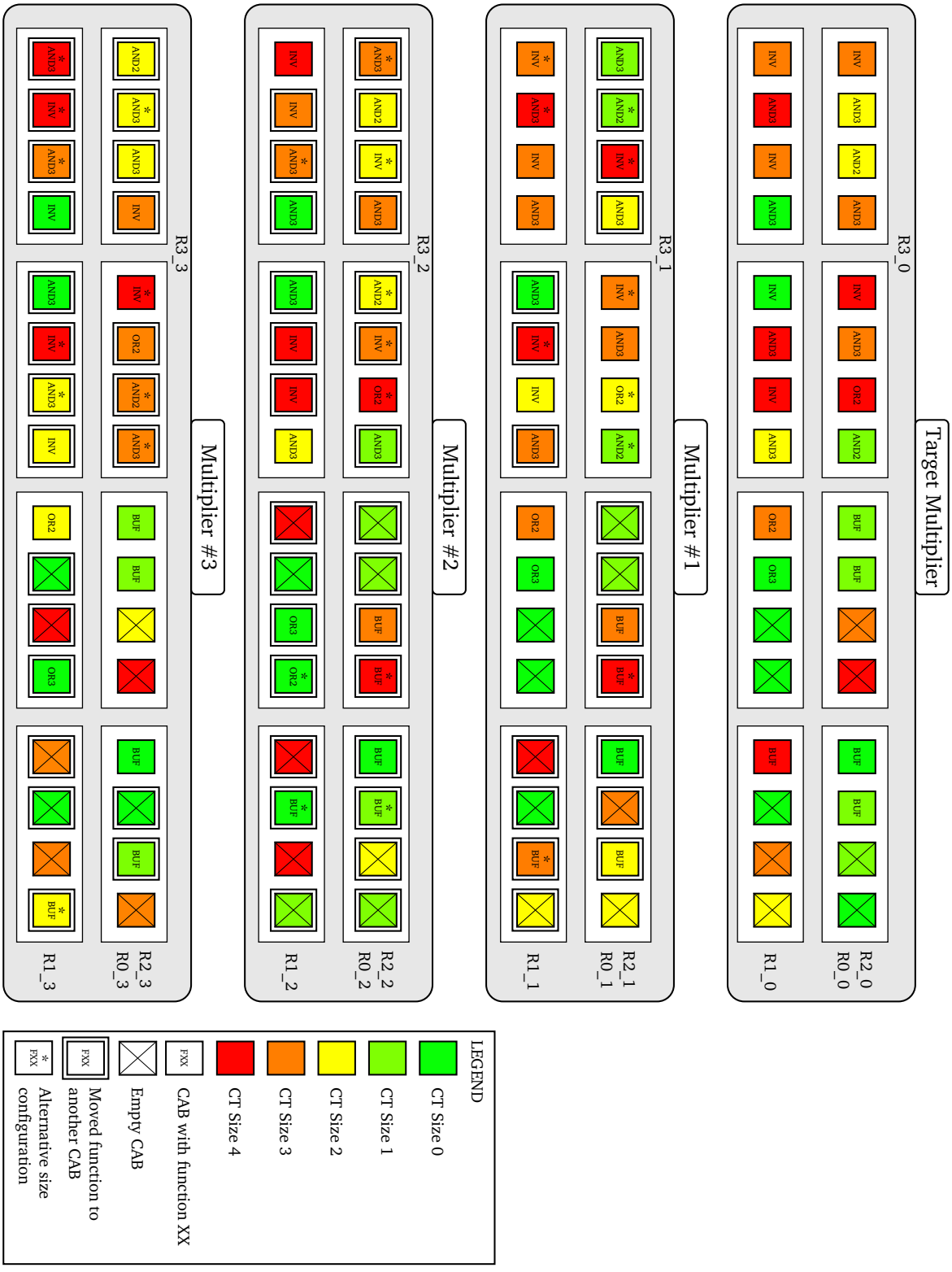


Figure 6.9: The evolved circuit after 600 generations, reducing the maximum difference in propagation delay of multipliers 1, 2 and 3 to the target from 20% to 3%. The GA has come up with considerably different solutions for each multiplier, making use of the local variations to find common ground between them.

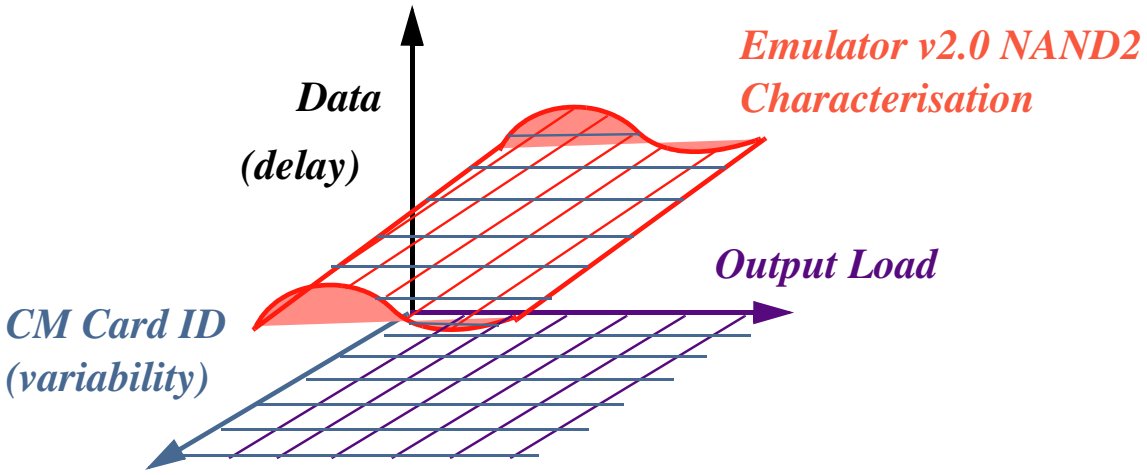


Figure 6.10: With the Emulator v2, the varying CT sizes are replaced by a varying output load. The CTs in every CAB are fixed at a size which minimises the variation in rising- and falling-edges.

achieve the same end result, which is what this experiment had as an initial goal: to create homogeneity from heterogeneity.

6.4 PAnDA Emulator v2

As Chapter 5 demonstrated, the behaviour of the circuits modelled using the Emulator and a full SPICE simulation was not in agreement, and Figure 5.10 illustrated this point by directly comparing measurements taken from the Emulator with the same measurements that would be generated in SPICE and showing that, using the Pearson correlation as a measure of fit, a value of around 75% was the average across a few instances of a 2-bit multiplier.

Figure 5.12 further stressed this point by showing how the largest mismatch between the Emulator and SPICE in a 3-stage ring-oscillator circuit occurs when the CABs in each stage comprise very different CT sizes.

In an attempt to reduce the impact of this mismatch, a revision of the Emulator was created, replacing the modelling of varying CT sizes with varying loads resulting from different numbers of CABs connected to the output of any given CAB, as illustrated in Figure 6.10.

This revision of the Emulator aims to model the load seen by each CAB more accurately, so that post-fabrication optimisation efforts can be applied to the model with a more meaningful connection with SPICE, and ultimately with the actual PAnDA device.

Previously unaccounted-for varying output loads are not the sole contributors to the mismatch verified in previous experiments, however. As mentioned in Section 4.3, the propa-

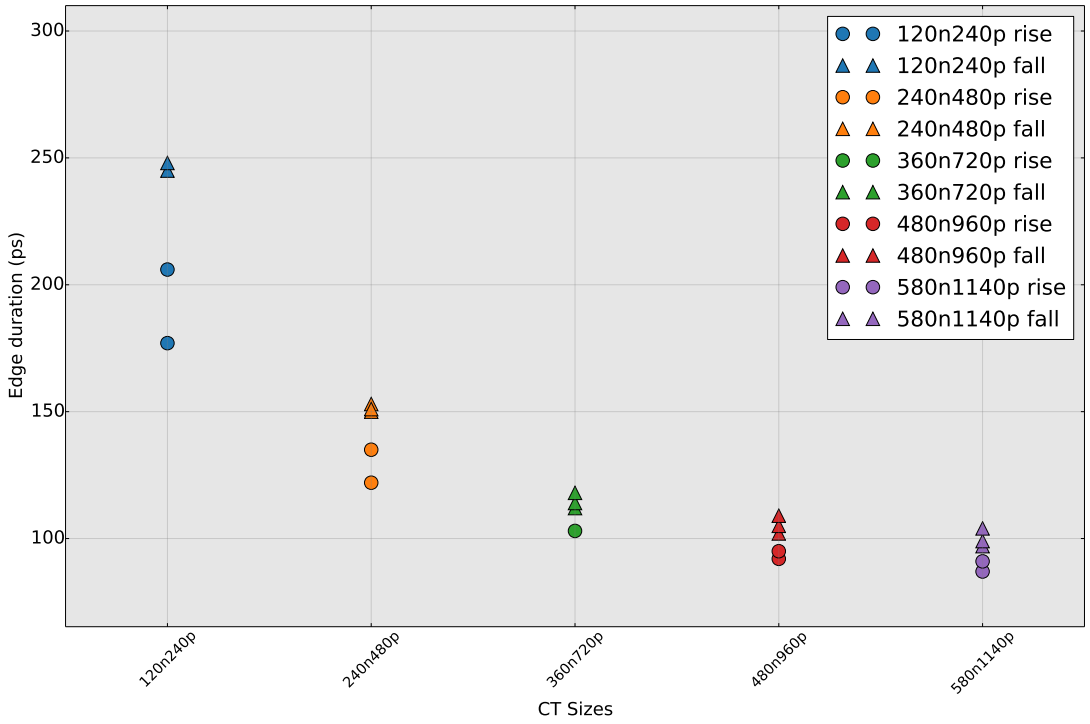


Figure 6.11: The duration of rising- and falling-edges of a NAND2 gate for each of the modelled CT sizes.

gation delay extraction for a particular CAB was done by applying a set of varying input combinations, designed to cause every possible transition pertaining to the function being characterised. The **slew rate** of the stimulating inputs was generated by connecting 3 CABs in series, stimulating the first stage with a random input pattern, and taking the worst-case slew rate of the middle stage. As explained in Chapter 5, the function and CT size of the CABs involved in this chain will clearly affect the load seen by the two first stages. In addition to this, it has been made clear by Figure 4.5 that, depending on the function being performed by a given CAB, different transitions will use a different set of CTs, which in turn will not only cause a different propagation delay, but also a different slew rate of the output signal.

Figure 6.11 illustrates the variation in rising-edge and falling-edge durations measured on a CAB configured as a two-input NAND gate, for each of the modelled CT sizes, suggesting that the variation between the duration of these edges becomes more substantial as CT sizes are decreased. As discussed in Section 4.3, a constant value of $6.25mV/ps$ – which translates to an edge duration of $160ps$ – was taken as the reference slew rate for the stimulating inputs used in the CM Card-generation modelling stage. Based on the values plotted in Figure 6.11,

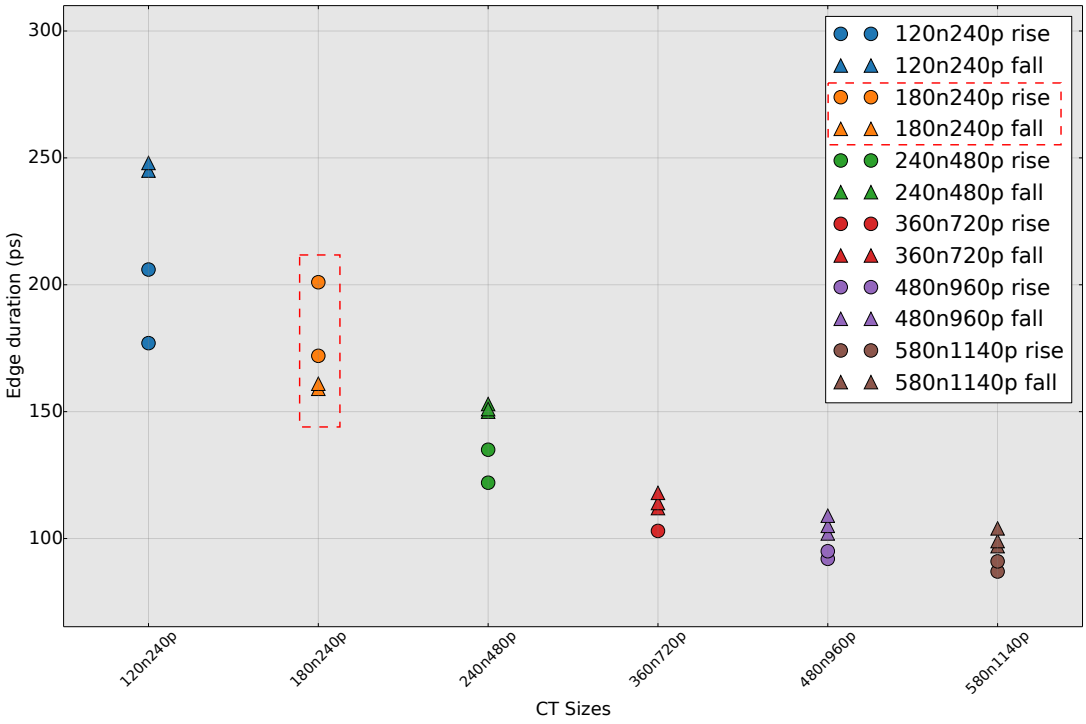


Figure 6.12: The duration of rising- and falling-edges of a NAND2 gate for each of the modelled CT sizes, with the addition of the $180n240p$, following a CMOS ratio of (3:4) rather than the previously used (1:2).

it is possible to see that this will be a considerable over-estimation for most of the CT sizes in the case of the two-input NAND gate.

In summary, the slew rate of the input signals used in the CM Card generation should, if the model is to be as accurate as possible, take into account not only the function being implemented by the CAB which would generate that signal, but also the sizes of the CTs it comprises and the number of CTs driving its output. Adding these new dimensions to the already large measurement space of the CM Cards would not be feasible, as it would require too much memory space to store these values. For these reasons, an approximate slew rate will be used even though it is understood that is will introduce a mismatch between the Emulator and SPICE.

This mismatch can be minimised, however, by choosing a CMOS ratio which reduces the gap between the varying slew rates associated with different transitions. The CT size that will be used in this revision of the Emulator was chosen by selecting the smallest size previously modelled (120n240p) and increasing the size of the nMOS CTs such that the variance between the edge durations would be reduced. Figure 6.12 illustrates the varying CT sizes previously modelled, highlighting the reduced variance of the falling- and rising-edge durations of the

CT size chosen for the Emulator v2 with respect to the smallest size modelled for the first revision.

By using this CT size, the PAnDA Emulator v2 makes use of a large intrinsic variation in propagation delay inherent to reduced size CTs (as illustrated in Figures 4.7, 4.8, and 4.9) whilst reducing the difference between rising- and falling-edge durations through the adjustment of the CMOS ratio.

6.5 Mitigating Variability Across Large Numbers of VPIs

In an attempt to illustrate how the Emulator could be used to investigate optimisation mechanisms for a particular circuit, a large set of VPIs are instantiated, and then a Genetic Algorithm is run on each instance in an attempt to find a configuration which minimises its *worst-case propagation delay*, thus minimising the impact that variability has on a particular design.

The input sequence required to stimulate each test-circuit was constructed and stored in an array in the C program implemented on the microBlaze processor included in the Control Module, previously described in Section 5.3. Along with this array, another was created that indicated which outputs of the Emulator should be targeted for the measurement of a propagation delay of a given transition. This array would specify which addresses should be read in the memory populated by the FSM, as described in Section 5.4.

Figure 6.13 summarises the procedure carried out in each experiment. Initially, both the Emulator and the Control Module are configured via JTAG, and then the latter relays the PAnDA bitstream through the GPIO port to the Emulator. The Emulator reads this configuration and passes it along to to its configuration-chain.

According to the mapped design, the Control Module then applies an input stimulus to the Emulator through the GPIO port, which the Emulator then passes along to its A and B inputs. This causes the FSM to be triggered, and after the FSM memory is fully populated, the Emulator sends the measurements back to the Control Module, which writes these to a file stored on a PC through a serial connection.

The virtual device creation and subsequent measurement extraction loop is repeated $n_{devices}$ times. For the experiments carried out in this section, this number was set to 1000, which was large enough to provide a wide variation in the performance of the virtual devices to give

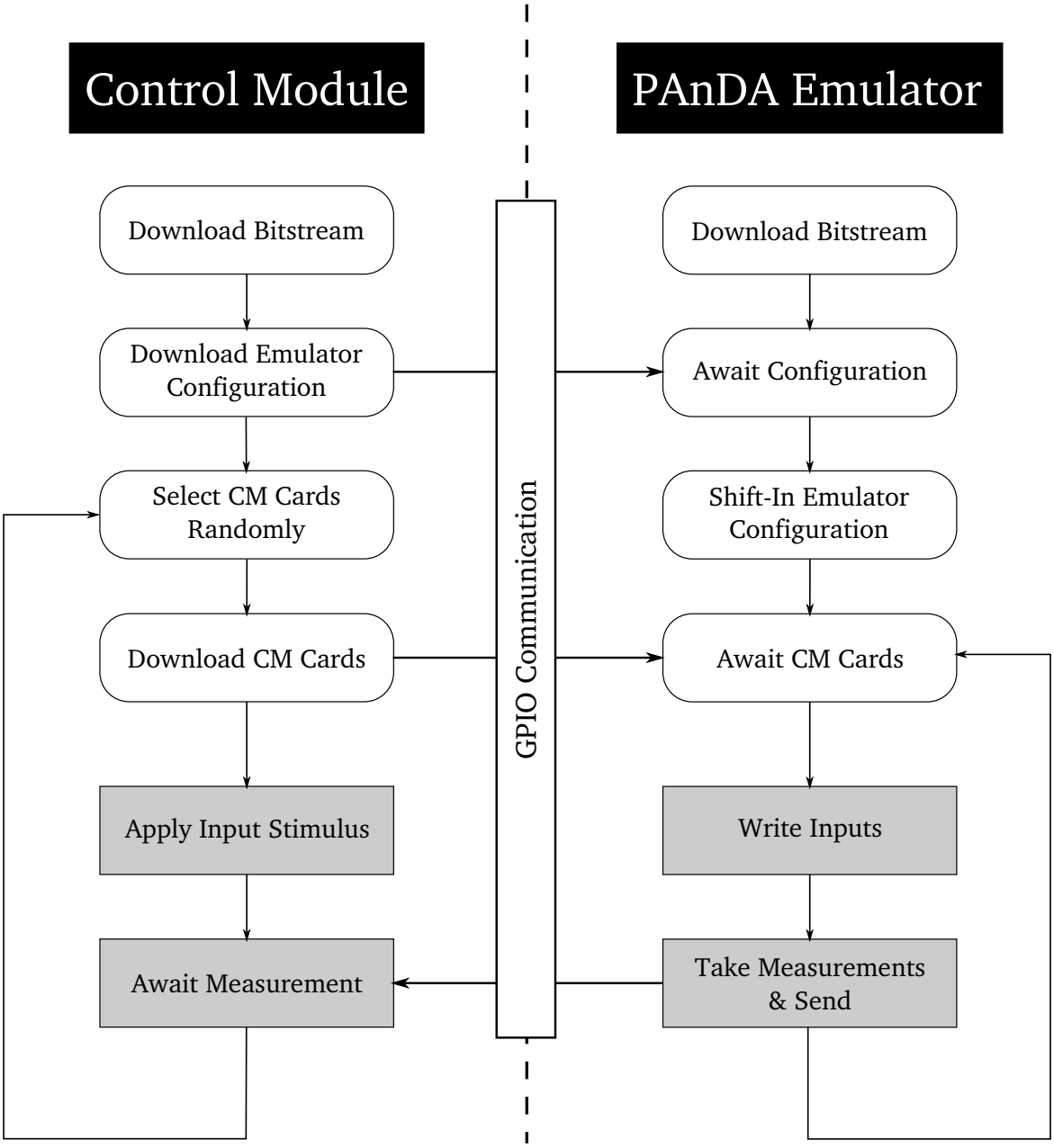


Figure 6.13: Flowchart describing the experiments carried out for this work. Described in text.

room for optimisation, whilst at the same time being small enough to allow for a parallel SPICE simulation of the transparent latches.

6.5.1 Test-Circuits

A few additional test-circuits were designed with varying degrees of complexity to serve as the object of the investigation on the effects of variability on the PAnDA architecture, and

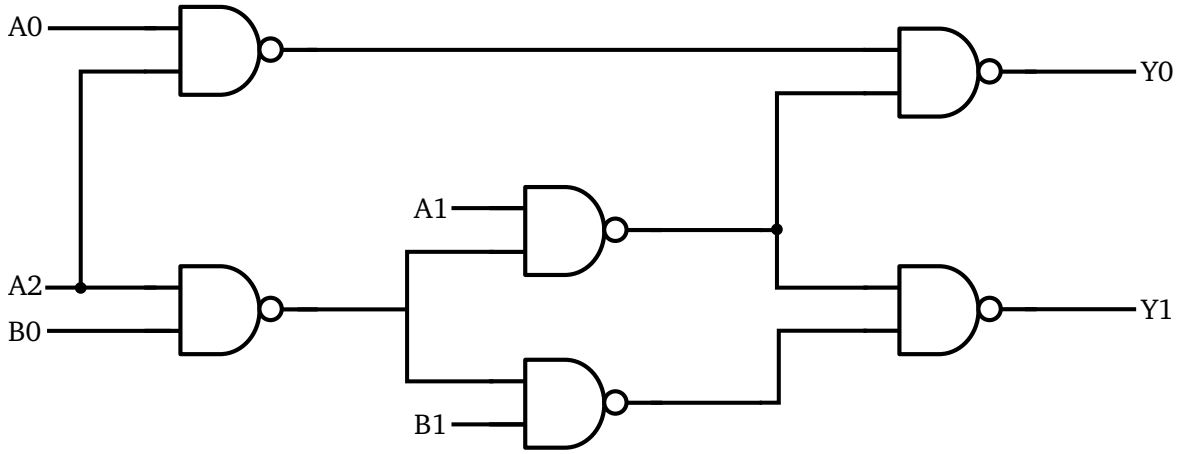


Figure 6.14: The ISCAS '87 C17 benchmark circuit, implemented with 6 NAND2 gates. The circuit takes in five inputs, $A2, A1, A0, B1$ and $B0$, and generates two outputs, $Y1$ and $Y0$.

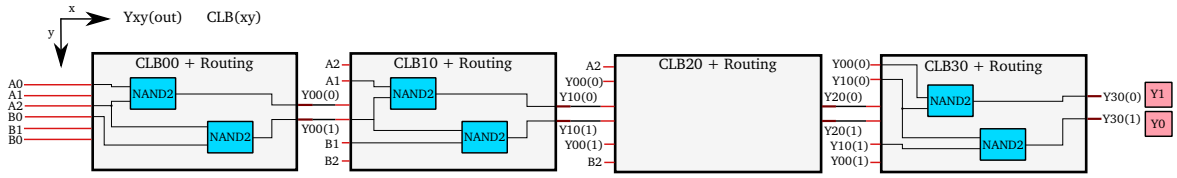


Figure 6.15: A C17 design mapped to the Emulator fabric, using 2-input NAND gates.

this Section describes not only their operation but also how they were implemented on the Emulator.

ISCAS C17 Benchmark

In order to further reduce the mismatch between the PAnDA Emulator and an equivalent RandomSPICE simulation whilst allowing for the study of the effects of variability and how these can be mitigated against, a test circuit was implemented on the Emulator consisting only of 2-input NAND gates. This circuit is part of the ISCAS '85 benchmark set, as is known as **C17**. The homogeneity in the function used across the CABs will provide a lower mismatch with the SPICE simulation in terms of signal slew-rates between them, improving the accuracy of the Emulator with respect to SPICE. It comprises 5 inputs and 2 outputs, connected through a network of NAND2 gates, as Figure 6.14 illustrates. A similar experiment has been carried out in [27], using only RandomSPICE simulations with a reduced set of input stimuli. The implementation of the design on the PAnDA Emulator is illustrated in Figure 6.15.

In this circuit there are two types of NAND2 CABs: those with a fan-out of 1, and those with a fan-out of 2. To account for this varying load – and further increase the accuracy

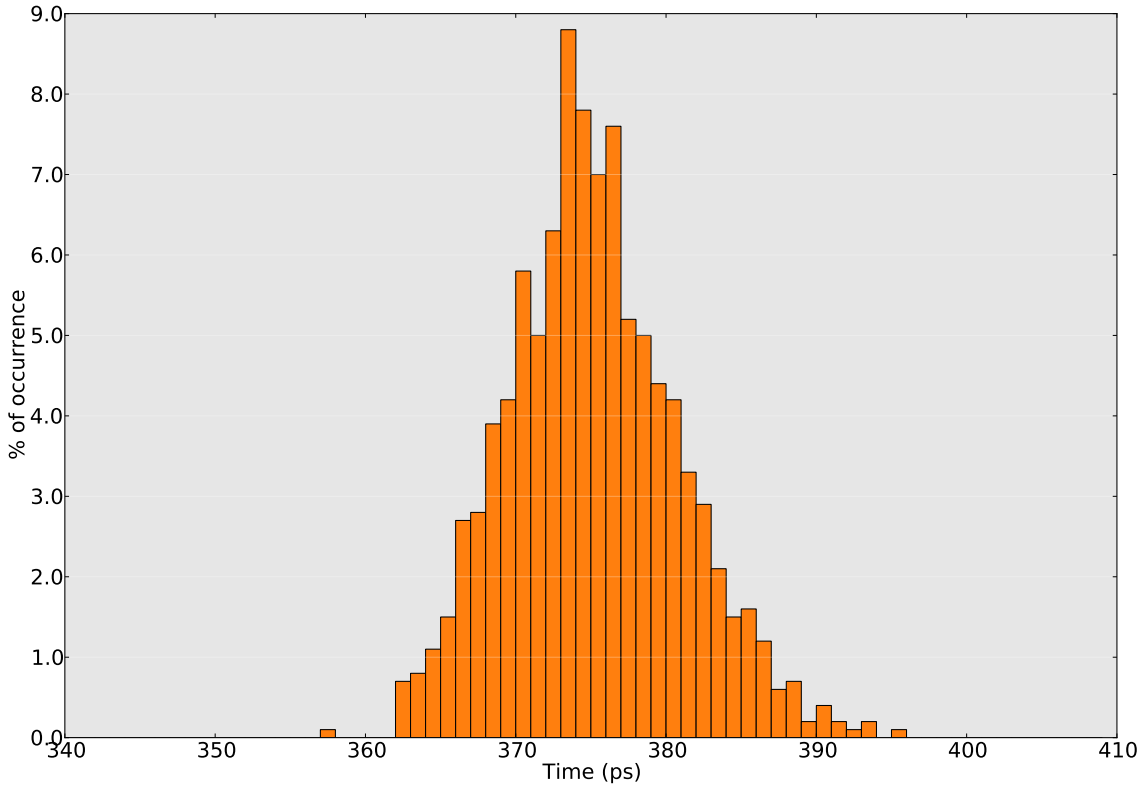


Figure 6.16: Distribution of the worst-case propagation delay of 1000 instances of a C17 circuit, implemented on the Emulator.

of the Emulator with respect to SPICE – two sets of CM Cards were created, taking into account this load variation. When the design is mapped to the Emulator, it will take this into account and load the appropriate CM Card to each CAB, depending on its fan-out.

To fully characterise the circuit, as in every previous experiment, every transition of each output must be accounted for in the input stimulus. In total, 1007 output transitions need to be measured for this circuit.

Table 6.3 presents the truth table for the C17 circuit. For each output, every transition (either from a logic one to a logic zero or vice-versa) and the corresponding input combination must be included in the input stimulus. Since some of the input combination sequences will cause transitions on both outputs, they can be measured in parallel, reducing the length of the input stimulus from 1007 to 727 input combinations.

As previously mentioned, a total of 1007 measurements were taken from each Emulator run of a virtual instance of a C17 circuit. Figure 6.16 illustrates the variation in *worst-case propagation delay* for a set of 1000 virtual instances. It should be pointed out that each Emulator run takes roughly 7 seconds to run, whereas a full SPICE simulation takes just

Table 6.3: Truth table for the C17 function.

Inputs					Outputs	
A2	A1	A0	B1	B0	Y1	Y0
0	0	0	0	0	0	0
0	0	0	0	1	0	0
0	0	0	1	0	1	0
0	0	0	1	1	1	0
0	0	1	0	0	0	0
0	0	1	0	1	0	0
0	0	1	1	0	1	0
0	0	1	1	1	1	0
0	1	0	0	0	1	1
0	1	0	0	1	1	1
0	1	0	1	0	1	1
0	1	0	1	1	1	1
0	1	1	0	0	1	1
0	1	1	0	1	1	1
0	1	1	1	0	1	1
0	1	1	1	1	1	1
1	0	0	0	0	0	1
1	0	0	0	1	0	1
1	0	0	1	0	1	1
1	0	0	1	1	0	0
1	0	1	0	0	0	1
1	0	1	0	1	0	1
1	0	1	1	0	1	1
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	0	1	0	0
1	1	0	1	0	1	1
1	1	0	1	1	0	0
1	1	1	0	0	1	1
1	1	1	0	1	0	1
1	1	1	1	0	1	1
1	1	1	1	1	0	1

under 4 hours. The total run-time for this experiment in characterising variability on a C17 circuit using the Emulator was just under 30 minutes, whereas it would take a full SPICE simulation a total of 3806 hours to complete the same task, representing a speed-up of almost four orders of magnitude.

D-Type Latch

With the aim of highlighting the ability of the Emulator to also model circuits of a **sequential** nature, a NAND2-based d-type latch circuit was mapped to the fabric, as illustrated in Figure

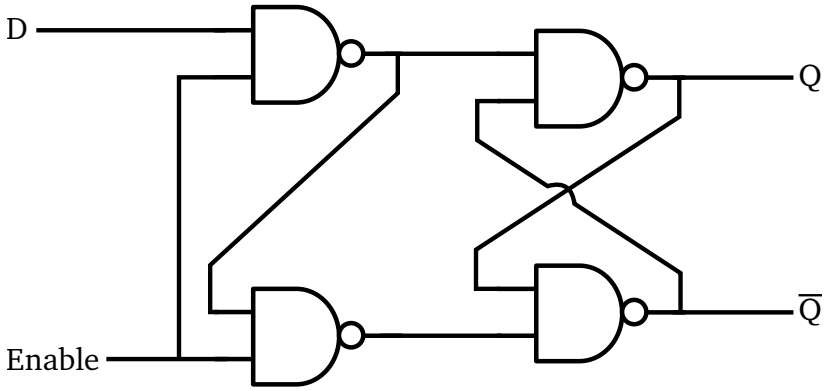


Figure 6.17: A d-type latch circuit, designed using four NAND2 gates.

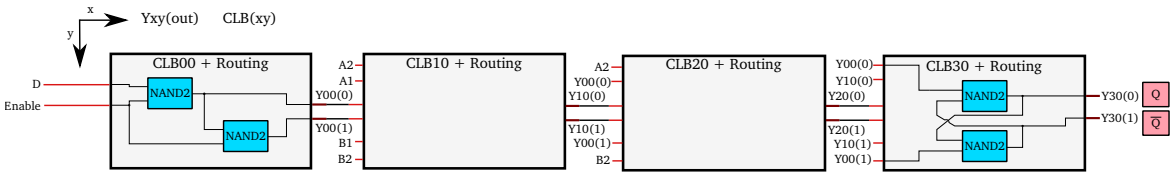


Figure 6.18: A d-type latch design mapped to the Emulator fabric, using 2-input NAND gates. Two unused CLBs are represented in the figure.

6.18. The use of the same function across the design is, as in the case of the C17, useful to reduce the mismatch between the Emulator and SPICE.

Since it is a much simpler circuit than the C17, it requires only 12 measurements for a full circuit characterisation. As with the C17 circuit, a fan-out of one or two is observed in this design, and the Emulator will load the appropriate CM Card based on the fan-out of each CAB.

The C17 will serve as a larger, more complex circuit, and the d-type latch will serve as a smaller circuit which can be simulated in SPICE in parallel with the Emulator experiments due to its size (and therefore simulation feasibility). This smaller circuit will then be useful to measure the accuracy of the revised Emulator model with respect to SPICE.

The d-type latch, often referred to as *transparent latch*, loads the value of D to its output Q – and its complementary to $\bar{Q}$ – when the *enable* signal is high. Otherwise, it holds the previous value of its outputs, Q and $\bar{Q}$, as detailed in Table 6.4.

The same experiment was carried out on the transparent latch, again creating 300 virtual instances of this circuit and extracting the worst-case propagation delay. The result of this experiment is illustrated in Figure 6.19.

The speed-up achieved in this case with respect to SPICE is much less significant, mostly due to the small size of the circuit. Each instantiated PAnDA VPI can simulate four latches

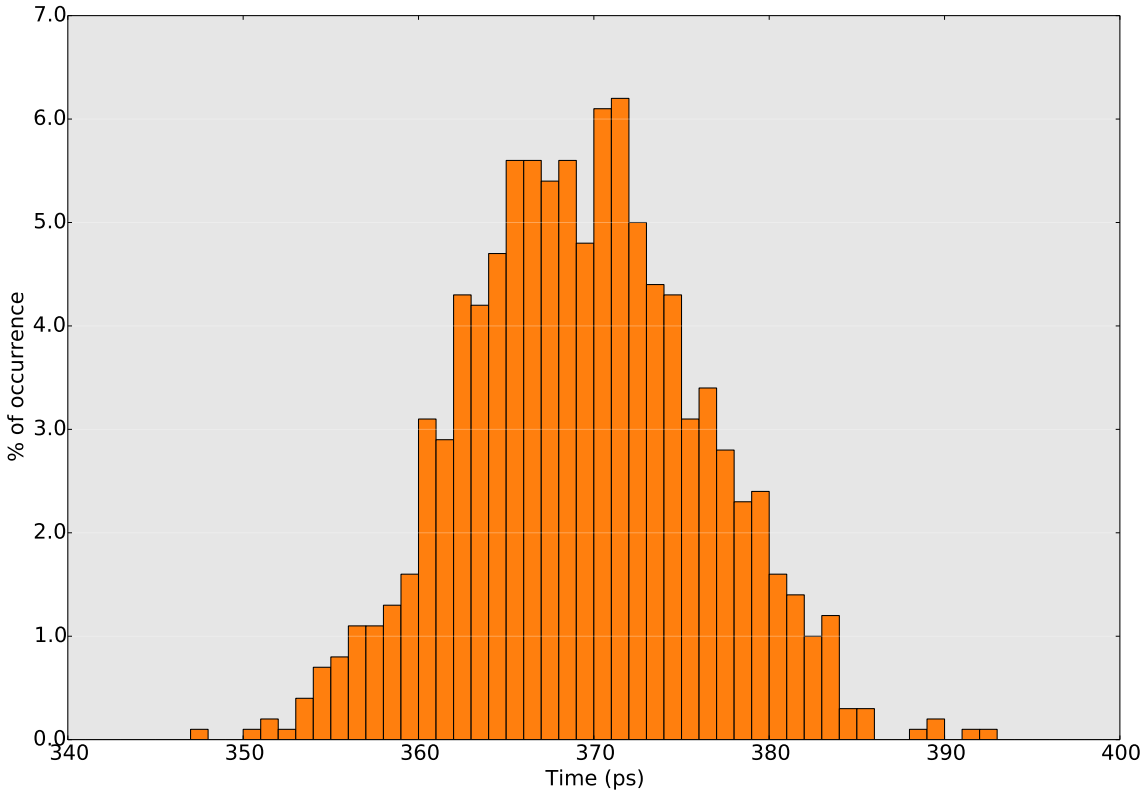


Figure 6.19: Distribution of the worst-case propagation delay of 1000 instances of a transparent latch, implemented on the Emulator.

in parallel, evaluating them in just under 2 seconds. Therefore, 1000 latches were evaluated in just over 8 minutes. In SPICE, each simulation takes 144 seconds, meaning that the simulating 1000 devices (using a single thread) took 40 hours of computational time, representing a speed-up of two orders of magnitude of the Emulator approach over SPICE.

6.5.2 Correlation with SPICE

As with previous experiments, the simulation speed-up will come at the expense of accuracy, but the latter is expected to increase with the refined load modelling. The issue of varying slew-rates was still left unresolved, and therefore a mismatch is still expected.

Table 6.4: Truth table for the d-type latch, or Transparent Latch.

Inputs		Outputs	
D	Enable	Q	$\bar{Q}$
0	0	Q_{prev}	$\bar{Q}_{prev}$
0	1	0	1
1	0	Q_{prev}	$\bar{Q}_{prev}$
1	1	1	0

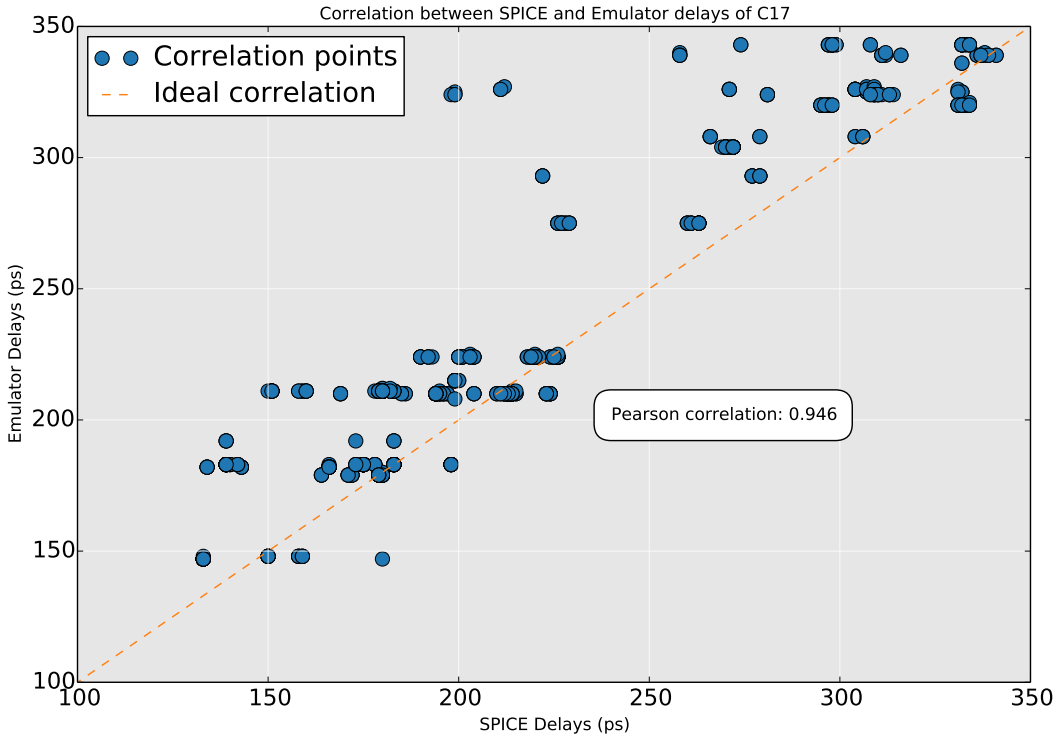


Figure 6.20: Correlation between the 1007 different propagation delays measured with the Emulator, on the y-axis, and those taken from the equivalent SPICE simulation, on the x-axis. Each plotted point corresponds to a particular measured transition of the C17 circuit. Some measurements give the same result, and therefore appear overlapped on the graph.

As an initial comparison, the worst-performing virtual instance of a C17 circuit was used as the subject of a full SPICE simulation. The CM Cards which were used in its mapping were extracted and compiled into a single netlist. This netlist was run in SPICE, and the same 1007 measurements extracted.

Having mapped both the C17 and transparent latch to the PAnDA Emulator, the number of instances that could be implemented on one device were maximised, so as to allow for parallel measurements. Using the 8-row by 4-column Emulator, 4 instances of a C17 could be implemented in parallel, as well as 4 instances of a transparent latch. Although both designs only occupy one row, the arrangement of the inputs A and B on the Emulator required each instance to be separated by two rows. This parallelism allows for a further speed-up over the SPICE implementation.

As in the case of the 2-bit multiplier, the Emulator and SPICE measurements were plotted against each other to analyse the correlation between them, as Figure 6.20 illustrates. The calculated Pearson correlation of 0.95 is found to be considerably higher than the average of 0.75 observed in the 2-bit multiplier mapped on the first version of the Emulator. The

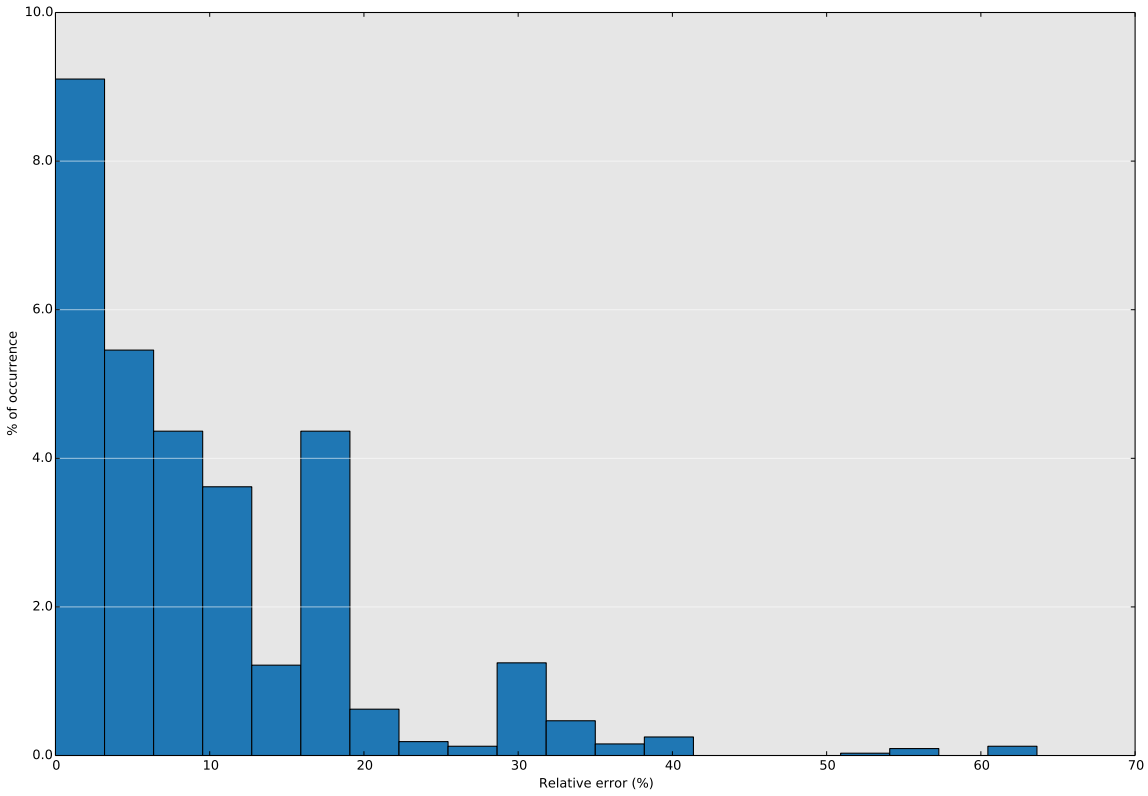


Figure 6.21: Distribution of the relative error between measurements taken with the Emulator and those taken from the equivalent SPICE simulation of a C17 circuit. The error is plotted on the x-axis, and its percentage of occurrence on the y-axis.

Emulator v2 is still overestimating its propagation delays, as most of the correlated points sit above the ideal correlation line. This high correlation value suggests that the Emulator is correctly capturing the analogue behaviour of the circuit, but with modelling inaccuracies and approximations that drive it away from the ideal curve.

Figure 6.21 illustrates how the relative error between the Emulator and SPICE measurements is distributed. A large majority is focused below the 20% mark, but in some instances it can go up to 80%, meaning that overall the Emulator comes relatively close to what a SPICE simulation would yield, but is considerably wrong in a small set of cases.

It should be noted that when the CM Cards were created for the Emulator v2, the input slew-rate was approximated to be constant, and set to $5mV/ps$. In the more dynamic environment that is a full SPICE simulation, the slew-rates will not be static. This will have a direct effect on the propagation delay measured, since a faster-rising signal will cause a lower delay. Although the revised version of the Emulator captures the varying load more accurately, it does not incorporate a changing slew-rate into the model. This is one of the possible causes for the persisting mismatch between the two implementations.

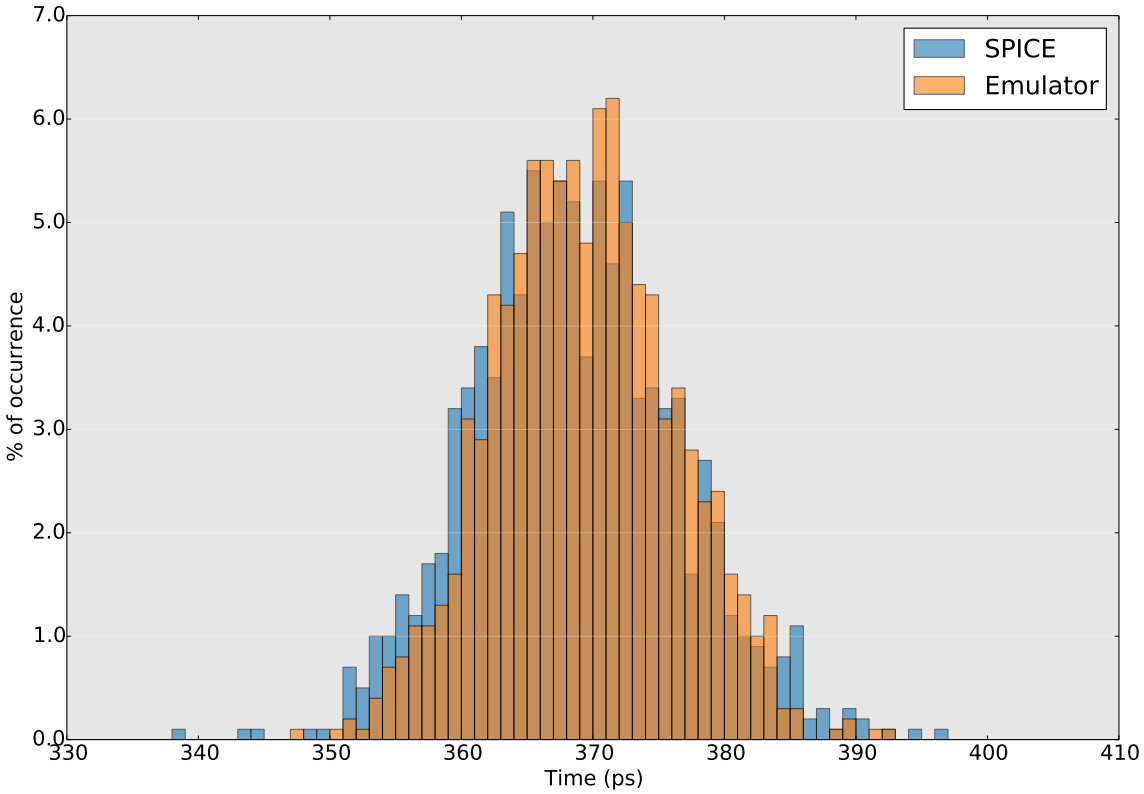


Figure 6.22: Comparison between the worst-case propagation delay of each of the 1000 virtual transparent latch instances created on the Emulator, and those simulated in SPICE.

Figure 6.22 illustrates the comparison made between the worst-case propagation delay of each of the 1000 virtual instances of a transparent latch created on the Emulator and those instantiated in SPICE.

The similarity between the distributions seems to suggest that with the appropriate constraints added to the model, such as the more accurate load modelling on the Emulator v2, this approach *can* provide a good performance estimation for *some* designs across a number of virtual devices along with a considerable run-time acceleration provided by the hardware.

6.5.3 Performance Optimisation with Emulator v2

As Figures 6.16 and 6.19 illustrated, the worst-case propagation delay for the evaluated circuits suffers from a variation of around ±5%. In the test circuits used for the first version of the Emulator, the multiplier and the ring-oscillator, the magnitude of the variations in performance were quite similar. The former presented a maximum variation between equivalent propagation delays on four different virtual instances of around 20%, and the latter showed a maximum variation of around 8% between oscillators.

Although the Emulator does not quite simulate each virtual instance as accurately as SPICE, Figures 6.20 and 6.22 showed that it comes quite close and also that, more importantly, *it introduces variability* to the performance of the virtual instances. As previously stated, Genetic Algorithms are heavily based on random search methods, but they add bio-inspired concepts which influence the direction of the search. Where there is no variation, evolution can not happen, and therefore neither can optimisation.

Having verified the existence of variability between each of the modelled test-circuits, optimisation was allowed to take place on each version of the Emulator. For this set of experiments, optimisation takes place *after* a PAnDA VPI is created, by applying functionally-neutral operations to the mapped design.

This set of experiments aimed to recreate the scenario in which a particular user – company, university, or an individual – acquires a batch of 1000 PAnDA devices, and maps their design of choice to each. Intrinsic variability then causes variations in their performance, such as the worst-case propagation delay.Given that for both of the testbenches implemented on this version were made up exclusively from NAND2 gates, and only one CT size was modelled, the optimisation efforts in this case were focused on the *CAB permutations*.

If the user is aware of the performance requirements for their design, they can then establish a *maximum value* for the propagation delays of their circuit. An experiment was carried out on the C17 benchmark, taking the delay distribution illustrated in Figure 6.16 as reference. An arbitrary value of 380 was chosen as the maximum value for any propagation delay of one of the generated virtual devices. Figure 6.23 illustrates the operation of this experiment, building upon the set-up described in Figure 6.4 by closing the outer loop which concerns the creation of virtual devices with the Emulator.

As only one CT size was being used, and no alternative CT configurations were included, each individual in the GA was made up of the permutations in each CLB, considerably reducing the size of the search-space with respect to the experiments carried out for the Emulator v1. Each genotype was only 3 genes long, one for each CAB permutation associated with the 3 CLBs that the design uses, as illustrated in Figure 6.15. For each created PAnDA VPI, the GA was allowed to run for a maximum of 10 generations, each comprising a population of 10 individuals.

Figure 6.24 illustrates the optimised distribution on worst-case propagation delays plotted against the original distribution. The algorithm succesfully explored the local variations of the CABs to minimise the propagation delay of the C17 instance mapped to each VPI.

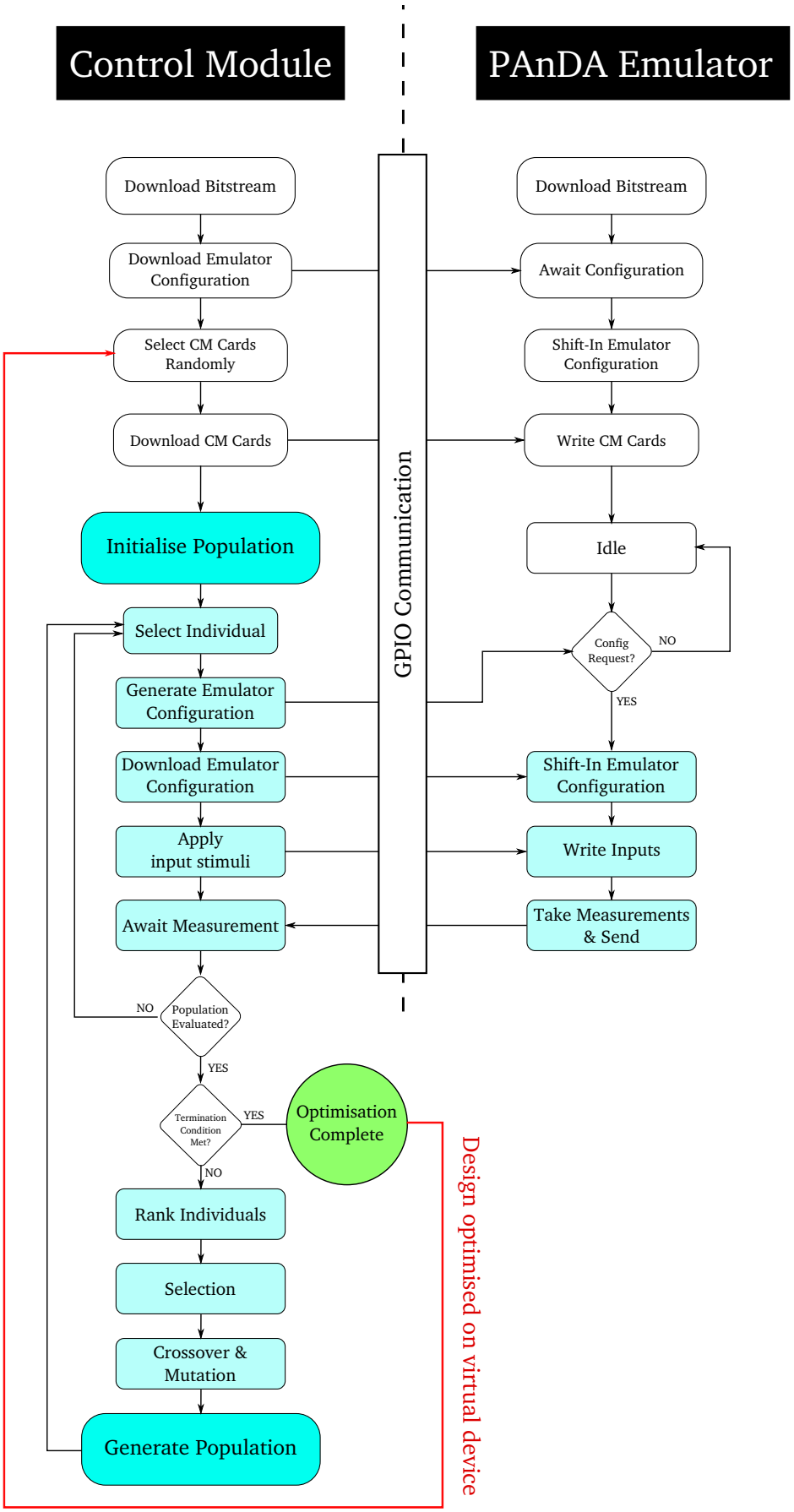


Figure 6.23: Integrating the GA-based optimisation with the PAnDA VPI generation from random sampling of CM Cards. The loop represented by the red arrow is repeated once for every VPI that is instantiated.

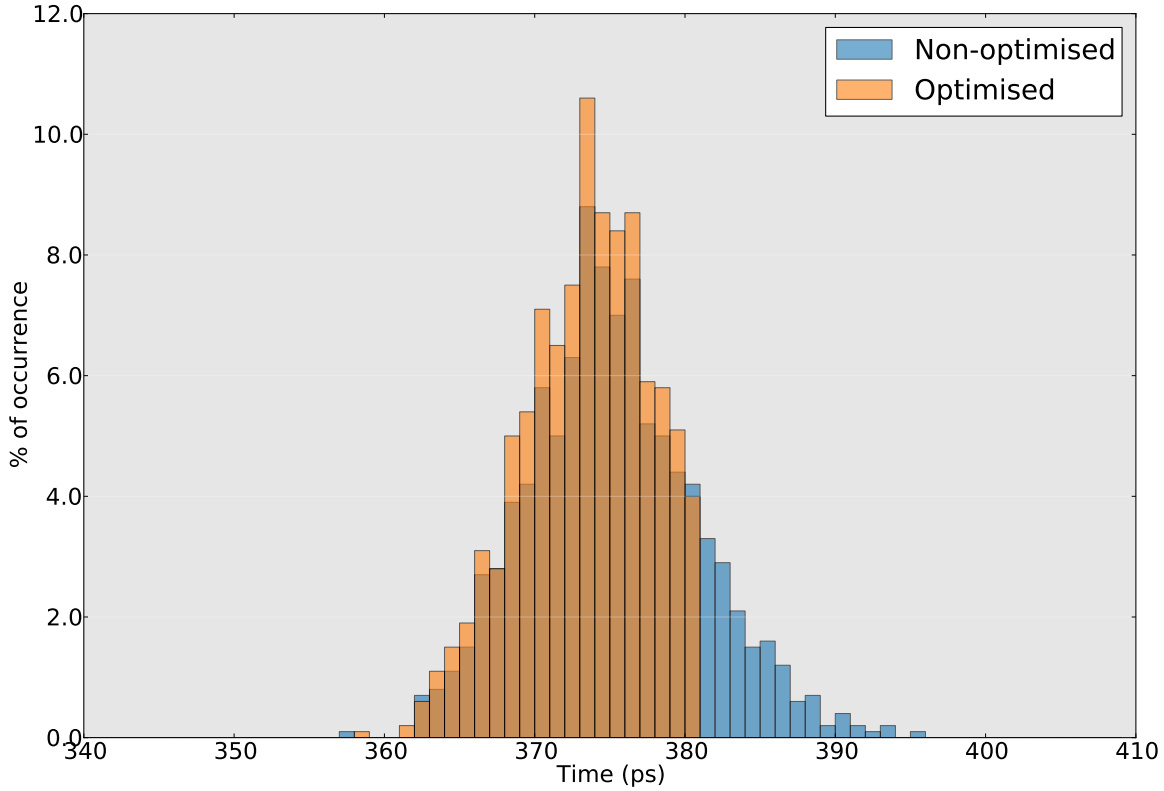


Figure 6.24: Comparison between the worst-case propagation delay of C17 instances, measured before and after optimisation, along with the approximate normal distribution curve-fit parameters for each. The GA was allowed to run for a maximum of 10 generations for each VPI, or until the worst-case propagation delay was measured below 380ps.

To further illustrate the optimising algorithm's ability to explore the local timing variations to minimise overall circuit propagation delay, another experiment was carried out which did not set a particular optimisation limit. Instead of exiting the optimisation loop once the propagation delay has been reduced below the user-defined figure, each optimisation run would be allowed the full 10 generations before being terminated.

Figure 6.25 illustrates the results of the experiment, plotting the optimised distribution of propagation delays next to the original distribution. The maximum observed delay has been reduced from 395ps to 379ps, representing a 4% improvement over the original case. Although the worst-case delay is almost exactly the same as the previous experiment where the user was allowed to set a particular value as the objective, the distribution of the values is entirely different. Allowing the GA to run for the full 10 generations without interruption seems to give it enough space to shift the (approximate) mean of the distribution by 2.6%, whilst at the same time reducing the (approximate) standard deviation by almost 50%, as well as the coefficient of variation σ/μ. In other words, the GA finds a way to minimise the

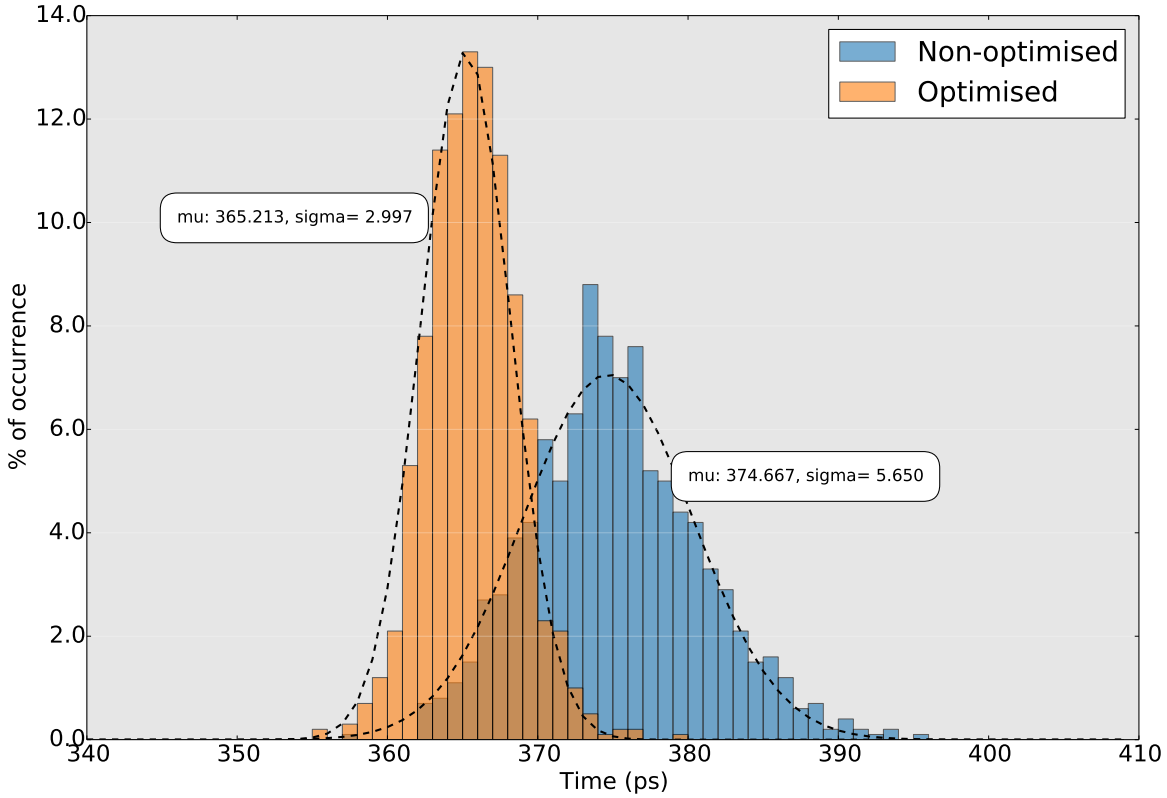


Figure 6.25: Comparison between the worst-case propagation delay measured before and after optimisation of 1000 instances of a C17 circuit, along with the approximate normal distribution curve-fit parameters for each. The GA was allowed to run for the full generations for each PAnDA VPI.

variation in worst-case delay on the C17 circuit, and to reduce both the average measured value and the maximum worst-case delay. A Vargha-Delaney [148] non-parametric test, typically used to compare the effect size of a particular solution, was carried out to calculate the effect size of the optimisation with regard to the non-optimised distribution, resulting in a A-value of 0.936. This value indicates that around 93% of the time, a given optimised circuit will outperform a non-optimised one. It should be noted that these results were achieved by only investigating different CAB permutations. Once varying CT sizes are introduced, the variations will become much greater, and so will the impact of optimisation.

6.6 Summary

This chapter focused on using a bio-inspired algorithm to achieve performance optimisation on circuits mapped to the Emulator, in two different shapes: performance standardisation and overall performance maximisation. It also focused on the Emulator's ability to quickly

generate and evaluate a very large number of PAnDA VPIs, along with the evolutionary optimisation that can be applied to each of these in order to mitigate the effects of variability.

The Genetic Algorithm implemented on the Emulator is used to do operation-point matching on two circuits: a 2-bit multiplier and a 3-stage ring oscillator.

In the case of the ring-oscillators, the frequency variation between a set of 16 is reduced from 8% to around 1.5%. The maximum propagation delay difference between four 2-bit multipliers is reduced from just below 20% to around 3%. In the presence of variability, the optimisation algorithm is able to quickly explore the search-space provided by the reconfiguration options of PAnDA to find a better solution for that particular VPI.

A second version of the Emulator is then introduced, which discards different CT sizes and alternative CT configurations to free-up memory and to give way to more accurate load modelling. The Emulator is then used to evaluate the performance of 1000 VPIs of D-type latches and C17 benchmark circuits. The changes applied to this revision of the Emulator prove to be beneficial toward increasing the accuracy of the circuits simulated on the Emulator with respect to their SPICE equivalents. The worst-case propagation delay distribution of a transparent latch is used to compare the two implementations, and distributions show a great deal of similarity.

Additionally, a final experiment is carried out which tightens the distribution of the worst-case propagation delay of 1000 PAnDA VPIs. The reduction in both mean value and width of the optimised distribution suggests that not only is it possible to optimise the performance of a particular circuit mapped to a VPI, but also that the spread induced by intrinsic variability can be reduced, increasing the reliability of a design mapped to the architecture.

Chapter 7

Conclusions and Further Work

Contents

7.1 Introduction

This thesis focuses on circuit performance optimisation in the presence of process and substrate variations. The modelling efforts that have been described are used to predict the impact of variability on the PAnDA architecture, and optimisation techniques have been developed which can make use of the reconfiguration resources of the architecture to improve circuit performance by making use of the existing variations.

7.2 Hypothesis

The first sub-hypothesis outlined in Chapter 1 read:

It is possible to use digital reconfiguration of the PAnDA architecture to optimise the performance of a circuit.

The second outlined hypothesis stated that:

It is possible to reduce the impact of variability on a circuit mapped to the PAnDA architecture making use of its digital reconfiguration resources.

The first sub-hypothesis has been confirmed through the use of the PAnDA Emulator, although the modelling inaccuracies are a limiting factor in the fidelity of the obtained results with respect to a physical device. The experiments carried out in Chapter 5 and the first half of Chapter 6, concerning the performance standardisation and reduction of worst-case propagation delay of different test circuits, have shown that the implemented Genetic Algorithm can make use of digital reconfiguration to optimise the performance of a design.

The second sub-hypothesis has also been confirmed, although with the same limitations as the first. Based on the developed model, the methodology described in the second half of Chapter 6 allows for the evaluation and subsequent optimisation of a large number of physical instances of a design mapped to a PAnDA device. By making use of the optimisation methodology developed for the first sub-hypothesis, the spread in circuit performance can be reduced, consequently reducing the impact of variability on the design.

The hypothesis outlined in Chapter 1 read:

It is possible to mitigate the effects of atomistic variability on the PAnDA architecture at the circuit-level through the use of digital reconfiguration, whilst making use of the substrate variations to allow for circuit performance optimisation.

This work has shown, through the use of a model which combines hardware acceleration and statistically-enhanced performance data, that digital reconfiguration can in fact be used to minimise the impact of intrinsic variability, as well as to improve the performance of a given design by making use of variations present in the fabric.

The developed embedded model includes a feature block, which contains information about a logic block's performance, characterised with statistically-enhanced tools. Although the work described in this thesis only focuses on delay information, this could be expanded to include other circuit performance features, such as power consumption. This model was then used to generate large numbers of virtual physical instances of particular circuits mapped to the architecture, and the impact of variability was observed and subsequently reduced through the use of a bio-inspired optimisation algorithm.

In this chapter, each of the major themes in the thesis is analysed and summarised, identifying the main contributions and the main limitations in each theme. A connection is also made to each of the objectives outlined in Chapter 1.

7.3 PAnDA & Modelling

As atomistic variability is certain to increase with future technology nodes, even in the case of beyond CMOS technology such as plastic electronics and carbon nano-tubes, it appears as though both designers and manufacturers will have to accept it as part of the manufacturing process, and will need to take it into account for the design of reliable devices and systems. With the continuing increase in transistor count of modern electronic devices, even small percentages of faulty transistors due to both process and environment variations can represent a large enough number to cause critical sections to fail. Chapter 2 introduces the main sources of intrinsic variability and how it can affect the performance of circuit, as well as overall device yield. Small random variations in the number of dopant atoms in a transistor's channel, and other inaccuracies in the manufacturing process lead to significant variations in threshold voltage and leakage current, causing timing and power consumption violations which can result in critical device failure. The models used for this work take these variations into account to produce accurate random device behaviour. Chapter 3 presents a range of approaches to tackle variability at different stages in the design cycle of VLSI devices, from pre-fabrication modelling, through manufacturing approaches, and up to post-fabrication adaptive strategies such as reconfigurable hardware platforms. One of the identified platforms, the PAnDA architecture, comprises transistors of configurable width, which allow for the adjustment of performance that is necessary in the face of variability.

The PAnDA architecture provides a promising platform not only to study the effects of variability in deep sub-micron processes, but also to tackle those effects at a post-fabrication stage. Its low-level analogue reconfiguration resources are designed to also enable it to cope with environmental (or temporal) variations which can be critically damaging for a design. These resources also allow for a fine-grained control over transistor properties which is not achievable through standard digital circuit design. By allowing the control over transistor geometry, this architecture bridges the gap between digital and analogue circuit design.

In order to allow for the characterisation of the impact of variability on PAnDA devices, statistically-enhanced models of the architecture are created through the use of the Random-

SPICE tool. Chapter 4 describes the PAnDA hierarchical architecture from the transistor- to the chip-level, along with the hierarchical models that are created for each abstraction layer. As the complexity of the model grows, so does the computation required to extract performance metrics from it. To bring the effects of variability on transistor performance to the chip-level, a novel approach is described in which the upper-level layers (chip-, CLB-levels, along with reconfiguration resources) are modelled in VHDL and implemented on an FPGA, and variability-aware data from low-level layers (CAB-, CT-layers) are then incorporated into the model through the use of digital timing control mechanisms. The variability-aware data is generated for a set of Configurable Transistor sizes across the function-set offered by the PAnDA architecture at the Configurable Analogue Block level, resulting in a library of CM Cards which can be invoked by the top-level model to generate *virtual physical instances* of a PAnDA device. The model which includes both the top-layers implemented in VHDL and bottom-layer variability-aware data is called the PAnDA Emulator v1.

The creation of CM Cards is in some respects analogous to BSIM models generated by the GSS toolkit, in that the latter creates transistor models through different combinations of values of device parameters, and the former creates more abstract CAB models through different combinations of RandomSPICE-generated transistors. The use of the same seed for every RandomSPICE run ensures that the same "virtual transistors" are used for every function in a CM Card, making digital reconfiguration a more meaningful mechanism. When a purely statistical approach is taken to delay extraction, such as SSTA, the correlation between functions is not necessarily maintained when it is applied to reconfigurable fabrics, and would require additional efforts to design mechanisms to preserve it.

The modelling efforts described in this work were mainly focused on providing a platform which emulated both the *functionality* of a PAnDA device and also the *variations* in performance that are caused by random parameter fluctuation. Limiting factors in the implementation of the PAnDA Emulator – most importantly the limited memory resources on the Virtex-5 serving as the Control Module – made it necessary to include modelling simplifications which contributed toward an inaccuracy with respect to a full SPICE simulation. There is, however, a decoupling between the feature extraction stage and the FPGA-based Emulator. More sophisticated methods can be applied to the former to increase the accuracy of the extracted values, which might include varying slew-rates and a wider range of load capacitances. These modelling upgrades carry with them an increase in memory requirements, which can be fulfilled by including external dedicated hardware in the design of

the Emulator. Another approach to increasing the dimensionality of the model would be to reduce the function set currently implemented by each CAB, since at the present version of the model each function requires approximately 1/8 of the memory used by each CM Card.

7.4 PAnDA Emulator & VPIs

In a novel approach, Chapter 5 takes the developed platform, the PAnDA Emulator v1, and describes how it can be used to create *virtual physical instances* of circuits, making use of hardware parallelism to allow for faster assessment of the impact of variability. A 3-stage ring oscillator and a 2-bit multiplier are mapped to the Emulator, and variations in performance due to intrinsic variability are observed through the use of a custom designed hardware module, embedded in the Emulator v1, which extracts frequency and propagation delay measurements for the mapped circuits. This module consists of a finite-state machine which makes use of the artificial time created by the Emulator v1, as part of the novelty of the approach. A study on the frequency variation observed in ring oscillators constructed using different-sized CTs is carried out, and wider CTs show a larger variation than smaller ones, as expected. Four multipliers are also evaluated, with input vectors applied which activate every possible path in the circuit. These performance metrics are extracted for the emulated circuits, and equivalent simulations are carried out in SPICE, for modelling validation. Inaccuracies are identified due to modelling simplifications such as uniform input slew-rate and output load. In the case of the ring oscillator, modelling adjustments are included which correct the input slew-rate and output load to account for varying CT sizes in each stage of the oscillator. These adjustments prove to reduce the inaccuracy of the measured frequencies with respect to SPICE from a worst-case of 34.76% error down to 3.17%.

Coming back to the objectives outlined in Chapter 1, the inclusion of statistically-enhanced models in a reconfigurable model of the PAnDA architecture have allowed for the evaluation of the performance of the PAnDA architecture under the effect of intrinsic variability, the first objective on the list. The limitation of this approach is that generic varying fan-out and fan-in models for each function have not been completed, and is within the scope of future work.

The second objective on the list concerned the acceleration of the performance characterisation process, something which is achieved through the parallel nature of the hardware side of the PAnDA model, implemented in VHDL, and therefore this objective is also met.

The third objective on the list, and the last which concerns the development of the platform on which optimisation mechanisms can be explored, required that methodology which allows for the performance evaluation of large numbers of devices be available. This objective is also met, given that VPIs can be created on the Emulator simply by loading different CM cards when instantiating a device. Characterising the performance of each device is a matter of loading the appropriate input stimulus and using the output logging mechanisms of the Emulator to extract the required measurements. These measurements are currently limited to either propagation delay or oscillating frequency (if applicable), but could easily be expanded to include other performance features such as power consumption, paving the way for multi-objective optimisation mechanisms to also be implemented.

7.5 Exploiting variability for optimisation

With limited memory resources, a version of the Emulator is created which replaces the varying CT sizes with varying input-slew rates and output loads, to increase modelling accuracy with respect to SPICE, referred to as PAnDA Emulator v2. Chapter 6 starts by introducing this updated platform, and describes how the digital reconfiguration resources of the Emulator can be used for performance optimisation. A new set of test-circuits, the ISCAS'87 C17 benchmark and a transparent latch are then implemented on the Emulator v2. Experiments are carried out which demonstrate the increase in modelling accuracy with respect to SPICE, at the expense of non-varying CT sizes. These experiments also show how large numbers of virtual devices can be created in a significantly shorter runtime than full RandomSPICE simulations, as part of the novelty of the approach. Having verified the impact of variability on the distribution of performance metrics on the designs mapped to both version of the PAnDA Emulator, the digital reconfiguration operations which can be applied to the platform are then introduced, making use of two flip-flop chains with custom control circuitry, also part of the novelty of this work. The Emulator v1 can make use of both varying CT sizes and function permutations, whereas the Emulator v2, with fixed CT sizes, can only make use of function permutations.

Genetic Algorithms, which take inspiration from the biological process of evolution, are then presented as an efficient optimisation approach, which make as few assumptions about the implemented circuits as possible. These take in performance measurements and make use of bio-inspired mechanisms to generate bitstreams which are used to reconfigure the Emulator. Much of the novelty of this work comes from the application of these algorithms to create bitstreams which cannot damage the device, and perform only functionally-neutral operations.

Operating-point matching experiments are then carried out on the Emulator v1, using the ring oscillator and 2-bit multiplier circuits. In each experiment, multiple instances of each circuit are mapped to the Emulator, with randomly assigned CT sizes. One instance is then randomly chosen as the target for performance, and digital reconfiguration is used on the others in order to minimise the differences in frequency for the ring oscillator, and propagation delay for the multiplier. In the case of the ring oscillator, the maximum difference in frequency is reduced from 8.2% down to 1.3%. For the multiplier, the maximum difference in propagation delay is reduced from 19.5% down to 3.2%. If a given manufactured circuit faces timing violations due to intrinsic variability, having the ability to alter its characteristics in order to eliminate these violations can provide a significant increase in device yield.

Worst-case propagation delay minimisation of a C17 circuit is then carried out on the Emulator v2, making use of the function permutations exclusively. Each of the 1000 instances previously characterised is optimised, with the maximum propagation delay observed being reduced by 4%. The coefficient of variation (σ/μ) is also reduced by around 50%, suggesting that not only can performance be improved by variation-aware reconfiguration, but variability can indeed be reduced by making use of these resources.

Although more configuration flexibility – resulting from control over the sizes of Configurable Transistors on PAnDA – is likely to make performance optimisation even more effective, standard function permutations which are applicable to more traditional FPGA architectures can provide significant benefits for variability-tolerance, as this thesis has demonstrated.

The fourth item on the list of objectives outlined in Chapter 1 described the need for an automated method for the performance optimisation process. This has been achieved by means of the algorithm implemented on the Control Module which performs optimisation across large numbers of virtual physical instances, as described in Figure 6.23.

The final item on the list of objectives required that substrate variations and digital reconfiguration resources should be used to both improve circuit performance and to allow for

performance standardisation across large numbers of devices. The GA applied to the test circuits on the Emulator v1 has shown that performance standardisation can be achieved, but the limitations of the approach should again be considered: as the resulting optimised circuits exhibit some disagreement with full SPICE simulations, the models would need refinement before the findings can be translated to actual hardware performance predictions. An initial effort in model refinement is undertaken out for the experiments carried out on the Emulator v2, which narrows the gap between the model's predictions and a full SPICE simulation. Although this gap was not completely eliminated, it is still possible to say that circuit performance optimisation across large numbers of devices has been achieved, as demonstrated by the optimisation experiments carried out on the C17 and transparent latch test-circuits.

7.6 Future Directions

As a continuation of the modelling work undertaken for the Emulator, efforts can be allocated towards increasing the accuracy of the model, by extracting performance characteristics (*i.e.* delay, power) for various input slew-rates and output loads. With the proposed approach, an increase in accuracy will likely always be supported by an equivalent increase in data collected, and therefore also an increase in the amount of memory required to store a CM Card. This issue could be relaxed by including large buffers outside of each CAB to standardise output load, although this introduces a potentially unwanted hardware overhead.

Assuming an increase in modelling accuracy, CT configurations which can potentially minimise the impact of variability can be studied, *e.g.* combinations of large and small *versus* combinations of medium-sized transistors. An increase in accuracy is necessary also to allow for the study of potential updates to the architecture.

Another future direction which can be identified is the development of automatic mapping tools for both the Emulator and PAnDA, as the configuration bitstream is still done by hand, and it can become a very time-consuming process for the implementation of complex designs on both platforms. Since the Emulator mimics the functionality of the PAnDA architecture, it can also be used for bitstream validation, protecting the PAnDA devices from potentially damaging configurations.

Given that the first fully functional version of the PAnDA architecture has recently been made available, the performance optimisation methods developed in this work are currently

being exported so that similar experiments can be carried out in hardware, and the results can be compared with the those described in this work.

Overall, variability is a growing concern among both designers and manufacturers, and PAnDA provides a substrate on which reliable circuits can be implemented. The experiments described in this work aimed to improve the performance of circuits mapped to the PAnDA architecture, which has been achieved through the use of an Emulator.

With more variability-aware mapping techniques currently being developed by the both academia and industry, it is likely that these efforts can be expanded in the next decade to allow for online performance characterisation and dynamic reconfiguration to manipulate the behaviour of implemented circuits. As programmable devices such as FPGAs begin to close the performance gap with ASICs, digital reconfiguration is likely to become a major area of interest in the field of embedded systems, as inevitable device variations make it possible to constantly explore the fabric to find better implementations of a given design.

Appendix A

Source files

The USB drive and CD-ROM included with the thesis contain the relevant files necessary to implement the Control Module on a Virtex-5 LX110T FPGA, and the PAnDA Emulator on a Virtex-6 XC6VLX760.

The folders are organised as follows:

- **Control Module (Virtex5)**
  - **Hardware** – Various files necessary to rebuild the Control Module hardware project.
  - **Software**
    - **control.c** – This source file contains the data extracted from RandomSPICE simulations, the functions used in the Genetic Algorithm, and the communications protocol for data exchange with the PAnDA Emulator;
    - **control.h** – Header file for control.c;
    - **panda_control.c** – Source file containing main function.
- **PAnDA Emulator (Virtex6)**
  - **Hardware**
    - **Various files** necessary to rebuild the PAnDA Emulator hardware project;
    - **Emulator IP Core** – Files which generate the hierarchical design of the Emulator, written in VHDL;
  - **Software**

  - **panda_eins_functions.c** – Source file containing the functions used for Emulator reconfiguration, as well as communications protocol for data exchange with the Control Module;
  - **panda_eins_functions.h** – Header file for panda_eins_functions.c;
  - **panda_eins.c** – Source file containing the main function.

- **Thesis.pdf** – A digital copy of this thesis.

Bibliography

[1] K. J. Kuhn, "Moore's Law Past 32nm: Future Challenges in Device Scaling," in *2009 13th International Workshop on Computational Electronics*. IEEE, may 2009, pp. 1–6.

[2] R. Jaramillo-Ramirez, J. Jaffari, and M. Anis, "Variability-aware design of subthreshold devices," *Proceedings - IEEE International Symposium on Circuits and Systems*, pp. 1196–1199, 2008.

[3] A. Asenov, A. R. Brown, G. Roy, B. Cheng, C. Alexander, C. Riddet, U. Kovac, A. Martinez, N. Seoane, and S. Roy, "Simulation of statistical variability in nano-CMOS transistors using drift-diffusion, Monte Carlo and non-equilibrium Greens function techniques," *Journal of Computational Electronics*, vol. 8, no. 3-4, pp. 349–373, sep 2009.

[4] F. Schellenberg, "A little light magic," *IEEE Spectrum*, vol. 40, no. 9, pp. 34–39, sep 2003.

[5] Y. Tsividis and C. McAndrew, "Proximity Effect Modeling," in *Operation and Modeling of the MOS Transistor*, 3rd ed. Oxford University Press, 2011, ch. 9.

[6] D. T. Reid, "Large-scale simulations of intrinsic parameter fluctuations in nano-scale MOSFETs," Ph.D. dissertation, University of Glasgow, 2010.

[7] A. Asenov, "Random Dopant Induced Threshold Voltage Lowering and Fluctuations in Sub 50 nm MOSFETs: a Statistical 3D 'Atomistic' Simulation Study," *Nanotechnology*, vol. 10, no. 2, pp. 153–158, jun 1999.

[8] S. M. M. A. Alam, "A comprehensive model of PMOS NBTI degradation," *Microelectronics Reliability*, vol. 47, no. 6, pp. 853–862, 2007.

[9] J. A. Walker, M. A. Trefzer, and A. M. Tyrrell, "A Reconfigurable Architecture for Current and Future Challenges in Electronic Design and Technology," in *Workshop on Variability modelling and mitigation techniques in current and future technologies (VAMM) DATE 2012*, 2012.

[10] K. Bernstein, D. J. Frank, A. E. Gattiker, W. Haensch, B. L. Ji, S. R. Nassif, E. J. Nowak, D. J. Pearson, and N. J. Rohrer, "High-performance CMOS variability in the 65-nm regime and beyond," *IBM Journal of Research and Development*, vol. 50, no. 4.5, pp. 433–449, jul 2006.

[11] Gilson Wirth, "Reliability and Yield of MOS Devices and Circuits," 2011.

[12] S. Kothawade, K. Chakraborty, and S. Roy, "Analysis and mitigation of NBTI aging in register file: An end-to-end approach," in *12th International Symposium on Quality Electronic Design*. IEEE, mar 2011, pp. 1–7.

[13] M. Merrett, P. Asenov, M. Zwolinski, D. Reid, C. Millar, S. Roy, S. Furber, and A. Asenov, "Modelling circuit performance variations due to statistical variability: Monte Carlo static timing analysis," in *Design, Automation & Test in Europe*. IEEE, mar 2011, pp. 1–4.

[14] G. S. Simulations, "RandomSPICE," 2015. [Online]. Available: http://www.goldstandardsimulations.com/services/service-simulations/circuit-simulation/random-spice/

[15] R. Rudolf, R. Wilcock, and P. R. Wilson, "Reliability improvement and online calibration of ICs using configurable analogue transistors," in *2012 IEEE Aerospace Conference*. IEEE, mar 2012, pp. 1–8.

[16] P. Wilson and R. Wilcock, "Yield improvement using configurable analogue transistors," *Electronics Letters*, vol. 44, no. 19, p. 1132, 2008.

[17] ——, "Optimal sizing of configurable devices to reduce variability in integrated circuits," in *2009 Design, Automation & Test in Europe Conference & Exhibition*. IEEE, apr 2009, pp. 1385–1390.

[18] C. Forzan and D. Pandini, "Statistical static timing analysis: A survey," *Integration, the VLSI Journal*, vol. 42, no. 3, pp. 409–435, jun 2009.

[19] S. Garg and D. Marculescu, "Impact of manufacturing process variations on performance and thermal characteristics of 3D ICs: Emerging challenges and new solutions," in *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*. IEEE, may 2013, pp. 541–544.

[20] D. Ernst, N. S. K. N. S. Kim, S. Das, S. Pant, R. Rao, T. P. T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: a low-power pipeline based on circuit-level timing speculation," *Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36.*, 2003.

[21] Xilinx, "Xilinx UG364 - Virtex-6 FPGA Configurable Logic Block User Guide," 2012.

[22] S. Hauck and A. DeHon, *Reconfigurable Computing: The Theory and Practice of FPGA-Based Computation*. Morgan Kaufmann Publishers Inc., nov 2007.

[23] E. Stott, Z. Guan, J. M. Levine, J. S. J. Wong, and P. Y. K. Cheung, "Variation and Reliability in FPGAs," *IEEE Design & Test*, vol. 30, no. 6, pp. 50–59, dec 2013.

[24] J. Langeheine, J. Becker, S. Folling, K. Meier, and J. Schemmel, "A CMOS FPTA chip for intrinsic hardware evolution of analog electronic circuits," in *Proceedings Third NASA/DoD Workshop on Evolvable Hardware. EH-2001.* IEEE Comput. Soc, 2001, pp. 172–175.

[25] J. A. Walker, M. A. Trefzer, S. J. Bale, and A. M. Tyrrell, "PAnDA: A reconfigurable architecture that adapts to physical substrate variations," *IEEE Transactions on Computers*, vol. 62, no. 8, pp. 1584–1596, 2013.

[26] M. A. Trefzer, J. A. Walker, and A. M. Tyrrell, "A Programmable Analogue and Digital Array for Bio-inspired Electronic Design Optimization at Nano-scale Silicon Technology Nodes," in *45th Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, 2011, pp. 1537–1541.

[27] J. A. Walker, M. A. Trefzer, S. J. Bale, and A. M. Tyrrell, "Exploiting the reconfigurability of the PAnDA architecture to overcome physical substrate variations," in *2013 IEEE International Conference on Evolvable Systems (ICES)*. IEEE, apr 2013, pp. 37–42.

[28] M. Trefzer, A. Tyrrell, and J. Walker, "Field-programmable gate array," sep 2013. [Online]. Available: http://www.google.com.ar/patents/WO2013064839A3

[29] K. Agarwal and S. Nassif, "Characterizing process variation in nanometer CMOS," *Proceedings - Design Automation Conference*, pp. 396–399, 2007.

[30] M. A. Trefzer, "PAnDA Homepage," 2015. [Online]. Available: http://www.panda.ac.uk/index.html

[31] P. B. Campos, D. M. R. Lawson, S. J. Bale, J. A. Walker, M. a. Trefzer, and A. M. Tyrrell, "Overcoming Faults using Evolution on the PAnDA Architecture," in *2013 IEEE Congress on Evolutionary Computation*. Ieee, jun 2013, pp. 613–620.

[32] P. B. Campos, M. A. Trefzer, J. A. Walker, S. J. Bale, and A. M. Tyrrell, "Optimising ring oscillator frequency on a novel FPGA device via partial reconfiguration," in *2014 IEEE International Conference on Evolvable Systems*. IEEE, dec 2014, pp. 93–100.

[33] N. Dahir, P. B. Campos, G. Tempesti, M. A. Trefzer, and A. M. Tyrrell, "Characterisation of Feasibility Regions in FPGAs under Adaptive DVFS," in *IEEE International Conference on Field-programmable Logic and Applications (FPL)*. IEEE, 2015.

[34] M. Orshansky, L. Milor, P. Chen, K. Keutzer, and C. Hu, "Impact of spatial intrachip gate length variability on the performance of high-speed digital circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 5, pp. 544–553, 2002.

[35] Y. Cao and L. T. Clark, "Mapping Statistical Process Variations Toward Circuit Performance Variability: An Analytical Modeling Approach," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 10, pp. 1866–1873, oct 2007.

[36] X. Zhang and X. Bai, "Process variability-induced timing failures A challenge in nanometer CMOS low-power design," *2008 IEEE International Conference on Integrated Circuit Design and Technology and Tutorial*, pp. 159–162, jun 2008.

[37] S. Chandra, "Variability-Aware System-Level Design and Analysis," Ph.D. dissertation, University of California, San Diego, 2009.

[38] S. Nassif, "Within-chip variability analysis," *International Electron Devices Meeting 1998. Technical Digest*, pp. 283–286, 1998.

[39] D. Sylvester, K. Agarwal, and S. Shah, "Variability in nanometer CMOS: Impact, analysis, and minimization," *Integration, the VLSI Journal*, vol. 41, no. 3, pp. 319–339, may 2008.

[40] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, no. 8, 1965.

[41] R. Dennard, F. Gaensslen, V. Rideout, E. Bassous, and A. LeBlanc, "Design of ion-implanted MOSFET's with very small physical dimensions," *IEEE Journal of Solid-State Circuits*, vol. 9, no. 5, pp. 256–268, oct 1974.

[42] S. Thompson, M. Armstrong, C. Auth, M. Alavi, M. Buehler, R. Chau, S. Cea, T. Ghani, G. Glass, T. Hoffman, C.-H. Jan, C. Kenyon, J. Klaus, K. Kuhn, Z. Ma, B. Mcintyre, K. Mistry, A. Murthy, B. Obradovic, R. Nagisetty, P. Nguyen, S. Sivakumar, R. Shaheed, L. Shifren, B. Tufts, S. Tyagi, M. Bohr, and Y. El-Mansy, "A 90-nm Logic Technology Featuring Strained-Silicon," *IEEE Transactions on Electron Devices*, vol. 51, no. 11, pp. 1790–1797, nov 2004.

[43] A. Asenov, "Random dopant induced threshold voltage lowering and fluctuations in sub-0.1 μm MOSFET's: A 3-D "atomistic" simulation study," *IEEE Transactions on Electron Devices*, vol. 45, no. 12, pp. 2505–2513, 1998.

[44] P. E. Allen and D. R. Holberg, *CMOS Analog Circuit Design (The Oxford Series in Electrical and Computer Engineering)*. Oxford University Press, 1987.

[45] K. Mistry, C. Allen, C. Auth, B. Beattie, D. Bergstrom, M. Bost, M. Brazier, M. Buehler, A. Cappellani, R. Chau, C. H. Choi, G. Ding, K. Fischer, T. Ghani, R. Grover, W. Han, D. Hanken, M. Hattendorf, J. He, J. Hicks, R. Huessner, D. Ingerly, P. Jain, R. James, L. Jong, S. Joshi, C. Kenyon, K. Kuhn, K. Lee, H. Liu, J. Maiz, B. McIntyre, P. Moon, J. Neirynck, S. Pae, C. Parker, D. Parsons, C. Prasad, L. Pipes, M. Prince, P. Rarade, T. Reynolds, J. Sandford, L. Shifren, J. Sebastian, J. Seiple, D. Simon, S. Sivakumar, P. Smith, C. Thomas, T. Troeger, P. Vandervoorn, S. Williams, and K. Zawadzki, "A 45nm Logic Technology with High-k+Metal Gate Transistors, Strained Silicon, 9 Cu Interconnect Layers, 193nm Dry Patterning, and 100% Pb-free Packaging," in *2007 IEEE International Electron Devices Meeting*. IEEE, 2007, pp. 247–250.

[46] A. Wong, R. Ferguson, and S. Mansfield, "The mask error factor in optical lithography," *IEEE Transactions on Semiconductor Manufacturing*, vol. 13, no. 2, pp. 235–242, may 2000.

[47] T. A. Brunner, "Impact of lens aberrations on optical lithography," *IBM Journal of Research and Development*, vol. 41, no. 1.2, pp. 57–67, jan 1997.

[48] J. Watts, N. Lu, C. Bittner, S. Grundon, and J. Oppold, "Compact Modeling Modeling FET Variation within a chip as a Function of Circuit Design and Layout Choices," in *Technical Proceedings of the 2005 Workshop on Compact Modeling*, 2005.

[49] T. Hook, J. Brown, P. Cottrell, E. Adler, D. Hoyniak, J. Johnson, and R. Mann, "Lateral ion implant straggle and mask proximity effect," *IEEE Transactions on Electron Devices*, vol. 50, no. 9, pp. 1946–1951, sep 2003.

[50] J. V. Faricelli, "Layout-dependent proximity effects in deep nanoscale CMOS," in *IEEE Custom Integrated Circuits Conference 2010*. IEEE, sep 2010, pp. 1–8.

[51] K. J. Kuhn, "Reducing Variation in Advanced Logic Technologies: Approaches to Process and Design for Manufacturability of Nanoscale CMOS," in *2007 IEEE International Electron Devices Meeting*. IEEE, 2007, pp. 471–474.

[52] J. Croon, G. Storms, S. Winkelmeier, I. Pollentier, M. Ercken, S. Decoutere, W. Sansen, and H. Maes, "Line edge roughness: characterization, modeling and impact on device behavior," in *Digest. International Electron Devices Meeting,*. IEEE, 2002, pp. 307–310.

[53] A. R. Brown, G. Roy, and A. Asenov, "Poly-Si-Gate-Related Variability in Decananometer MOSFETs With Conventional Architecture," *IEEE Transactions on Electron Devices*, vol. 54, no. 11, pp. 3056–3063, nov 2007.

[54] T. Herrmann, W. Klix, R. Stenzel, S. Duenkel, R. Illgen, J. Hoentschel, T. Feudel, and M. Horstmann, "Line Edge and Gate Interface Roughness Simulations of Advanced VLSI SOI-MOSFETs," in *Simulation of Semiconductor Processes and Devices*. Springer Vienna, 2007, pp. 101–104.

[55] E. Baravelli, M. Jurczak, N. Speciale, K. De Meyer, and A. Dixit, "Impact of LER and Random Dopant Fluctuations on FinFET Matching Performance," *IEEE Transactions on Nanotechnology*, vol. 7, no. 3, pp. 291–298, may 2008.

[56] X. Wang, B. Cheng, A. R. Brown, C. Millar, J. B. Kuang, S. Nassif, and A. Asenov, "Impact of statistical variability and charge trapping on 14 nm SOI FinFET SRAM cell stability," in *2013 Proceedings of the European Solid-State Device Research Conference (ESSDERC)*. IEEE, sep 2013, pp. 234–237.

[57] L. Mark, "ECE 612 Lecture 19: Device Variability," nov 2008. [Online]. Available: https://nanohub.org/resources/5856

[58] J. J. Kim, R. Rao, S. Mukhopadhyay, and C. T. Chuang, "Ring oscillator circuit structures for measurement of isolated NBTI/PBTI effects," *Proceedings - 2008 IEEE International Conference on Integrated Circuit Design and Technology, ICICDT*, pp. 163–166, jun 2008.

[59] A. Bansal, R. Rao, J. J. Kim, S. Zafar, J. H. Stathis, and C. T. Chuang, "Impact of NBTI and PBTI in SRAM Bit-cells: Relative Sensitivities and Guidelines for Application-Specific Target Stability/Performance," *IEEE International Reliability Physics Symposium Proceedings*, pp. 745–749, 2009.

[60] S. Mahapatra, *Fundamentals of Bias Temperature Instability in MOS Transistors*, S. Mahapatra, Ed. Springer India, 2016.

[61] J. Keane and C. H. Kim, "Transistor Aging," *IEEE Spectrum*, 2011. [Online]. Available: http://spectrum.ieee.org/semiconductors/processors/transistor-aging

[62] W. Dai, "Timing analysis taking into account interconnect process variation," in *6th International Workshop on Statistical Methodology*. IEEE, 2001, pp. 51–53.

[63] F. Liu, L. Pileggi, and S. Nassif, "Modeling Interconnect Variability Using Efficient Parametric Model Order Reduction," in *Design, Automation and Test in Europe*. IEEE, 2005, pp. 958–963.

[64] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter variations and impact on circuits and microarchitecture," in *Proceedings 2003. Design Automation Conference (IEEE Cat. No.03CH37451)*. IEEE, 2003, pp. 338–342.

[65] Y. Wang, M. Zwolinski, A. Appleby, M. Scoones, S. Caldwell, T. Azam, P. Hurat, and C. Pitchford, "Analysis, quantification, and mitigation of electrical variability due to layout dependent effects in SOC designs," in *SPIE Advanced Lithography*, M. E. Mason, Ed. International Society for Optics and Photonics, mar 2012.

[66] S. G. Narendra, "Effect of MOSFET threshold voltage variation on high-performance circuits," Ph.D. dissertation, Massachusetts Institute of Technology, 2002.

[67] J. Srinivasan, S. Adve, P. Bose, and J. Rivers, "The impact of technology scaling on lifetime reliability," in *International Conference on Dependable Systems and Networks, 2004.* IEEE, 2004, pp. 177–186.

[68] B. Greskamp, S. R. Sarangi, and J. Torrellas, "Threshold Voltage Variation Effects on Aging-Related Hard Failure Rates," in *IEEE International Symposium on Circuits and Systems.* IEEE, may 2007, pp. 1261–1264.

[69] M. Eisele, J. Berthold, D. Schmitt-Landsiedel, and R. Mahnkopf, "The impact of intra-die device parameter variations on path delays and on the design for yield of low voltage digital circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 5, no. 4, pp. 360–368, 1997.

[70] R. Wang, P. Friedberg, A. Vladimirescu, and J. Rabaey, "Yield optimization with energy-delay constraints in low-power digital circuits," *2003 IEEE Conference on Electron Devices and Solid-State Circuits*, pp. 285–288, 2003.

[71] P. Pouyan, E. Amat, and A. Rubio, "Process variability-aware proactive reconfiguration technique for mitigating aging effects in nano scale SRAM lifetime," *Proceedings of the IEEE VLSI Test Symposium*, pp. 240–245, apr 2012.

[72] B. Rebaud, M. Belleville, C. Bernard, M. Robert, P. Maurine, and N. Azemard, "A comparative study of variability impact on static flip-flop timing characteristics," *Proceedings - 2008 IEEE International Conference on Integrated Circuit Design and Technology, ICICDT*, pp. 167–170, jun 2008.

[73] J. A. Hilder, "Evolving Variability Tolerant Logic," Ph.D. dissertation, University of York, 2010.

[74] B. Harish, N. Bhat, and M. B. Patil, "Bridging technology-CAD and design-CAD for variability aware Nano-CMOS circuits," in *IEEE International Symposium on Circuits and Systems.* IEEE, may 2009, pp. 2309–2312.

[75] L. W. Nagel and D. Pederson, "SPICE (Simulation Program with Integrated Circuit Emphasis)," EECS Department, University of California, Berkeley, Tech. Rep., 1973.

[76] H. Shichman and D. Hodges, "Modeling and simulation of insulated-gate field-effect transistor switching circuits," *IEEE Journal of Solid-State Circuits*, vol. 3, no. 3, pp. 285–289, sep 1968.

[77] D. P. Foty, *MOSFET modeling with SPICE: principles and practice*. Prentice-Hall, Inc., jan 1997.

[78] N. Paydavosi, T. H. Morshed, D. D. Lu, W. M. Yang, M. V. Dunga, X. J. Xi, J. He, W. Liu, M. C. Kanyu, X. Jin, J. J. Ou, M. Chan, A. M. Niknejad, and C. Hu, "BSIM4v4.8.0 MOSFET Model - User's Manual," 2013.

[79] P. Nenzi and H. Vogt, "Ngspice User Manual, version 25," 2013.

[80] G. S. Simulations, "Gold Standard Simulations - News."

[81] G. Roy, A. R. Brown, F. Adamu-Lema, S. Roy, and A. Asenov, "Simulation Study of Individual and Combined Sources of Intrinsic Parameter Fluctuations in Conventional Nano-MOSFETs," *IEEE Transactions on Electron Devices*, vol. 53, no. 12, pp. 3063–3070, dec 2006.

[82] D. Reid, C. Millar, S. Roy, G. Roy, R. Sinnott, G. Stewart, G. Stewart, and A. Asenov, "Enabling cutting-edge semiconductor simulation through grid technology." *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, vol. 367, no. 1897, pp. 2573–84, jun 2009.

[83] G. Pitcher, "GSS makes multimillion dollar sale to GlobalFoundries," *NewElectronics*, jul 2014.

[84] M. A. Trefzer, A. Tyrrell, S. J. Bale, and J. A. Walker, "High-Sigma Performance Analysis using Multi-Objective Evolutionary Algorithms," in *Design Automation Conference*, 2015.

[85] Shupeng Sun, Xin Li, Hongzhou Liu, Kangsheng Luo, and Ben Gu, "Fast Statistical Analysis of Rare Circuit Failure Events via Scaled-Sigma Sampling for High-Dimensional Variation Space," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 7, pp. 1096–1109, jul 2015.

[86] T. McWilliams, "Verification of Timing Constraints on Large Digital Systems," pp. 139–147, 1980.

[87] A. Dunlop, V. Agrawal, D. Deutsch, M. Jukl, P. Kozak, and M. Wiesel, "Chip Layout Optimization Using Critical Path Weighting," in *21st Design Automation Conference Proceedings.* IEEE, 1984, pp. 133–136.

[88] K. Butler, "Deep submicron: is test up to the challenge?" in *Proceedings of 1995 IEEE International Test Conference (ITC).* Int. Test Conference, 1995, p. 923.

[89] M. Escalante and N. Dimopoulos, "A probabilistic timing analysis for synthesis in microprocessor interface design," in *IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing. Proceedings.* IEEE, 1995, pp. 277–280.

[90] S. Tsukiyama, M. Tanaka, and M. Fukui, "A statistical static timing analysis considering correlations between delays," in *Proceedings of the ASP-DAC 2001. Asia and South Pacific Design Automation Conference 2001 (Cat. No.01EX455).* IEEE, 2001, pp. 353–358.

[91] S. Tsukiyama, "Toward stochastic design for digital circuits - statistical static timing analysis," in *ASP-DAC 2004: Asia and South Pacific Design Automation Conference 2004 (IEEE Cat. No.04EX753).* IEEE, 2004, pp. 762–767.

[92] I. Nitta, T. Shibuya, and K. Homma, "Statistical Static Timing Analysis Technology," *FUJITSU Scientific & Technical Journal*, vol. 43, no. 4, pp. 516–523, 2007.

[93] C. Visweswariah, K. Ravindran, K. Kalafala, S. Walker, S. Narayan, D. Be, J. Piaget, N. Venkateswaran, and J. Hemmett, "First-Order Incremental Block-Based Statistical Timing Analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 10, pp. 2170–2180, oct 2006.

[94] D. Blaauw, K. Chopra, A. Srivastava, and L. Scheffer, "Statistical Timing Analysis: From Basic Principles to State of the Art," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 4, pp. 589–607, apr 2008.

[95] M. Kwak and L. Guo, "Phase-Shift Lithography," in *Encyclopedia of Microfluidics and Nanofluidics SE - 1729-4*, D. Li, Ed. Springer US, 2014, pp. 1–10.

[96] B. J. Lin, "Lithography till the end of Moore's law," in *Proceedings of the 2012 ACM international symposium on International Symposium on Physical Design - ISPD '12.* New York, New York, USA: ACM Press, mar 2012, p. 1.

[97] Y. Cai, Q. Zhou, X. Hong, R. Shi, and Y. Wang, "Application of optical proximity correction technology," *Science in China Series F: Information Sciences*, vol. 51, no. 2, pp. 213–224, feb 2008.

[98] H. Zhang, "Reduction of CD variance by using optical proximity correction for patterning with DUV phase shift mask," *Microelectronic Engineering*, vol. 30, no. 1-4, pp. 119–122, jan 1996.

[99] B. Yu and D. Z. Pan, *Design for Manufacturability with Advanced Lithography*. Springer International Publishing, 2016.

[100] J. W. Tschanz, J. T. Kao, S. G. Narendra, R. Nair, D. a. Antoniadis, A. P. Chandrakasan, and V. De, "Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 11, pp. 1396–1402, 2002.

[101] S. Narang and A. P. Srivastava, "NBTI detection methodology for building tolerance with respect to NBTI effects employing adaptive body bias," in *2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015]*. IEEE, mar 2015, pp. 1–7.

[102] T. Chen and S. Naffziger, "Comparison of adaptive body bias (ABB) and adaptive supply voltage (ASV) for improving delay and leakage under the presence of process variation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 5, pp. 888–899, oct 2003.

[103] S. Garg and D. Marculescu, "System-Level Leakage Variability Mitigation for MPSoC Platforms Using Body-Bias Islands," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 12, pp. 2289–2301, dec 2012.

[104] W. Davis, J. Wilson, S. Mick, C. Mineo, A. Sule, M. Steer, and P. Franzon, "Demystifying 3D ICs: The Pros and Cons of Going Vertical," *IEEE Design and Test of Computers*, vol. 22, no. 6, pp. 498–510, jun 2005.

[105] S. Garg and D. Marculescu, "System-level process variability analysis and mitigation for 3D MPSoCs," in *2009 Design, Automation & Test in Europe Conference & Exhibition*. IEEE, apr 2009, pp. 604–609.

[106] G. Estrin, "Reconfigurable computer origins: The UCLA fixed-plus-variable (F+V) structure computer," *IEEE Annals of the History of Computing*, vol. 24, no. 4, pp. 3–9, oct 2002.

[107] ——, "Organization of computer systems," in *Papers presented at the May 3-5, 1960, western joint IRE-AIEE-ACM computer conference on - IRE-AIEE-ACM '60 (Western)*. New York, New York, USA: ACM Press, may 1960, p. 33.

[108] I. Kuon and J. Rose, "Quantifying and exploring the gap between FPGAs and ASICs: Measuring and exploring," in *Quantifying and Exploring the Gap Between FPGAs and ASICs: Measuring and Exploring*. New York, New York, USA: ACM Press, feb 2010, pp. 1–180.

[109] Xilinx Inc., "MicroBlaze Soft Processor Core," 2015. [Online]. Available: http://www.xilinx.com/tools/microblaze.htm

[110] Z. Guan, "Variation-aware and adaptive timing optimisation methods in Field Programmable Gate Arrays," Ph.D. dissertation, Imperial College London of Science, Technology and Medicine, 2013.

[111] P. Sedcole and P. Y. K. Cheung, "Parametric yield in FPGAs due to within-die delay variations," in *Proceedings of the 2007 ACM/SIGDA 15th international symposium on Field programmable gate arrays - FPGA '07*. New York, New York, USA: ACM Press, feb 2007, p. 178.

[112] H. Yu, Q. Xu, and P. H. W. Leong, "Fine-grained characterization of process variation in FPGAs," in *Proceedings - 2010 International Conference on Field-Programmable Technology, FPT'10*. IEEE, dec 2010, pp. 138–145.

[113] L. Scheffer, "Explicit computation of performance as a function of process variation," in *Proceedings of the 8th ACM/IEEE international workshop on Timing issues in the specification and synthesis of digital systems - TAU '02*. New York, New York, USA: ACM Press, dec 2002, p. 1.

[114] E. Stott, J. M. Levine, P. Y. Cheung, and N. Kapre, "Timing Fault Detection in FPGA-Based Circuits," in *2014 IEEE 22nd Annual International Symposium on Field-Programmable Custom Computing Machines*. IEEE, may 2014, pp. 96–99.

[115] N. Julai, A. Yakovlev, and A. Bystrov, "Error detection and correction of single event upset (SEU) tolerant latch," in *2012 IEEE 18th International On-Line Testing Symposium (IOLTS)*. IEEE, jun 2012, pp. 1–6.

[116] G. Lucas, C. Dong, and D. Chen, "Variation-Aware Placement With Multi-Cycle Statistical Timing Analysis for FPGAs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 11, pp. 1818–1822, nov 2010.

[117] C. Dong, S. Chilstedt, and D. Chen, "Variation Aware Routing for Three-Dimensional FPGAs," in *2009 IEEE Computer Society Annual Symposium on VLSI*. IEEE, 2009, pp. 298–303.

[118] J. Langeheine, S. Fölling, K. Meier, and J. Schemmel, "Towards a Silicon Primordial Soup: A Fast Approach to Hardware Evolution with a VLSI Transistor Array," in *Evolvable Systems: From Biology to Hardware, Third International Conference, ICES 2000*, 2000, pp. 123–132.

[119] A. Stoica, D. Keymeulen, R. Zebulum, A. Thakoor, T. Daud, Y. Klimeck, R. Tawel, and V. Duong, "Evolution of analog circuits on field programmable transistor arrays," in *Proceedings. The Second NASA/DoD Workshop on Evolvable Hardware*. IEEE Comput. Soc, 2000, pp. 99–108.

[120] Martin Albrecht Trefzer, "Evolution of Transistor Circuits," Ph.D. dissertation, Rupertus Carola University of Heidelberg, Seminarstrasse 2, 69120 Heidelberg, dec 2006.

[121] M. Merrett, Y. Wang, M. Zwolinski, K. Maharatna, and M. Alioto, "Design metrics for RTL level estimation of delay variability due to intradie (random) variations," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. IEEE, may 2010, pp. 2498–2501.

[122] Xilinx, "Xilinx UG702 - Partial Reconfiguration User Guide," 2012.

[123] A. Bassi, A. Veggetti, L. Croce, and A. Bogliolo, "Measuring the effects of process variations on circuit performance by means of digitally-controllable ring oscillators," *IEEE International Conference on Microelectronic Test Structures*, pp. 214–217, 2003.

[124] R. Rao, K. A. Jenkins, and J. J. Kim, "Completely digital on-chip circuit for local-random-variability measurement," in *Digest of Technical Papers - IEEE International Solid-State Circuits Conference*, vol. 51, 2008, pp. 412–414.

[125] C. 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.

[126] T. Higuchi, M. Iwata, I. Kajitani, H. Yamada, B. Manderick, Y. Hirao, M. Murakawa, S. Yoshizawa, and T. Furuya, "Evolvable hardware with genetic learning," in *1996 IEEE International Symposium on Circuits and Systems. Circuits and Systems Connecting the World. ISCAS 96*, vol. 4. IEEE, 1996, pp. 29–32.

[127] H. de Garis, "Evolvable Hardware: Genetic Programming of a Darwin Machine," in *Artificial Neural Nets and Genetic Algorithms*. Springer Vienna, 1993, pp. 441–449.

[128] A. Tyrrell and G. Greenwood, *Introduction to Evolvable Hardware: A Practical Guide for Designing Self-Adaptive Systems*, Wiley-IEEE Press, Ed. Wiley-IEEE Press, 2006.

[129] J. Johnson and Y. Rahmat-samii, "Genetic algorithm optimization and its application to antenna design," in *Proceedings of IEEE Antennas and Propagation Society International Symposium and URSI National Radio Science Meeting*, vol. 1. IEEE, 1994, pp. 326–329.

[130] D. Linden, "Using a real chromosome in a genetic algorithm for wire antenna optimization," in *IEEE Antennas and Propagation Society International Symposium 1997. Digest*, vol. 3. IEEE, 1997, pp. 1704–1707.

[131] J. Lohn, D. Linden, G. Hornby, and W. Kraus, "Evolutionary design of an X-band antenna for NASA's Space Technology 5 mission," in *IEEE Antennas and Propagation Society Symposium, 2004.*, vol. 3. IEEE, 2004, pp. 2313–2316 Vol.3.

[132] J. Robinson and Y. Rahmat-Samii, "Particle Swarm Optimization in Electromagnetics," *IEEE Transactions on Antennas and Propagation*, vol. 52, no. 2, pp. 397–407, feb 2004.

[133] Y. Yang, S. Yan, J. Liu, and J. Liang, "Genetic-Ant Colony Optimization algorithm and its application to design of antenna," in *2014 10th International Conference on Natural Computation (ICNC)*. IEEE, aug 2014, pp. 611–616.

[134] a. Thompson, "On the Automatic Design of Robust Electronics Through Artificial Evolution," *International Conference on Evolvable Systems: from Biology to Hardware*, pp. 13–24, sep 1998.

[135] J. F. Miller and P. Thomson, *Genetic Programming*, ser. Lecture Notes in Computer Science, R. Poli, W. Banzhaf, W. B. Langdon, J. Miller, P. Nordin, and T. C. Fogarty, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, vol. 1802.

[136] J. F. Miller, D. Job, and V. K. Vassilev, "Principles in the Evolutionary Design of Digital Circuits - Part I," *Genetic Programming and Evolvable Machines*, vol. 1, no. 1-2, pp. 7–35, 2000.

[137] Z. Vasicek and L. Sekanina, "A global postsynthesis optimization method for combinational circuits," in *2011 Design, Automation & Test in Europe*. IEEE, mar 2011, pp. 1–4.

[138] L. Sekanina, "Towards evolvable IP cores for FPGAs," in *NASA/DoD Conference on Evolvable Hardware, 2003. Proceedings.* IEEE Comput. Soc, 2003, pp. 145–154.

[139] R. Dobai and L. Sekanina, "Towards evolvable systems based on the Xilinx Zynq platform," in *2013 IEEE International Conference on Evolvable Systems (ICES)*. IEEE, apr 2013, pp. 89–95.

[140] ——, "Image filter evolution on the Xilinx Zynq Platform," in *2013 NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2013)*. IEEE, jun 2013, pp. 164–171.

[141] M. Sikulova, G. Komjathy, and L. Sekanina, "Towards compositional coevolution in evolutionary circuit design," in *2014 IEEE International Conference on Evolvable Systems*. IEEE, dec 2014, pp. 157–164.

[142] A. Stoica, "Toward evolvable hardware chips: Experiments with a programmable transistor array," in *Proceedings of the Seventh International Conference on Microelectronics for Neural, Fuzzy and Bio-Inspired Systems.* IEEE Comput. Soc, 1999, pp. 156–162.

[143] J. Walker and J. Miller, "The Automatic Acquisition, Evolution and Reuse of Modules in Cartesian Genetic Programming," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 4, pp. 397–417, aug 2008.

[144] J. A. Walker, K. Völk, S. L. Smith, and J. F. Miller, "Parallel evolution using multi-chromosome cartesian genetic programming," *Genetic Programming and Evolvable Machines*, vol. 10, no. 4, pp. 417–445, oct 2009.

[145] J. A. Walker, R. Sinnott, G. Stewart, J. A. Hilder, and A. M. Tyrrell, "Optimizing electronic standard cell libraries for variability tolerance through the nano-CMOS grid," *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, vol. 368, no. 1925, pp. 3967–81, aug 2010.

[146] J. A. Walker, J. A. Hilder, D. Reid, A. Asenov, S. Roy, C. Millar, and A. M. Tyrrell, "The evolution of standard cell libraries for future technology nodes," *Genetic Programming and Evolvable Machines*, vol. 12, no. 3, pp. 235–256, apr 2011.

[147] J. H. Holland, *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* University of Michigan Press, 1975.

[148] A. V. Delaney and H. D., "A Critique and Improvement of the "CL" Common Language Effect Size Statistics of McGraw and Wong," *Journal of Educational and Behavioral Statistics*, vol. 25, no. 2, pp. 101–132, 2000.