Multi-objective Digital VLSI Design Optimisation

Linan Cao

Doctor of Philosophy

University of York

Electronic Engineering

August 2021

Abstract

Modern VLSI design's complexity and density has been exponentially increasing over the past 50 years and recently reached a stage within its development, allowing heterogeneous, many-core systems and numerous functions to be integrated into a tiny silicon die. These advancements have revealed intrinsic physical limits of process technologies in advanced silicon technology nodes. Designers and EDA vendors have to handle these challenges which may otherwise result in inferior design quality, even failures, and lower design yields under time-to-market pressure. Multiple or many design objectives and constraints are emerging during the design process and often need to be dealt with simultaneously. Multi-objective evolutionary algorithms show flexible capabilities in maintaining multiple variable components and factors in uncertain environments. The VLSI design process involves a large number of available parameters both from designs and EDA tools. This provides many potential optimisation avenues where evolutionary algorithms can excel.

This PhD work investigates the application of evolutionary techniques for digital VLSI design optimisation. Automated multi-objective optimisation frameworks, compatible with industrial design flows and foundry technologies, are proposed to improve solution performance, expand feasible design space, and handle complex physical floorplan constraints through tuning designs at gate-level. Methodologies for enriching standard cell libraries regarding drive strength are also introduced to cooperate with multi-objective optimisation frameworks, e.g., subsequent hill-climbing, providing a richer pool of solutions optimised for different trade-offs.

The experiments of this thesis demonstrate that multi-objective evolutionary algorithms, derived from biological inspirations, can assist the digital VLSI design process, in an industrial design context, to more efficiently search for well-balanced trade-off solutions as well as optimised design space coverage. The expanded drive granularity of standard cells can push the performance of silicon technologies with offering improved solutions regarding critical objectives. The achieved optimisation results can better deliver trade-off solutions regarding power, performance and area metrics than using standard EDA tools alone. This has been not only shown for a single circuit solution but also covered the entire standard-tool-produced design space.

Table of Contents

List of Tables

List of Figures

Acknowledgements

I would like to thank my supervisors Dr. Martin Trefzer, Dr. Simon Bale, and my thesis advisor Professor Andy Tyrrell for all of their tireless guidance and assistance throughout the PhD life. I would also like to thank my parents and my girlfriend Jinhui for their love, support, understanding and patience.

Declaration

I declare that this thesis is a presentation of original work and I am the sole author. This work has not previously been presented for an award at this, or any other, University. All sources are acknowledged as References.

Publications

- L. Cao, S. J. Bale, and M. A. Trefzer, "Instrumenting parametric physical layout for multi-objective optimisation," in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2018, pp. 1339–1345
- L. Cao, S. J. Bale, and M. A. Trefzer, "Multi-objective optimisation of digital circuits based on cell mapping in an industrial eda flow," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2021, First Revision
- L. Cao, S. J. Bale, and M. A. Trefzer, "Multi-objective digital design optimisation via improved drive granularity standard cells," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2021, in Press/Accepted
- L. Cao, S. J. Bale, and M. A. Trefzer, "Multi-objective design optimisation for handling complex floorplan constraints," *Proc. Design, Automation and Test in Europe (DATE)*, 2022, Under Review

Linan Cao

August 2021

Chapter 1

Introduction

1.1 Motivation

The recent advancement of semiconductor technologies has revolutionised the use of and capabilities of smart electronic devices. State-of-the-art electronic design now allows the integration of numerous functions into a complex system on a single chip comprised of billions of transistors.

These achievements are accomplished by pushing technology to its physical limits. Transistor shrinking has succeeded with continuous improvements in the physical dimension, switching frequency and power efficiency of integrated circuits (ICs), allowing embedded electronic systems to be used in more and more real-world automated applications. However, as modern semiconductor technologies come ever closer to the atomic scale, the transistor scaling challenge and stochastic performance variations intrinsic to fabrication emerge [7] [8].

Electronic system architectures has moved to system-on-chip (SoC), many-core and heterogeneity. Such physical limits and variations further complicate the design process to ensure the quality of results (QoRs), e.g., system resilience, reliability, performance, power overheads, etc. Electronic design automation (EDA) tools handle the growing size and complexity of modern electronic designs by breaking down systems into smaller blocks or cells, introducing different levels of abstraction. In the field of digital very large scale integration (VLSI) design, comprehensive and mature industry-standard design flows are available to tape out chips. This complex process consists of several steps including logic design, logic synthesis, physical implementation (place and route) and pre-silicon physical verification [9] [10]. Each step has a dedicated EDA tool capable of coping with the design issues related to the specific task [11].

However, in this staged, hierarchical design approach, where each step is optimised independently, overheads and inefficiency can accumulate in the resulting overall design. The semiconductor devices (i.e., typically transistors) are not straightforwardly used in EDA tools for building digital VLSI circuits since significant computation efforts might

be consumed for design analysis and evaluation. Abstraction models are therefore created to speed up the design process. For instance, a standard cell library (containing building blocks, i.e., logic gates, for digital ICs) provided by a foundry has already contained two levels of abstraction, where cells are built with abstracted transistor models while each cell is also modelled and all modelled cells compose a whole library. Therefore, the abstraction error margins may be introduced in each step within the whole design flow.

In addition, the EDA design kits have not realised the automation of a full flow for building electronic designs from devices to systems. Human efforts from engineers are still required in cases where the tool cannot completely solve the problem or fail to meet design goals such as clock frequency, power consumption.

These limits prevent EDA tools from making full use of the full capability of semiconductor technologies. Furthermore, the revealed challenges result in a shift in VLSI design optimisation since it is hard to simultaneously improve all objectives to a significant degree. Instead, seeking appropriate trade-off solutions between different objectives (i.e., typically power, performance and area (PPA)) while satisfying all design constraints (i.e., most derived from physical level such as design rules, power delivery, signal integrity, etc.) is becoming the main design target for given specifications. This often costs more design resources (e.g., engineer headcounts, the number of licenses) and postpones the time to release next-generation chips for an IC company.

The modern integrated circuits design process and optimisation are still far behind biological organisms which have long since accomplished the feat of not only operating reliably with highly variable components, but also maintaining and tuning themselves in changing environments, when faults occur or they are otherwise perturbed. Biological mechanisms have co-evolved with their organisms, hence, they are perfectly adapted to the requirements of their embodiment. In this context, getting inspirations from biology with natural evolution as Nature's guiding "design and optimisation" principle

could be a promising methodology to obtain high-quality solutions regarding such design complexities and constraints.

The proposed research of this PhD project focuses on using bio-inspired techniques to create multi-objective optimisation frameworks combined with circuit design enabling a wide range of trade-off solutions for use in different case scenarios. The application of evolutionary algorithms (EAs) is helpful for optimising complex circuits and configurations, which can potentially enlarge the feasible solution space from a more global viewpoint. The proposed optimisation approaches are compatible with state-of-the-art digital EDA flows and industrial silicon technologies.

1.2 Hypotheses and Objectives

The work presented in this thesis establishes how bio-inspired optimisation techniques can improve digital circuits by augmenting industry-standard practice and design environments, and how improved trade-off solutions can be achieved in several critical design objectives at the physical implementation level.

In this thesis I propose and evaluate evolutionary computation inspired methods for digital very large scale integration (VLSI) designs based on the following hypotheses:

Hypothesis: "Combining multi-objective evolutionary algorithms with digital VLSI design processes can achieve solutions with improved performance down to physical layout level, expand feasible design space, and handle complex physical layout constraints more efficiently via refining standard cell mapping and improving standard cell granularity."

With the following supporting sub-hypotheses:

Sub-hypothesis 1.1: "Multi-objective evolutionary algorithms can optimise the drive strength mapping of logic gates in digital VLSI designs for superior performance with a wide spread of feasible trade-off solutions better than standard tools."

Sub-hypothesis 1.2: "Multi-objective evolutionary algorithms in conjunction with an industrial digital IC flow can achieve better Pareto-driven search space coverage across various circuit topologies than standard tools alone."

Sub-hypothesis 1.3: "Multi-objective evolutionary algorithms can explore a larger feasible solution (objective) space to deal with complex physical floorplan constraints more efficiently than standard tools can."

Sub-hypothesis 1.4: "Fine-grained drive strength resolution of standard cells can optimise digital VLSI designs for over-design mitigation and push performance of silicon technologies."

To verify these hypotheses, the objectives of this thesis are presented as follows:

Objective 1: Develop an automated multi-objective VLSI design optimisation framework allowing the manipulation of digital circuit building blocks down to physical layout level.

Objective 2: Demonstrate that framework capable of improving performance of VLSI designs (including complex physical corner cases) offering a range of Pareto-optimised solutions and better design space coverage over industrial-flow-generated ones.

Objective 3: Investigate the application to library level optimisation via improving drive strength granularity of standard cells for better use of foundry technology nodes and better resulting quality of VLSI design solutions.

1.3 Contributions

During the undertaking of this work, the following research contributions have been made to the field of digital VLSI design:

- The creation of an automated physical design flow for multi-objective VLSI optimisation using parameterised layout templates.
- The creation of a multi-objective (MO) EDA framework for generating trade-off solutions with improved performance through standard cell (re)mapping in an industrial flow.
- A methodology of using the standard industrial flow to generate a set of initial solutions with different topologies and different performance covering a large design space, and seeding the MOEA with them to more efficiently sample the entire feasible design space.
- The knowledge of how different physical floorplan constraints affecting the performance of VLSI designs during the standard industrial flow, and the application of the methodology of seeding the MOEA with diverse initial solutions to effectively handle the most complex constraint.
- A methodology to broaden the drive-granularity of standard cell libraries, which helps to further exploit capabilities of a foundry technology node, enhances design quality and reduces design margins.
- The application of fine-grained drive strength cells into an industrial flow to produce VLSI designs with improved PPA metrics over the original resolution library. This also has been shown to improve efficiency of the MOEDA flow in optimising circuit designs.
- The standard design tools demonstrate significant variations for producing high-quality results particularly when timing constraints become stringent; The results from this thesis show that the standard design tools can be run more efficiently with the assistance of MOEAs.

1.4 Thesis Structure

This thesis consists of eight chapters and is structured as follows:

- Chapter 2 gives a background review of the modern VLSI design process from device to physical layout before fabrication, and summarises the current design challenges and future trends.
- Chapter 3 explores the multi-objective optimisation literature to gain the understanding on how multi-objective evolutionary algorithms (MOEAs) are promising to solve the problems that need to deal with several conflicting objectives.
- Chapter 4, a feasibility investigation, introduces an automated design flow to perform multi-objective design and optimisation at physical layout level for a CMOS VLSI circuit.
- Chapter 5 presents an optimisation framework - MOEDA cooperating with commercial design tools for enhancing the quality of design solutions regarding critical metrics through refining drive strength mapping of standard cells.
- Chapter 6 illustrates a methodology to seed the MOEA with the entire design space, accessible by standard tools, across various circuit topologies for multi-objective design space exploration (MODSE) including dealing with different die shapes and pin places of physical floorplans.
- Chapter 7 proposes an interpolation methodology to expand the drive strength granularity of standard cells, and applies enriched libraries to standard digital flow for evaluation and to the proposed MOEDA flow for further trading off feasible solutions.
- Chapter 8 reviews all proposed frameworks and methodologies used in this thesis for digital VLSI design optimisation, and suggests improvements and future work that could be undertaken based on current findings.

Chapter 2

Digital Integrated Circuit Design in EDA Flows

2.1 Overview

Modern electronic design complexity has enabled high-density integrated circuits (ICs) to comprise billions of transistors at the physical layout level. Progress in semiconductor technology has made this possible through down-scaling semiconductor devices and introducing new device architectures. The technology development has resulted in a large shift for designers to advance from manual circuit design to automated specification-based design flows in a few decades. Since the first EDA tool introduced in the 1960s, the development of EDA tools has always geared towards automating the entire design process and linking each separate design step into a complete flow. Such a hierarchical design methodology, constructing designs from transistors up to large systems, heavily relies on the abstractions of every design step. The post-fabrication design effects thus need to be accurately accounted for early in the design cycle. In response to this challenge, modern EDA vendors start blurring boundaries of separate steps towards a deeper level of integration within the digital EDA flow, providing seamless transition across synthesis, place-and-route and sign-off steps. Such integration, however, is difficult since some steps need additional design freedom allowing designers to tackle independently, introducing engineering change order (ECO) efforts from engineers.

This chapter will overview the modern digital IC design process from transistor scaling to the pre-fabrication design stage and outline the current challenges of commercial EDA flows. The chapter is structured as follows: Section 2.2 introduces transistor scaling including its trends and challenges. Standard cell library design for digital IC flow using is then explored in Section 2.3. Section 2.4 introduces the industry-standard digital VLSI design flow and its challenges for modern circuit designs. The related optimisation techniques for augmenting the standard flow to tackle these challenges are discussed in Section 2.5. Section 2.6 summarises the chapter.

2.2 Transistor Evolution

2.2.1 Transistor Scaling

The technology node (also process node, process technology or simply node) refers to a specific semiconductor manufacturing process and its design rules. In the semiconductor industry, the miniaturisation of technology devices (i.e., transistors) has continuously enabled the next-generation process node, circuit and system architecture over the last few decades. The transistor scaling allows ICs to obtain greater device density following the well-known Moore's law (shown in Figure 2.1) which projects a doubling of transistors on a single chip about every two years [12]. Such scaling targets drive the industry to push the semiconductor physical limits towards many process technology innovations introducing new materials and new structures to fulfil Moore's law.

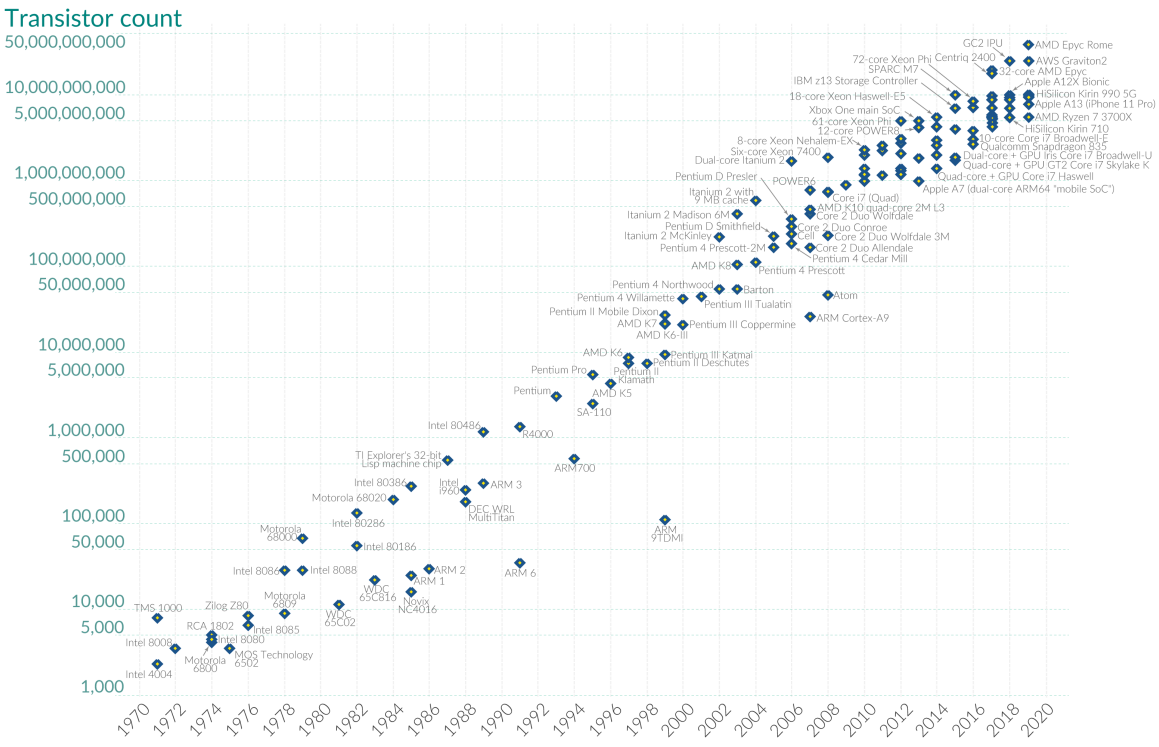
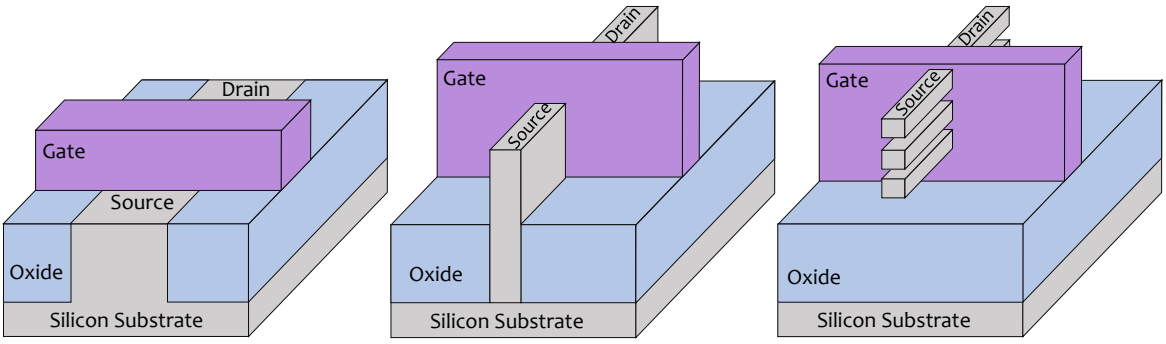


Figure 2.1 Moore's law: The number of transistors on microchips doubles every two years [1].

The metal-oxide-semiconductor field-effect transistor (MOSFET) is by far the most commercially successful semiconductor device used in process technology, which is created based on a sandwich-like metal-oxide-semiconductor (MOS) structure by superimposing several layers of conducting and insulating materials. Electric fields control the transistor operation so the devices are called field-effect transistors (FETs) [13]. The MOSFETs have been used for building logic functions in Complementary MOS (CMOS) technology. The architecture of MOSFETs is planar at its primary development stage as shown in Figure 2.2 (a), and the first MOSFET was successfully fabricated in the late 1950s.



(a) Planar FET (b) FinFET (c) GAAFET

Figure 2.2 Transistor architecture evolution from planar to GAA

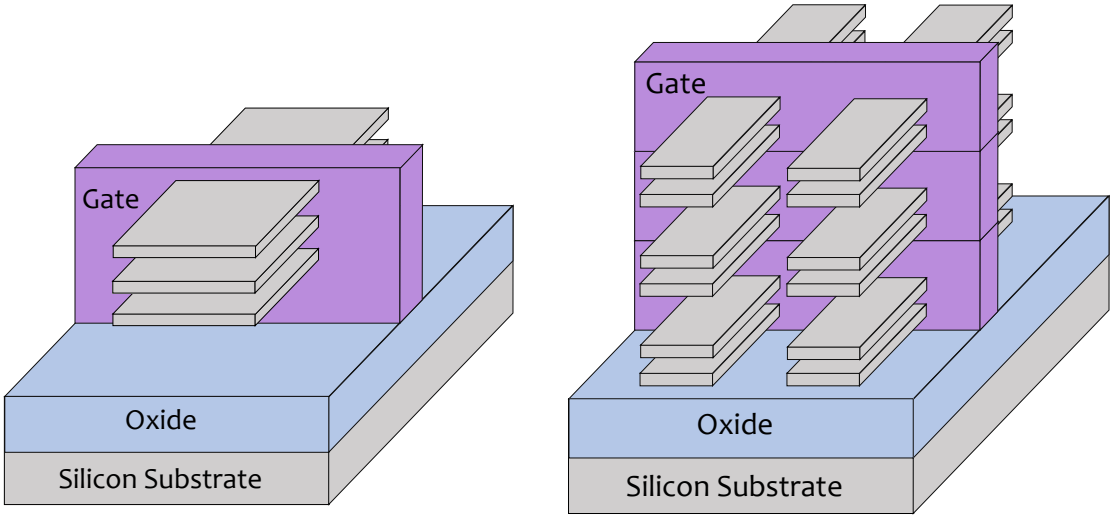
In the early 2000s, before the 130nm node, MOSFETs rewardingly employed the Dennard scaling methodology [14], in which the transistor size could be scaled by a constant while delivering consistent improvements in transistor area, performance (e.g., delay) and power reduction [15]. However, this transistor shrinking trajectory has broken down and can no longer be followed in advanced technology nodes because the power cannot be dropped without simultaneously decreasing either the transistor performance or increasing current leakage. The leakage has been particularly significant beyond 65nm node, where it poses a greater proportion of overall power consumption and causes thermal issues on the chip [16]. The architecture innovation of transistor has been shifted from the planar FET to FinFET and to the cutting-edge gate-all-around

(GAA) FET to reach the goal of increasing the control of channel for leakage reduction and operating at lower power with good performance. Figure 2.2 presents the evolution of transistor architecture from planar device to GAA.

Since the reporting of International Technology Roadmap for Semiconductors (ITRS) in 2001 revealed the promise of FinFETs (as illustrated Figure 2.2 (b)) for CMOS technology scaling limits elimination [7], FinFET technology has vested interests from the semiconductor industry. Foundries have successfully rolled out FinFETs for commercial production from the 2010s onwards, and they became mainstream devices at 14nm, 10nm and 7nm process nodes [8]. Moreover, as stated in the latest report of the Institute of Electrical and Electronics Engineers (IEEE) International Roadmap for Devices and Systems (IRDS) in 2020, the FinFET architecture remains promising for mainstream logic devices to sustain until 2025 [8]. Nowadays's state-of-the-art 5nm technology node is still using FinFET architecture. Both giant semiconductor manufacturers Samsung and Taiwan Semiconductor Manufacturing Company (TSMC) entered their volume production in 2020.

The 2020 IEEE IRDS report further projects beyond 2022, where a transition to lateral GAA devices for the next die shrink below 5nm and GAAFET will become the mainstream device after 2025, taking the place of FinFETs [8]. A lateral GAAFET architecture example, shown in Figure 2.2 (c), shows how the gate material surrounds the source to drain channel region (i.e., using the nanowire structure in this case) on all sides.

Most recently, Samsung has launched the plan to develop its own novel variant (i.e, using lateral nanosheet structure) of GAAFET, called MBCFET ^{TM} shown in Figure 2.3 (a), for 3nm process node. From a long-term perspective for the next 15 years, the projected evolution of device architectures is expected to potentially include vertically stacked fine-pitch 3D GAA devices in hybrid formed with the lateral GAAFETs, presented in Figure 2.3 (b) [8].



(a) MBCFET^{TM} (b) Vertical stacked GAAFETs

Figure 2.3 (a) A lateral GAAFET using nanosheet structure proposed by Samsung. (b) Projected GAA device architecture for 3D VLSI beyond 2030.

2.2.2 Scaling Challenge

A process technology is typically labelled with a node name indicating the device dimension. However, the industry "Node Range" labelling scheme of modern process technologies, particularly in FinFETs, starts losing their actual meaning. The node names used to represent physical features of a transistor, such as the gate length or metal half-pitch. Most recently, due to how the transistor architecture changed dramatically from how it used to be, the "Node Range" labels simply become commercial names for a generation of a certain size and its technology, and does not represent any geometry of the transistor [17].

The projected few process nodes in the 2020 IEEE IRDS roadmap are defined with labelling "3", "2.1", "1.5", "1.0 eq" and "0.7 eq" from 2022 to 2034 [8]. These numbers look like they are continuously shrinking, whereas the physical gate length of each corresponding process node is not constantly dropping down. The expected physical gate length of the "5nm" node starts with 18nm, and decreases to 12nm at "1.5nm" node and stays constant for the following nodes [8]. Figure 2.4 presents the CMOS

scaling evolution, in which it can be observed that gate length or metal pitch is no longer shrinking significantly with each process node generation [2].

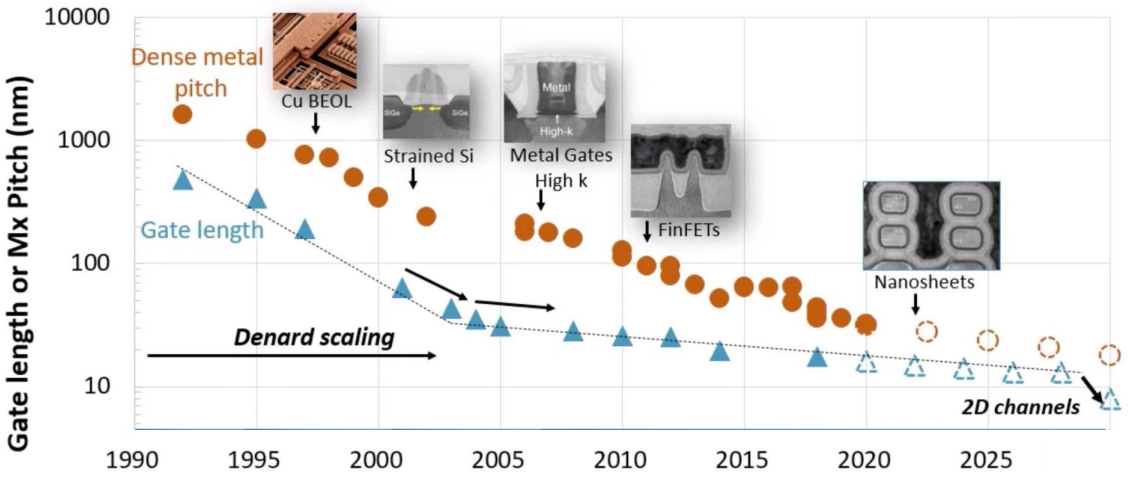


Figure 2.4 CMOS technology scaling evolution: The gate length or maximum metal pitch is hard to shrink in advanced technology nodes [2].

These demonstrate that transistor scaling becomes extremely hard due to its physical constraints. Novel 3D stacked GAA device architecture is indeed expected to feature in the roadmap of 2020 IEEE IRDS as already shown in Figure 2.3 (b), which can further increase the transistor density on the die area. This makes it possible to continuously fulfil Moore's law in future chip evolution. However, a single transistor's physical dimension (e.g., gate length, metal pitch) will not significantly change (scale-down) in the upcoming MOSFET process nodes.

In such a background, one of the potential optimisation opportunities could be how to make the most use of the full capability of current technology nodes for better design performance. The transistors are the fundamental components used to build large systems using a hierarchical design methodology. It is hard to pass transistors' realistic effects through each level of abstraction with one hundred percent accuracy. Significant overheads and inefficiencies will be accumulated in the later stages of the design flow. Enhancing design correlation between early and late design steps can mitigate the impact caused by abstraction margins or errors on the overall design results. So there

still exists a space in current process technology and EDA tools/flows for pushing modern designs to obtain improved overall performance.

2.3 Standard Cell in Digital Integrated Circuits

2.3.1 Standard Cell Library

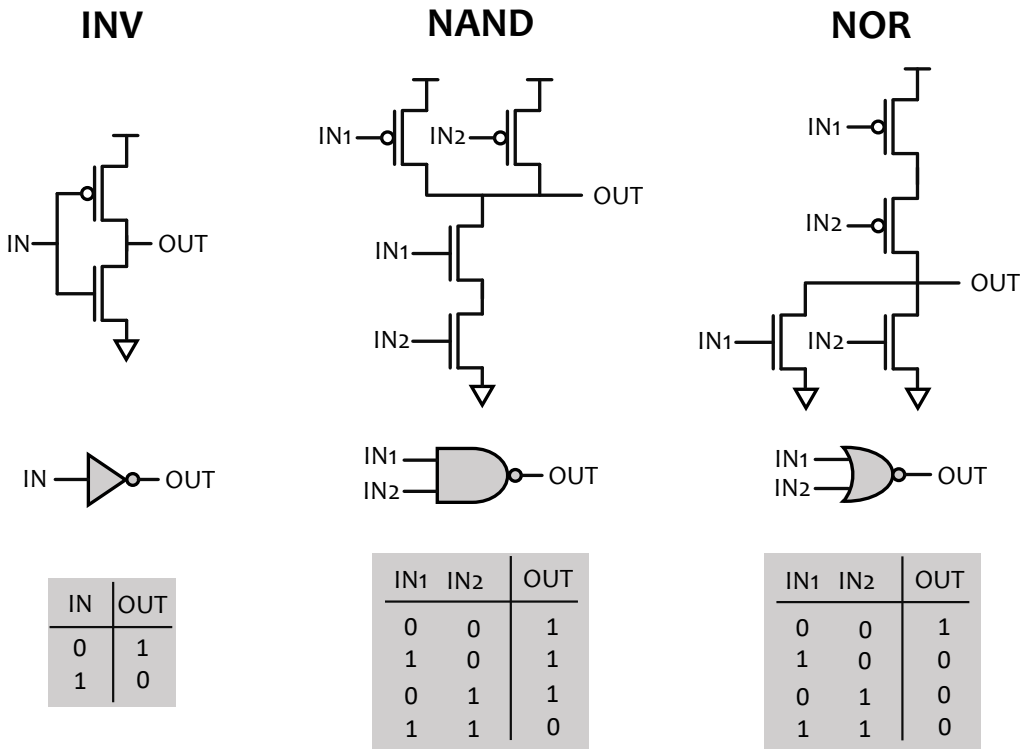


Figure 2.5 Examples of common Boolean logic cells

A standard cell is an implemented single function block through manipulating transistors and interconnects to form a complete structure. Standard cells are pre-defined for digital IC design and distributed in libraries often provided by foundries and pre-qualified for manufacturing. A standard cell library typically offers a wide range of Boolean logic (e.g., NOT, AND, OR, NAND) and storage or sequential (e.g., Flip-Flop, Latch) functions so that they can meet universal design specifications and ensure

overall functionality. Figure 2.5 presents a few basic logic cells including schematics, symbols and their behaviour.

However, circuits not only must satisfy behavioral functionality but also have to meet the constraints or requirements derived from physical level when taping out chips. Foundries therefore create each standard cell function with multiple options in drive strength, and each library provides multiple versions in routing track, threshold voltage (V_{th}) and supply voltage (Vdd).

Drive strength of a logic gate refers to its relative capability to charge or discharge the capacitance presented at its output. Large drive strength featuring bigger transistor sizes has a larger drive force to speed up a logic cell's performance (transition time) but can consume more power and die area, and vice versa. Thus, the multiple drive strength options for a single cell are used to drive different required loads of circuit paths.

For the whole library, the cell height (e.g., 9-track or 12-track) of standard cell layouts implies how many route channels can be used later at the circuit-level physical routing stage. More tracks allowing more routing space above the cells could relax routing congestion, which can reduce potential design rule violations. Higher cells also provide larger drive capabilities for better circuit performance. However, this would also consume more power and increase the die area significantly.

The threshold voltage (V_{th}) is the minimum voltage at the transistor gate (V_G) required to form an inversion layer (channel) in between source and drain so that can turn the transistor on. Different threshold voltages can be achieved via tuning manufacture parameters of a transistor such as doping concentration. Foundries usually provide standard V_{th} (SVT), high V_{th} (HVT) and low V_{th} (LVT) cell libraries aiming to effectively control the leakage power in digital ICs. Because Higher V_{th} can reduce the leakage but cells require larger transition time, and vice versa.

Multiple Vdd libraries are an important technique to typically save dynamic power of digital ICs, allowing using different power domains. Different blocks having different

supply voltages can be integrated into a single system-on-chip (SoC) chip. Thus, some blocks can use lower voltages or even be completely shut off for a specific operation mode so that power-efficient systems can be obtained. This method increases power planning complexity in terms of laying down the power rails and power grid structure. Level shift cells are necessary to interface between different blocks.

The provision of cells in the library having different layout architectures and characteristics tries to make the most of the physical features of transistors. The standard cell library is the middle abstraction layer which bridges process technology and common logic blocks. Achievable well-optimised libraries have therefore become crucial, which could determine the overall quality of results (QoRs) of VLSI designs.

2.3.2 Standard Cell Design Flow and Automation

The commercial digital IC design flow requires pre-characterised cell libraries for circuit analysis and physical implementation. Figure 2.6 demonstrates the overall standard cell design flow.

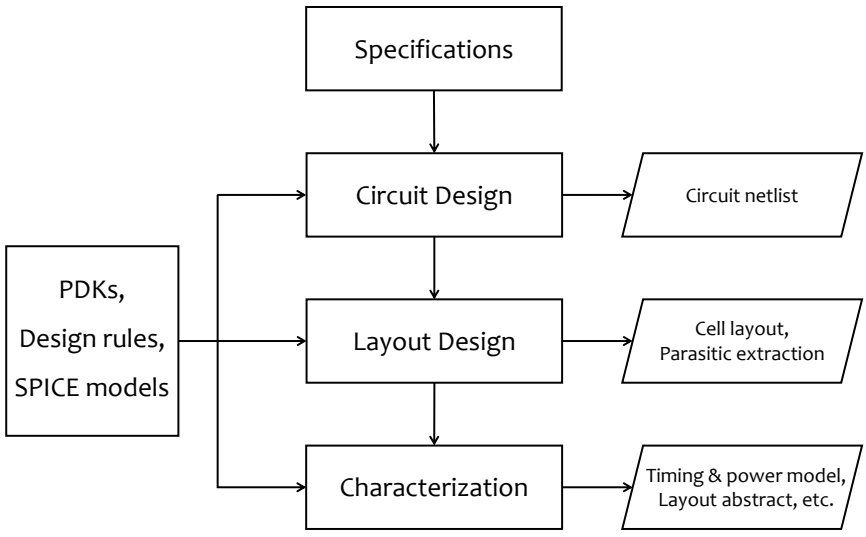


Figure 2.6 Standard Cell Design Flow

A standard cell is designed with a schematic or hardware description language (HDL) entry based on a cell specification. The transistors in physical formats, provided in process design kits (PDK), are then placed and routed in the cell layout. If all required cell layouts are created and verified, the standard cell library creation is finished. However, it will consume much evaluation time and simulation effort if straightforwardly manipulating physical layouts of cells to complete a circuit in EDA tools. This is because all pre-fabrication verification is performed based on the parasitic extracted design. So the extraction effort will be significant if all transistors of a design are processed simultaneously, and the circuit analyser needs to deal with extracted information of all elements at the same time.

The pre-processed timing and power models, typically Liberty (.lib) format, are generated for each cell through simulation based on its parasitic extraction. These characterised models can speed up the evaluation process of circuits. Thus, the full physical layouts no longer need to retain all interconnects and transistor structures, and only the top layer metal including input/output (I/O) pin positions is required for the subsequent circuit-level place and route. The abstract view (.lef) containing the geometry information (normally metal 1) of cells is produced. The process of transforming a standard cell library into pre-processed formats (i.e., timing, power models and layout abstract) is referred to as library characterisation. Once the overall design layout is complete, all standard cells used will then be replaced by the full layout ones for fabrication.

Creating standard cell libraries might take much human effort in a turnaround design cycle for producing cell layouts (i.e., transistor placement and interconnects). In the past two decades, automated layout generators of standard cells (or called transistor synthesis tool) have been investigated to accelerate this iterative process. In the early 2000s, EDA vendors started offering full standard cell design flow kits (e.g., Prolific ProGenesis^{TM}, Synopsys Cadabra^{TM} and NanGate Library Creator^{TM}) for automating optimised CMOS gate creation, including cell circuit design, physical layout and library characterisation. However, most of them are no longer active and available excepting

NanGate Library Creator™ (acquired by Silvaco in 2018). Its latest library platform cello™ [18] supports advanced process technology down to 7nm FinFET node for standard cell library creation, migration and optimisation. It now excels in technology migration and layout optimisation for further PPA gains based on legacy libraries for optimised cell variants generation.

In addition, few standard cell creators were introduced for research in the early 2000s. A home-brewed tool from IBM, called C-cell, could generate optimised cell layouts based on primitive cells and was adopted for high-performance microprocessor design [19]. A layout generation system from Kyoto University called VARDS [20] could produce a cell layout with variable drive strength. It had been successfully employed for 130nm, 180nm and 350nm library generation [21], on-demand library generation in the full digital IC flow [22] and post-layout transistor sizing for chip power reduction [23]. More recently, a dedicated layout generator for area-efficient standard cells was proposed by the same research team [24]. However, these research-purpose cell creators all need to operate based on primitive cells and symbolic layouts, which means each logic functional cell needs to create a corresponding layout template manually created by human effort.

Automating the creation of standard cell libraries from scratch is an extremely challenging task. In particular, custom specifications on cell design such as special requirements of Vdd, V_{th} (typically near-threshold operation [25–28]), or special drive strength [29], are still in a dire need of experience-based designer efforts.

2.4 Digital VLSI Design Flow

The process of designing a digital VLSI circuit is highly complex. It starts with a system specification, following a series of steps and eventually produces a packaged chip. A typical design flow is represented by the flow chart shown in Figure 2.7. The system specification defines the overall goals and high-level requirements of the system

such as functionality, performance, physical dimension and production technology [9]. A basic architecture must then be determined to meet these specifications. Example decisions include the number and types of computation cores, usage of memory, usage of intellectual property (IP) blocks, power requirements, etc [9].

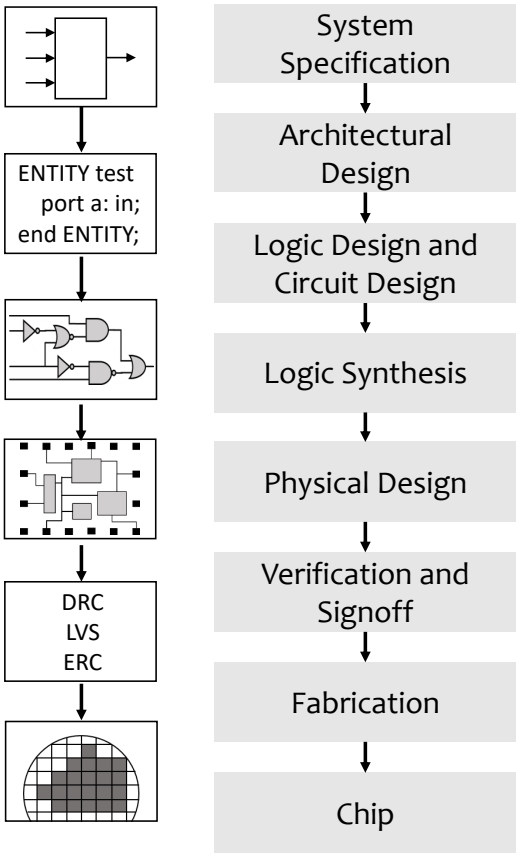


Figure 2.7 VLSI Design Flow

The emphasis of the work discussed in this thesis mainly involves logic design to physical design, which refers to a register-transfer level (RTL) to graphic design system II (GDSII) flow, also called digital flow in the EDA community.

2.4.1 Logic Design and Circuit Design

Logic design is performed at the RTL using an HDL, which defines the functional behaviour. Two common, widely used HDLs are Verilog and VHDL (i.e., VHSIC (very

high speed integrated circuit) hardware description language). All RTL modules must be simulated and verified for the use of consequent design steps.

In addition, an IC does not only include logic designs but also some critical macros like memory blocks, analogue circuits, and I/O cells, which are normally manually designed at the transistor level by engineers. These macros have to be complete before running the logic synthesis. They are also required to be characterised in advance, including timing and power models, and physical layout abstracts need to be created.

2.4.2 Logic Synthesis

Logic synthesis is a process that automatically converts HDL designs into a list of signal nets and low-level circuit elements. In general, the synthesis process, shown in Figure 2.8, has two main steps: 1) a given HDL functionality description is firstly transformed into a netlist comprised of generic logic gates (e.g., and, or, not, universal sequential elements). The modern EDA synthesizer provides few optimisation options for designers to manipulate design hierarchy and logic structure transformations in the RTL during the generic synthesis step; 2) The generic netlist is then mapped into logic gates from a given technology standard cell library. The library used in this step is pre-characterised in terms of timing, power (.lib file) and layout abstract (.lef file), as discussed in Section 2.3. The technology-specified gates that defined their drive strength, threshold voltage (corresponding to a physical view from the library) and their inter-connectivity refer to a gate-level netlist.

The synthesised design also needs to be checked whether it meets the constraints like timing, power, etc. If not, the synthesis tool will perform optimisation through remapping logic or resizing gates in an iterative loop until design metrics improved. Incremental optimisation is being operated while synthesising the design concurrently. In addition, the synthesis tool usually provides different optimisation levels (e.g., low, medium, high, ultra), but engineers have to make a choice between runtime and QoRs.

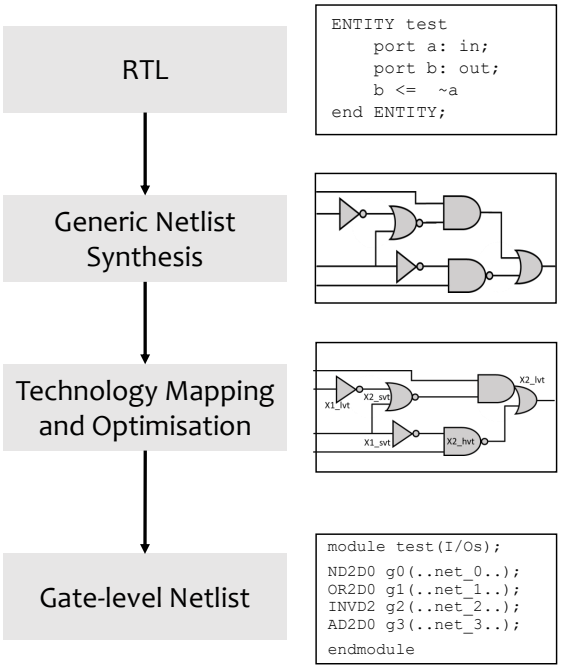


Figure 2.8 A general Synthesis Flow

However, it is not unreasonable that the obtained synthesised design failed to meet some design constraints, particularly the critical one - timing, although the "try hard" synthesis mode - ultra optimisation effort is enabled. So the failed timing paths then might be best fixed manually in the RTL design by engineers. It can cause iterations of the whole synthesis flow and exacerbates the design effort challenge.

2.4.3 Physical Design

The obtained synthesised gate-level netlist (i.e., an abstract circuit description) of a design will then be transformed into a detailed geometric representation - layout for fabrication [30]. This process refers to the physical design or physical implementation, which is a crucial step in the digital VLSI flow to make designs manufacturable. During physical layout generation, all components (macros and cells) are assigned spatial locations (placement) and have appropriate interconnections (routing) completed in multiple fabricating technology metal layers. The result of physical design, typically GDSII stream, is a set of manufacturing specifications for fabrication. Nowadays's

physical implementation tools can complete the whole process in an automated way. Figure 2.9 presents each distinct step of physical design.

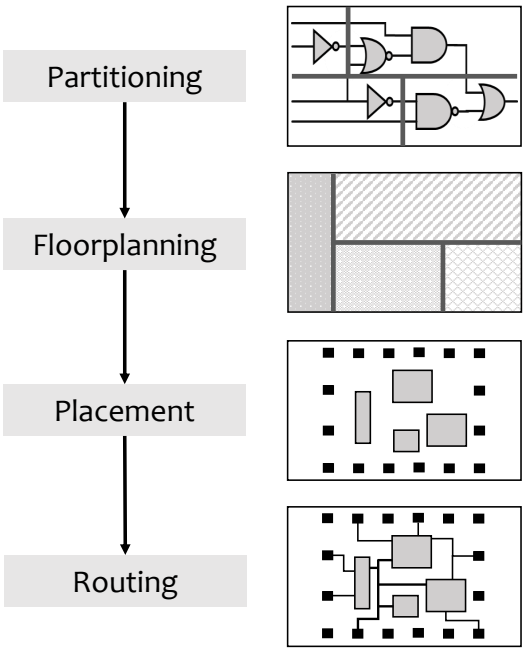


Figure 2.9 Physical Design Flow

Partitioning. The chip-level partitioning is a common strategy to lessen the complexity of physical design by dividing the whole circuit into smaller subcircuits called modules or blocks. Each block then can be designed or analysed independently. But this process needs to be operated while considering other partitions to minimise connections between subcircuits, which may otherwise cause performance degradation [30].

Floorplanning. After the circuit partitioning phase, each block has a known hard or soft shape. Hard blocks have fixed dimensions and areas, while a soft block has a fixed area but the aspect ratio can be changed. The entire arrangement of all blocks including their shapes and positions without any design rule violations (e.g., no overlap) is floorplanning. The determined topology of a circuit layout is necessary for the subsequent placement and routing steps [31]. Particularly for routing, an poor-floorplanned layout would significantly affect the routing quality (e.g., heavy

routing congestion), which could pessimistically impact the overall performance of the design.

Placement. It seeks to determine the spatial locations of standard cells or logic elements within each block on the layout die surface. All elements need to be placed on their legal sites. During this step, it also requires addressing optimisation such as minimising the distance between the cells or total wirelength of interconnections while meeting timing constraints and keeping routability. A poor placement consumes a larger die area and results in more timing violations. In addition, during the placement step, detailed locations of elements could enable more accurate estimates of circuit delay for earlier-stage timing optimisation [9].

Routing. Following the placement step, routing is to complete interconnections between standard cells or blocks. The routing process needs to specify wiring segments and topologies to connect all elements for a given placement and netlist while respecting constraints such as design rules, routing resources (cell tracks, metal layers) [9]. The simultaneous routing optimisation goal is minimising total wirelength and maximising timing slack.

Physical design directly impacts final circuit timing, area, power and reliability. Particularly meeting timing is of the most importance when completing the physical layout generation. So timing evaluation is performed at each step of physical design flow, and any timing violations must be solved before carrying on to the next step. Modern physical design EDA tools offer incremental optimisation techniques to fix these problems automatically through gate resizing (drive strength remapping), buffer insert/delete, logic refinement, instance movement, etc. These local optimisations might not be able to consider the design globally, and limited in trading off design metrics well.

The complete circuit layout must be fully verified to ensure behavioural and electrical functionality before fabrication. Few changes on layouts may be required for solving problems exposed at physical verification step. This is normally achieved through

manual engineering efforts of experienced designers, called engineering change order (ECO). It allows inserting logic directly into the gate-level netlist corresponding to a change in the RTL due to design error fixes or a change request from the customer. ECO flow is usually preferred as they save time and money in comparison to a full chip re-spin. However, the changes of a design using an ECO flow should not in a significant amount, which may otherwise require layout replacing and rerouting, often lead to worse final QoRs [32].

The full flow of RTL-to-GDSII is an iterative process in practice. Commercial tools have iterative mechanisms inside for optimisation in some dedicated steps. Although the commercial design kit is indeed powerful, significant human efforts are still involved in the design process. IC engineers normally check design quality at each step to meet all constraints. If violations can not be solved at the current step, engineers turn back to an earlier design stage for design adjustments to achieve design closure. Such a cycle is time-consuming.

In addition, with the increasing complexity of VLSI design, the modern EDA tools are required to manipulate a fast algorithm to deliver a feasible solution against the time-to-market pressure. So deterministic algorithms, which can always deliver the same solution for a particular given input, are in demand and have been developed for most sub-design steps of digital flow. These algorithms only require one execution for producing solutions but algorithm designers need to determine a mathematical function mapping the specific problem domain for computing. Such methods used might be limited to obtain a well compromised solution from a global point of view.

Therefore, to find possible optimal trade-off solutions regarding multiple design requirements using appropriate library cells while consuming less turnaround time is the challenge of design optimisation.

2.5 Modified Digital Flow

Modern chip complexity requires VLSI designers to simultaneously consider a growing list of constraints and objectives including performance, power, signal integrity, reliability and yield [10]. Design closure is a process that an IC design is modified from its initial description to meet target constraints and objectives [10]. To achieve this goal, industry engineers and academic researchers propose novel optimisation techniques to augment designs during the digital flow.

Custom design methodologies are efficient to improve the QoRs of designs achieved by experienced engineers. In the early 2000s, W. Dally and A. Chang evidenced the role of custom design in application specific integrated circuit (ASIC) chips [33]. They proposed to selectively apply a number of custom design techniques in the digital flow, including custom floor-planning, place and route critical signals to achieve the most compact layout structure. This manual design process enables reducing load on paths, better density and ultimately achieves better PPA, but custom design requires significant manual effort and is therefore not scalable to handle the complexity of large designs.

Furthermore, D. Chinnery and K. Keutzer stated that there is a gap between full-custom design and standard digital flow regarding speed and power [34] [35]. Digital ICs implemented using the standard design flow may significantly reduce design cycle time but have lost possible optimal trade-off solutions, which full-custom design can achieve. However, the current extreme design complexity, as well as the time-to-market pressure to continuously produce new generations of chips, designers in industry still focus on synthesis-centred methodology to save design efficiency and resource budgets. Therefore, implementing extra custom design and optimisation techniques as enhancements to the standard digital flow is promising to achieve higher QoRs [36].

To accelerate custom design in the digital flow, H. Onodera et al. introduced an ASIC design methodology with on-demand library generation during design loop. It can

produce cells with tailored drive strength from a set of symbolic layouts [22]. This enabled tunability of the drive strength of cells, which is in contrast to the conventionally used set of cells with fixed drive strengths. In [19] IBM also raised a similar semi-custom design flow for microprocessor design. The method continuously iterates the whole flow using pre-defined parameterized cells to recover the performance of designs through auto-generating compensated cells into a fixed cell library. In [23] a post-layout transistor down-sizing method was proposed for power reduction while preventing interconnect modifications, so that straightforwardly save the design turnaround time.

Most recently, EDA vendors offer latest digital full flow solution, such as Cadence iSpatial^{TM}, Synopsys Fusion Compiler^{TM}, to unify the power of logic synthesis and physical implementation tools. The key enhancement of novel design methodologies is migrating a part of physical implementation functions to logic synthesis for early-stage accurate evaluation to reduce design margins, so as to enable faster throughput time and improved PPA metrics.

Industry-standard IC design flow, to a large extent, is a closed-source design process. Limited engineer change order (ECO) opportunities can only be executed by the designer within the flow's constraints at place and route stage, which is at a late stage in the flow making efficient optimisation impossible. To address this issue, a number of academic open source tools in logic synthesis, physical implementation and verification have been developed [37]. The OpenROAD project, led by University of California San Diego, seeks to develop an open source RTL-to-GDSII EDA flow through integrating to achieve 24-hour, No-Human-In-The-Loop layout design with no PPA loss [38, 39]. This opens up new possibilities to modify or hook into the design flow at a more detailed level, enabling the application of popular techniques, including artificial intelligence (AI) and machine learning (ML).

2.6 Summary

This chapter provides an overview of digital integrated circuit design in EDA flows from transistor scaling to physical layout implementation. Some current challenges in the modern digital IC design are explored both in terms of EDA tools and designers.

Process technology scaling trend is moving towards novel transistor structure (3D stacked) investigation instead of further shrinking the absolute physical size of transistors, because significant variability has been introduced in small-scale transistor so simultaneous improvements on power and performance are almost impossible to achieve. This implies that power, performance and area gains for overall electronic system optimisation no longer heavily depends on transistor scaling.

Each design step of digital flow introduces its own level of abstraction, so any margin or error will accumulate and propagate. Hence, achieving a good solution in each step is crucial for the success of subsequent design steps and the quality of the overall solution. Increasing the correlations between front-end and back-end during the IC design cycle is vital to reduce margins across different levels of abstraction.

Thus in today's silicon-based IC industry, designing and optimising an electronic system requires achieving design closures in regard to multiple constraints consuming less throughput time. The potential avenues to make this possible are: 1) to make both full use of current technology nodes and EDA design flows; 2) to trade off multiple objectives and constraints, which become more challenging in modern designs. So there is a demand to provide designers with choices and allow to select designs with the most appropriate trade-off solutions for different application cases.

The next chapter will give a literature background of multi-objective problems and commonly-used methodologies, as well as multi-objective optimisation techniques applied in the VLSI design processes.

Chapter 3

Multi-objective Optimisation

3.1 Overview

The VLSI design process is often involved with multiple conflicting objectives and constraints as discussed in Chapter 2. Investigating possible optimum trade-offs of a design to satisfy all objectives and constraints is the ultimate goal that needs to be achieved. This process heavily relies on the experience of designers, and they are required to be familiar with the problem domain and design challenges. Therefore, VLSI design can be considered as a multi-objective problem (MOP).

This chapter provides an introduction of the multi-objective problem at first in Section 3.2. Decomposition-based method to solve MOPs is then discussed including its limitations in Section 3.3. The evolutionary multi-objective optimisation techniques are explored in Section 3.4. Section 3.5 introduces the evolutionary technique's application in VLSI design including its limitations, challenges and trends. Section 3.6 summarises the chapter.

3.2 Multi-objective Problem

Most real-world engineering optimisation problems are inherently multi-objective. Finding solutions has always been a challenge for researchers and engineers since they must simultaneously satisfy several objectives. A general multi-objective problem [40] with n decision variable vectors and m objectives can be defined as:

$$\mathbf{y} = \mathbf{f}(\mathbf{x}) = \quad [f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_m(\mathbf{x})]^T \tag{3.1}$$

where $\mathbf{x} = [x_1, x_2, ..., x_n]^T \in \mathbf{X} \subset \mathbf{R}^n$ is an n-dimensional decision vector and $\mathbf{X}$ is the n-dimensional design space. The $\mathbf{y} = [y_1, y_2, ..., y_m]^T \in \mathbf{Y} \subset \mathbf{R}^m$ is an m-dimensional objective vector and $\mathbf{Y}$ is the m-dimensional objective space. The $\mathbf{x}$ defines m functions mapping $\mathbf{X}$ to $\mathbf{Y}$.

In most optimisation problems, there are restrictions imposed by particular characteristics of environment resources available (e.g., physical limitations, time requirements, etc.). All these restrictions, called constraints in general, must be complied with in order to deliver a certain solution that is both feasible and acceptable. So the constraints [41] can be expressed in the form of inequalities:

$$g_i(\mathbf{x}) \leq 0 \quad i = 1, ..., p \tag{3.2}$$

or equalities:

$$h_j(\mathbf{x}) = 0 \quad j = 1, ..., q \tag{3.3}$$

Thus a MOP consists of m objectives, $p + q$ constraints on the objective functions and its optimisation is performed on n decision variables.

Such optimisation problems, often constrained, are not unreasonable in VLSI design flows. Power, performance and area are fundamental objectives for a chip design. Thus digital IC design optimisation (in terms of speed) normally aims to reduce the cost of power and area without performance degradation. In addition to the physical design, each step needs to complete the corresponding task while considering multiple specific constraints, such as timing slack, design rules, power delivery, signal integrity, etc., to ultimately ensure design reliability.

3.2.1 Pareto Optimality

The optimisation goal changes in MOPs since several objective functions and constraints exist. It is rarely the case that there is a single solution point which simultaneously optimises all the objective functions to a significant degree. Therefore, the optimisation process is aimed at finding a potential set of compromises (or trade-offs) rather than a single solution instead.

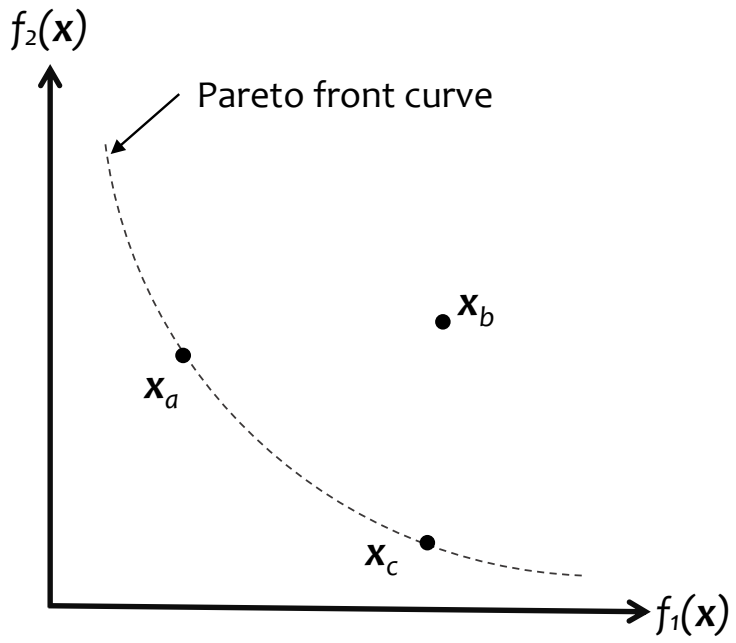


Figure 3.1 Pareto optimality: Solution $\mathbf{x_a}$ and $\mathbf{x_c}$ exist on Pareto front and both dominate solution $\mathbf{x_b}$.

The most commonly adopted notion of optimality is called *Pareto optimum* or *Pareto optimal* [42]. In a multi-objective *minimisation* problem, one decision variable vector $\mathbf{x_a} \in \mathbf{X}$ is said to *dominate* another $\mathbf{x_b} \in \mathbf{X}$ if for all i

$$f_i(\mathbf{x_a}) \leq f_i(\mathbf{x_b}) \quad i = 1, ..., m. \tag{3.4}$$

In addition, if and only if there does not exist a $\mathbf{x}' \in \mathbf{X}$ that dominates $\mathbf{x_a}$, the solution $\mathbf{x_a}$ is said to be non-dominated or Pareto optimal. All Pareto optimal solutions exist on the Pareto front when $m = 2$ or the Pareto surface when $m > 2$. However, if vectors $\mathbf{x_a}$ and $\mathbf{x_c}$ both exist on Pareto front, the two vectors are not comparable. Figure 3.1 illustrates these concepts.

In other words, the Pareto optimality finds a possible solution that one objective better off without making others worst off. In many real-world problems, the Pareto front is

hard (probably impossible) to achieve as problem solvers need to guarantee that no solution behind it can exist.

3.3 Decomposition of Multi-objective Problems

To straightforwardly solve a problem involving multiple objectives and constraints is quite complex. Decomposing the complexity of a MOP is a commonly adopted strategy to tackle this kind of problems. The primal problem is often facilitated into a problem abstraction resulting in a reduced search complexity. Using such a method normally employs a scalarising function to aggregate all the objectives into one scalar objective function. The scalarised function consists of a set of single objective sub-problems that correspond to the objectives of the primal problem. The optimal solutions to the one scalar objective optimisation problem are the Pareto optimal solutions to the multi-objective optimisation problem. A set of parameters (i.e., scalarising coefficients) for the scalarisation can be used to search for Pareto optimal solutions and different parameter combinations can produce different solutions. Thus the definition of scalarising coefficients is crucial for the optimisation process and it is normally determined by decision makers according to specific problems.

For example, the weighted sum method (or linear scalarisation) is a frequently used scalarising approach due to its lower computation efforts and high search efficiency. As a common concept in multi-objective optimisation, the weighted sum method has been discussed prominently [43] [44] since it was introduced by Zadeh [45]. The method linearly aggregates all the single objective functions into one objective function by using a weight vector $\mathbf{w}$. So the decomposed MOP can be expressed as:

$$\mathbf{y} = \sum_{i=1}^{m} \mathbf{w}_i f_i(\mathbf{x}), \quad \text{s.t. } \mathbf{x} \in \mathbf{X} \tag{3.5}$$

It is a feasible methodology that can improve the optimisation efficiency with obtaining compromised solutions. However, in the weighted sum method, aggregating all objective functions needs to deal with weighting coefficients adjustment. One obvious problem that may be hard is to precisely and accurately select a set of weights to scale all objectives fairly [46]. Defining appropriate scalarising weights requires practitioners to be familiar with the specific problem domain. This method also has problems with selection. For instance, two solutions may have the same overall objective result, but each single objective function may contribute completely different values, introducing ambiguity in the selection process. Furthermore, the weighted sum method increases the difficulty of producing the entire Pareto optimal sets because it may be difficult or even impossible to obtain trade-offs concerning all proposed objectives only according to overall objective function values.

This drawback of the linear scalarisation has been theoretically evidenced [47] [48]; it cannot find the Pareto front on non-convex regions [49] [50]. However, many modern complex systems (e.g., deep neural networks, the performance of semiconductor devices, etc.) imply non-convexities that contain local minima during optimisation. In respect of a convex problem, every local minimum is a global minimum. Figure 3.2 basically illustrates convex and non-convex problems.

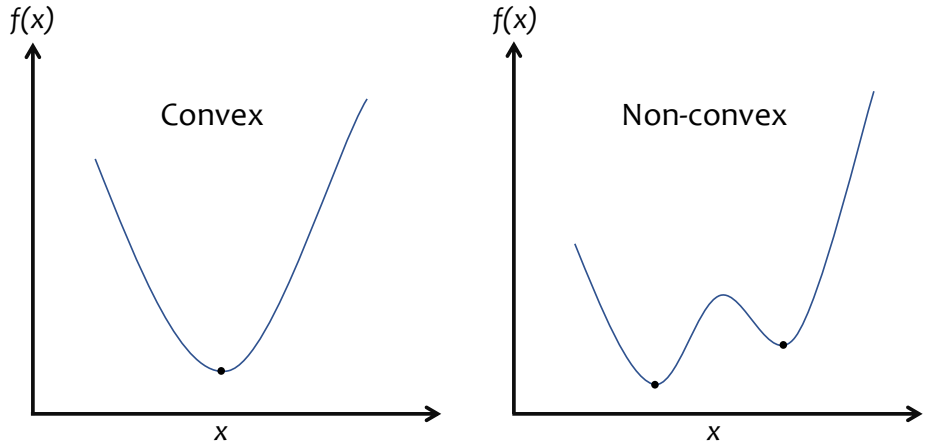


Figure 3.2 Examples of convex and non-convex problems

If the problem is identified as convex, the scalarised MOP could be mapped into the convex problem class, such as linear programming, geometric programming, etc., which have been widely used in electronic circuit modelling and design optimisation [51] [52] [53] [54]. Most of these techniques are inherently adapted to handle continuous version problems.

However, the FinFET and the recent GAAFET technologies introduce additional discreteness in transistor sizing. The increased complexity in modern semiconductor devices causes non-convex delay functions, which lead to non-convexities in device simulation results and non-linear delay model (NLDM) tables. In addition, capacitance and slew constraints will further complicate the problem [55].

Kashfi in [56] empirically studied the modelling of power and delay for VLSI circuits through comparing convex and non-convex analytical models in the multi-objective optimisation process. This work concluded that the non-convex circuit model has much more chance of finding the global optimum, which also demonstrates the non-convexity of realistic circuit behaviour in power and delay. In contrast, although the convex model could guarantee the optimum, it is possibly local and may have more modelling errors. The weighted sum method also has been evidenced that it is less effective for solving MOPs in non-convex functions [56].

Therefore, it is reasonable to consider modern VLSI design, particularly at the physical level, as a non-convex problem to hold optimisation results close to design practice behaviours.

3.4 Evolutionary Multi-objective Optimisation

Solving MOPs using evolutionary algorithms (EAs) is an alternative approach that can provide global solutions for large complex problem space comprised of many potential local minima.

Evolutionary algorithms are a class of population-based stochastic optimisation methodology inspired by Darwin's evolutionary theory [57]. The optimisation, through genetics, recombination and selection, allows populations to improve with evolutionary cycles. Although many variants of EAs have been proposed over time, the general underlying principle remains the same. That is, a population of individuals encoding the problem evolves naturally over generations to result in better-adapted solutions eventually.

3.4.1 Operation of a Basic Evolutionary Algorithm

In detail, a *population* normally required in an EA contains a number of *individuals* which are the candidate solutions of a problem. The natural environment is represented as a cost function, called *fitness*, which allows evaluation and assignment of a fitness score to each individual. Under the pressure of the environment, individuals are reproduced through variation operations and involved in an iterative evolutionary loop with a number of *generations*. At the end of each generation, better fitness individuals survive the natural selection process for the subsequent generation until the termination of the evolution process [58]. All poor fitness individuals will die out. Figure 3.3 illustrates a generic flow of EAs.

Representation. Implementing an EA to solve a real-world problem requires a representation mapping a genetic encoding to the problem. This is a preparation step before routinely executing the EA. The representation usually refers to a specific data structure (the genetic encoding) that the EA manipulates during optimisation. It describes the problem in a set of necessary parameters or variables. This is called a set of *genes* or a *chromosome* representing an individual. The genes normally need to be defined with a feasible range according to the specific problem. A legal combination of specified genes (chromosome) can represent an individual or a solution to a problem.

Initialisation. This establishes an initial population which can be either initialised randomly or seeded with a set of specific configurations. The *population size*, i.e., the number of individuals of the population, is variable and needs to be defined.

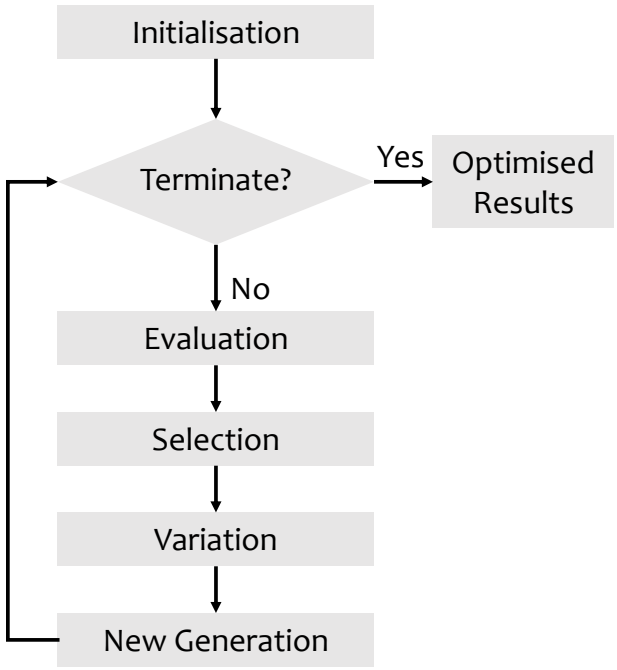


Figure 3.3 A generic flow of EAs

Evaluation. Once genes have been specified and mapped to the chromosomes of individuals can be inserted into an environment for fitness measuring (here, actual hardware or a realistic simulation in the electronic design). So the evaluation is an assessment scheme to assign performance results to individuals. A fitness function, or cost function, needs to be defined in regard to problem objectives to calculate the numerical fitness score of each individual.

Selection. Following evaluation, the fitness scores are used during the selection process to determine which individuals should survive to form the population for the next generation. The goal of selection is to promote the individuals which receive high rewards from the fitness function and discard others.

However, one issue with selection is that, when an individual is found which has an advantage over the other individuals, it can often be lost in the next generation since its advantage may be removed by genetic variations [59]. So elitism strategy is commonly

used because it can ensure elitist individuals (i.e., having best fitness) to be preserved unchanged and carried through to the next generation [60].

Variation. *Crossover* and *mutation* are often used variation operators. Since the variations are performed on the chromosome of an individual, they are also called genetic operators.

The crossover operator combines subsets of the chromosomes of usually two individuals and mixes them to form two new chromosomes representing two new individuals. Figure 3.4 shows an example of two offspring individuals that are produced through separating and recombining the paternal chromosomes. Not all offspring individuals will be reproduced through the crossover operation, and some of them will be copied to the next generation without modifications. To do this, the *crossover rate* (probability) refers to determine the number of times a crossover occurs for individuals in one generation, which is the probability that two chromosomes exchange some of their parts [61]. The 100% crossover rate means that all offspring individuals are produced through crossover. If the crossover rate is 0%, then the offspring is exactly copied from the old population.

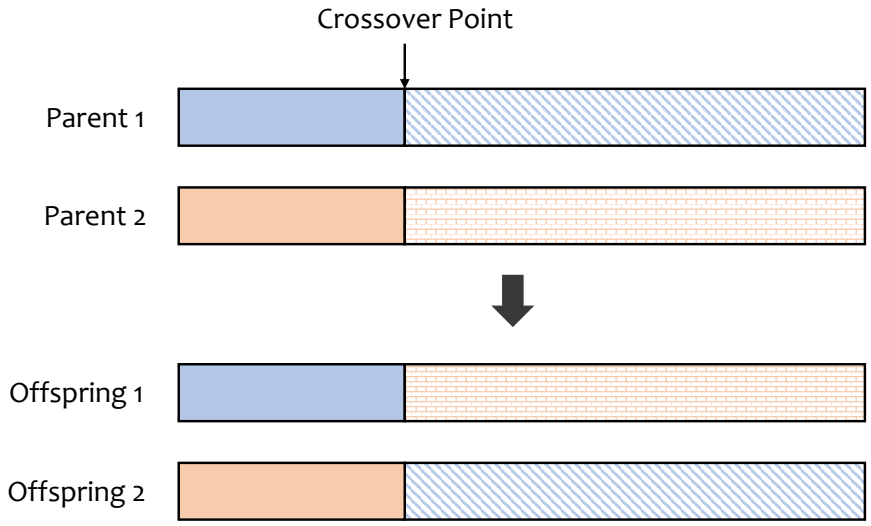


Figure 3.4 Example of a crossover operation, here single point crossover.

Mutation normally takes place after crossover is completed. The operator randomly applies modifications to one or more genes of a chromosome or an individual. In electronic design, the mutation is usually restricted with a modification range that covers the valid parameter (ranges for the genes) to make the design realistic and feasible. For example, shown in Figure 3.5, in a binary-coded chromosome, one or more bits randomly chosen can be switched from '1' to '0' or from '0' to '1'. The number of genes to be altered in a chromosome of an individual refers to the *mutation rate*.

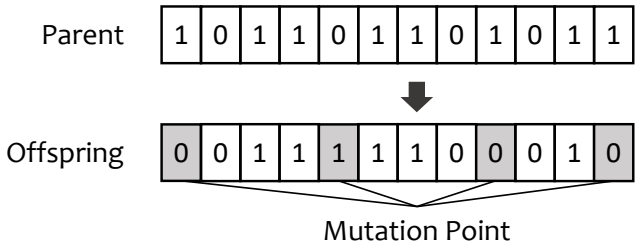


Figure 3.5 Example of mutation operation, here random bit-flip.

In the EA community, it is a common view that the crossover is the primary search mechanism and that mutation is a secondary operator. In recent years, it has been found that mutation is a much more important operator specifically in the context of evolutionary strategies, and some of EAs developed only exclusively use mutation [59]. Crossover operation is not often used in the electronic design (typically in hardware) due to its implementation difficulty. Since circuit topologies in electronic hardware are varying during the design process, the overall design functionality will be modified if performs crossover operator between different circuit topologies. Relying instead on mutation is more commonly adopted in electronic hardware design [59].

Termination. Termination of the evolution process is triggered when specific criteria are met, such as:

1) The quality of the solution is sufficient for use.

2) The maximum number of generations set by the decision-maker is reached.

3) The performance of EA produced solutions is stagnating during the evolution process.

3.4.2 Multi-objective Evolutionary Algorithm

Evolutionary techniques have been widely used in solving MOPs. Before the emergence of multi-objective evolutionary algorithms (MOEAs), it is common to decompose the complexity of MOPs using scalarising methods in the EA community. The single-objective evolutionary algorithm can then apply to solve the aggregated multi-objective problem. This approach was very popular and adopted in the early stage of MOEA development history [62], but its incapability of dealing with non-convex Pareto fronts was soon reveled [63].

To cope with the real-world scientific and engineering MOPs that are often irregular due to their high-dimensionality, discontinuance, and multi-modality [40], the development of MOEAs derived from basic EAs provides an alternative approach to solving these irregular problems. The ultimate optimisation goal of MOPs is to obtain a set of Pareto-optimal solutions. Although a few compromised feasible solutions that the scalarisation methodology can achieve, MOEAs instead explore the whole fitness landscape approaching the entire set of the globally optimal solutions [64].

The first MOEA, called vector evaluated genetic algorithm (VEGA), is proposed by David Schaffer [65]. This area has attracted a lot of interests from researchers around the world since then. Multi-objective genetic algorithm (MOGA) [66] and Non-dominated sorting genetic algorithm (NSGA) [67] were introduced and drawn much attention in the EA community. Both dealing with selection problem are based on the Pareto optimality ranking mechanism, but the diversity of Pareto fronts is hard to maintain in these algorithms. This is because the Pareto optimal set may eventually converge to a single solution [62]. To preserve diversity, the strength Pareto evolutionary algorithm (SPEA) [68] was proposed using the idea of elitism to retain all the best solutions. Thus its produced Pareto front diversity heavily relies on the large

archived non-dominated solution set, which would slow down the search. A revised version of SPEA called SPEA2 [69] overcame the drawback of its predecessor through introducing a nearest neighbour density estimation and an archive truncation method.

The non-dominated sorting genetic algorithm-II (NSGA-II) [70] is the most popular MOEA that has been widely applied to solving many real-world MOPs, and it is still widely used today by researchers. It features a fast *non-dominated sorting* approach and a *diversity preservation* mechanism that does not require any user-defined parameters. These provide selection with elitism as well as a good spread of solutions.

Figure 3.6 illustrates the principle of both non-dominated sorting and diversity preservation running in an example population.

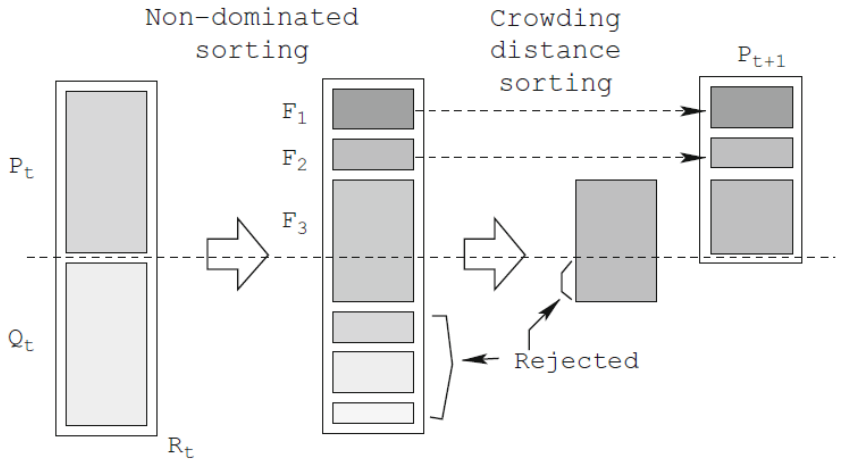


Figure 3.6 The overall NGSA-II procedure including non-dominated sorting and diversity preservation mechanism in a population evolution.

Non-dominated sorting. As stated earlier in this chapter, if one individual p performs better than another q in at least one objective while not performing worse in any other objectives, then p is said to dominate q. In non-dominated sorting, each individual (e.g., p) has two entities: the first is domination count, the number of solutions that dominate p; the second is a set of solutions that p dominates. The individuals are grouped based on their domination count into multiple fronts $\mathbf{F} = (\mathbf{F}_1, ..., \mathbf{F}_i)$. The non-dominated individuals which have the lowest domination counts (i.e., zero) form

Algorithm 1 NSGA-II

Procedure: NSGA-II (N, M, $f_m(\mathbf{x}_n)$). ▷ N individuals evolved M generations to solve $\mathbf{f}_m(\mathbf{x}_n)$.

```
Initialize random parent population P_t in size N
Evaluate objective values
Non-Dominated-Sorting (P_t)
Generate offspring population
    Binary-Tournament-Selection (P_t)
    Q_t ← Crossover (P_t)
    Q_t ← Mutation (Q_t)
for t ← 1 to M do
  for each population R_t ← P_t ∪ Q_t in size 2N do
    F ← Non-Dominated-Sorting(R_t) ▷ Assign Rank (level) based on Pareto dominance.
    P_{t+1} ← ∅
    i ← 1
    while |P_{t+1}| + |F_i| ≤ N do
      Crowding-Distance-Assignment(F_i)
      P_{t+1} ← P_{t+1} ∪ F_i
      i ← i + 1
    end while
    F_i ← Descend-Sort(F_i)
    P_{t+1} ← P_{t+1} ∪ F_i[1 : (N − |P_{t+1}|)] ▷ Less crowded individuals from the first to the (N − |P_{t+1}|)th of F_i to fill P_{t+1}.
    Create next generation
        Binary-Tournament-Selection (P_{t+1})
        Q_{t+1} ← Crossover (P_{t+1})
        Q_{t+1} ← Mutation (Q_{t+1})
  end for
end for
```

the first front $\mathbf{F}_1$. The individuals which have the second lowest domination counts form the second front $\mathbf{F}_2$ and this will continue to the third and following fronts until all individuals are assigned.

Diversity Preservation. This *crowding distance sorting* algorithm estimates the solution density in the vicinity of each individual based on the Euclidean distance to their nearest neighbours [57]. It mainly has two steps: the first is to calculate the distance of each individual to others, and assign the value to each individual; the second is to descendingly re-sort the $\mathbf{F}$ according to their distance values. So that if

two individuals belong to the same non-dominated front, the one that resides in the less crowded region is preferred.

Algorithm 1 illustrates the NSGA-II algorithm in detailed. Since the thesis work investigates whether MOEAs can enhance the digital VLSI design processes instead of finding the best EA for VLSI design, NSGA-II, a population-based algorithm, will be selected as the main searching and optimisation technique.

All EA parameters set (population size, generation count, mutation rate) in this work are widely-used in MOEA literature [59].

3.5 MOEA Application in VLSI Design Flow

Optimisation methods for VLSI computer-aided design (CAD) cooperating with evolutionary algorithms firstly appeared in the late 1980s [71] [72]. The EA-based VLSI CAD became an active and popular research area since then, which particularly attracted much attention in the 1990s and 2000s.

In the VLSI design flow, a number of EA-based techniques made impressive contributions to logic synthesis [73–76], partitioning [77–79], floorplanning [80–83], placement (standard cell placement [84, 85], macro cell placement [86, 87]) and signal routing [88, 89], which were even cover the whole VLSI design flow. Most of these works were presented around 20 years ago. The complexity of VLSI design at that time (e.g., around 100 million transistors on a chip) was not as high as today's chips (i.e., normally dozens of billion transistors), which also has not been further complicated with too many modern design constraints. The EA-based techniques had achieved competitive high-quality VLSI circuits than the previously published excel works at that moment.

However, the EA-based VLSI design corresponding to each sub-step started to reveal its drawbacks compared to the modern EDA tools that can deliver a reasonably good and deterministic solution as speedy as tools can. The weak capabilities are that

(1) Since EAs are inherently stochastic search methods, randomness might influence the solution quality. In other words, the solutions produced by EAs for multiple executions are often different to each other. (2) The runtime of EAs are not competitive comparing to the deterministic algorithms.

Most EAs' applications in the VLSI start shifting to perform multi-objective design optimisation and design space exploration, which are often out the scope of typical RTL to GDSII design flow. At the library level, the work in [90] introduced a library-reduction strategy based on an EA only using mutation, which allows saving logic synthesis efforts while keep improving PPA of designs. At system or architecture level, many publications are looking at evolutionary design space optimisation. The authors in [91] introduced a GA-based approach to automatically tune the parameters of hardware intellectual property (IP) generator, which successfully optimised a network-on-chip (NoC) router and a fast Fourier transform for higher quality results (PPA) at a lower computational cost. Both [92] and [93] proposed the use of a multi-objective evolutionary approach to search for Pareto optimal configurations of a highly-parameterized system-on-chip (SoC) architecture model, where computation, communication and memory elements can be changed based on parameter values. This method has both improved quality of results (QoRs) (i.e., power/speed trade-off) and exploration efficiency. Erbas in [94] compared a mathematical model (i.e., weighted Chebyshev method) with two typical MOEAs: SPEA2 and NSGA-II to optimise the processor network mapping onto multiprocessor SoC architectures. Dey in [95] applied multi-objective design space exploration for power grid networks of SoC using NSGA-II.

For VLSI design reliability, the authors in [96] applied NSGA-II to optimise the soft error (transient faults) through refining gate drive strengths while compromising area and delay of the circuits. Instead of relying on a general EA, a cooperative co-evolutionary algorithm (CCEA) (i.e., decompose n-D decision vector into n 1-D subcomponents) is proposed in [97] and applied in the tolerance to transient faults in multiprocessor systems for better search efficiency.

Moreover, MOEAs were used to augment high-level synthesis (HLS) efficiency and outcomes. In [98] a GA was proposed to combine two sub-tasks running concurrently during HLS of datapaths. It showed that the productivity of datapath generation was enhanced significantly while considering area constraints. Ferrandi in [99] presented a multi-objective genetic algorithm optimisation framework in HLS for area and latency optimisation. Evaluation of candidate solutions was performed using a fast estimation model, which saved evaluation time and ultimately improved the design exploration efficiency.

Previous research confirms that MOEA based approaches could achieve superior QoRs in completing the digital flow from logic synthesis to physical design. Due to revealed limitations (e.g., runtime, randomness) of MOEAs, many MOEA applications shift to system-level or even high-level synthesis. Early exploration of the design space plays a crucial role as it allows evaluating alternative solutions without the burden of low-level primitives [94, 100]. The existing research demonstrates that MOEAs are well-suited for multi-objective design space exploration.

However, an MOEA developed for VLSI design might not create many interests within the VLSI or EDA community unless its performance and robustness are competitive in real-world large VLSI design problems. In addition to the academic results from evolutionary-based design, they should not be separated from comparing the algorithms with commercial EDA tools.

Researchers in [72] provide guidelines for developing MOEAs for VLSI design.

(1) It is of paramount importance to investigate the randomness of the solution quality given the probabilistic nature of MOEAs. Designers will hesitate to use an algorithm or method that requires running several times.

(2) The results produced from MOEAs neglect the practicability that solutions need to be implemented and tested in the context of realistic constraints.

(3) Both runtime and quality have to be taken into account when using MOEAs.

3.6 Summary

This chapter provides an overview of multi-objective problems, and two general methodologies (i.e., decomposition-based and MOEA-based), as well as MOEA applications in VLSI design, are discussed.

It is highlighted that MOEAs are able to solve MOPs, particularly for discrete, nonlinear, non-convex problems. Although MOEAs have not been widely commercialised in standard EDA tools/flows, some related global stochastic optimisation techniques (simulated annealing is typical) have been used in state-of-the-art EDA tools/flows. The critical issues of developing EAs for VLSI design are high-computing efforts and randomness of results. So EDA vendors prefer to use deterministic algorithms for completing VLSI designs, in which all decisions made by the algorithm are repeatable (i.e., not random) [9].

However, the ever-growing complexity, also showing discontinuities, non-linearities and non-convexities, of modern VLSI design make feasible solutions hard to be achieved by EDA flows in a single execution cycle. Many cases need human efforts from engineers to resolve the problems, and some might be beyond the experience of a specific designer. Moreover, as explored in Chapter 2, modern hierarchical design flows might bring significant overheads and pessimistic overall results (unable to reach the possible optimum) due to the potential margins from multiple levels of abstraction.

In such a situation, it is a promising opportunity to reconsider applying MOEAs across different abstraction levels as the global optimiser to assist the design process of VLSI, particularly in balancing multiple objectives and constraints. The next chapter will present an initial work that combines MOEAs and VLSI physical layout design to perform multi-objective optimisation.

Chapter 4

Multi-objective Circuit Optimisation using Layout Templates

4.1 Overview

This chapter presents an automated physical design approach to perform multi-objective optimisation on a CMOS VLSI design. An MOEA is applied to optimise the circuit by performing adjustments on a parameterised layout template. This aims to produce a set of layout instances representing optimised trade-off solutions regarding propagation delay, energy consumption and layout area.

In order to accurately characterise the performance of a circuit close to a real fabrication process, it is necessary to consider the design down to the physical level. Therefore, in this initial work, the focus is on designing and optimising circuits based on their layouts, enabling high-performance solutions both realistic and feasible. The specific goal of this chapter is to provide a feasibility study of using parameterised layout approach in the context of evolutionary multi-objective optimisation, and results for a simple test case are shown.

The remaining parts of the chapter are as follows: Section 4.2 provides an overview of the proposed automated physical design flow. Section 4.3 gives a detailed description of the parametric layout. Thereafter, Section 4.4 introduces the multi-objective circuit optimisation setup on a test case. Experimental results for a 1-bit full adder example circuit are reported and discussed in Section 4.5. Section 4.6 outlines the summary of this chapter.

4.2 Automated Multi-Objective Design Flow

This chapter delivers an initial feasibility study showing that MOEAs can help VLSI design by straightforwardly modifying physical layouts while simultaneously improving multiple objectives. The proposed automated design flow covers circuit specifications, circuit layouts, and circuit evaluations. Figure 4.1 presents the design flow for the parametric layout engine. It consists of four steps as follows:

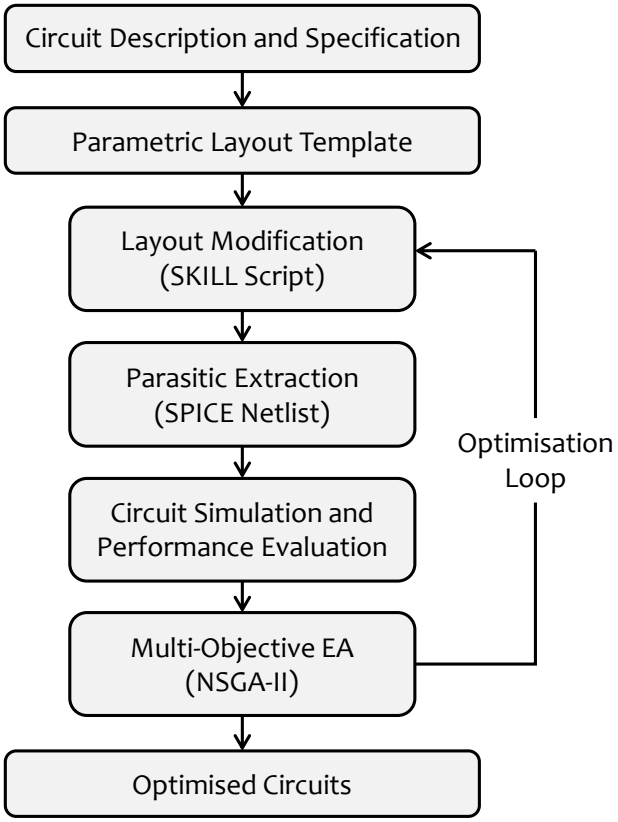


Figure 4.1 The multi-objective physical design flow with a parametric layout engine.

(1) An initial, parameterised, layout template is created using a standard design tool (Cadence® Virtuoso® [101]) according to the circuit description and specification.

(2) The layout template combines with an MOEA which performs function-preserving structural modifications on the template. Such layout modification is selecting appropriate standard cells for use in this case, and it is performed through using a Cadence® SKILL® script.

(3) A parasitic extraction (PEX) is performed on each distinct resulting layout instance using Mentor® Calibre® [102]. The PEX netlist is then evaluated for several performance metrics using a standard SPICE simulator (Mentor® ELDO® [103]).

(4) The evaluation results feed the MOEA to update the layout with modifications continuously. The MOEA ultimately finds possible optimised trade-off solutions for a given circuit when the optimisation is complete.

The test case considered in this work is a 1-bit full adder. The parametric layout template for the 1-bit adder is constructed from fixed-size NAND gates, each with two variable-sized inverters at its output. Each inverter cell has a unique integer ID (genes), and the MOEA is then used to perform cell selection. In this initial work, drive strengths are optimised for the NAND function's output inverters by choosing appropriate drive strengths from a given set of cells.

4.3 Parametric Physical Layout

Schematics do not fully model all circuit characteristics during the IC design process, so performing design or simulation based on the schematic level are useful for behavioural verification. However, manufacturing an IC requires a physical layout created by transforming the circuit level representation (devices and interconnections) into geometric representations of shapes in multiple layers. The parasitic extraction of physical layouts can take wire lengths and more detailed physical device characteristics into account.

For the purposes of optimisation in this chapter, circuit layouts must be parameterised to provide representations for the MOEA using. A parametric layout template for an example circuit has been created using building blocks of physical layout structures. This allows modification of the design parameters such as the component sizes and cell spacing. By adjusting the relevant design parameters, it is possible to swap instances of the building blocks and thereby change the circuit's physical layout characteristics and performance while the circuit function is preserved. For the experiment in this work, to initially keep the setup simple, positions of the different building blocks within the layout are fixed, but their relative positions and wire lengths are automatically adjusted depending on which instances are selected. Hence, the parametric layout aims to more conveniently and efficiently adjust the circuit layout to achieve higher performance designs.

In this work, a number of standard logic cells (building blocks) are created (prior to running the optimisation) in a commercial 65nm process technology. Each cell has its physical characteristics defined by specific transistor sizes and the cell width. The SKILL® scripting language in Cadence® Virtuoso® is a powerful EDA tool which can be used to realise design automation on schematics and layouts in complex design flows. SKILL in this case is used to select cells from the custom-created cell library and automatically construct the physical layout based on the parameters supplied from the outer-loop MOEA optimisation algorithm. A basic placement and routing process is then performed based on these parameters and the layout template, as shown in Figure 4.3. The relevant design parameters are provided as arguments to a SKILL script which selects, places and routes the required cells from the library making it ready for parasitic extraction and simulation.

4.3.1 Pre-designed Standard Logic Cells

The pre-designed library, containing logic cell schematics and layouts, includes one fixed-size function cell (NAND gate) and ten inverters which are used to realise the different drive strengths for each instance of the NAND function. The inverter's NMOS transistor widths are increased linearly from 120nm (minimum MOSFET width in the process technology used) to 550nm. The PMOS widths are determined by the paired NMOS transistor widths, specifically $\text{PMOS}_{\text{width}} = 1.4 \times \text{NMOS}_{\text{width}}$ to account for slower mobility of the charge carriers in the PMOS devices. All cells are designed with the same height, 1.8um in this work, and some of them use a multi-fingered layout style to fit larger-width transistors into the fixed cell layout height.

A layout example of one of the inverters is shown in Figure 4.2, the two vertical strips on both sides near the cell boundary are the input and output pin physical extensions which are used to make inter-cell routing more convenient and efficient for the experiments in this case.

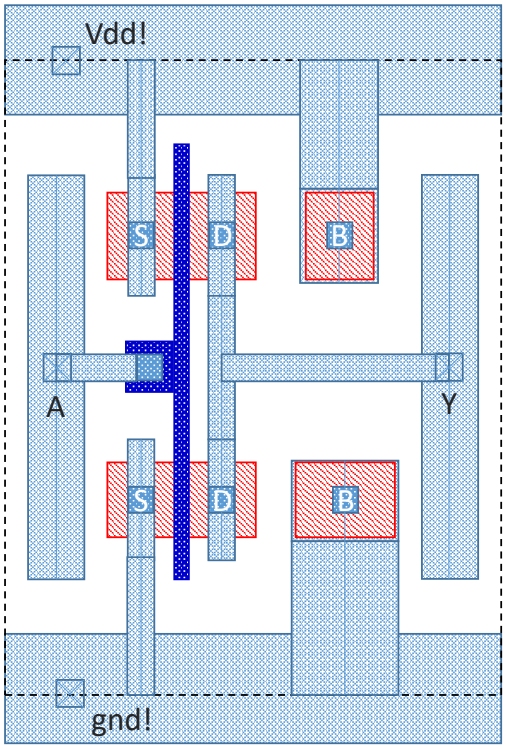


Figure 4.2 An inverter layout example. Input A and output B on both sides of the layout are in the shape of strips corresponding to metal layer 1 (metal-1).

4.3.2 Layout Generation

Figure 4.3 shows an example of parametric layout generation. A layout template is shown at the top, with the logic function cells (green blocks) and conceptual routing in place to implement the overall function. After each logic function, there are two reserved empty spaces where the optimisation algorithm can insert any combination of inverters (orange blocks) from a cell library provided (shown on the right) for fine-grained tuning of output drive strength. Each inverter in the cell library ($\mathbf{G}$) has a unique integer identity (gene), and the string of integers representing the set of inverters to be inserted into the template forms the chromosome ($\mathbf{g}$). To instantiate a circuit layout is to look up the requested inverter layouts in the cell library and place them into the circuit floorplan.

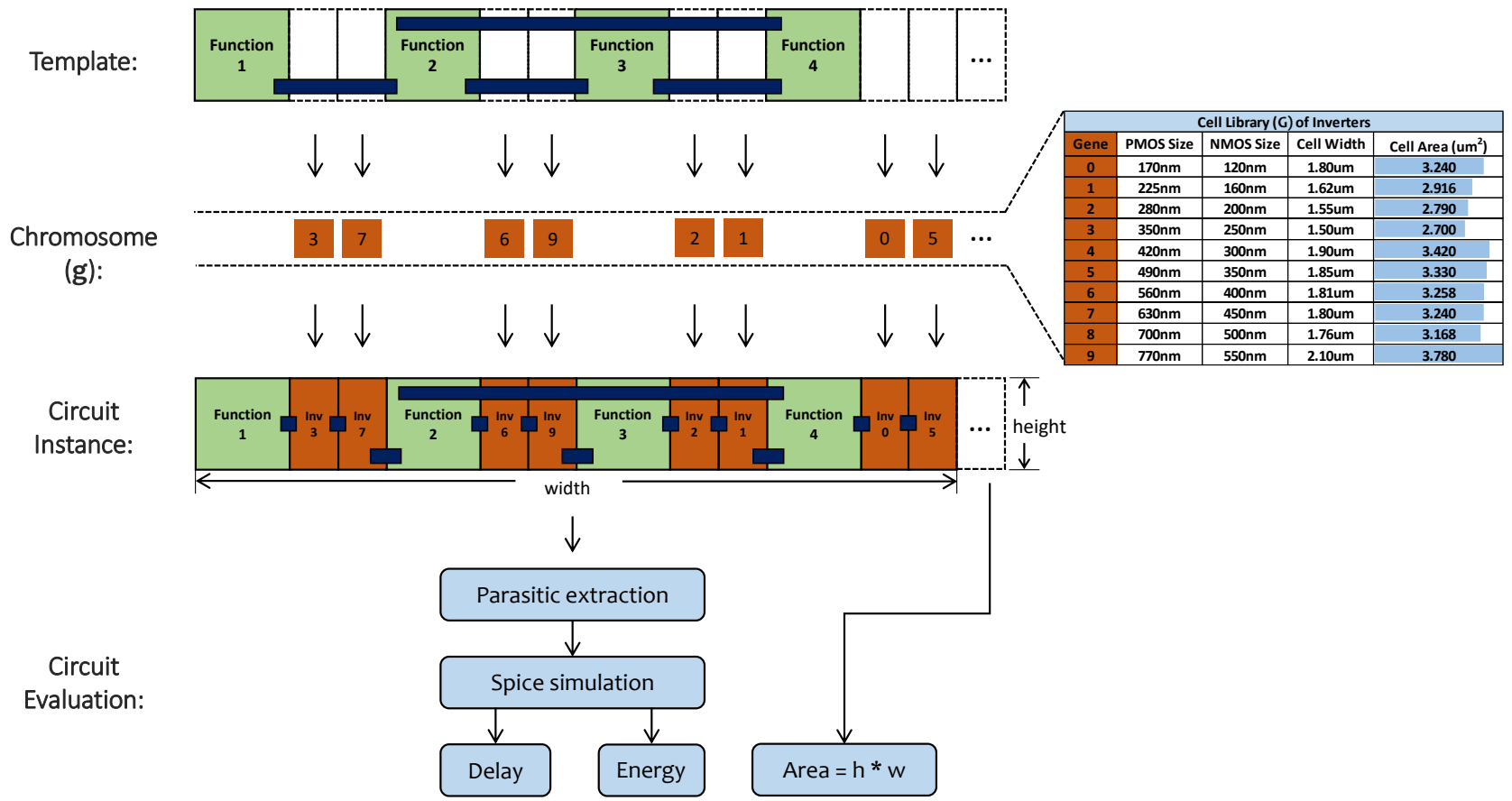


Figure 4.3 Example of circuit instance generation using parametric layout template with subsequent circuit evaluation. A layout template is firstly defined according to circuit specifications. The layout is then instantiated through inserting inverters from a custom-designed cell library onto the template. The produced layout instance is evaluated in regard to delay, energy and area.

Once the inverters are chosen and put in place, the routing is automatically adjusted for the different cell widths resulting in a completed layout instance of the template circuit. Subsequently, the circuit instance is evaluated on the desired performance characteristics (objectives) through parasitic extraction and SPICE simulation. All steps are automated within standard EDA tools using the SKILL scripting language.

In this initial work, the parametric layout generation is focused on varying the combination of inverters while the location and size of the NAND gates remain fixed. Furthermore, the layout floorplan presents all cells next to each other in the shape of a strip without gaps. This is to ensure that a simple routing solution is possible. The upper layer metal paths (dark blue strips) connect the cells to form a complete circuit.

4.3.3 SKILL Script

To automatically generate the layouts, specifically the template creation and circuit instantiation, a SKILL script is used. For placement and routing efficiency, a database of all the inverters' information is stored in this script which includes the inverters identifier, cell width and the input/output (I/O) pin positions. During instantiation, a determined circuit chromosome is placed into this script which then performs cell placement, inter-cell routing and top-level I/O pin creation (See Appendix A for an example).

(1) Cell placement. In order to perform cell placement, the SKILL script locates the cell dimensions for each inverter cell by performing a look-up using the integer identifiers. All cells (NAND and inverter) are then sequentially placed with reference to their size in order to ensure the cells exactly and precisely touch each other in the strip of the layout.

(2) Interconnection. The inter-cell routing process uses the stored I/O pin locations for each cell. A total of five fixed routing channels are available due to the design rules of the cell height, and an upper metal layer (metal-2) trace is created in each channel

to connect the cells. Metal-2 to metal-1 via placement is also performed at this stage. All cells are currently in a fixed height, so routing is only performed in the horizontal dimension.

(3) Top-level Pin creation. The final step of layout generation requires the creation of the top-level pins for the whole circuit. These are placed in a fixed position in the layout and allow the completed circuit to be connected to an external SPICE testbench.

4.3.4 Parasitic Extraction

Once layout generation is completed, evaluating the performance of a circuit from the physical layout requires the extraction of parasitic information. Parasitic extraction is the modelling and calculation of the parasitic effects in the circuit interconnect and devices, which can include the parasitic capacitance, resistance and inductance [104][105]. It can create a more accurate analogue model of a circuit. In this work, Mentor® Calibre® [102] runs the parasitic extraction (PEX). The extracted netlist is then combined with a set of testbenches which evaluate the circuit performance for three metrics: worst-case propagation delay, energy consumption and circuit area.

4.4 Multi-objective Circuit Optimisation

As illustrated in Figure 4.1, the optimisation loop performs modifications on the parametric layout template to generate new layout instances with different properties, such as different area, energy consumption, propagation delay. More importantly, the MOEA, taking charge of such modifications above through defining different chromosomes in the SKILL script, can automatically and more efficiently select cells to perform function-preserving optimisation for producing a set of Pareto-driven solutions.

4.4.1 Algorithm

NSGA-II, a popular multi-objective optimisation algorithm, is adapted in this case. It is a desirable method used in VLSI circuit design given the multiple conflicting objectives, particularly in trading off circuit power, speed and area.

During the optimisation, NSGA-II takes charge of the optimisation loop involving parametric layout generation and eventually reaching a more optimised set of trade-offs through comparing all generated solutions. In detailed, NSGA-II is now automatically performing circuit instantiation and circuit evaluation in Figure 4.3.

Algorithm 2 Adapted NSGA-II for MO Layout Optimisation

Procedure: NSGA-II (N, M, $f(\mathbf{g})$). ▷ N individuals evolved M generations to solve the fitness function $f(\mathbf{g})$. The vector $\forall \mathbf{g} \in \mathbf{G}$ (pre-defined cell library) is a chromosome consisting of a set of genes (inveters) representing an individual (layout instance).

Randomly initialize parent population $\mathbf{P}_t = \{\mathbf{g_1}, \mathbf{g_2}, ..., \mathbf{g_N}\}$ in size N
Offspring population $\mathbf{Q}_t \leftarrow$ Mutation($\mathbf{P}_t$)
for $t \leftarrow 1$ to M **do**
  for each population $\mathbf{R}_t \leftarrow \mathbf{P}_t \cup \mathbf{Q}_t$ in size $2N$ **do**
    Fitness evaluation $\leftarrow f(\mathbf{R_t})$ ▷ Call fitness function $f(\mathbf{g})$ for each individual evaluation.
    $\mathbf{F} \leftarrow$ Non-Dominated-Sorting($\mathbf{R}_t$)
    $\mathbf{P}_{t+1} \leftarrow \emptyset$
    $i \leftarrow 1$
    while $|\mathbf{P}_{t+1}| + |\mathbf{F}_i| \leq N$ **do**
      Crowding-Distance-Assignment($\mathbf{F}_i$)
      $\mathbf{P}_{t+1} \leftarrow \mathbf{P}_{t+1} \cup \mathbf{F}_i$
      $i \leftarrow i + 1$
    end while
    $\mathbf{F}_i \leftarrow$ Descend-Sort($\mathbf{F}_i$)
    $\mathbf{P}_{t+1} \leftarrow \mathbf{P}_{t+1} \cup \mathbf{F}_i[1 : (N - |\mathbf{P}_{t+1}|)]$ ▷ Less crowded individuals from the first to the $(N - |\mathbf{P}_{t+1}|)$th of $\mathbf{F}_i$ to fill $\mathbf{P}_{t+1}$.
    $\mathbf{Q}_{t+1} \leftarrow$ Mutation($\mathbf{P}_{t+1}$)
  end for
end for

Both the fast non-dominated sorting approach and diversity preservation scheme of NSGA-II are adopted, but only mutation operator is implemented in this case. The

used notations about the EA setup are: N is population size; M is the maximum number of generations; ρ is the mutation rate. The adapted algorithm, in this case, is presented in Algorithm 2.

4.4.2 Objectives

The three optimisation objectives considered in this work are: propagation delay, energy consumption and circuit area. So the fitness function is:

$$f(\mathbf{g}) = \min \quad [delay, \quad energy, \quad Area] \quad \text{s.t.} \quad \forall \mathbf{g} \in \mathbf{G} \tag{4.1}$$

Measuring these objectives properly and observing any improvements is a key task during the optimisation.

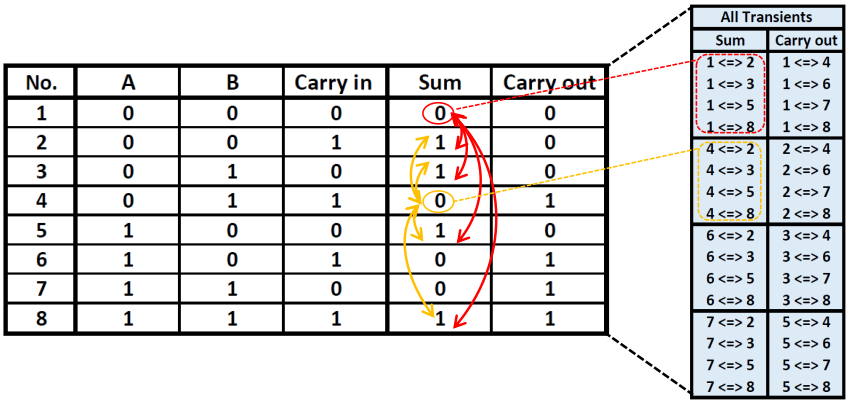


Figure 4.4 Truth table for the Full Adder circuit. The expanded view on the right show all of the possible transitions for each output.

(1) Propagation Delay. Figure 4.4 shows the truth table for a full adder. It has three inputs: *A*, *B* and *Carry-in (Cin)* and two outputs: *Sum* and *Carry-out (Cout)*. The expanded view on the right shows all of the possible transitions for each output. For example, in the case of row one there are four possible input combinations which can change the sum output from a zero to a one. In total, there are 32 possible transitions.

The proposed propagation delay which should be measured and minimised is the worst case among all transitions.

(2) Energy Consumption. The energy consumption for each of the 32 transitions was calculated from the transient simulation results using Equation (4.2).

$$E_n = \int_{t_{start}}^{t_{stop}} i(t)dt \times vdd \qquad (Joules) \tag{4.2}$$

where t_{start} and t_{stop} represent the start and end time of the transient simulation and $i(t)$ is the current drawn from the DC power supply. The total energy consumption for the circuit instance was then calculated by summing the energy used during each transition as shown in Equation (4.3). However, some internal transitions which the intermediate gates that do not necessarily cause the primary output's transients are not included for power measurement.

$$E_{total} = \sum_{n=1}^{n=32} E_n \tag{4.3}$$

This calculation combines both the dynamic and static power consumption of the circuit over all transitions. In future work, it may be desirable to extract these as separate objectives.

(3) Circuit Area. For efficient placement and routing, standard logic cells are typically in a fixed height. The full adder is laid out in a single strip. Therefore, the circuit area can be calculated directly as $width \times height$ as show in Figure 4.3. The width value for each cell is stored directly in the SKILL script.

4.4.3 Circuit Simulation

To evaluate the performance of each circuit, a SPICE testbench containing one transient simulation for each of the transitions outlined in Figure 4.4 was created. Each circuit was simulated using Mentor® ELDO® [103]. SPICE measurement expressions were

used to extract each of the three objectives outlined above. The input stimuli consisted of a periodic pulse with a 2.5ns pulse duration (i.e., 400MHz clock frequency) and 1ps edge rise/fall time, which is set according to the 65nm node performance. In terms of the load capacitance, 2fF, 5fF and 10fF connected to each of the outputs for experiments were used, and only the best optimisation results is shown in this chapter. Extraction of the worst case delay, total energy consumption and circuit area is implemented in a post-processing step using a python script and then passed to the NSGA-II optimisation algorithm.

4.5 Experimental Results

In this work, a 1-bit full adder was chosen as the proof-of-principle test circuit. As described previously, the cell library is currently constrained to minimum-sized, 2-input NAND gates and a selection of inverters with varying drive strengths. This allows, initially, to keep the circuit structure simple and understand what will be required to scale to larger cell libraries and designs. Based on these building blocks, a parametric layout of a 1-bit full adder is created and multi-objective optimisation is carried out.

4.5.1 Full Adder Parametric Layout

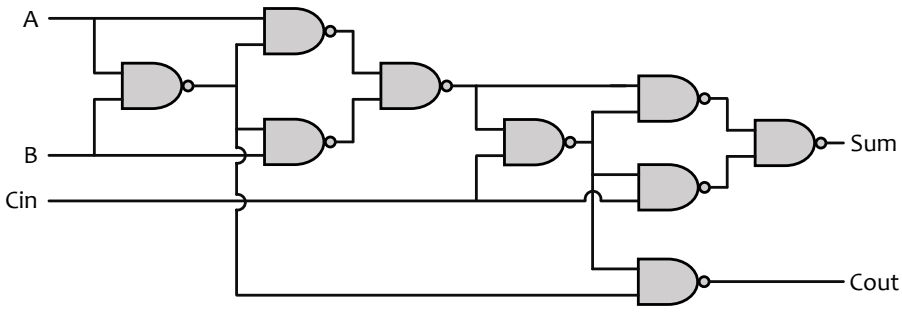


Figure 4.5 A schematic of the full adder circuit showing the NAND gates. In this experiment, each NAND gate contains the base logic function and two series connected inverters.

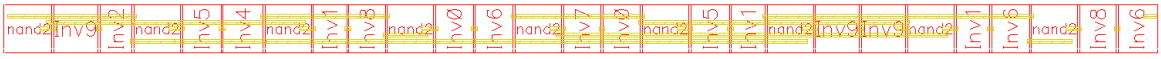


Figure 4.6 Example parametric layout of the full adder circuit, some layers have been hidden for easy legibility. The yellow wires are interconnections in metal layer 2 (metal-2).

In order to realise the 1-bit full adder circuit, as shown in Figure 4.5, nine NAND gates are required and each of these NAND gates is equipped with two inverters at its output. Hence, there are a total of eighteen inverters presented in a design. Figure 4.6 shows a physical layout example for one design taken from the optimisation population where the 'inv' and 'nand' instances are labelled. It is these 'inv' instances that are parametrised and can be swapped with other 'inv' instances from the cell library during circuit optimisation. The SKILL script ensures that all layout design rules are met, all cells are abutted to each other without gaps and that the metal-2 layer paths (yellow) are properly routed between cells to ensure overall function as a 1-bit full adder.

4.5.2 Optimisation Results and Discussion

During optimisation, a population size N of 200 individuals is used and the maximum run generations M is set to 150. One inverter out of total eighteen will be randomly modified during each mutation process, so the mutation rate ρ is 1/18 in this case. Each optimisation run took approximately 50 hours using a four-core 2.4 GHz Xeon processor. The results shown in Figure 4.7 are for the experiment using 5fF output load capacitance, which presents a Pareto-optimised solution spread in this case. The sub-plots in the left column show the initial (blue) and final (red) populations for each pair of objectives (delay, energy and area). The right column of plots shows only the final population for a pair of objectives but includes the third objective as a colour map.

With reference to plot 4.7 (b), it can be seen that larger circuits typically consume more energy but provide higher speed and smaller circuits consume less energy but are slower. It should be noted that in this plot, there is also a cluster of higher speed,

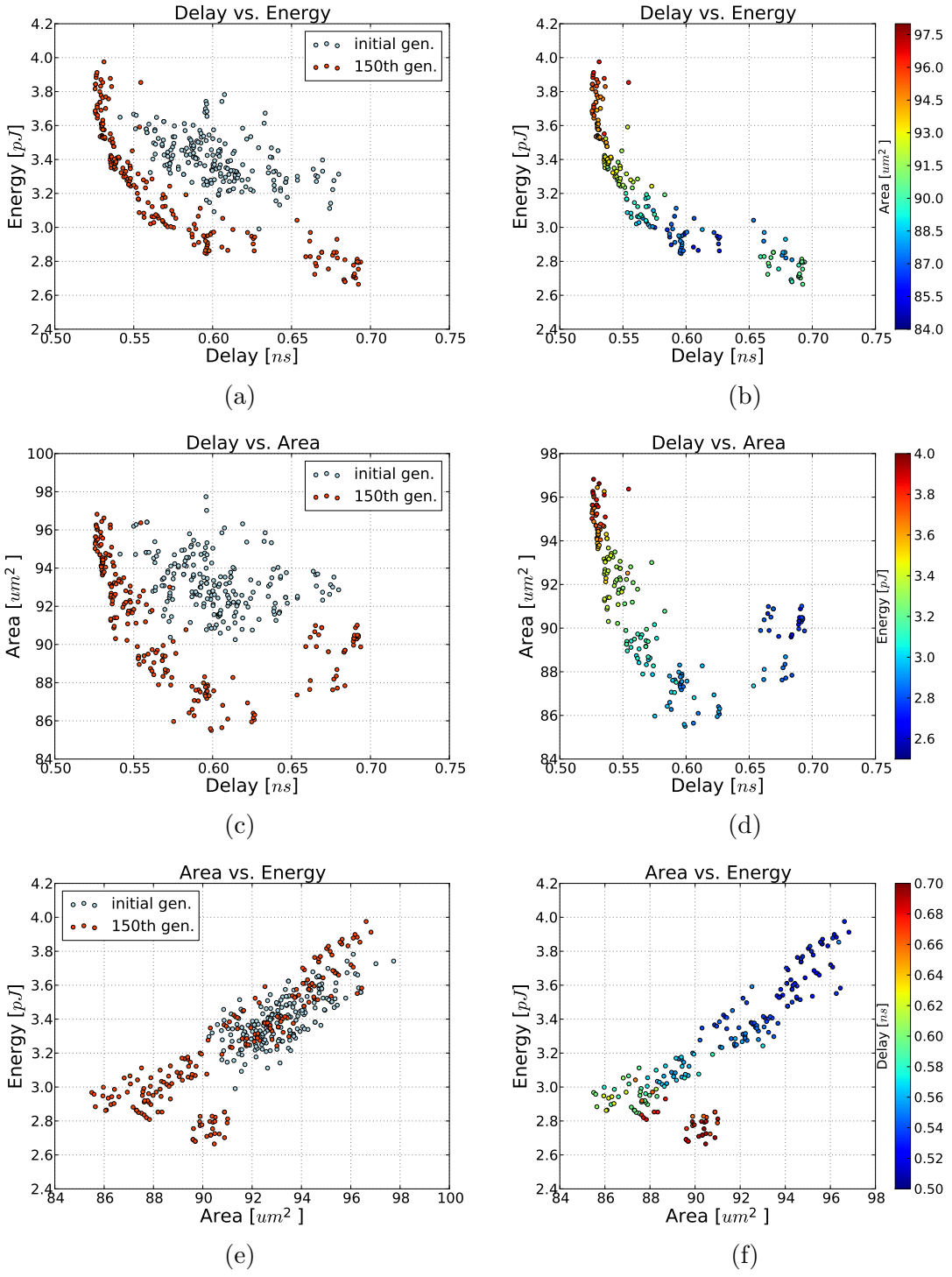


Figure 4.7 Optimisation results for the 1-bit full adder ($N = 200, M = 150, \rho = 1/18$, output load=5fF). Plots (a), (c) and (e) show scatter graphs of the initial (blue) and final (red) populations for each pair of objectives. Plots (b), (d) and (f) only show the final population but include the third objective as a diverging colour map.

lower energy individuals in the bottom right, from 0.65ns to 0.7ns on delay axis. The reason for this is that the minimum-size transistors do not have the smallest layout area due to design constraints imposed by the foundy and PDK. Although solutions in this cluster do not feature the smallest total circuit area, the specific transistors widths are in fact smaller. Plots (d) and (f) also have separate clusters due to design constraints. The cluster between 0.65ns and 0.7ns on delay axis in plot (d), delay vs area, includes some slower speed circuits which are using the smallest transistors but the total circuit areas are again not the smallest. A similar situation is also visible in the plot (f), area vs energy.

4.5.3 Optimisation Advancement at Physical Level

In this part, an empirical comparison study is performed to present the difference in circuit performance between schematic level and physical level during the optimisation.

The same multi-objective optimisation approach is applied at the schematic level of the 1-bit full adder. The population size N is 200, the number of generations M is 150, and the mutation rate ρ is set for 1/18, the same as in the layout level experiment. Two sets of optimised results are achieved, and both final populations are plotted in Figure 4.8. There are two clusters in the plot. The blue represents the optimised results based on the schematic level, and the red cluster shows the results (same as shown in Figure 4.7) that the optimisation is performed at the physical layout level. From the plot, the blue cluster (schematic level) has better delay and lower energy consumption than the red cluster's (layout level). The margin looks significant between both levels, but the layout level results have better diversity. This is because the simulation at the schematic level does not include the parasitic effects of circuits, which could make the circuit performance of different schematics not distinct from each other and the resulting solution diversity limited.

To compare the optimisation efficiency, few individuals are selected from both clusters for extra simulation and analysis. The five red crosses, at the lower left corner in

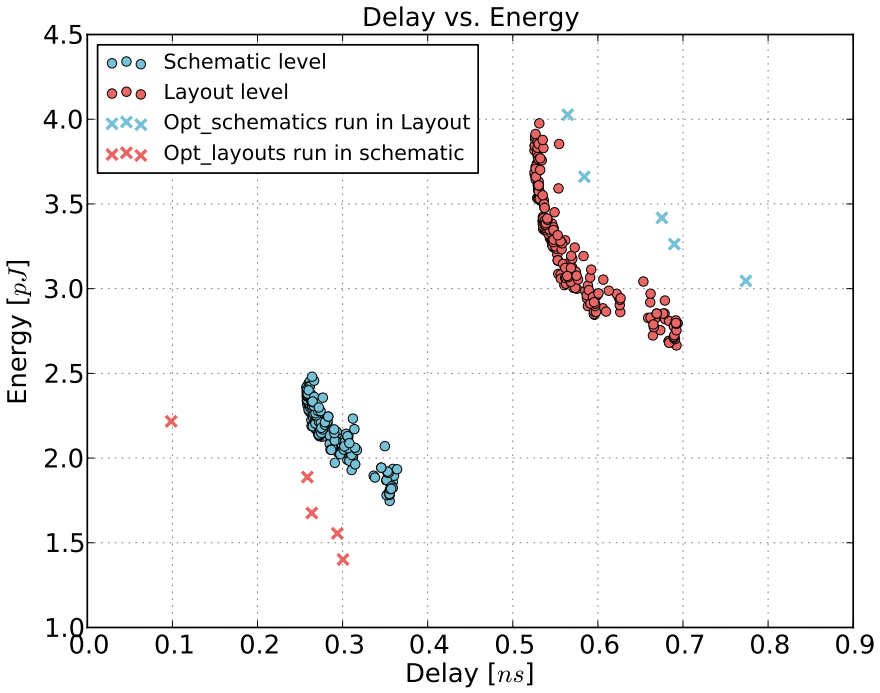


Figure 4.8 Example of parametric layout generation with subsequent circuit evaluation.

Figure 4.8, are taken from the first non-dominated front of the layout level optimisation results (red cluster). These five selected individuals were evenly distributed at the original front. Thereafter, they are additionally run the simulation based on the netlists of their schematics without parasitic extraction using the same chromosome (i.e.,the selection of inverters).

Regarding the five blue crosses, at the upper right corner in the plot, they are the schematic level (blue cluster) optimised solutions which also evenly positioned at their first non-dominated front. The new results (blue crosses) are achieved by implementing the schematic level optimised solutions to the physical layouts with running parasitic simulation.

Based on the observation of this comparison study, operating optimisation at the physical level, which includes all parasitic effects and layout design rules, not only makes design realistic and feasible but also has superior optimised results (better

performance and wider solution diversity) than at the schematic level under the same MOEA evaluation budget.

4.6 Summary

This chapter describes a methodology to maintain circuit building blocks (standard cells) on parametric physical layouts for multi-objective circuit optimisation. The fully automated approach is compatible with standard EDA design tools, making use of the built-in scripting language SKILL to generate layout instances in the loop which produces Pareto-optimised solutions for a CMOS VLSI design.

Such an optimisation keeps looking at the physical layout level, aiming to ensure that the optimised circuits can be fabricated and that the measured performance metrics are realistic. Although this might take more simulation efforts and runtime, it can quite hold design quality practical. Specifically, this chapter demonstrated the feasibility of the proposed parametric layout approach in the context of multi-objective optimisation, and results for a 1-bit full adder are shown. In addition, an empirical study is performed using the same test case to demonstrate that the optimisation at the physical level advances the results compared to the schematic level optimisation.

The following chapter scales up the multi-objective optimisation framework to be compatible with foundry libraries and industrial auto-synthesis, place and route flows. The methodology and its experiments are presented in Chapter 5.

The optimisation avenue in this chapter is tuning standard cell selection. Particularly when combined this with transistor sizing later (the work presented in Chapter 7), this is something current approaches cannot exploit, as instead they are constrained by the limited choices of pre-sized cells available in a library. This may lead to the automatic place and route tool adding margin and selecting cells that are too large or vice versa.

Chapter 5

Multi-objective (MO) EDA Framework

5.1 Overview

The procedure of building a digital integrated circuit using pre-processed blocks or cells from a foundry is a common and mature methodology in modern VLSI design. Comprehensive industry-standard flows are available to tape out digital chips. Technology down-scaling enables high-density circuits and the EDA tools therefore need to handle a large quantity of cells within the flow. Meanwhile, multiple versions of each logic function, in terms of drive strength, cell height, threshold voltage, etc., are typically offered by the foundry in cell libraries. The possible design space is thus huge and complex because a circuit can be composed of millions of gates. Different selections and combinations of cells can directly determine the power, performance and area (PPA) metrics of a circuit. Such a search space will be further complicated with practical design rules and constraints in physical implementation. This can lead to the rise of optimisation difficulty that designs must meet multiple conflicting objectives simultaneously while satisfying all rules and constraints at the layout level, which might be beyond what experienced engineers can manually handle.

In this chapter, a fully-automated, multi-objective (MO) EDA flow is introduced to address this issue. It specifically tunes drive strength mapping, preceding physical implementation, through an multi-objective population-based search algorithm. Designs are evaluated with respect to their PPA metrics. The proposed approach is capable of expanding the design space, offering a set of Pareto-optimised solutions for different case-specific utilisation. The proposed MOEDA framework has been applied to ISCAS-85 and EPFL benchmark circuits using a commercial 65nm standard cell library. The experimental results demonstrate how the MOEDA flow enhances the solutions initially generated by the standard digital flow, and how simultaneously a significant improvement in PPA metrics is achieved.

The specific contributions made in this chapter are summarised as follows: 1) A multi-objective (MO) EDA optimisation framework, fully-compatible with an industrial digital flow from logic synthesis to physical implementation. 2) Global tuning of standard cell

drive strength mapping using parameterised gate-level netlists. 3) Enhanced trade-off design solutions with improved PPA metrics. 4) A comparison study of optimisation efficiency between the MOEA search and the stochastic search.

The remaining parts of this chapter are structured as follows: Section 5.2 gives a related background review of discrete gate sizing. Section 5.3 introduces the proposed MOEDA flow. Experiment setup is described in Section 5.4. Section 5.5 presents experimental results using a reduced cell library and a commercial full cell library, and a statistical evaluation of MOEDA flow is included. A comparison experiment between the used MOEA and the random search is performed in Section 5.6. Section 5.7 outlines summary of this chapter.

5.2 Discrete Gate Sizing for PPA Optimisation

Gate sizing is a crucial step for achieving timing closure and power minimisation of digital ICs. It originally refers to determining transistor widths inside of logic gates to make designs meet constraints. Modern digital EDA flows synthesize designs using a set of pre-designed cells [10]. The optimisation problem thus is shifted to focusing on cell selection, in respect to drive strengths and often including threshold voltage V_{th} assignment, from discretised gate libraries. This process exists in the technology mapping step during logic synthesis and the gate resizing optimisation during physical implementation. It also often happens in manual design adjustment to achieve timing closures by engineers, particularly when EDA tools failed to meet the timing goal.

The optimisation objective in gate sizing is to minimise power consumption (mainly leakage power) while meeting the timing constraints [10]. To achieve this goal, many methods have been proposed to facilitate the gate sizing problem. Lagrangian Relaxation (LR) has been recently used for gate sizing optimisation, which moves the timing constraints to the objective function weighted by Lagrange multipliers to penalise the

overall results of the objective function. The problem is then simplified to find the solution of weight factors.

In regard to the optimisation objectives in the related research using LR, the work in [106] derived LR associated with finding trade-off between leakage power and circuit timing. The authors in [107] expanded the primal objective function (power minimisation) by adding the area objective using an extra weight factor while meeting the timing constraint. More recently, A. Sharma in [108] considered more additional realistic constraints, such as maximum load, maximum slew, of gates for simultaneous gate sizing and clock skew scheduling. However, the LR theory is typically formulated for continuous problems and might not naturally handle the discrete gate-sizing [109]. In addition, the LR often assumes convexity, which might not hold for practice circuit delays (i.e., nonlinear, non-convex) [110].

Alternative multi-objective gate sizing frameworks, like geometric programming [53] [54], simulated annealing [111], have been investigated using the weighted sum scalarizing method to handle the multi-objective functions. In [109], J. Hu proposed a different way to scalarise the objectives of leakage power and slacks into a sensitivity guided function for solution ranking (non-dominated), and a heuristic-based stochastic searching method was applied. The proposed method included two stages: global timing recovery seeks violation-free solutions by up-sizing gates and down-sizing threshold voltage (V_{th}), and then power minimisation with feasible timing reduces leakage power on the gates (gate down-sizing and V_{th} up-scaling) that are oversized during the first stage.

Given the background review above, limited research completes the gate sizing simultaneously handling all critical objectives (PPA) through an industrial physical design flow and libraries to investigate how beneficial these methods can be in practice [32] [107] [110]. Yella in [32] stated that significant changes in cell sizes of design netlists, after applying gate-sizing optimisation, require re-placement and re-routing for new wire load parasitics. Therefore, optimising designs with timely updating corresponding layouts can make evaluations realistic, and achieved solutions feasible.

In earlier works, typical heuristic techniques like genetic algorithms were applied to solving gate sizing problems. The methods for multi-objective optimisation in [112] and [113] both are still based on scalarized cost functions. More recently, gate-sizing-based soft error optimisation using MOEAs is proposed in [96] but its objectives are soft error rate, critical path delay and area.

The gate sizing problem with its optimisation is multi-objective in nature. Most introduced methods are scalarising based (e.g., a popular one, weighted sum function) to decompose the optimisation complexity since its high search efficiency [50]. However, as stated in Chapter 3, the device physics of ICs imply non-convexities and non-linearity [55] where the weighted sum method is not sufficient to search for feasible Pareto-optimal solutions [50].

Solving discrete gate sizing problems still lacks the theoretical guarantee and it is not clear how far the current optimised solutions are away from the optimum [10]. Therefore, it is worthwhile to apply global search methods to optimise such a problem. MOEAs excel in handling multiple design parameters and objectives inter-independently particularly when designers are faced with a large, complex design space.

5.3 MOEDA Optimisation Framework

5.3.1 Algorithm

In this case, NSGA-II [70] has been adapted as the search tool. Particularly, the fast non-dominated sorting approach and diversity preservation strategies used ensure convergence while achieving a uniform spread of Pareto-optimal solutions. Only mutation operator is adopted.

The used notations about the EA setup are: N is population size; M is the maximum number of generations; ρ is the mutation rate. The adapted NSGA-II will be introduced in the following subsection in detail.

5.3.2 Multi-objective (MO) EDA Flow

The MOEDA flow, illustrated in Figure 5.1, is a fully-automated multi-objective design framework using compatible with an industrial digital flow. The industrial flow is tapped between the logic synthesis and the physical implementation stage, where the MO evolutionary optimisation loop is inserted. The novelty here lies in the additional level of abstraction that can automatically fine-tune drive strength mapping during the process of the flow. The proposed flow involves:

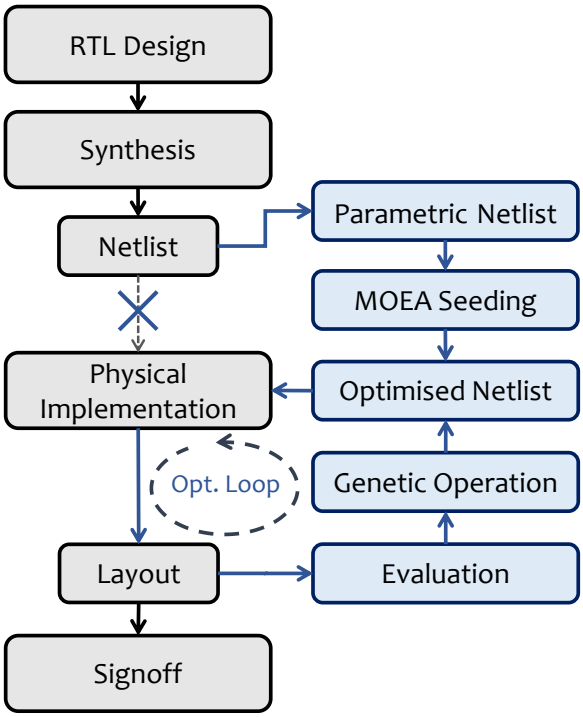


Figure 5.1 MOEDA Flow. The flowchart on the left side is the standard digital flow and on the right side the MO extension is shown. The blue cross indicates the position where the standard flow is broken.

(1) Parametric netlist. Synthesised netlists are composed of technology-specific logic gates and their connectivity. The MOEA representation encodes the drive strengths of gates into a set of genes, in this case a string vector $\mathbf{g}$ (i.e., instance names), defining each gate function and its drive strength. This information is used to produce a parametric netlist from the synthesis results. Integer EA representations are a commonly used

encoding method, but in this work industrial logic gate libraries (a wide range of gate types and driving options) are integrated into the loop, so it requires the representations to cope with different gate types and their respective drive strengths.

(2) MOEA seeding. In this work, initial populations are seeded from a set of solutions obtained from the synthesis tool. This is achieved by converting the output netlists from the standard tool to parametric netlists, allowing the MOEA to modify them.

(3) Genetic operations. Only mutation is used in this work. The mutation operation modifies the drive strength of components based on a given probability ρ. This results in a new netlist, which is then ready for physical implementation. With the pressure to promote beneficial mutations and discard the others, the evolutionary loop continues to keep producing increasingly optimised solutions.

(4) Evaluation. This calculates the fitness scores of each individual. MOEA-optimised netlists are propagated to place and route in the physical implementation step, producing layout instances for accurate evaluation metrics. Three objectives are used here which are worst case delay (D_{wc}), total consumption power (P_{total}) and area of all logic gates (A_{gate}), and fitness scores are evaluated at post-route stage from the place and route tool. Fitness scores are then fed back to the MOEA for ranking and selection.

The optimisation goal in this work is to simultaneously minimise D_{wc}, P_{total} and A_{gate} so the fitness function is:

$$\begin{aligned} f(\mathbf{g}) = \min \quad & [D_{wc}(\mathbf{g}), \quad P_{total}(\mathbf{g}), \quad A_{gate}(\mathbf{g})] \\ \text{s.t.} \quad & \mathbf{g} = (g_1, ..., g_i), \quad \forall g_i \in \mathbb{G} \end{aligned} \tag{5.1}$$

where the chromosome vector $\mathbf{g}$ is the input variables to the fitness function, which are drive strengths of gates (g_i) selected from a standard cell library ($\mathbb{G}$).

Figure 5.2 demonstrates a population example where $\mathbf{P}_t$ consists of N layout individuals $(\mathrm{L}_1, \mathrm{L}_2, ..., \mathrm{L}_\mathrm{n})$. Each L has a chromosome $\mathbf{g}$ consisting of a set of genes $(g_1, g_2, ..., g_i)$.

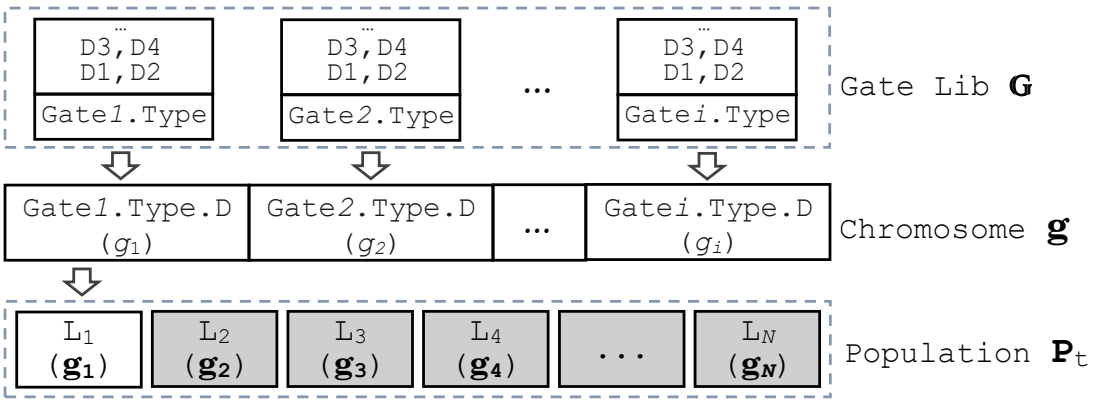


Figure 5.2 A chromosome example of an individual in a population and how each gene is mutated using a logic gate library. "D" represents the drive strength.

The chromosome overall represents all the logic gates of a netlist. Each single g (Gate.Type.D) represents the drive strength of a logic gate. When mutation is triggered, the gates to be mutated are randomly selected according to the mutation rate ρ. For each selected gate, it will first identify its function (Gate.Type) and then perform an online look-up to achieve all drive strength options (D) of this gate function from $\mathbb{G}$, and randomly choose one drive strength from them to replace the previous one.

The optimisation process is continuously producing different circuit layout instances by adjusting the netlists and keeping improved solutions generation-by-generation. The adapted NSGA-II algorithm is explained in Algorithm 3. Figure 5.3 presents the overall process of MOEDA in the context of the NSGA-II.

Algorithm 3 Adapted NSGA-II for MOEDA flow

Procedure: NSGA-II (N, M, $f(\mathbf{g})$). ▷ N individuals evolved M generations to solve $f(\mathbf{g})$.

1: Initialize parent population $\mathbf{P}_t$ in size N. ▷ Seed with a specific synthesis-optimised solution generated by the tool.
2: Offspring population $\mathbf{Q}_t \leftarrow$ Mutation($\mathbf{P}_t$)
3: **for** $t \leftarrow 1$ to M **do**
4: **for** each population $\mathbf{R}_t \leftarrow \mathbf{P}_t \cup \mathbf{Q}_t$ in size $2N$ **do**
5: Fitness evaluation $\leftarrow f(\mathbf{R_t})$ ▷ Call fitness function $f(\mathbf{g})$ for each individual evaluation.
6: $\mathbf{F} \leftarrow$ Non-Dominated-Sorting($\mathbf{R}_t$)
7: $\mathbf{P}_{t+1} \leftarrow \emptyset$
8: $i \leftarrow 1$
9: **while** $|\mathbf{P}_{t+1}| + |\mathbf{F}_i| \leq N$ **do**
10: Crowding-Distance-Assignment($\mathbf{F}_i$)
11: $\mathbf{P}_{t+1} \leftarrow \mathbf{P}_{t+1} \cup \mathbf{F}_i$
12: $i \leftarrow i + 1$
13: **end while**
14: $\mathbf{F}_i \leftarrow$ Descend-Sort($\mathbf{F}_i$)
15: $\mathbf{P}_{t+1} \leftarrow \mathbf{P}_{t+1} \cup \mathbf{F}_i[1 : (N - |\mathbf{P}_{t+1}|)]$ ▷ Less crowded individuals from the first to the $(N - |\mathbf{P}_{t+1}|)$th of $\mathbf{F}_i$ to fill $\mathbf{P}_{t+1}$.
16: $\mathbf{Q}_{t+1} \leftarrow$ Mutation($\mathbf{P}_{t+1}$)
17: **end for**
18: **end for**

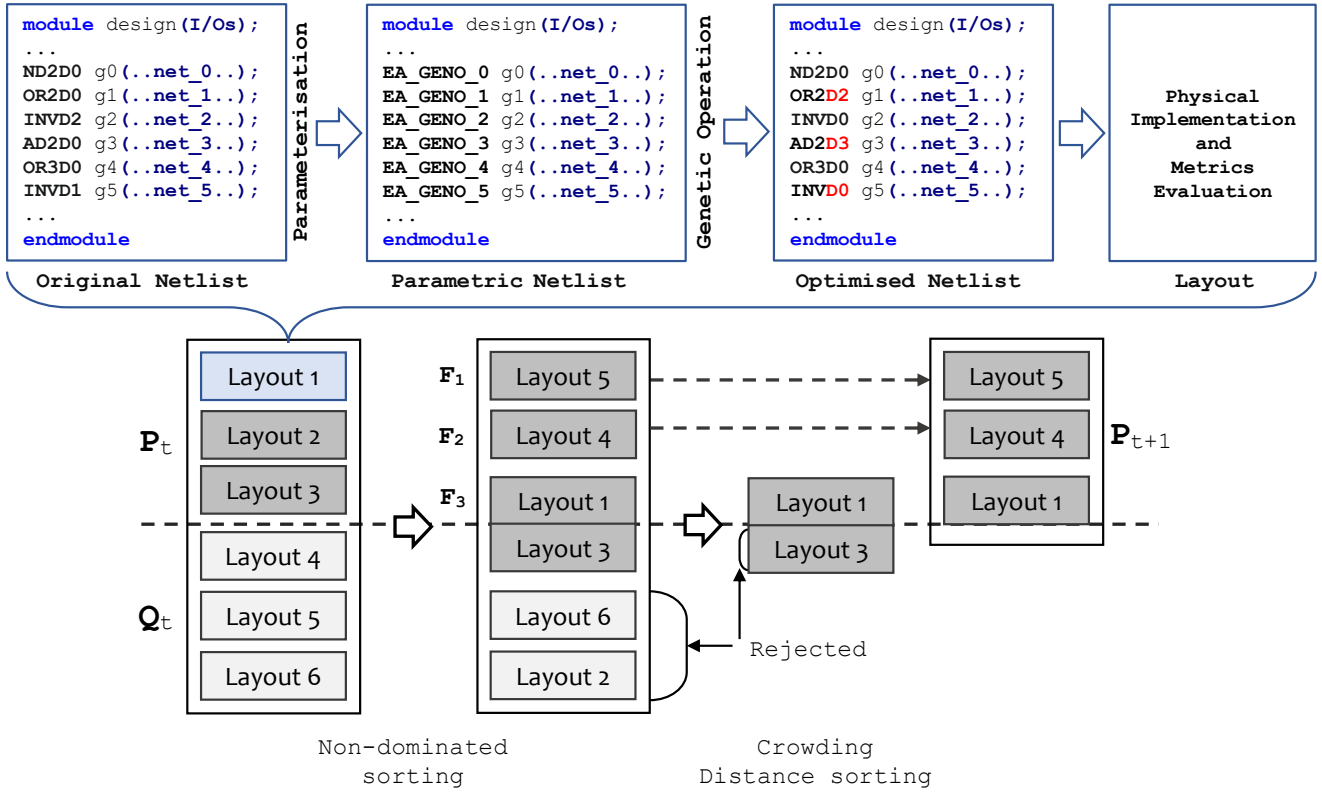


Figure 5.3 It shows the concept of the overall MO evolutionary optimisation process including original, parametric and optimised netlist examples. The highlighted texts in the optimised netlist is mutated gates. Individuals are represented by their circuit layout. For illustration, only a few individuals are shown in $\mathbf{P}_t$ and in each non-dominated sorting rank. Hundreds of individuals are typically used when running experiments. Layout 1 and Layout 3 are in the $\mathbf{F}_3$, the Layout 1 is in a less-crowed region so included into the $\mathbf{P}_{t+1}$ and Layout 3 is rejected during the crowding-distance sorting.

5.4 Experimental Setup

The proposed algorithm is implemented in C++ and the proposed MOEDA design flow experiments is run on a 2.2GHz Xeon E5-2650 CPU. The ISCAS-85 benchmark suite [114] and EPFL benchmark circuits [115] are implemented and optimised using the Cadence® digital flow suite. Benchmark circuits in the form of RTL designs are synthesised into gate-level netlists using Genus™ (v17.11) [116]. These netlists are then optimised using the proposed flow in tandem with the physical implementation tool Innovus™ (v17.11) [117] to generate the layouts from the optimised netlists. The versions of used EDA tools represented the most up-to-date flow when we performed the experiments. We also have full optimisation licences of Cadence® digital flow.

All experiments are using a TSMC 65nm low-power core cell library (TCBN65LP) in standard threshold voltage containing about 400 combinational cells.

5.4.1 Tool Environment Setup

The MOEDA flow is applied to further enhance designs which are already well-optimised by the industrial tools. In order to take advantage of the Genus™ synthesis tool as much as possible, it is necessary to push it to the limit of what it can achieve with the user options available. Hence, the *synthesis compile effort* is set to high and *ultra optimisation* is enabled. Apart from that, each benchmark is repeatedly synthesised, tightening its timing constraint bit-by-bit until it fails timing. The last working solution before timing failure is the best in speed, delay or slack that the tool can achieve. This solution is then chosen as a seed for initialising the MOEA.

In the timing constraint setup, an ideal general clock for all inputs and outputs of a circuit is created, which means all paths are clocked with two ideal flip-flops at the beginning and the end of each path. The benchmarks used are all combinational circuits, so that the ideal clock was not applied with any uncertainties or transition delays.

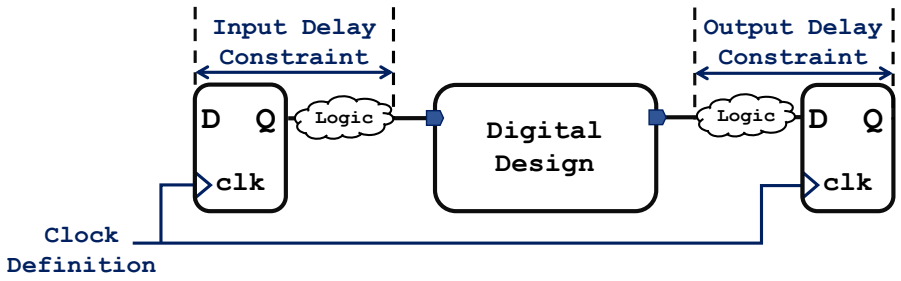


Figure 5.4 Conceptual testbench to define timing constraints in EDA tools. Virtual logic parts and flip-flops allows end users to specify delays and clocks. The design under test is *Digital Design* in the middle.

To tighten the timing constraint, the output delay constraint (T_{od}) (illustrated in Figure 5.4) is gradually increased for a given clock period (T_c). So the required time (T_r) (illustrated in Figure 5.5) is:

$$T_r = T_c - T_{od} \tag{5.2}$$

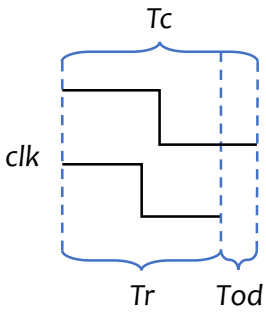


Figure 5.5 Conceptual waveform diagram to illustrate the relationship between clock period (T_c), the output delay constraint (T_{od}) and the required time (T_r).

The circuit path arrival time (T_a) should be less than the required time (T_r) to the meet the timing constraint. The settings of both synthesis step and physical implementation step are summarised in Table 5.1.

The timing constraint covers different general clocks for different test circuits and they will be specified in corresponding experiments. In the physical design flow, all die area is shaped in the ratio of 1.0, and core utilisation is 70%. Timing-driven placement

and routing, and signal integrity (SI) driven routing are enabled for better timing performance.

Table 5.1 Tool Settings in Digital Flow

Synthesis Setup	Place & Route Setup
syn_generic_effort = high iopt_ultra_optimisation = true	aspect ratio = 1.0 core utilisation = 0.7 timing-driven placement = true timing-driven routing = true SI-driven routing = true

5.4.2 Objective Evaluation in Tools

Evaluation for all objectives takes place after place-and-route with Innovus^{TM} as follows:

(1) D_{wc}, worst-case arrival time T_a which is the value of required time T_r minus the worst negative slack (WNS) amongst all path delays. Static timing analysis (STA) is performed at the post-route stage. STA is an important method to validate the timing performance of a design by checking all possible paths for timing violations. Specifically, STA breaks a design down into timing paths, calculates the signal propagation delay along each path, and checks thoroughly for violations (total negative slacks, TNS) of timing constraints inside the design. STA is much more efficient because it is not necessary to enumerate all possible combinations of state and given input vectors to perform full circuit simulation (also referred as dynamic timing analysis) [9].

(2) P_{total}, which is the result from power analysis in Innovus^{TM}. This is an approach that estimates the switching activity of the circuit without running a costly detailed simulation.

$$P_{total} = P_{switching} + P_{internal} + P_{leakage} \tag{5.3}$$

The total power consumption, shown in Equation (5.3), includes three parts: Switching power ($P_{switching}$) consumed in the charging and discharging of interconnect and load capacitance; Internal power ($P_{internal}$), or called short-circuit power, is the power dissipated by an instantaneous short-circuit current flowing between the supply voltage and the ground at the time the gate switches state. Leakage power ($P_{leakage}$) consumed by devices when not switching.

Both internal and leakage power are calculated based on power tables from the Liberty (.lib) file, which contains the specifications and characterisations of the standard cells. Switching power in the power analyser is calculated based on Equation (5.4),

$$P_{switching} = 0.5 * C_L V^2 F * A, \tag{5.4}$$

where C_L is the output capacitive loading, V is the voltage, F is frequency, and A is the average switching activity (the value 0.2 used in this work is the default from Innovus^{TM}).

(3) A_{gate}, which is calculated by adding the areas of each single gate used. This is directly reported by Innovus^{TM}.

All evaluations above are performed on a single mode under typical corner conditions (PVT: TT, 1.2V, 25°C). PVT means process, voltage and temperature. TT refers to both typical corner for PMOS and NMOS.

5.4.3 Multi-threads Running and Runtime

According to the computing resources and licenses, all experiments in this work were run in parallel using 24 threads in an MOEDA run for evaluating individuals.

The evolutionary multi-objective approach requires a larger number of evaluations, which increases the runtime of the algorithm. The majority of runtime is spent on completing place and route. This aims to achieve accurate metrics as close as possible

to sign-off. However, due to the inherent parallelism of the population-based approach, this can be overcome (sped up) using a larger number of licenses and high-performance computing (HPC) resources. In addition, the MOEDA algorithm feature is able to deliver a set of trade-off solutions spanning the feasible design space in one go, rather than a single, case-specific solution. Therefore, the runtime is not the primary concern in this work.

5.5 Multi-objective Optimisation Experiments

5.5.1 Initial Experiments with a Reduced Library

The first step is to quickly evidence the argument that the proposed optimisation method is capable of enhancing designs in the context of an industrial digital flow. To achieve that, initially circuits are implemented using a reduced set of standard cells only including two functions: a two input nand (ND2) and inverters (INV) which are taken from the TSMC library. The aim is to initially reduce the complexity of the problem and to make analysis of results simpler, before moving to the full real-world cell library. In terms of drive strengths of both functions, only the nand gate with the smallest drive strength D0 is included, but inverters feature 11 different drive strengths shown in Table 5.2. The nand gate is a universal gate capable of realising complete overall behavioural function and the various inverters can meet different drive strengths required in the different timing paths.

Table 5.2 A Reduced Experimental Standard Cell Library

Function	Drive Strength
ND2	D0
INV	D0 D1 D2 D3 D4 D6 D8 D12 D16 D20 D24

At first, optimisation is performed exclusively on drive strengths of inverters. Synthesising with only one minimum nand gate is biasing the synthesis tool towards using a

large number of different inverters. This, in turn, creates a larger and richer search space for optimisation with the MOEDA.

In this experiment, all ISCAS-85 benchmark circuits have been implemented and optimised with the proposed MOEDA digital flow using a population size of 200 individuals. Designs are optimised over 400 generations for the largest circuits C5315, C6288 and C7552. For all others, the number of generations is 200. In addition, the output load constraints have not been applied in this case.

The netlist parameterisation is performed on the inverters of the synthesised netlist, converting their drive strength setting into genes for optimisation. The algorithm is seeded with the best tool-generated solution in terms of delay to provide a starting point where PPA over standard flow can be improved from the beginning of the search.

Tables 5.3 and 5.4 present each testing case with its required timing (T_r), the total number of synthesised gates (# Syn Gates), and the total number of genes (# Genes) which are the inverters in this experiment. The Syn-Opt. column is the best solution (using Genus synthesis tool) of each circuit in terms of delay. The general clock is set to 250MHz in order to make the tool deliver working solutions for most of the benchmark circuits as timing constraints are easily met when they are first synthesised. The timing limit of each circuit is found by gradually tightening the timing constraints by $0.05ns$ increments. However, the general clock of C6228 benchmark has to be lowered to 200MHz ($5ns$ clock period), because it fails the $4ns$ clock period even without any further output delay constraints applied.

Under the MOEDA solution in Tables 5.3 and 5.4 there are four columns, where the first three report solutions with the best scenarios in delay, power and area, respectively. Each solution is the best improvement on one of objectives that can be achieved strictly without worsening any others. Column four takes all objectives into account simultaneously and shows the optimised trade-off solution, which is defined here as the individual from the final generation that is positioned at the shortest Euclidean

distance to the origin. The trade-off solutions demonstrate the optimisation capability of achieving improvements in all objectives simultaneously.

The result shows that each circuit can be improved (up to 2.2% in D_{wc}, 35.3% in P_{total} and 26.9% in A_{gate} of C17 circuit trade-off solution) using the MOEDA flow. The runtime of the largest circuit C7552 is about 12 hours. The complexity (in terms of gate count and function) of the benchmarks shown in the tables increases from top to bottom. It can be observed that PPA of the smaller circuits can be improved to a larger degree. The reason for this may be a result of the smaller design space allowing the optimisation to achieve results approaching an exhaustive search. For this reason, the focus will be on the larger benchmark circuits in subsequent experiments.

These initial experiments suggest that the MOEDA flow is a promising and viable approach to tackling multi-objective problems in a standard digital flow. However, with the constraint of only using two types of logic functions, i.e., nand and inverter, it can not make full use of all features of the EDA tool's own optimisation algorithms or the process technology.

Table 5.3 MOEDA design flow using the reduced library for full ISCAS-85 benchmark suite

$N = 200$, $M = 200$, $M^* = 400$, $\rho = 1\%$

Test Case (T_r)	#Syn Gates #Genes	Syn-Opt. Solution	MOEDA Solution			
			Best D_{wc} ($\Delta\%$)	Best P_{total} ($\Delta\%$)	Best A_{gate} ($\Delta\%$)	Trade-off ($\Delta\%$)
C17 ($0.10ns$)	10 3	D_{wc}: **0.092** P_{total}: **1.324** A_{gate}: **18.72**	**0.084** (8.7%) 0.900 14.04	0.090 **0.856** (35.3%) 13.68	0.090 0.856 **13.68** (26.9%)	**0.090** (2.2%) **0.856** (35.3%) **13.68** (26.9%)
C432 ($1.50ns$)	316 138	D_{wc}: **1.459** P_{total}: **37.81** A_{gate}: **401.76**	**1.388** (4.9%) 37.61 401.40	1.444 **35.40** (6.4%) 386.64	1.453 35.55 **385.20** (4.1%)	**1.417** (3.6%) **35.80** (4.7%) **387.00** (2.7%)
C499 ($1.20ns$)	650 214	D_{wc}: **1.167** P_{total}: **112.2** A_{gate}: **944.64**	**1.104** (5.4%) 111.4 942.84	1.167 **105.2** (6.2%) 883.08	1.162 105.9 **880.92** (6.7%)	**1.143** (2.1%) **105.9** (5.6%) **884.52** (6.4%)
C880 ($1.10ns$)	674 243	D_{wc}: **1.019** P_{total}: **89.44** A_{gate}: **875.16**	**0.980** (3.8%) 87.89 874.08	0.993 **86.49** (3.3%) 873.00	1.014 87.32 **871.92** (0.4%)	**0.993** (2.6%) **86.49** (3.3%) **873.00** (0.2%)
C1355 ($1.30ns$)	669 224	D_{wc}: **1.201** P_{total}: **109.5** A_{gate}: **929.88**	**1.159** (3.5%) 0.1093 929.52	1.194 **105.1** (4.0%) 893.88	1.194 105.1 **893.88** (3.9%)	**1.194** (0.6%) **105.1** (4.0%) **893.88** (3.9%)
C1908 ($1.30ns$)	366 168	D_{wc}: **1.281** P_{total}: **86.43** A_{gate}: **749.52**	**1.191** (7.0%) 84.10 729.00	1.248 **81.19** (6.1%) 711.36	1.273 83.13 **709.56** (5.3%)	**1.231** (2.9%) **81.77** (5.4%) **713.52** (4.8%)

Units: D_{wc} $[ns]$ P_{total} $[uW]$ A_{gate} $[um^2]$

Table 5.4 MOEDA design flow using the reduced library for full ISCAS-85 benchmark suite (cont.)

$N = 200$, $M = 200$, $M^* = 400$, $\rho = 1\%$

Test Case (T_r)	#Syn Gates #Genes	Syn-Opt. Solution	MOEDA Solution			
			Best D_{wc} ($\Delta\%$)	Best P_{total} ($\Delta\%$)	Best A_{gate} ($\Delta\%$)	Trade-off ($\Delta\%$)
C2670 (1.00ns)	948 314	D_{wc}: **0.988** P_{total}: **136.0** A_{gate}: **1248.12**	**0.938** (5.1%) 136.0 1246.68	0.967 **133.0** (2.2%) 1247.4	0.969 135.2 **1239.84** (0.7%)	**0.943** (4.6%) **134.3** (1.3%) **1245.6** (0.2%)
C3540 (2.00ns)	1311 478	D_{wc}: **1.890** P_{total}: **275.5** A_{gate}: **1706.04**	**1.809** (4.3%) 268.0 1701	1.809 **268.0** (2.7%) 1701	1.809 268.0 **1701** (0.3%)	**1.809** (4.3%) **268.0** (2.7%) **1701** (0.3%)
*C5315 (1.40ns)	2075 632	D_{wc}: **1.359** P_{total}: **318.7** A_{gate}: **2723.4**	**1.319** (2.9%) 314.0 2719.44	1.354 **311.4** (2.3%) 2718.72	1.354 314.3 **2709.72** (0.5%)	**1.319** (2.9%) **314.0** (1.5%) **2719.44** (0.15%)
*C6288 (4.50ns)	4221 1403	D_{wc}: **4.478** P_{total}: **1946** A_{gate}: **5270.4**	**4.296** (4.2%) 1915 5270.04	4.369 **1911** (1.8%) 5270.04	4.39 1925 **5268.6** (0.3%)	**4.296** (4.2%) **1915** (1.6%) **5270.04** (0.0%)
*C7552 (1.85ns)	2403 753	D_{wc}: **1.700** P_{total}: **438.0** A_{gate}: **3090.24**	**1.652** (2.8%) 435.1 3087.36	1.684 **434.5** (0.8%) 3087.36	1.691 435.7 **3086.64** (0.1%)	**1.652** (2.8%) **435.1** (0.7%) **3087.36** (0.09%)

Units: D_{wc} $[ns]$ P_{total} $[uW]$ A_{gate} $[um^2]$

5.5.2 Experiments with a Full Commercial Library

The proposed MOEDA flow is now scaled up to optimise designs using the full TSMC library. Instead of only adjusting the drive strength of inverters, the designs are synthesised using the full library and the MOEDA is handling drive strength optimisation for all types of logic cells.

The selected three benchmarks from ISCAS-85 suite, in different structures and functions, are a 16-bit error detector/corrector (C1908), a 9-bit ALU (C5315) and a 16x16 multiplier (C6228). One large circuit used from EPFL benchmark suite is an arithmetic function for log2 calculation. The statistics of benchmarks are summarised in Table 5.5. The reason that only combinational circuits were used is all large squential circuits are built from basic combinational blocks, and the optimisation of a sequential circuit will eventually collapse into the optimisation of its combinational parts [115]. In addition, although most gate-sizing related research optimises sequential circuits, they still only manipulate on combinational components [106][109, 108, 110].

Table 5.5 Statistics of benchmarks for MOEDA using the full TSMC library

Test Case	#Inputs	#Outputs	# Primal Gates
C1908	33	25	880
C5315	178	123	2307
C6288	32	32	2406
log2	32	32	32060

In the previous experiment using the reduced library it was possible to efficiently explore the feasible design space starting from a single seed pushed to the timing limit of what the standard tools can achieve. This is no longer sufficient in this case, where the use of the full library causes a dramatic increase in both complexity of the design space and the behaviour of the greedy optimisation algorithms built into the standard flow. In addition, the largest circuit has around 30,000 gates. Each gate has five drive strength options on average, so the search space will be approximately comprised of 5^{30000} alternative solutions. So the MOEA is used to handle the search complexity.

Three different seeds are used here to initialise the MOEDA algorithm, which are obtained from running synthesis and implementation under three different timing constraints for each benchmark: the first (named *a*) is the tightest constraint that can just be met, resulting in a solution with the best delay. In the second case (c), the timing constraint is relaxed so that it can be easily met, allowing the standard flow room to optimise for power and area. The third timing constraint (b) is chosen in the middle between the first and second. The three different solutions obtained will be used as seeds for three independent runs of the MOEDA flow. This aims to investigate how the synthesis tool optimises solutions in trading off PPA metrics when setting different timing goals, and how the MOEDA flow further compensates these tool-generated solutions.

Table 5.6 MOEDA design flow with using the full commercial library

$N = 200$, $M = 200$, $\rho = 1\%$

Test Case	clock (T_c)	(#) T_r	#Syn Gates #Genes	Syn-Opt. Solution	MOEDA Solution		
					Best D_{wc} ($\Delta\%$)	Best P_{total} ($\Delta\%$)	Best A_{gate} ($\Delta\%$)
C1908	250MHz	(*a*) 0.60*ns*	299 299	D_{wc}: **0.580** P_{total}: **222.9** A_{gate}: **1452.96**	**0.569 (1.9%)** 221.9 1451.88	0.580 **211.0 (5.3%)** 1388.16	0.580 211.0 **1388.16 (4.5%)**
		(*b*) 0.76*ns*	178 178	D_{wc}: **0.697** P_{total}: **111.1** A_{gate}: **698.04**	**0.687 (1.4%)** 107.9 682.92	0.688 **107.5 (3.2%)** 682.2	0.696 0.1078 **678.96 (2.7%)**
		(*c*) 1.50*ns*	105 105	D_{wc}: **1.263** P_{total}: **42.1** A_{gate}: **344.52**	**1.234 (2.3%)** 39.69 344.52	1.249 **39.32 (6.6%)** 343.08	1.251 39.51 **342.72 (0.5%)**
C5315	250MHz	(*a*) 0.74*ns*	750 750	D_{wc}: **0.723** P_{total}: **472.9** A_{gate}: **2762.64**	**0.706 (2.4%)** 470.5 2755.44	0.715 **458.9 (3.0%)** 2729.16	0.72 461.2 **2724.48 (1.4%)**
		(*b*) 0.88*ns*	516 516	D_{wc}: **0.824** P_{total}: **310.9** A_{gate}: **1873.44**	**0.805 (2.3%)** 309.0 1869.48	0.819 **304.6 (2.0%)** 1859.76	0.823 305.7 **1852.56 (1.1%)**
		(*c*) 1.50*ns*	400 400	D_{wc}: **1.305** P_{total}: **225.2** A_{gate}: **1346.76**	**1.241 (4.9%)** 222.3 1343.52	1.289 **217.4 (3.5%)** 1343.16	1.302 220 **1336.68 (0.8%)**

Units: D_{wc} $[ns]$ P_{total} $[uW]$ A_{gate} $[um^2]$

Table 5.7 MOEDA design flow with using the full commercial library (cont.)

$N = 200$, $M = 200$, $\rho = 1\%$

Test Case	clock (T_c)	(#) T_r	#Syn Gates #Genes	Syn-Opt. Solution	MOEDA Solution		
					Best D_{wc} ($\Delta\%$)	Best P_{total} ($\Delta\%$)	Best A_{gate} ($\Delta\%$)
C6288	250MHz	(a) 2.34ns	2178 2178	D_{wc}: **2.225** P_{total}: **5509** A_{gate}: **9382.32**	**2.204 (0.9%)** 5495 9364.68	2.206 **5481 (0.5%)** 9377.28	2.204 5495 **9364.68 (0.2%)**
		(b) 2.90ns	1555 1555	D_{wc}: **2.726** P_{total}: **3829** A_{gate}: **6363.00**	**2.673 (1.9%)** 3785 6331.32	2.708 **3732 (2.5%)** 6278.76	2.708 3732 **6278.76 (1.3%)**
		(c) 4.00ns	1140 1140	D_{wc}: **3.591** P_{total}: **2824** A_{gate}: **4194.00**	**3.528 (1.8%)** 2821 4191.84	3.59 **2754 (2.5%)** 4183.92	3.585 2777 **4137.48 (1.3%)**
log2	40MHz	(a) 16.4ns	11838 11838	D_{wc}: **16.355** P_{total}: **19610** A_{gate}: **38547.0**	**15.839 (3.2%)** 19090 38728.1 (-0.5%)	16.24 **19070 (2.8%)** 38797.6 (-0.6%)	16.355 19610 **38547.0 (0.0%)**
		(b) 17.9ns	11272 11272	D_{wc}: **17.751** P_{total}: **18000** A_{gate}: **36623.9**	**17.364(2.2%)** 18000 36840.6 (-0.6%)	17.72 **17890 (0.6%)** 36726.8 (-0.3%)	17.751 18000 **36623.9 (0.0%)**
		(c) 18.8ns	11119 11119	D_{wc}: **18.435** P_{total}: **17590** A_{gate}: **35999.6**	**17.795 (3.5%)** 17510 36227.5 (-0.6%)	18.357 **17390 (1.1%)** 36203.0 (-0.5%)	18.435 17590 **35999.6 (0.0%)**

Units: D_{wc} $[ns]$ P_{total} $[uW]$ A_{gate} $[um^2]$

All circuits are optimised with running 200 generations with a population size of 200 individuals. The number of synthesised gates and the number of genes are the same as shown in Tables 5.6 and 5.7 because all gates are encoded into chromosomes, so that the MOEDA flow is optimising the drive strength of all gates. In terms of the number of synthesised gates in each circuit, it is much less than the number in original benchmarks shown in Table 5.5. The TSMC library has a large range of complex logic cells such as AOI (AND-OR-Inverter), IINR (NOR with 2 Inverted Inputs), full adders, etc., which are already comprised of few basic simple logic gates like XOR, NAND, OR, etc. In contrast, original benchmarks used basic simple generic gates. This makes the synthesis tool to automatically merge the simple gates into complex gates for the total transistor count and physical area reduction, so finally reduce the number of gates.

This may compact the design space and reduce the search complexity but still increases the difficulty of PPA extra optimisation. In real-world libraries, complex logic cells have less options of drive strengths (normally no more than 5) due to the physical design complexity, and a large number of complex cells are used by tools evidenced by the significant decreasing in gate numbers. This may block the optimisation results for achieving huge improvements.

Under such difficulties, the optimised results are still promising compared to the Syn-Opt. solutions which are obtained by running the synthesis tool with "try hard" mode. MOEDA solutions demonstrate significant improvements in most test cases (up to 4.9% in D_{wc}, 6.6% in P_{total} and 4.5% in A_{gate}) as shown in results tables above. The reported improvement of an objective does not (or slightly) sacrifice the metrics of other objectives. Although numbers might be small, even 1% or 2% of improvements could be helpful for designs under tight constraints and particularly when they just fail timing [110].

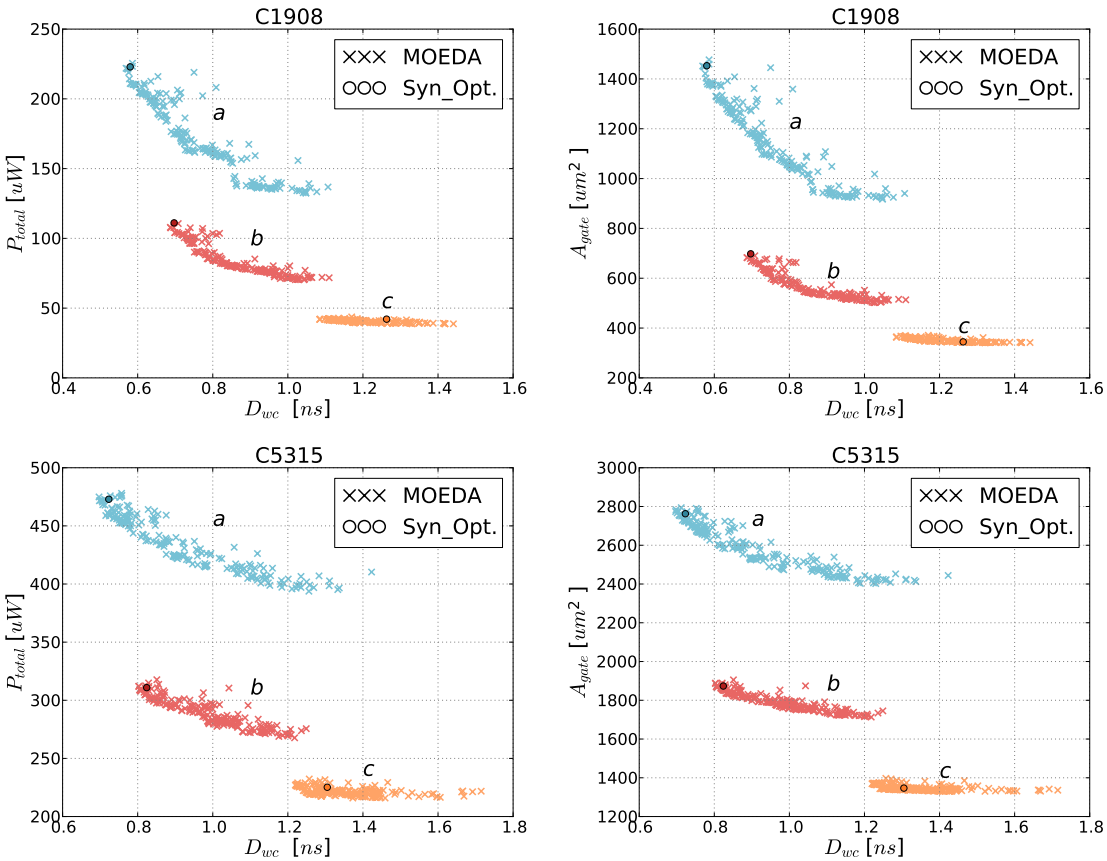


Figure 5.6 MOEDA flow optimisation results using the full TSMC library for C1908, C5315.

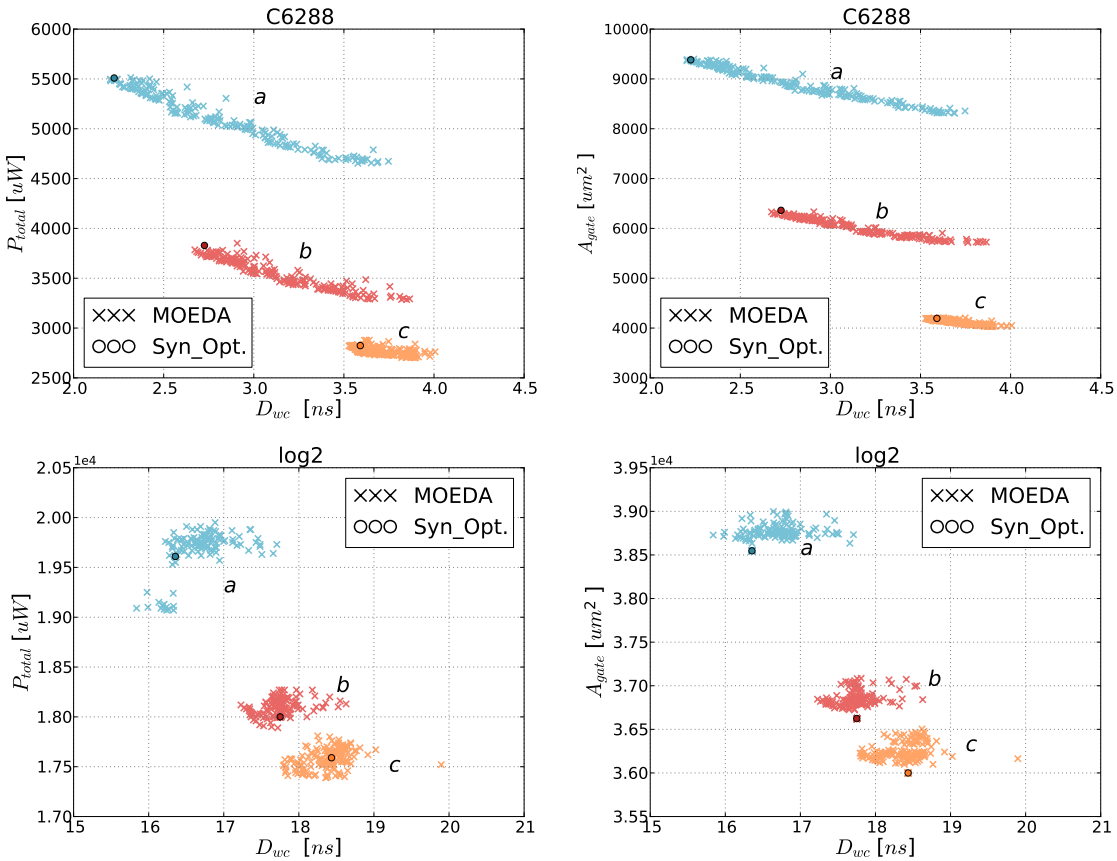


Figure 5.7 MOEDA flow optimisation results using the full TSMC library for C6288 and log2.

In Figures 5.6 and 5.7, the final generation of each circuit with three independent seeding runs is shown, plotting "D_{wc} vs. P_{total}" (left column), "D_{wc} vs. A_{gate}" (right column) and the corresponding Syn-Opt. reference solutions. The three clusters (a, b and c) correspond to the three seed timing constraints, listed in Tables 5.6 and 5.7. In all cases, the MOEDA produces a wide range of useful trade-off solutions, with reduced power consumption or area, within the boundaries of the given seed topology.

From these plots, a number of solutions are improved regarding all objectives in four test circuits. The MOEDA-generated Pareto-driven clusters of C1908, C5315 and C6288 are smooth with good solution spreads, whereas the log2 circuit's is not. This is because the algorithm in MOEDA flow needs to handle the increased size of design, where larger EA parameters (the number of generations M and population size N) are required for producing Pareto-driven results. The improved performance of log2 circuit is still promising and considerable in power and delay objectives under such an optimisation run with using the same EA parameters as other smaller test cases used. This implies that standard digital flow is also struggling to produce well trade-off solutions for a relatively larger design, so that the MOEDA flow has more optimisation room to get improved solutions run with relatively less iterations and smaller population.

From smaller cases of C1908, C5315 and C6288, the tool's performance can be further observed when different constraints are applied. For timing settings corresponding to clusters a and b, the tool is operating under tight timing requirements, causing the synthesis tool to spend the most effort on timing closure and less on power and area, so the MOEDA flow does not achieve significant improvements on delay (but much more trade-offs with less power and area). However, for relative relaxed timing settings corresponding to clusters c, the tool does not make the solution trade-off on timing too much but spend more efforts on power and area, where the MOEDA flow enhances the solution particularly in timing. It can be concluded that the MOEDA flow demonstrates the capability of balancing these three objectives to a greater extend while tools have not.

Furthermore, as the circuit size increasing, the improvement of area is hard to be achieved (particularly in log2). This explicitly shows that area optimisation needs to include tuning the circuit structure (i.e., reducing gate count or transforming between complex and simple gates) instead of only focusing on drive strength refinement. But it is still worthwhile to take the area as one of objectives in the optimisation, which otherwise may have much degradation on area when optimising other objectives.

The runtime for the largest case optimisation (log2.a) needs 138 hours. Although the proposed optimisation method is at the cost of longer computing time, this investment will be worthwhile when considering the enhancements in delay and savings in power consumption or area that could not otherwise be achieved, particularly for feasible circuit solutions that are produced in large numbers.

5.5.3 Statistics of MOEDA Flow Convergence

In this subsection, a statistical evaluation is performed for the proposed MOEDA flow, which specifically run the NSGA-II algorithm ten times using the same population size ($N = 200$), generation count ($M = 200$) and mutation rate ($\rho = 1\%$). The selected test case is C5315 circuit with its tightest timing constraint (a). Figure 5.8 shows how the PPA metrics (i.e., D_{wc}, P_{total}, A_{gate}) have been optimised and converged with the evolutionary process. Experiments confirm that the algorithm reliably converges to similar performance when run the MOEDA flow multiple times with the same evaluation budget. Slight variations (around 2% of each objective) are shown between different runs due to the inherent randomness of MOEAs.

Since the focus here is not on investigating the statistical analysis of MOEAs, the algorithm is run once for each benchmark to manage runtime.

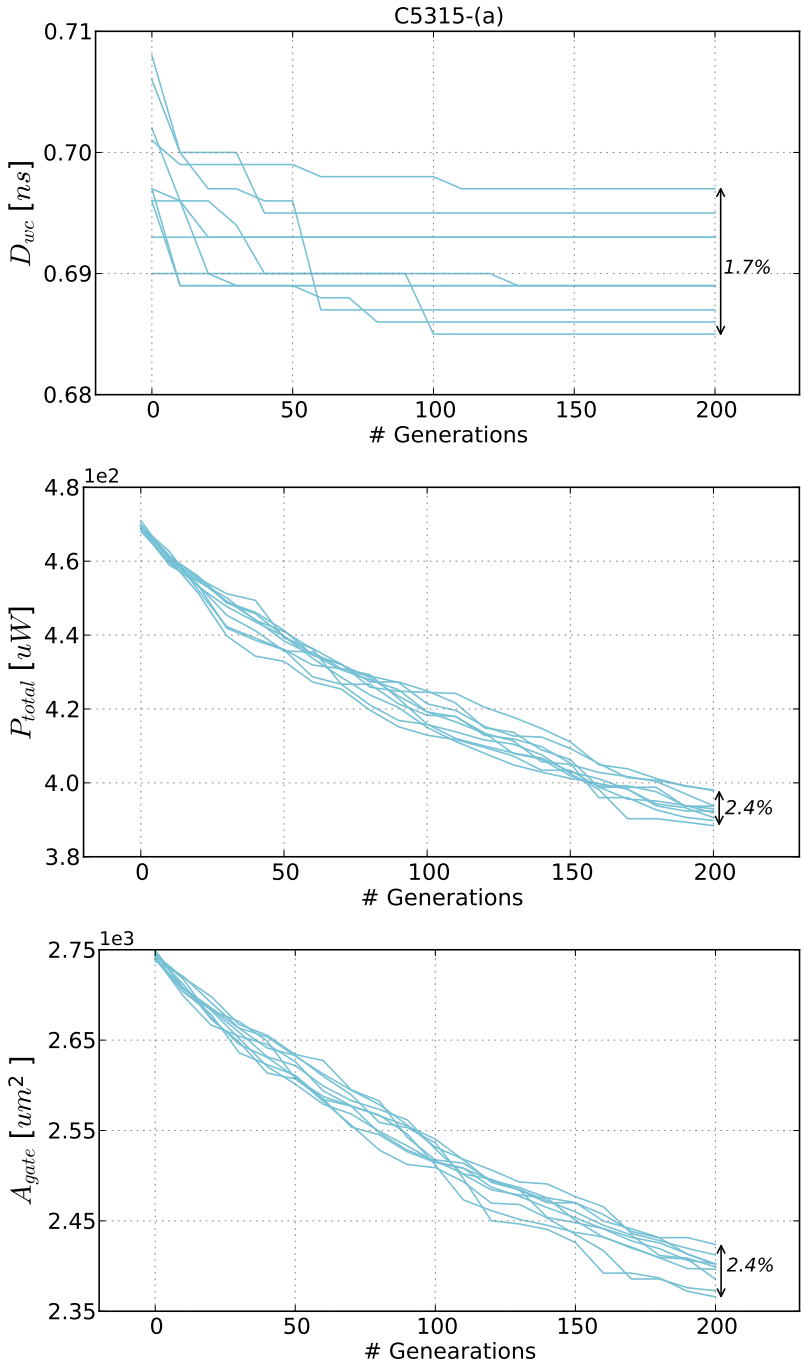


Figure 5.8 Statistics of MOEDA convergence. The annotations show the variations between different runs.

5.6 MOEA Search vs. Stochastic Search

This section performs an optimisation efficiency comparison between MOEA search and stochastic search. The selected test case is C5315-(a), a 9-bit ALU with a tight timing constraint (a). For MOEA search, the NSGA-II is initialised using 1% mutation rate in 200-individual population size for 200 generations, so 40000 evaluations are generated in total. The MOEA optimisation results (red cluster) shown in Figure 5.9 are seeded with the tool-optimised Syn_Opt solution (black round scatter).

Two stochastic search experiments are then run here for comparison. The first one, referred to a local stochastic search (grey cluster shown in Figure 5.9), is to randomly produce 40000 individuals seeding with the same Syn_Opt solution, and each of them is achieved by randomly mutating the chromosome using the same probability. The second one is completely randomised results referred to a global stochastic search, which produces 40000 individuals seeding with the same Syn_Opt solution but all genes (i.e., drive strength of logic gates) of each individual are modified (100% mutation rate). The results of global stochastic search are the blue cluster in Figure 5.9.

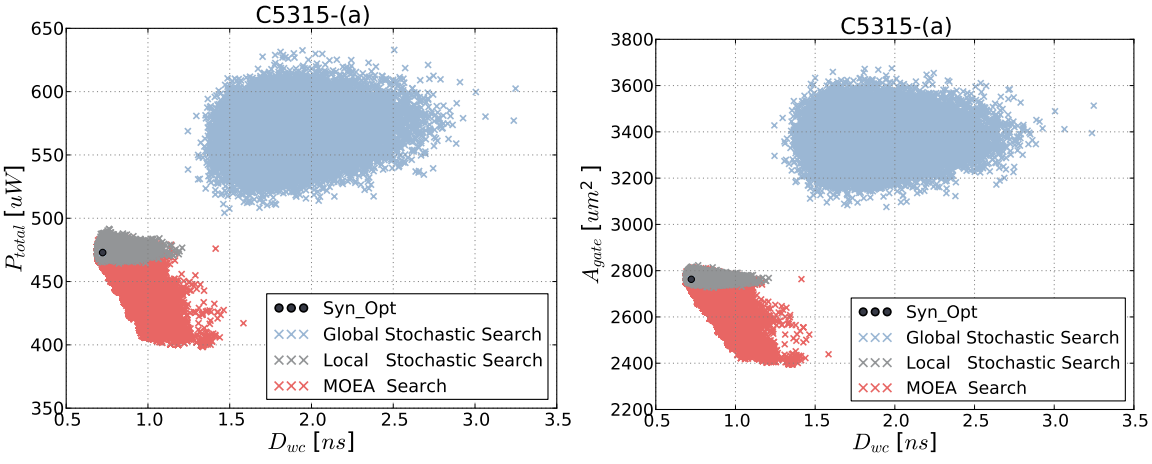


Figure 5.9 MOEA search compared to two stochastic search.

Based on the observations made from the plots, it demonstrates that MOEAs have superior optimisation performance than the stochastic search when evaluation budgets

are the same. Since the focus of this thesis is not on investigating which MOEA is the best to achieve the optimum results in VLSI design optimisation, only NSGA-II is used for experiments.

5.7 Summary

This chapter proposes a fully-automated multi-objective electronic design automation (MOEDA) flow extension to enhance the current industry-standard logic synthesis and physical implementation process. The MOEDA flow is fully compatible with commercial design tools and specifically optimises drive strength of gates during technology mapping in such a way that the subsequent physical implementation stage can achieve designs with better PPA metrics. The proposed method has been successfully applied to the optimisation of ISCAS-85 benchmark suite and a large circuit from EPFL benchmark set using TSMC 65nm low power standard cell library.

The MOEDA framework is the main novel contribution which exploits more optimisation opportunities within the standard digital flow. Experimental results show that the proposed MOEDA flow has operated design optimisation gaining significant improvements on PPA over the standard design tool's solutions. It can be concluded that optimising technology mapping to refine drive strength selection of cells is beneficial to the critical performance of VLSI circuits.

In addition, optimising circuits for a given timing constraint with one circuit topology solution (single seed) is not capable enough to offer a larger design space when circuit structures are changing. Therefore, the next chapter will investigate how running synthesis multiple times can be harnessed to expand, access and explore the design space with respect to different circuit topologies.

Chapter 6

Design Space Exploration in Large-scale Designs

6.1 Overview

The experiments in Chapter 5 confirm that MOEDA framework is able to assist standard flows to more efficiently produce trade-off solutions. However, this only has been achieved through seeding the MOEA with a single synthesis-optimised solution (single topology). Such an optimisation is not compatible with various circuit topologies to offer a larger coverage of optimised design space.

To deal with this issue, this chapter performs multi-objective design space exploration (MODSE) of VLSI designs, which allows the MOEA to handle circuit optimisation with multiple solution seeds (i.e., different topologies). Such an approach also has been applied to tackle the complex physical floorplan constraints.

The specific contributions are summarised as follows: 1) An analysis of tool-generated design space comprised of a set of initial solutions with different circuit structures and different performance. 2) A methodology to seed the MOEA with these initial solutions to more efficiently sample the entire feasible design space for better PPA metrics and design space coverage. 3) An investigation of how different floorplan constraints affecting the overall results of VLSI designs during physical implementation. 4) The application of the proposed methodology of seeding the MOEA with a solution population to effectively handle the most complex physical floorplan constraint.

The following sections in this chapter are structured as: Section 6.2 gives the related work review on design space exploration using standard digital flows. Section 6.3 introduces the MODSE approach and the experiment setup is presented in Section 6.4. Section 6.5 gives an analysis of the design space generated by standard digital flow. The optimisation results of MODSE are demonstrated in Section 6.6. Section 6.7 provides summary of this chapter.

6.2 Design Space Exploration using Standard Digital Flow

Design space exploration (DSE) aims at searching for feasible solutions in the objective space (i.e., often multiple dimensions). Optimisation using the steps of a standard digital flow from RTL to GDSII in the loop can be viewed as black-box design space exploration. While many of the algorithms used in commercial EDA flows are proprietary and not accessible by end users, logic synthesis and physical design tools provide a range of parameters and optimisation options for designers to choose from such as logic reconstruction, area constraints, synthesis effort level, place and route with timing or power optimisation, etc. These parameters can be tuned with an optimisation method like machine learning or any other intelligent auto-search approaches to fully utilise the optimisation potential that the tools are capable of.

Efficient design space exploration approaches promise to simultaneously balance multiple design objectives. As stated in Chapter 3, MOEAs are often used for DSE at the VLSI architecture level and up to high level synthesis. However, within the full digital flow, alternative intelligent techniques have been adopted such as automated design-parameter tuning [118–120] and machine learning [121–123].

A. Kahng presented in [120] that there is unpredictable "noisiness" in tool-generated solutions causing variability in the resulting PPA metrics. Multi-armed bandit (MAB) sampling strategies, a probability theory, was applied in a fully-automated digital flow, which aims to determine the optimal utilisation (parameter settings) of EDA tools to "de-noise" the design results. In [119], Ziegler proposed an automated method to explore the search space via tuning parameters at the synthesis step for multi-objective optimisation. This is a rank-based iterative process using an aggregate cost function composed of the normalised weighted sum of PPA metrics.

Running through the whole design flow leads to more computing resource consumption. To improve the exploration efficiency, in [118], the authors exploited an automated

selection mechanism based on searching the design space in parallel. Generated solutions were evaluated at each step of the digital flow and those that were not competitive in terms of PPA were pruned early rather than being propagated through the entire design flow. This approach can improve search efficiency and save computing time. More recently, Kwon in [123] introduced a learning-based "recommender system" for auto-tuning design flows, whereby a previously trained learning network suggests what parameter settings would be beneficial for a specific design. This can dramatically save exploration and runtime but gathering and archiving useful and sufficient training data is still time consuming. Circuits from which training data could be derived are often proprietary, which represents another major obstacle to creating a sufficiently large and useful training data set.

Growing design complexity and aggressive technology scaling make it hard to evaluate realistic, post-fabrication metrics without running the design flow through to the physical level. In [121] [122], machine learning approaches were employed to bridge the synthesis solution space to the physical solution space, with the goal to enable Pareto-driven exploration for high speed and power efficient adder designs. In their design evaluation they used a weighted sum cost function. The results showed that learning approaches could trade-off solutions in terms of PPA and find Pareto frontiers.

It is challenging to minimise computational effort in the process of design space exploration while balancing multiple design objectives due to the large number of evaluations necessary. Accurately predicting results of designs can dramatically save the effort of evaluations where the implementation and simulation time could be reduced or even completely removed. Machine learning based methodologies are promising for prediction but this requires a high prediction accuracy of learning algorithms [124] [125]. Even a small margin might result a significant variation in overall implemented design results, and also achieving appropriate training data set is difficult. These further leads to a wide research domain beyond the scope of this study.

6.3 Multi-obejctive Design Space Exploration Flow

6.3.1 Algorithm

The NSGA-II including its fast non-denominated sorting approach and diversity preservation scheme will be considered here to address the problem of design space exploration. Only mutation operator is adopted in this case. The used notations about the EA setup still are that: N is population size; M is the maximum of generations; ρ is the mutation rate. The adapted NSGA-II algorithm for MODSE will be introduced in the next subsection.

6.3.2 MODSE using Multiple Seed Designs

The multi-obejctive design space exploration (MODSE) method is altered from MOEDA framework (introduced in Chapter 5). Instead of starting with seeding the initial population using a single synthesis-optimised solution, how the proposed algorithm can explore the design space simultaneously using a set of multiple different seeds is investigated. The seeds are a range of different solutions (circuit topologies) generated using the standard digital flow under a number of different timing constraints.

Therefore, in this case, a more fine-grained range of timing constraints are applied in 100 increments from minimum (a constraint that the tool can easily meet) to maximum (solutions start to fail timing) in order to investigate how the standard tools deal with the different constraints and what design space coverage they can achieve. Each benchmark has been synthesised once for each timing constraint setting to generate the 100 solutions for seeding. The tool-generated design space (with enabling synthesis optimisation) comprised of 100 seed solutions, in different circuit topologies, is the baseline for the MODSE to perform optimisation on.

With regard to the specific execution steps of MODSE, the majority of the steps are same as in the MOEDA framework, excepting the MOEA seeding (MOEA initialisation)

step that installs an initial population using 100 seeds. This allows the algorithm to evaluate and select the best-suited seeds (circuit topologies) for breeding the optimised design space. Figure 6.1 illustrates the MODSE flow using seed population.

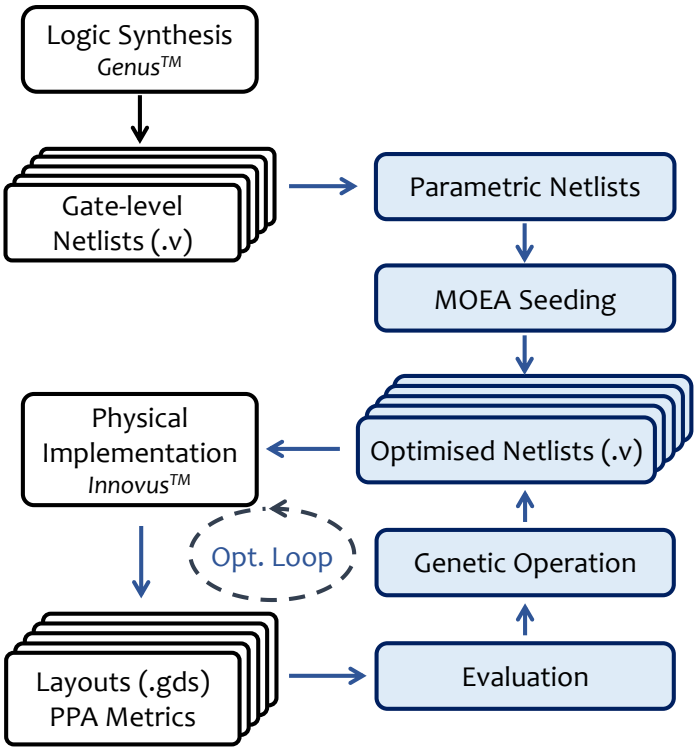


Figure 6.1 MODSE flow using multiple circuit topology seeds.

The optimisation objectives and fitness function $f(\mathbf{g})$, shown in Equation (6.1) are the same that simultaneously minimise the worst case delay (D_{wc}), total power consumption (P_{total}) and all logic gate area (A_{gate}).

$$\begin{aligned} f(\mathbf{g}) = \min \quad & [D_{wc}(\mathbf{g}), \quad P_{total}(\mathbf{g}), \quad A_{gate}(\mathbf{g})] \\ \text{s.t.} \quad & \mathbf{g} = (g_1, ..., g_i), \quad \forall g_i \in \mathbb{G} \end{aligned} \tag{6.1}$$

The algorithm 4 introduces MODSE flow in detail.

Algorithm 4 Adapted NSGA-II for MODSE

Procedure: NSGA-II (N, M, $f(\mathbf{g})$). ▷ N individuals evolved M generations to solve $f(\mathbf{g})$. The vector $\mathbf{g} \in$ the cell library $\mathbf{G}$, is the chromosome consisting of a set of genes (logic gates) defining an individual.

1: Initialize parent population $\mathbf{P}_t$ in size N ▷ Seed with 100 different solutions (circuit topologies) generated by the standard flow.
2: Offspring population $\mathbf{Q}_t \leftarrow$ Mutation($\mathbf{P}_t$)
3: **for** $t \leftarrow 1$ to M **do**
4: **for** each population $\mathbf{R}_t \leftarrow \mathbf{P}_t \cup \mathbf{Q}_t$ in size $2N$ **do**
5: Fitness evaluation $\leftarrow f(\mathbf{R_t})$ ▷ Call fitness function $f(\mathbf{g})$ for each individual evaluation.
6: $\mathbf{F} \leftarrow$ Non-Dominated-Sorting($\mathbf{R}_t$)
7: $\mathbf{P}_{t+1} \leftarrow \emptyset$
8: $i \leftarrow 1$
9: **while** $|\mathbf{P}_{t+1}| + |\mathbf{F}_i| \leq N$ **do**
10: Crowding-Distance-Assignment($\mathbf{F}_i$)
11: $\mathbf{P}_{t+1} \leftarrow \mathbf{P}_{t+1} \cup \mathbf{F}_i$
12: $i \leftarrow i + 1$
13: **end while**
14: $\mathbf{F}_i \leftarrow$ Descend-Sort($\mathbf{F}_i$)
15: $\mathbf{P}_{t+1} \leftarrow \mathbf{P}_{t+1} \cup \mathbf{F}_i[1 : (N - |\mathbf{P}_{t+1}|)]$ ▷ Less crowded individuals from the first to the $(N - |\mathbf{P}_{t+1}|)$th of $\mathbf{F}_i$ to fill $\mathbf{P}_{t+1}$.
16: $\mathbf{Q}_{t+1} \leftarrow$ Mutation($\mathbf{P}_{t+1}$)
17: **end for**
18: **end for**

6.4 Experimental Setup

The proposed MODSE methodology is implemented in C++ and the experiments are conducted on a 2.2GHz Xeon E5-2650 CPU. Three selected benchmarks from ISCAS-85 suite [114] and a large design from EPFL benchmark suite [115] are used in this chapter. All benchmarks are implemented and optimised using the Cadence® digital flow suite. The experimental circuits in the form of RTL designs are synthesised into gate-level netlists using Genus™ (v17.11) [116]. These netlists are then optimised using the proposed flow in tandem with the physical implementation tool Innovus™ (v17.11) [117] to generate the layouts from the optimised netlists. The versions of

EDA tools used here represented the most up-to-date flow including full optimisation licences when experiments were performed.

All experiments are using a TSMC 65nm low-power core cell library (TCBN65LP), i.e., **G** in algorithm 4, in standard threshold voltage containing about 400 combinational cells.

6.4.1 Tool Environment Setup

Table 6.1 summarised specific settings for both logic synthesis and physical layout implementation, which are the same as the settings in Chapter 5. However, in the place and route process, the die area is initially shaped in the ratio of 1.0, but it will be customedly reshaped later in the experiment to investigate how different floorplan settings will affect the design results.

Table 6.1 Design Constraint and Tool Settings in Digital Flow

<table>
<tr><th>Synthesis Setup</th><th>Place & Route Setup</th></tr>
<tr><td>syn_generic_effort = high
iopt_ultra_optimisation = true</td><td rowspan="3">aspect ratio = 1.0
core utilisation = 0.7
timing-driven placement = true
timing-driven routing = true
SI-driven routing = true</td></tr>
<tr><td>Design Constraint</td></tr>
<tr><td>set_load = D1/D8
create_clock = 40MHz/250MHz</td></tr>
</table>

The set of timing constraints for multiple synthesis runs (to achieve the 100 seed solutions) is obtained through gradually increasing the output delay constraint (T_{od}) with a constant increment factor. The required time (T_r) equals to clock period (T_c) minus T_{od}. The arrival time (T_a) of the critical path of a design thus has to meet the required timing constraint (T_r) for achieving the timing closure.

The design constraints also cover the different clock frequencies (create_clock) and output load capacitance (set_load) applying to various benchmark circuits. These will be specified in the following experiments. The benchmarks used are combinational circuits, so that the ideal clock was not applied with any uncertainties or transition delays.

6.4.2 Objective Evaluation in EDA Tools

The fitness function is to simultaneously minimise all objectives: D_{wc}, P_{total} and A_{gate}. All specific evaluations take place after place-and-route with Innovus^{TM} based on typical corner conditions (PVT: TT, 1.2V, 25°C), which is same as in Chapter 5.

(1) D_{wc}: The arrival time of critical path calculated at the post-route stage by static timing analysis engine in Innovus^{TM}.

(2) P_{total}: The result from the average power analysis in Innovus^{TM}. It is the sum of leakage power, internal power and switching power.

(3) A_{gate}: The sum area of all logic gates and it is directly reported by the Innovus^{TM}.

6.4.3 Multi-threads Running and Runtime

According to the computing resources and number of licenses available, all experiments in this work are running 24 MODSE evaluation threads in parallel.

The runtime is still not the key focus in this work. The MODSE flow needs more computing resources due to the continuous generation of design layouts in a large quantity. This aims for accurate and real-world evaluation. It is easily to speed up the flow through making design evaluations at an earlier design stage, but what is investigated in this work is comprehensively evidence the proposed MODSE flow has generic optimisation capability in an industrial design environment.

6.5 Analysis of Tool-generated Design Space

This section firstly investigates the tool-generated design space. The selected benchmarks from ISCAS-85 suite, in different structures, are a 16-bit error detector/corrector (C1908), a 9-bit ALU (C5315) and a 16x16 multiplier (C6228). One large circuit from EPFL benchmark suite is an arithmetic function for log2 calculation.

Table 6.2 Test Case Summary

Test Case	Function	clock (T_c)	T_r (Increment Factor)	set_load	#Syn Gates #Genes	#Orig. Gates
C1908	16-bit error detector/corrector	250MHz ($4ns$)	$1.50ns - 0.51ns$ ($0.01ns$)	D1	105 - 445	880
				D8	105 - 468	
C5315	9-bit ALU	250MHz ($4ns$)	$1.50ns - 0.51ns$ ($0.01ns$)	D1	396 - 1323	2307
				D8	401 - 1287	
C6288	16x16 multiplier	250MHz ($4ns$)	$4.00ns - 2.02ns$ ($0.02ns$)	D1	1105 - 3208	2406
				D8	1123 - 3222	
log2	log2 calculation	40MHz ($25ns$)	$25.00ns - 15.10ns$ ($0.10ns$)	D1	10801 - 12561	32060
				D8	10797 - 12555	

Table 6.2 summarises specific timing constraint settings for each test case including the clock period (T_c), the required timing (T_r) range with output delay increment factor when tightening the timing. So the 100 different timing requirements determine the 100 seeds in the synthesis tool. Different output load scenarios, including loading with drive strength D1 and D8, are applied to all test cases under the same set of timing constraints. The output load values (D1 and D8) are specified as the input pin capacitance of inverters with drive strength D1 and D8 from the TSMC TCBN65LP cell library. The reason for selecting D1 and D8 as output loads is that D1 load is a nominal scenario in practice and D8 load with larger capacitance is the middle sized one from all available inverters.

The number of synthesised gates (i.e., also the number of genes for the MOEA) are presented from minimum to maximum of corresponding synthesised netlists. The original gate number of each test case stated in the benchmark suite report is also included in Table 6.2. The number of synthesised gates in most test circuits is much

less than the number in their original benchmarks. This is because the TSMC library has a large range of complex logic cells such as AOI (AND-OR-Inverter), IINR (NOR with 2 Inverted Inputs), full adders, etc. In contrast, original benchmarks used basic generic gates like and, not, or, etc. The synthesis tool is able to automatically merge the simple gates into complex ones. This is also consistent with the results in Chapter 5 Section 5.5.

In case of C6288, the number of synthesised gates is beyond its original number when tight timing constraints are applied. The likely reason is that the structure of C6288 (16x16 multiplier) requires a large number of adders to form an "array-like" topology. So the synthesis tool finds it hard to merge them, and more large drive strengths need to be mapped with large buffers inserted in circuit paths against stringent timing constraints.

Figures 6.2 and 6.3 illustrate the standard tool's design space for each benchmark circuit under D1 (left column) and D8 (right column) output load scenarios. From plots, all cross markers represent tool-generated solutions in "D_{wc} vs. P_{total}" and their face colours correspond to the colour bar relating to the area objective A_{gate} ranging from large (red) to small (blue). Solutions additionally marked with squares have failed to meet timing constraints. The red line highlights the synthesis optimised (Syn Opt.) "elite" solution front, which is calculated using the non-dominated sorting approach in three dimensions with regard to D_{wc}, P_{total} and A_{gate}. All solutions in the first domination rank are connected with a line to highlight the "Syn Frontier" more clearly. The "Syn Frontier"s shown in the figures are projections from the 3D objective space onto the 2D plots.

Looking at the design space of the standard flow, it can be observed that the 16-bit error detector/corrector (C1908), the 9-bit ALU (C5315) and the log2 calculation circuit can be synthesised and optimised well by the tool as the set of solutions forms a relative smooth Pareto frontier.

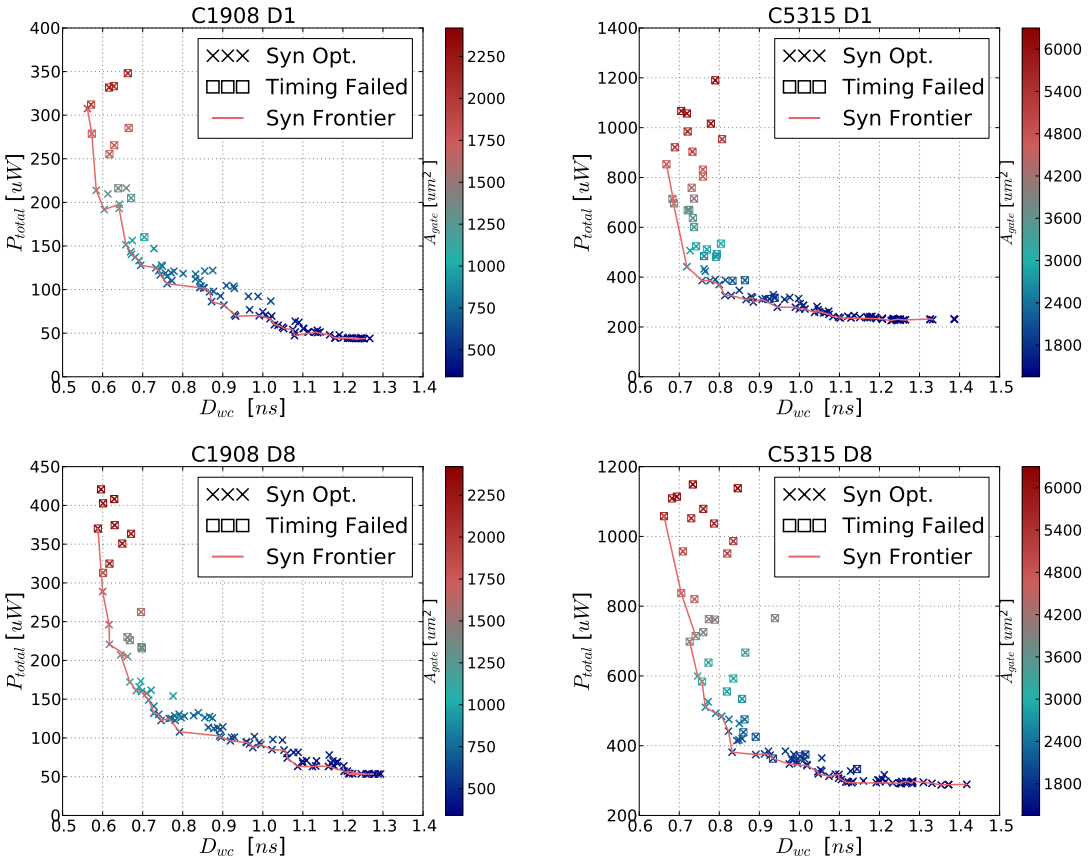
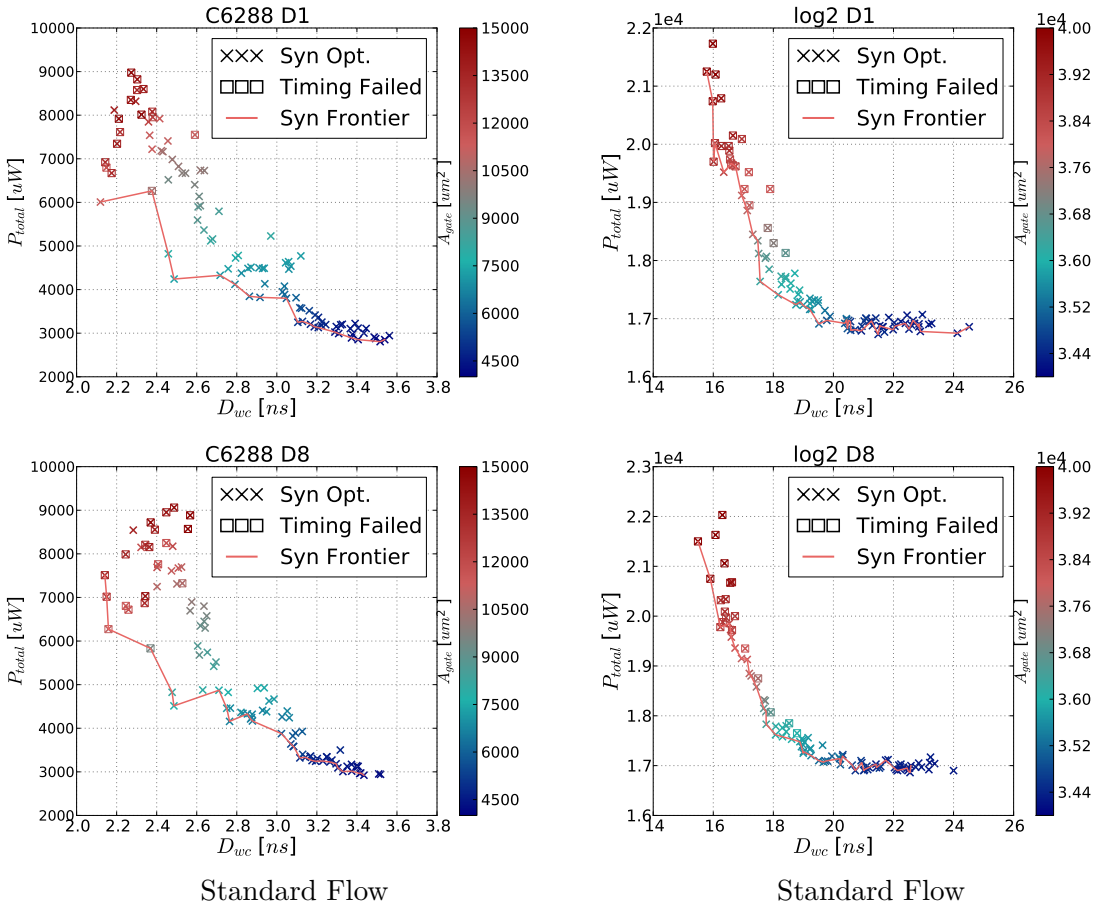


Figure 6.2 The tool-generated design space under the drive strength D1 and D8 output load scenarios for C1908 (16-bit error detector/corrector) and C5315 (9-bit ALU).

However, the 16x16 multiplier (C6288), which is a highly structured circuit, yields a less regular frontier with more clustered solutions. This indicates that the standard tools struggle to effectively trade-off multiple objectives when optimising a complex design with a relatively fixed circuit topology. So the design might not gain sufficient optimisation at the logic synthesis step.

6.5.1 Performance Variation in Synthesis Tool

In terms of the timing closures in tool-generated design space, some solutions start to fail timing when synthesising with tight constraints. To investigate this, for each



Standard Flow Standard Flow

Figure 6.3 The tool-generated design space under the drive strength D1 and D8 output load scenarios for C6288 (16x16 multiplier) and log2 calculation circuit.

benchmark under D8 output load scenario, 30 solutions are selected constrained by the tightest timing out of the primal 100 tool-generated ones. The performance (i.e., D_{wc} - worst case delay) of these 30 solutions (in blue crosses) are plotted with corresponding required timing T_r in Figure 6.4.

From these plots, the T_r is gradually tightened from right to left on the x-axis and the timing failed solutions are further marked with square. The circuit delay D_{wc} of tool-produced solutions should have been dropped when tightening the timing constraint. However, it can be observed that the synthesis tool starts delivering solutions with significant variations in terms of the circuit delay D_{wc} when applying strict constraints. The circuit delay D_{wc} of two adjacent solutions is often quite distinct from each other

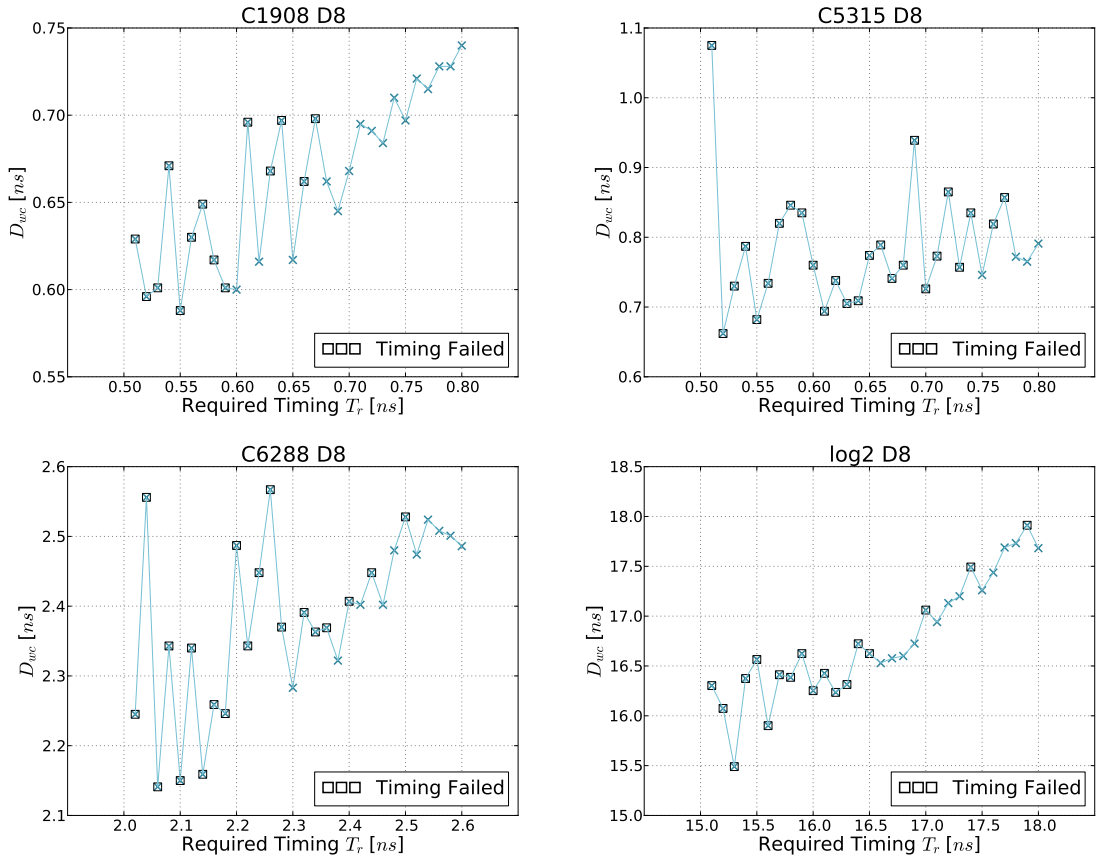


Figure 6.4 The variation investigation inside of EDA synthesis tools is presented by plotting the worst case delay D_{wc} of tool-generated solutions with corresponding timing constraints T_r.

although the required timing T_r difference between them is minor. These indicate the unpredictability or "noisiness" of tool outcomes, which is consistent with the conclusion in [120]. In such a situation, designers run a given flow for multiple times with small perturbations to constraints or other initial conditions, in order to take the best resulting solution forwards in the subsequent design steps. This might result in a time-consuming design cycle and it is hard for designers to obtain a globally-optimum solution.

6.6 Multi-objective Design Space Exploration

6.6.1 Squeeze Design Space for PPA Optimisation

The design space generated by standard digital flow is the baseline for the MODSE engine to perform global optimisation on. The 100 seed designs are loaded into the initial population of the MODSE flow and optimised with the evolutionary process. All test cases are optimised over 100 (M) generations using a population size N of 500, so the initial population comprises five copies of each seed circuit.

The plots in Figures 6.5 (C1908), 6.6 (C5315), 6.7 (C6288) and 6.8 (log2) show the improved solution space, plotting "D_{wc} vs. P_{total}" and "D_{wc} vs. A_{gate}". The red line presents the "Syn-Frontier"s from the baseline design space (tool-generated). All those seed solutions that have survived until the last generation, although with modified drive strengths, are marked with a red cross. The solutions shown as blue crosses are those produced by the MODSE flow comprising of all individuals of the final generation.

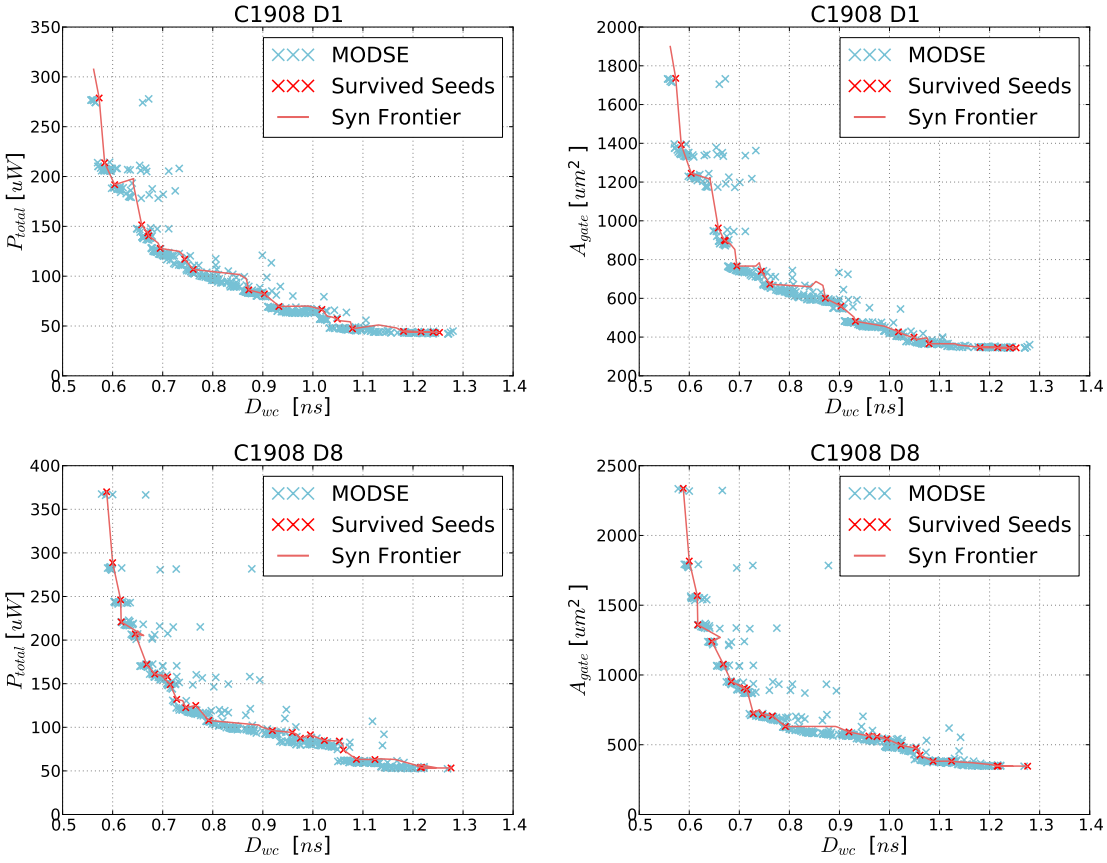


Figure 6.5 Design space optimisation results under the drive strength D1 and D8 output load scenarios for C1908 16-bit error detector/corrector. $N = 500$, $M = 100$, $\rho = 1\%$.

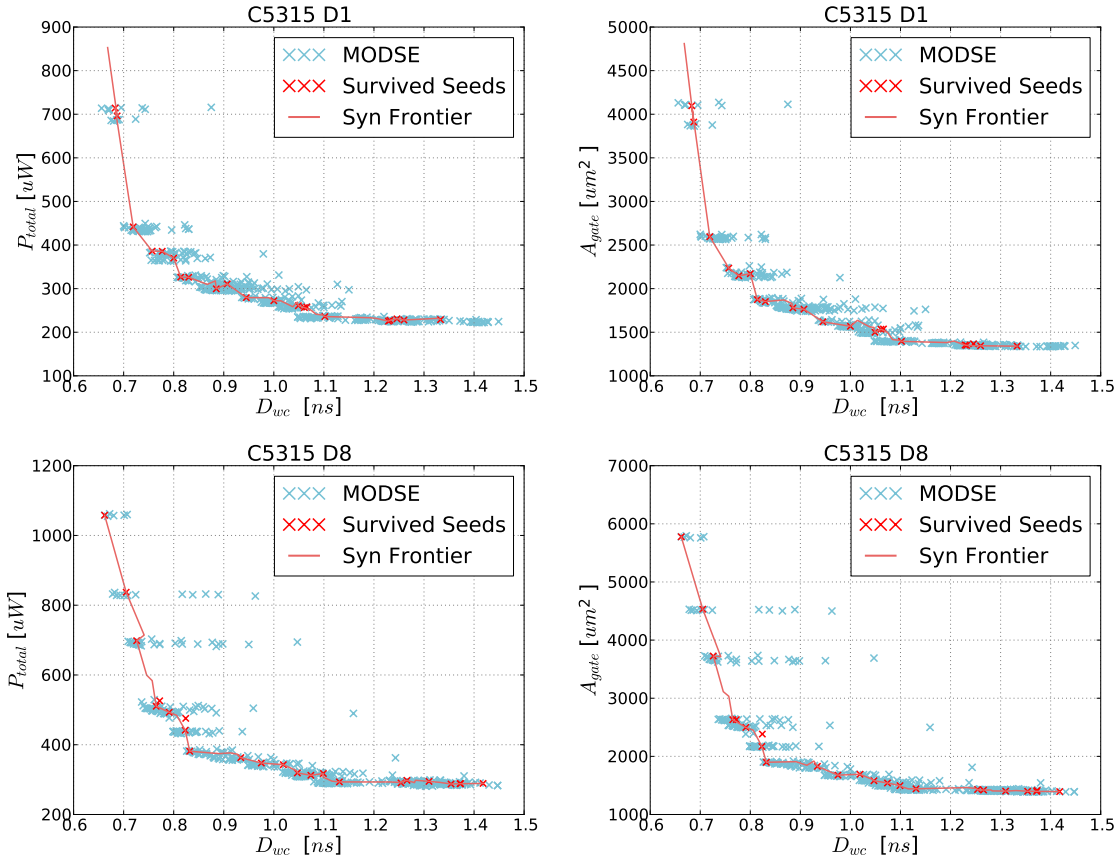


Figure 6.6 Design space optimisation results under the drive strength D1 and D8 output load scenarios for C5315 9-bit ALU. $N = 500$, $M = 100$, $\rho = 1\%$.

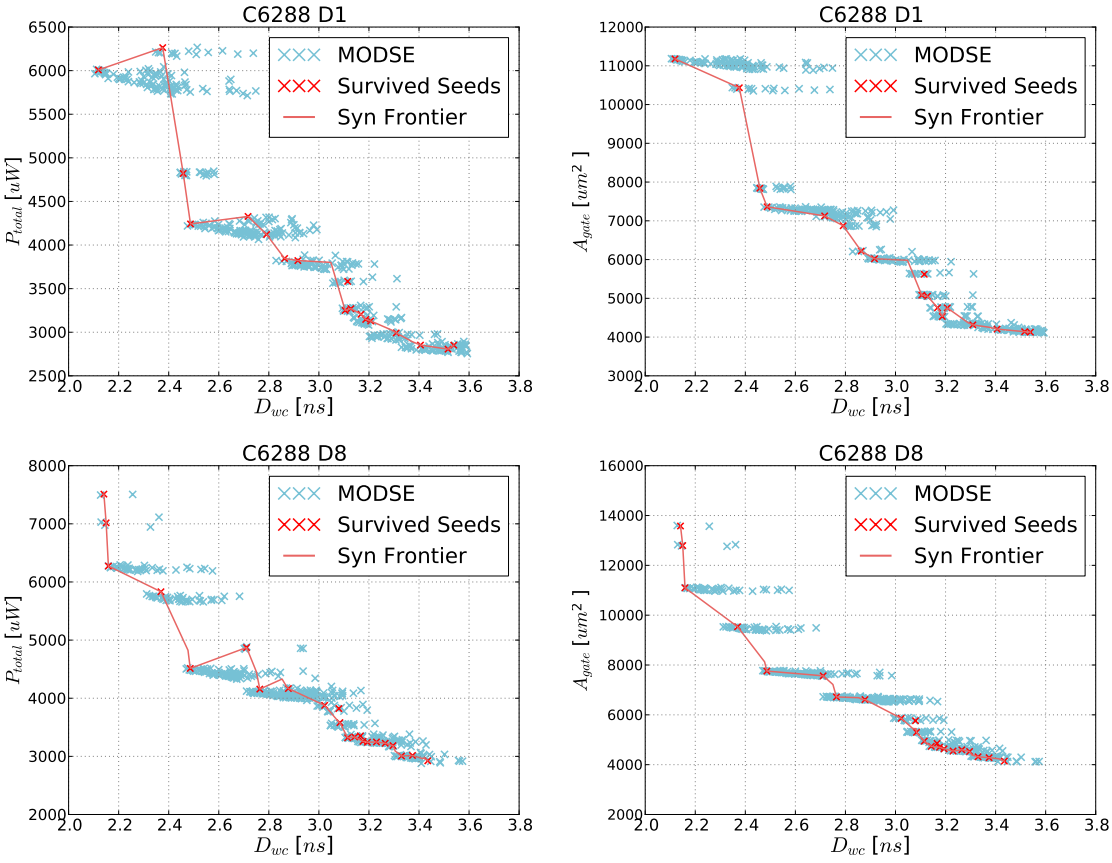
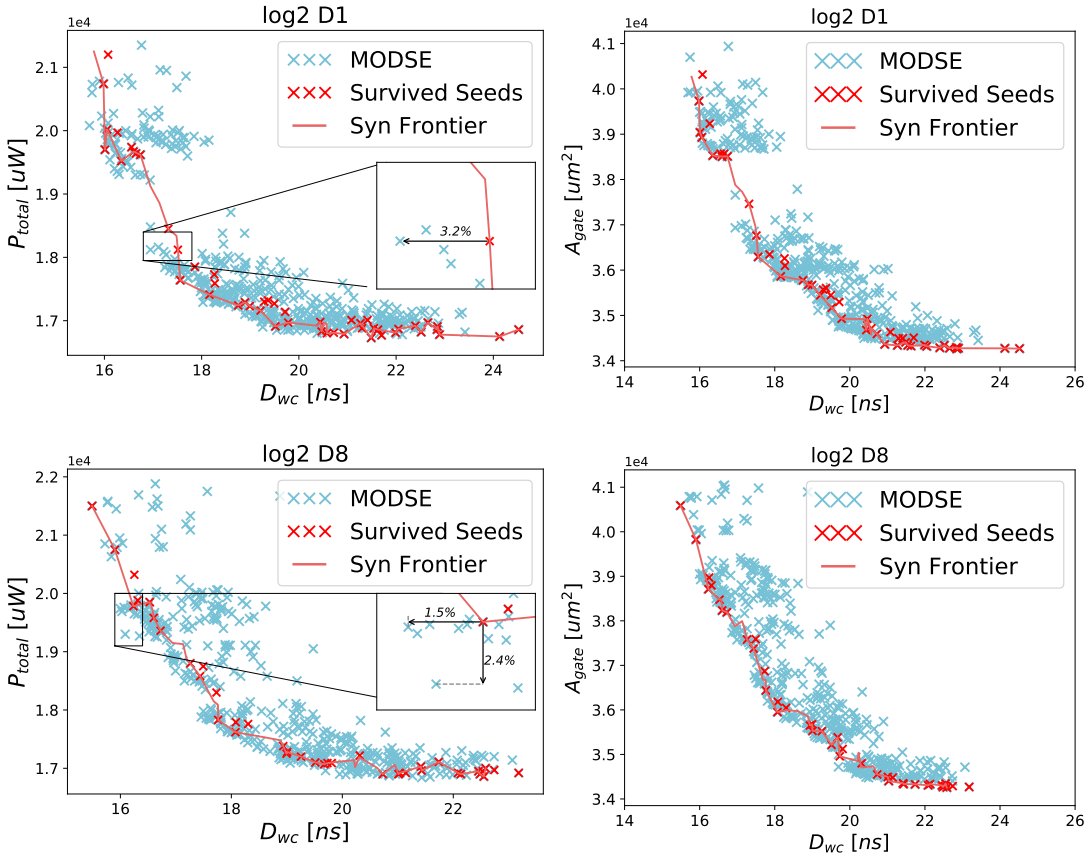


Figure 6.7 Design space optimisation results under the drive strength D1 and D8 output load scenarios for C6288 16x16 multiplier. $N = 500$, $M = 100$, $\rho = 1\%$.



MODSE Flow MODSE Flow

Figure 6.8 Design space optimisation results under the drive strength D1 and D8 output load scenarios for log2 calculation circuit. $N = 500$, $M = 100$, $\rho = 1\%$. The runtime of largest case (log2.D8) is 162 hours. The optimised design space of log2 with D1 and D8 loads is shown with zoom-in views to present the improvements clearly.

The results confirm that the MODSE flow can push the baseline frontier further to extend the design space of all test cases in all three objectives, across different circuit topologies. For the largest circuit log2, the optimised design space is shown with additional zoom-in views to present the quantified improvements which are still considerable. Furthermore, the relative improvement looks marginal from the plots due to the wide axis range, but the total absolute values for saved power and improved delay are significant.

In the case of circuit C1908, the optimised solutions that form a smooth Pareto frontier, whereas there are some gaps in the optimised design space of C5315, log2 and particularly of C6288 (see Figure 6.7). The gaps are artefacts from the baseline design space due to limitations of the tool's optimiser and properties of the circuit. Although the proposed MOEDA flow could not fully bridge these large gaps, it has been achieved that the optimised design space covers the baseline design space and beyond more uniformly. This makes better choices for design-specific using as a richer set of solutions is available.

Only about one-fourth of the initial seeds survive until the final generation in design C1908, C5315 and C6288, and about half of the initial seeds survive in log2 circuit until the final generation after applying MODSE. Most of the surviving seeds are positioned on the "Syn-Frontier"s, while others have been discarded in the evolution process. This further indicates that there is "noisiness" inside of standard flow tools and not all solutions generated by tools are presumably optimised, potentially losing some of good solutions. So design iterations within the flow with human engineer efforts are often required to obtain possible best solutions. The MODSE flow can auto-iterate designs without throughout the whole flow for better trade-offs in PPA metrics.

In the case of log2 calculation circuit (see Figure 6.8), the optimised solution cluster covers a larger space, where there a number of dominated solutions are behind the Pareto front. This is because the log2 circuit is comprised of more logic gates than other benchmarks, which requires more computing time for MODSE against the enlarged

design space. This also reveals the optimisation limit of exploited MODSE which is its optimisation scalability.

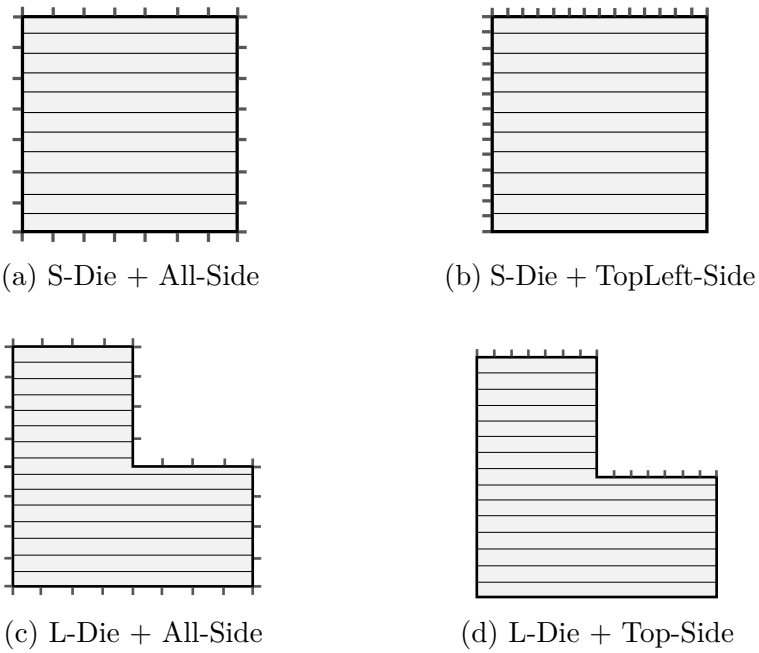
Although the limitations of the proposed MODSE is shown, the optimisation improvements are still considerable compared to the primal tool-generated design space.

6.6.2 Squeeze Design Space for Constrained Floorplan

In all prior experiments, the physical die is always shaped in a square without considering any placements of hard blocks, so the whole physical die is only utilised to place and route standard cells. In modern system-on-chip (SoC) design, the overall physical die is normally in a rectangle shape but consists of few hard blocks been placed first. The remaining limited core space then becomes a non-regular polygon. In addition, the input/output (I/O) pin positions are often constrained by the fixed location of hard blocks since some of I/O pins are required to connect with them. Some dedicated pins therefore should be placed as close to certain blocks to reduce routing wirelength.

In this set of experiments, the proposed MODSE method is employed to optimise the design space specifically when circuits need to be implemented to fit in a non-regular polygon die with constrained pin places. A generic function benchmark C5315 (9-bit ALU) is selected as the test circuit. Figure 6.9 conceptually presents four different floorplan settings. The first, (a) "S-Die + All-Side", represents square physical die with randomly placed pins at all sides. This is also the setup for all prior MODSE experiments. The others are (b) "S-Die + TopLeft-Side": square physical die with randomly placed pins only at top and left sides; (c) "L-Die + All-Side": "L" shape physical die with randomly placed pins at all sides; (d) "L-Die + Top-Side": "L" shape physical die with randomly placed pins only at the two top sides.

The area of "L-Die" is set to be the same as the "S-Die" 's and the core utilisation keeps consistent to 70%. This can ensure the congestion of the place and route is unchanged. The set of experiments in this case aims to investigate how the performance



(a) S-Die + All-Side (b) S-Die + TopLeft-Side

(c) L-Die + All-Side (d) L-Die + Top-Side

Figure 6.9 Four study cases: MODSE optimisation with different physical die shapes and pin location constraints.

of synthesised solutions will be in physical implementation when applying different floorplan settings.

The tool-generated design space of C5315 with the floorplan setting of "S-Die + All-Side" under the drive strength D8 output load scenario is presented in Figure 6.2.

For other floorplan settings, the same synthesis constraint setup is used, as shown in Table 6.3, in terms of clock (T_c), required timing (T_r) and output capacitive load (set_load). All tool run settings for logic synthesis and physical implementation are kept consistent for all experiments in this chapter (shown in Section 6.4 Table 6.1). So the synthesis outcomes (gate-level netlists) will not be changed because the extra floorplan constraints is only set for physical implementation.

Apart from the case of (a) "S-Die + All-Side", three new sets of seeds (i.e., still 100 for each) are then obtained through running standard digital flow for the cases (b),

Table 6.3 Design Constraint Setup for Different Floorplans

Test Circuit	Common Settings			No.	Die Shape	Pin Location
	clock (T_c)	T_r (Increment Factor)	set_load			
C5315	250MHz ($4ns$)	$1.50ns - 0.51ns$ ($0.01ns$)	D8	(a)	S-Die	All-Side
				(b)	S-Die	TopLeft-Side
				(c)	L-Die	All-Side
				(d)	L-Die	Top-Side

(c) and (d). The evaluation metrics are still the worst case delay (D_{wc}), total power (P_{total}) and the sum of all gate area (A_{gate}).

Four tool-generated design space is achieved. Each corresponding "Syn Frontier" (i.e., Pareto frontiers) is presented in Figure 6.10, plotting "D_{wc} vs. P_{total}" (left) and "D_{wc} vs. A_{gate}" (right). The "Syn Frontier" lines represent the "elite" solutions in the first domination rank of each design space. These are calculated by the non-dominated sorting with regard to all objectives.

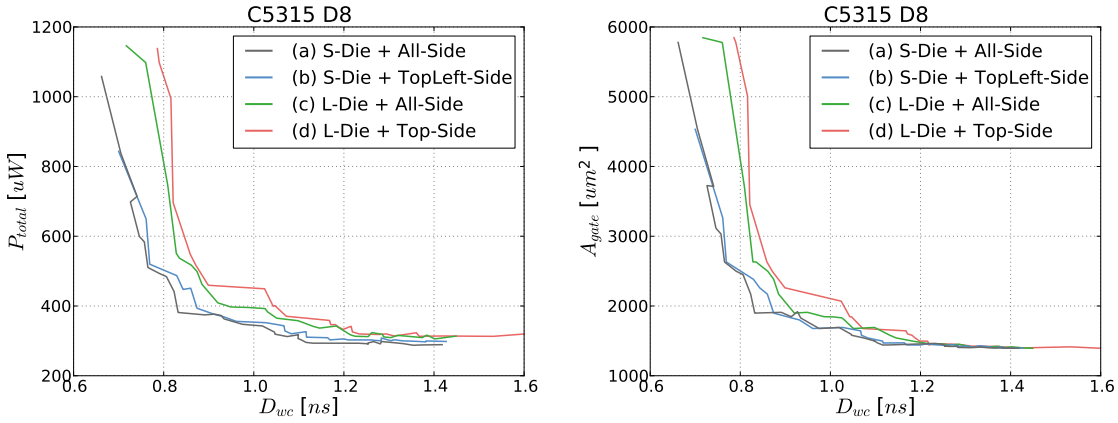


Figure 6.10 The "Syn Frontier"s of tool-generated design space of all study cases are illustrated.

From plots, it can be observed that the restricted die shape and pin locations pose a serious effect on the quality of results (QoRs) of the design. In addition, the non-regular polygon die has a greater negative influence on the design than the restrictions from pin locations. This indicates the physical implementation tool is struggling to fit the synthesised design (logic gates) into further constrained floorplans.

Based on the observation, the worst case in QoRs is the (d) "L-Die + Top-Side" (red line). This is also the baseline where the MODSE performs optimisation on. So the tool-generated 100 seeds from case (d) are loaded in MODSE for expanding design space. The optimisation runs with N - 500 individuals for M - 100 generations. The mutation rate ρ is still kept for 1%.

The MODSE-optimised outcome of the case (d) in terms of all objectives are shown in Figure 6.11. The "Syn Frontier" of the best case, (a) "S-Die + All-Side", is included for comparison. The light blue scatters, representing the expanded design space, demonstrate distinct improvements in PPA and better coverage beyond the baseline design space (red line), although the MODSE method has not been able to completely closed the gap from the worst case (d) to the best case (a).

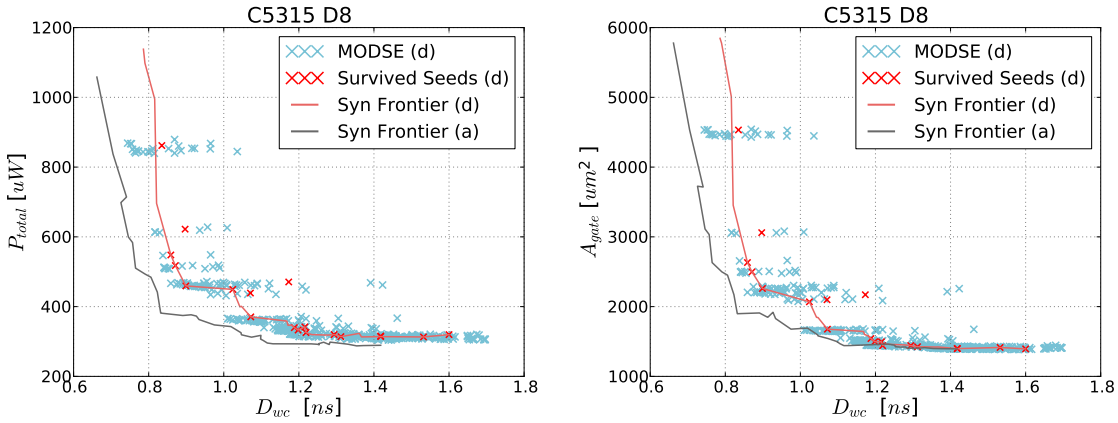


Figure 6.11 MODSE is applied to optimise the design space of case (d) "L-Die + Top-Side" generated by standard tools. The expanded design space (in light blue scatters) are plotted in "D_{wc} vs. P_{total}" (left) and "D_{wc} vs. A_{gate}" (right). The survived seeds after applying MODSE are shown as well. MODSE algorithm settings are $N = 500$, $M = 100$, $\rho = 1\%$.

6.6.3 Discussion

The experimental results confirm the capabilities of the MODSE flow in producing better design space coverage with significant PPA improvements (up to 3.2% in worst-

case delay in the largest log2 circuit) over the entire primal tool-generated design space. This has not only been demonstrated with a set of benchmark circuits covering different structures and functions, but has also been evidenced when designs are further constrained at physical flooplan step with the non-regular polygon die and restricted pin locations.

However, although the proposed method is capable of exploiting design opportunities by adjusting drive strengths at the gate-level netlists, circuit topology optimisation is currently not yet included. This current limitation is likely the reason that the MODSE-produced design space gaps cannot be fully closed, which would provide potential optimum trade-off design choices. It is particularly visible in the results of C6288 circuit, due to is fixed topology. From these results it can be concluded that including circuit toplogy modification in the MODSE approach could enable further design optimisation opportunities.

6.7 Summary

In this chapter, a fully-automated multi-objective design space exploration (MODSE) method is introduced to augment the standard-flow-generated design space. The proposed method optimises the design across various circuit topology solutions in terms of power, performance and area (PPA), and significant improvements with better design space coverage have been achieved.

The MODSE method is invented based on the MOEDA flow which also tunes drive strength mapping for logic gates, but the MODSE is able to handle different tool-generated circuit structures with considering different practical floorplan settings. The proposed method has been successfully applied to the optimisation of ISCAS-85 benchmark circuits and a large circuit from EPFL benchmark suite using the TSMC 65nm low power standard cell library.

The next chapter will perform an optimisation focusing on library level to enrich drive strength granularity for foundry standard cells. It is expected to present better final results (PPA) at physical layout level than using original drive strength granularity.

Chapter 7

Improved Drive Granularity Standard Cells

7.1 Overview

As discussed in Chapter 2, in a standard cell library, the provided drive strength options of a logic cell are limited and therefore of relatively coarse granularity and range. Although limiting and discretising drive options accelerates cell selection to handle modern, large complex designs fast, an optimum scenario would be that EDA tools could select cells of exact drive to meet load requirements thereby avoiding over-design in terms of power and area. Methods like improving drive strength resolution in adjacent most commonly used cells (i.e., most selected by standard tools) can potentially improve designs particularly for lowering power [21] [126]. The authors in [29] investigated drive granularity on small sizes for leakage power minimisation in planar MOSFET technology. The work in [127] performed exhaustive FinFET sizing to provide all possible drive options for few basic logic gates to achieve better PPA metrics in physical implementation.

Seeking to achieve richer cell libraries, implementing logic designs using mixed-height (i.e., routing tracks such as 9-track and 12-track) or double-row-height standard cells are recently proposed [128–131]. Smaller-height cells feature compact area and lower power dissipation, but are weaker in drive strength. Cells with a larger height provide higher cell drive capabilities, but consume more area and power. Mixing different-height cell libraries, available from foundries, is an alternative efficient approach to achieve richer drive options. However, current EDA tools cannot directly handle the mixed-height cell placement legalization so that dedicated place and route tools need to be developed for each case. Interpolating fine-grained drive strength of logic gates based on an existing cell library and inserting them to expand the original granularity can be straightforwardly implemented in standard tools. This approximates circuit optimisation close to transistor-level, although it might still require custom-design effort, but can ensure the design legalization for fabrication.

This work first investigates how to create standard cells featuring fine-grained drive options and an interpolation methodology is proposed. Two sets of custom-designed

standard cells, one in normal and another in fine-grained drive strength granularity, are then developed using a commercial 65nm technology to demonstrate the benefits of using fine-grained cells for logic circuits.

In addition, enriched standard cell libraries (larger cell quantity) lead to increasingly difficult logic synthesis for producing well-optimised technology-mapped netlists. The MOEDA optimisation framework is used to assist tools to further trade-off solutions in PPA when synthesising designs with an improved drive granularity library.

The remaining parts of the chapter are structured as follows: Section 7.2 illustrates the design of fine-grained drive strength library. Section 7.3 provides an introduction of adapted MOEDA framework in this chapter. Experiment setup is described in Section 7.4. Section 7.5 demonstrates experimental results and Section 7.6 outlines the summary of this chapter.

7.2 Drive Strength Design of Standard cells

7.2.1 Logic Design using Multiple Driving Options

To have multiple drive strengths for logic functions in a cell library is crucial to achieve timing closure. It is normally indicated using a post-fix after a cell function name, such as X1, X2, X3, etc., in a library. This can provide various drive capabilities to meet different loads when building real circuits at the physical level. Using larger drive strength cells generally consumes more electrical power, die area and pin capacitance, but it is able to drive larger loads or to speed up the circuit clock frequency.

Synthesis tools are mapping drive strength from a finite set of discrete drive options for a generic functional gate, and usually needs to select the smallest possible one to minimise power consumption and area while trying to meet the timing constraint simultaneously. Hence, a limited number of coarse-grained cell drive strengths will inherently lead to over-design. For example, when a wire delay corresponds to a drive

equivalent of X1.5, and choices available are only X1 and X2, then X2 would be selected in order to meet timing requirement [126]. Improving drive granularity of cells, such as adding some intermediate-sized cells X1.2, X1.5, X2.5, etc., thus could avoid this problem.

Industry-standard cell libraries are designed in a proprietary way. This limits design optimisation to whatever drive strength options are available in a given library to effect design-specific transistor sizing when implementing designs in the digital flow. Pre-designing a richer set of drive strengths for each functional cell can approach a more ideal scenario where the selected drive strength can meet required loads in a more accurate way. However, this would require a huge manual design effort for fab designers when creating cell libraries, and it is too expensive and time consuming to make libraries larger. This is because they have already contained 600-1000 logic gates typically in a library, and the design effort will be further increased significantly since different library versions, regarding threshold voltages, cell heights, supply voltages, are required be created.

In [29], it proposes using different drive strength compositions but keeping the original library resolution (i.e., the total number of drive options of each logic gate is fixed) to minimise leakage power consumption especially when circuits operating at relative lower clock frequency or in the sleep mode. This work particularly brings more smaller drive strength which are less than the typical drive strength X1.

However, limited research to date investigates how the synthesis tools deal with the different drive granularity of cells and how this would affect the final results of digital circuits.

7.2.2 Improved Drive Granularity Library Design

The new proposed design methodology for improving cell drive resolution is to interpolate custom-designed cells into the original library in the middle of two cells

with adjacent drive strengths. Instead of generating a large number of cells with fine-grained drive strength, this method aims to expand drive granularity of cells based on a well-optimised industrial library, and all newly produced cells are aligned with the original library in terms of logic cell drive capabilities.

Here, the TSMC 65nm technology is used, but its pre-designed standard cell library (TCBN65LP) including schematics and full layouts is proprietary and therefore unavailable. Hence, in order to create a representative test case for the 65nm technology used, a reduced library is firstly initialised including 11 inverters (INV) which have the same drive strengths as those in the TSMC TCBN65LP library, and one nand logic function (NANDX0) with minimum drive strength (transistors are of smallest width). This re-designed cell library that is modelled to match the original drive granularity of the commercial library is named "MINI_ORIG".

Subsequently, a set of inverters of more fine-grained drive strengths are interpolated into "MINI_ORIG" to form another library named "MINI_FINE". Both custom-designed libraries and their drive granularity are summarised in Table 7.1. To focus the investigation on the drive strength selection and simplify the problem, only drive strength expansion of inverters is considered in this case.

Table 7.1 Contents of Each Experimental Cell Library

Library Name	Functions	Inverters (INV)
MINI_ORIG	NANDX0	X0, X1, X2, X3, X4, X6, X8, X12, X16, X20, X24
MINI_FINE	NANDX0	X0, X0.5, X1, X1.5, X2, X2.5, X3, X3.5, X4, X5, X6, X7, X8, X10, X12, X14, X16, X18, X20, X22, X24

To define the drive strength of a gate needs to be based on its performance evaluation (i.e., the speed to drive a load capacitance). For example, if the X1 can drive a unit load capacitance C_{unit_load} in a period time T_{unit_load} (i.e., circuit delay), the X2 needs to be designed through iterative transistor-sizing until it can drive double unit load capacitance $2 \times C_{unit\_load}$ taking a near-exact same time T_{unit_load}. So the definition of drive strength:

$$X = \frac{C_{X\_load}}{C_{unit\_load}} \quad \text{s.t.} \quad T_X = T_{unit\_load} \tag{7.1}$$

In addition, the transistor size of drive strength X1.5 in "MINI_FINE" library is defined when it can drive the $1.5 \times C_{unit\_load}$ in the same time T_{unit_load}. The following drive strengths in both "MINI_ORIG" and "MINI_FINE", such as X2.5, X3, X3.5, X4, etc., are all created using the same approach.

The inverter drive strength X1 is defined by $PMOS_{size}$ = 230nm and $NMOS_{size}$ = 165nm, so the P/N ratio adapted in this work is 1.39 for all cells. The X0 cell is defined by the minimum-sized $NMOS_{min}$ and $PMOS_{size} = 1.39 \times NMOS_{min}$ according to the minimum design rules from the technology. The X0.5 inverter is then interpolated in the middle between X0 and X1 through transistor-sizing until it can drive a load capacitance value in the middle between X0 and X1's in T_{unit_load}. All created cells keep the same transistor length 60nm.

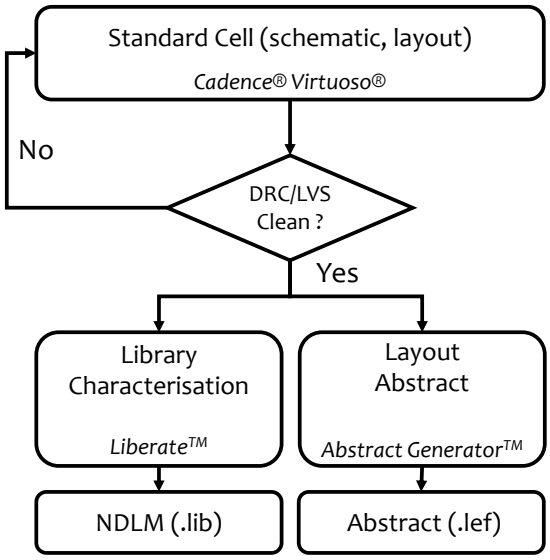


Figure 7.1 Standard cell design flow including library characterisation and layout abstract.

Both custom-designed "MINI_ORIG" and "MINI_FINE" libraries are implemented including schematics and layouts using Cadence® Virtuoso® [101]. All library cells are

designed in body tapped structure. The cell layouts are characterised respectively into timing and power models (Liberty file) and physical abstractions (LEF file, top layer view of layouts) using Cadence® Liberate™ [132] and Abstract Generator™ [133] tools. The standard cell design flow is illustrated in Figure 7.1.

The Non-Linear Delay Model (NLDM) is a 2D look-up-table-based model containing timing and power information of each gate. The two input indexes given are *input slew* and *load capacitance*. The output index is the circuit's delay or power under different compositions of the input slew and the load capacitance to separately produce delay and power look-up tables. Both delay and power are evaluated through a series of SPICE-based simulations run by the Liberate characterisation tool.

The Liberty (.lib) file contains two main parts: the first contains the technology library including all environment descriptions such as operating conditions, wire load mode, etc., and the second contains the cell descriptions obtained by running the library characterisation tool. The technology library, in this case, is using the typical corner (PVT: TT, 1.2V, 25°C) of TSMC65nm technology, which is the same as the TCBN65LP library uses, and the environment descriptions are kept the same as well. In addition, a 7x7 look-up-table NLDM is used for cell descriptions and characterisation input index values are inferred from the original TCBN65LP library.

The input slew in this case is a fixed range where the input signal transition ranges from a close-ideal step to a larger slew time. The same input slew set is used for all cells. Furthermore, each designed drive strength has a specific capacitive load set ranging from a small to large capacitance value. The inverters in "MINI_ORIG" library use the corresponding load capacitance indexes from the TCBN65LP library, but the load capacitance index for each fine-grained inverter is found through calculating the middle (or average) value of two adjacent cells' load capacitance indexes.

The characterised information of each gate includes both timing and power consumption tables. The timing tables of each gate include cell delay (i.e., measured from 50% to 50%) and transition time (i.e., measured from 30% to 70% in this case). Both the cell

delay and transition time are specified during the characterisation. Hence, four tables per input pin of a logic gate are generated, including cell rise, cell fall, rise transition and fall transition. The power consumption information in NLDM includes two parts: internal power and leakage power. The internal power, or called short-circuit power, is the power dissipated by an instantaneous short-circuit current flowing between the supply voltage and the ground at the time the gate switches state. The power dissipation table describes each cell's internal power consumption as the combination of energy consumed by output and input pin transitions with respect to a given clock frequency. The values provided represent the amount of energy consumed (in uW/MHz or pJ) within the cell when the corresponding output pin state changes. Input pin energy consumption is included to increase accuracy of estimated power consumption, where the consumed energy value is measured for each input pin toggle while output pin state remains unchanged. In order to obtain the internal power consumption, the consumed energy needs to be considered with a clock frequency applied in the EDA tool's power analysis. The average/minimum/maximum leakage power values are provided in nanowatts (nW) for immediate use.

7.2.3 The Performance of the Proposed Libraries

To verify whether the proposed libraries are designed in an appropriate way, and particularly the fine-grained drive inverters are properly interpolating into the original granularity, the delay and power of each gate are analysed to determine the relationship between two adjacent drive strengths and the overview of all cells.

Table 7.2 shows the transistor count, cell width and leakage power (i.e., specifically contains minimum, average and maximum values) of each cell for both "MINI_ORIG" and "MINI_FINE" libraries. Based on the characterisation results, the leakage power increases linearly when the transistor width increases with each drive strength. Since the transistor sizes of inverters X0, X0.5 and X1 are close to each other, the minimum leakage power is close to each other and does not increase as the transistors size up.

The likely reason is the inherent variation of transistors. All cells are created at the same height 1.8um, so the cell area also increases as the cell width increases from small to large drive strength, as transistors of increasing size need to be accommodated. Drive strengths X0 and X0.5 have the same width as X1 due to constraints of the physical design rules.

Table 7.2 Library Cell Information

PVT: TT, 1.2V, 25°C

Cell Name	Transistor Count	Cell Width [um]	Leakage Power [nW]		
			Min.	Ave.	Max.
INVX0	2	0.6	0.0106	0.0108	0.0111
INVX0.5	2	0.6	0.0114	0.0117	0.0120
INVX1	2	0.6	0.0112	0.0133	0.0153
INVX1.5	2	0.7	0.0184	0.0225	0.0265
INVX2	4	0.8	0.0282	0.0297	0.0311
INVX2.5	4	1.0	0.0388	0.0406	0.0423
INVX3	6	1.2	0.0511	0.0516	0.0522
INVX3.5	6	1.3	0.0627	0.0637	0.0646
INVX4	8	1.4	0.0704	0.0731	0.0759
INVX5	10	1.6	0.0894	0.0948	0.1002
INVX6	12	1.8	0.1070	0.1167	0.1264
INVX7	14	2.2	0.1301	0.1441	0.1581
INVX8	16	2.4	0.1497	0.2575	0.1853
INVX10	20	3.0	0.1931	0.2191	0.2451
INVX12	24	3.4	0.2325	0.2673	0.3022
INVX14	28	4.0	0.2766	0.3201	0.3636
INVX16	32	4.4	0.3165	0.3686	0.4207
INVX18	36	5.0	0.3605	0.4223	0.4841
INVX20	40	5.6	0.4044	0.4760	0.5475
INVX22	44	6.2	0.4482	0.5295	0.6109
INVX24	48	6.6	0.4885	0.5790	0.6695
NANDX0	4	0.8	0.0028	0.0152	0.0303

Figure 7.2 and Figure 7.3 respectively visualise delay (cell rise/fall tables) and internal (short-circuit) power (output pin rise/fall power) information of all inverters, and how the fine-grained drive strengths interpolate into the original granularity. The inverters only have output pin power consumption, because they only have one input and one output. To best visualise the data in these plots, two 2D figures "Load Capacitance vs. Delay" and "Load Capacitance vs. Power" are projected. For the third index, the input slew, 3 different slews out of 7 in total, are selected to show the gate circuit delay and power when driving various loads under different input slews. The smallest slew (slew 1), the medium slew (slew 4) and the largest slew (slew 7) are shown.

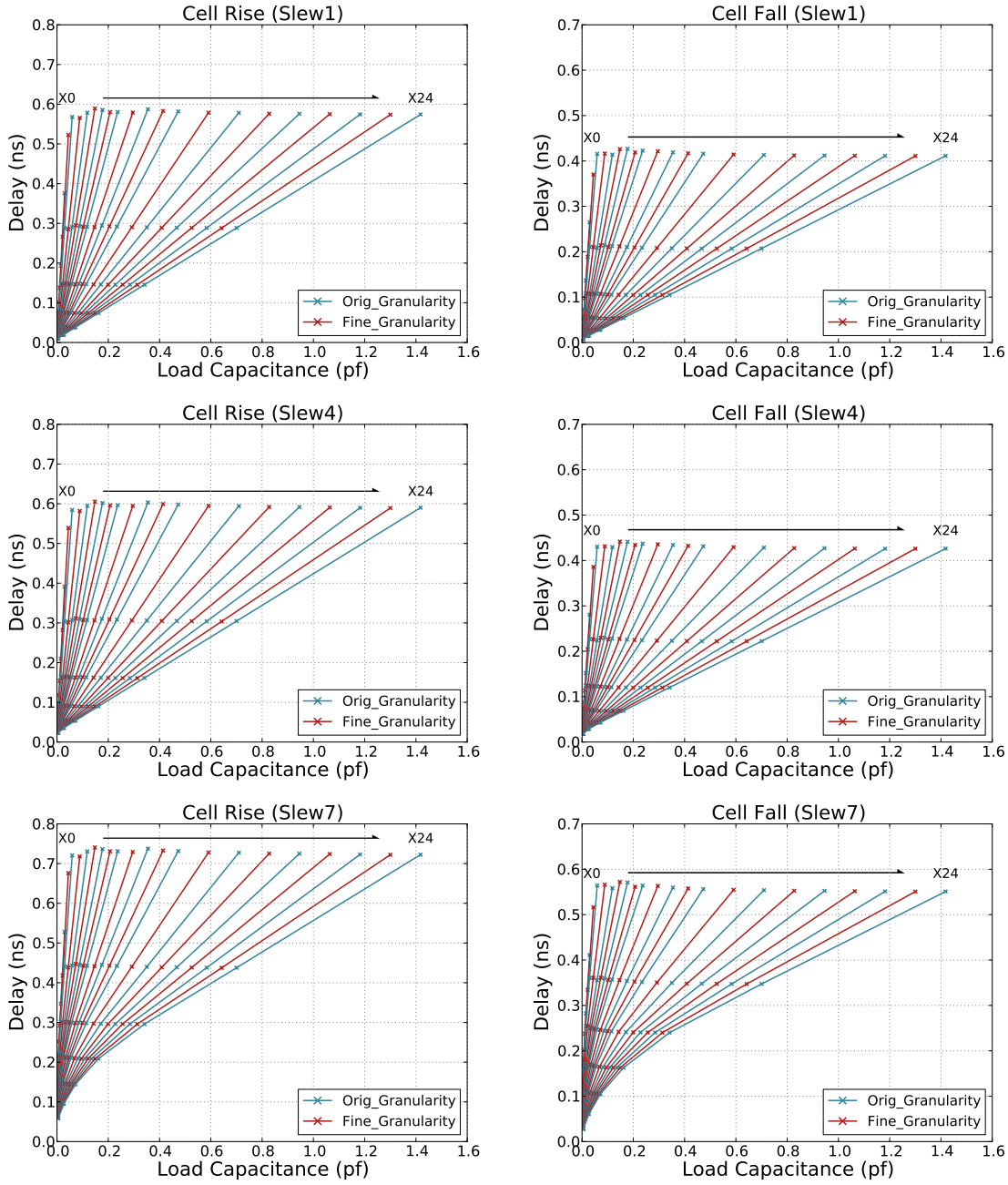


Figure 7.2 This plot shows cell rise/fall propagation delays of all inverters as "Load Capacitance vs. Delay" for three different input slew rates. The blue curves represent the original granularity inverters from the "MINI_ORIG" library, and the red lines illustrate the fine-grained "MINI_FINE" library's inverters

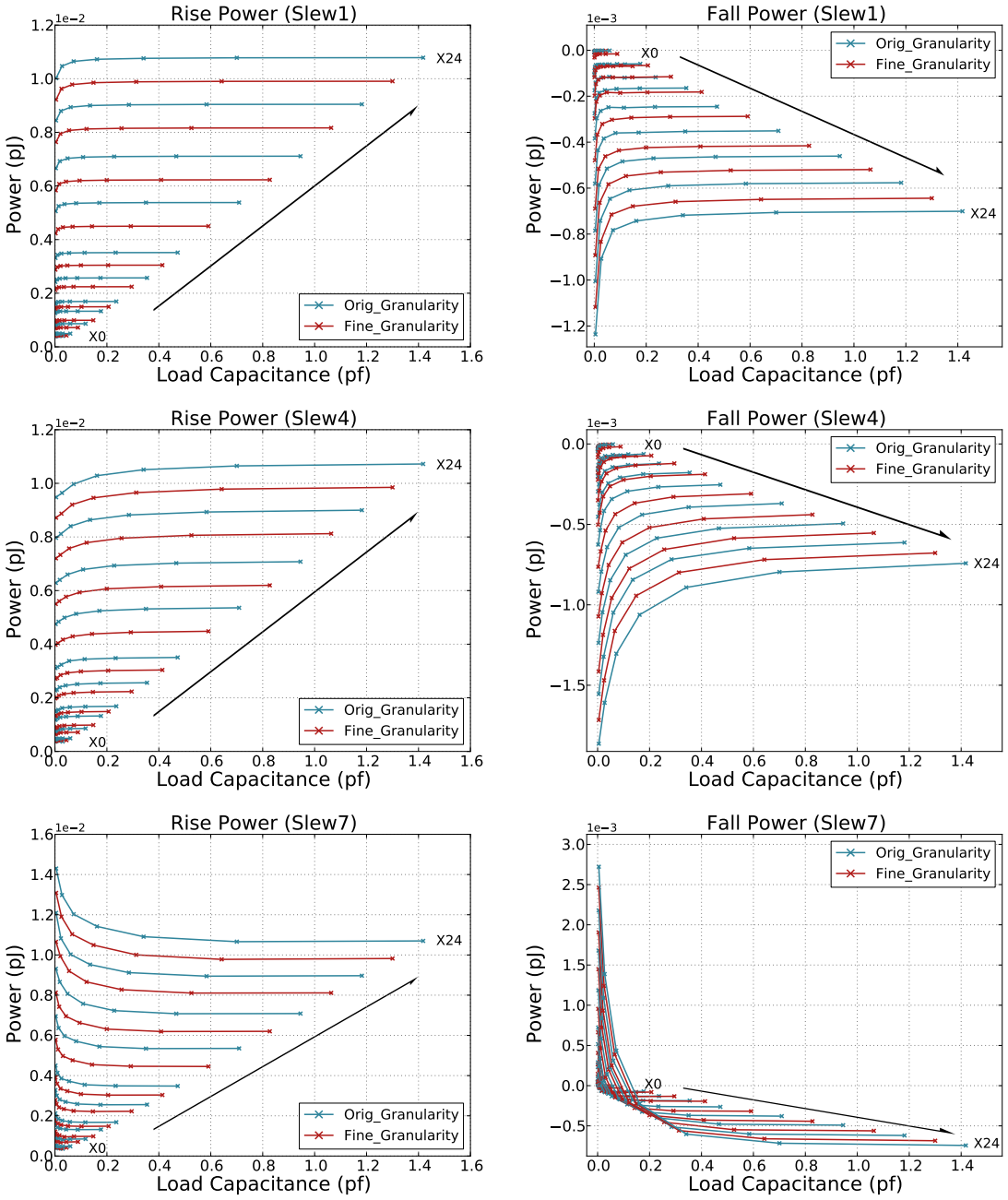


Figure 7.3 This plot shows rise/fall output pin power consumption of all inverters as "Load Capacitance vs. Power" for three different input slew rates. The blue curves represent the original granularity inverters from the "MINI_ORIG" library, and the red lines illustrate the fine-grained "MINI_FINE" library's inverters.

Both cell rise and fall propagation delays shown in the plots demonstrate that fine-grained drive inverters (red curves) are interpolating into the original drive granularity (blue curves) in an appropriate way. All inverters can drive the specified sets of loads with approximately same speed. Exceptions are X0 and X0.5 which, due to the minimum physical design constraints, cannot be down-sized further. Regardless of that, the X0.5 inverter is properly interpolated between X0 and X1. In terms of power consumption, the plots also show that all fine-grained drive inverters are positioned in the middle of adjacent original granularity inverters. Figure 7.4 shows the layouts of X0, X0.5, X1 and X1.5 inverters.

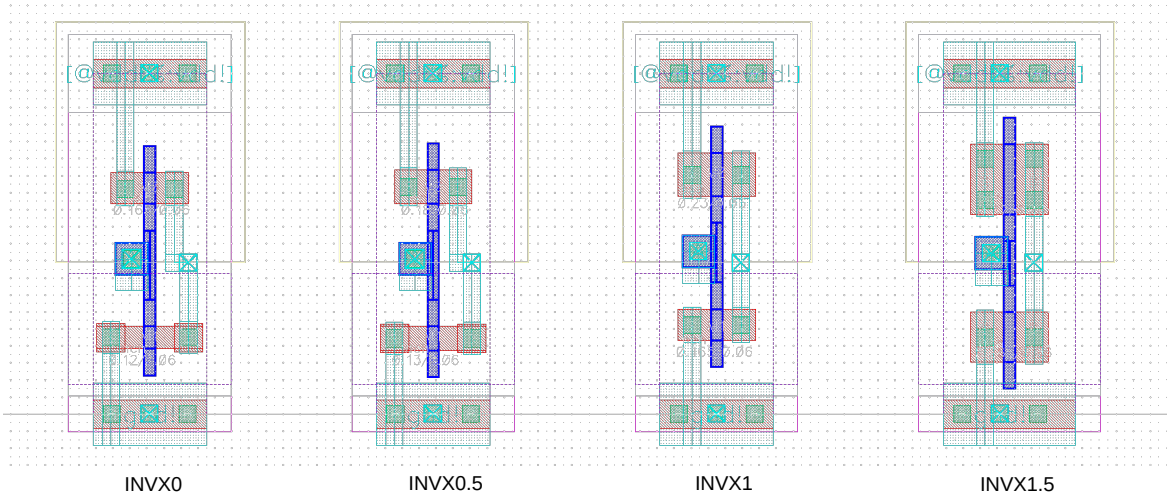


Figure 7.4 Layout examples of inverters created in this work. The inverter X0, X0.5 and x1 have the same cell width due to the physical design rules of the process technology used, but X1.5 is larger than others.

Following on from the analysis and discussion of both custom-designed libraries, these will be firstly loaded into the standard digital flow in order to investigate how the EDA tools trade-off design solutions when using a rich (finer-grained) drive strength library compared to the original, coarser-grained one. The MOEDA flow will then optimise drive strength selection based on the tool-optimised gate-level netlists in order to search for better solutions in PPA.

7.3 MOEDA in Fine-grained Cell Selection

The enriched "MINI_FINE" library is double the size of its original "MINI_ORIG" library. The synthesis tool then faces exponentially increased search space where the produced circuit solutions are not well-optimised trade-offs regarding PPA metrics.

Therefore, same adapted algorithm (NSGA-II) used in Chapter 5 with only using mutation operator is used for this work. The optimisation process is exclusively optimising selection on inverters, which is similar to the first initial experiment using reduced commercial library presented in Chapter 5 Section 5.5, but this work instead uses custom-designed fine-grained library.

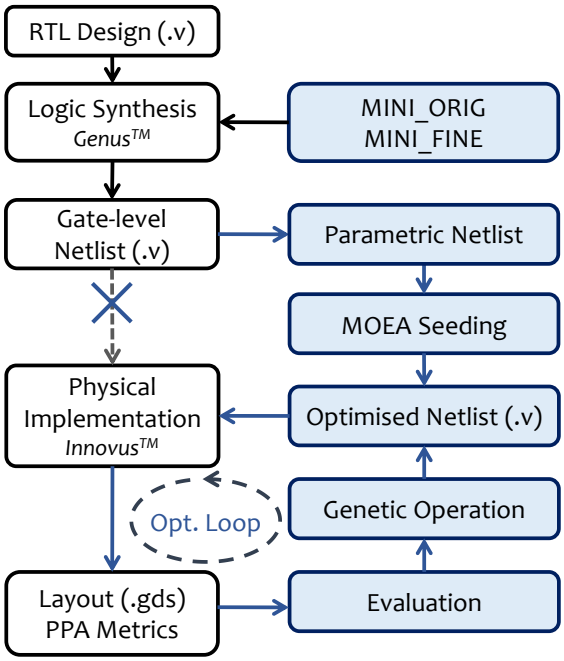


Figure 7.5 MOEDA framework works with custom-design "MINI_ORIG" and "MINI_FINE" libraries instead of using the foundry libraries.

Figure 7.5 specifically presents the MOEDA flow used in this chapter to cooperate with custom-designed different drive granularity cell libraries. From the figure, the MOEDA framework is the same as introduced in Chapter 5 included in the flowchart on the

left side (white boxes) which is the standard digital flow and the MO evolutionary optimisation engine (blue boxes) shown on the right.

The MOEDA engine still automatically performs fine-tuning on drive strength selection of logic gates (i.e., inverters in this case). The optimisation is operated on the synthesised gate-level netlists and they are then implemented into physical layouts for design evaluations.

Only inverters will be manipulated by the MOEA, so in this case a set of integer parameters (EA representations) define each inverter's drive strength to form a parametric netlist. Converting the solution netlist from the tool into a parametric netlist allows the MOEA to modify it. The modification is based on the "MINI_FINE" library (called **G** later) that includes all drive options of inverters.

The optimisation objectives are worst case delay (D_{wc}), total power consumption (P_{total}) and all gate area (A_{gate}). So the fitness function is to minimise them simultaneously, as shown in Equation (7.2). Evaluation metrics are calculated by the physical design EDA tools based on the physical layout instance.

$$\begin{aligned} f(\mathbf{g}) = \min \quad & [D_{wc}(\mathbf{g}), \quad P_{total}(\mathbf{g}), \quad A_{gate}(\mathbf{g})] \\ \text{s.t.} \quad & \mathbf{g} = (g_1, ..., g_i), \quad \forall g_i \in \mathbf{G} \end{aligned} \tag{7.2}$$

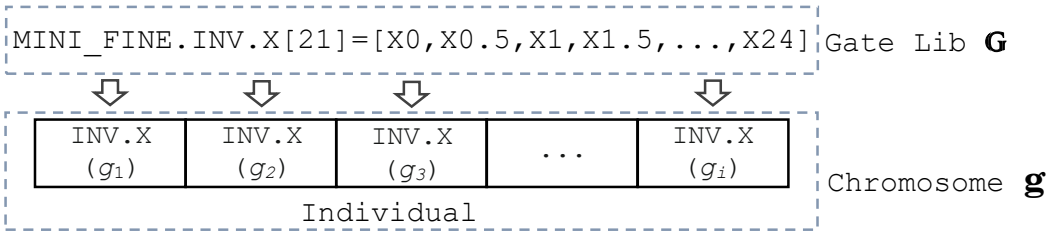


Figure 7.6 A chromosome example of an individual (i.e., layout instance in this case).

The chromosome vector **g** represents the input variables to the fitness functions, which in this case are drive strengths of inverters (g_i) available from the "MINI_FINE" library (**G**). Fig. 7.6 demonstrates a chromosome example of an individual where the $\mathbf{g} = (g_1, ..., g_i)$ represents all inverters of it. Each single g (INV.X) shows the drive

strength of an inverter. When mutation is triggered, the inverters to be mutated are randomly selected by the MOEA based on the given mutation rate ρ. For each selected inverter, the algorithm will randomly choose a new one from $\mathbf{G}$ (including all drive options) to replace the previous one.

7.4 Experiment Setup

The proposed algorithm is implemented in C++ and all experiments are running on a 2.2GHz Xeon E5-2650 CPU. Three ISCAS-85 benchmark circuits [114] are used as the test cases. The circuits (RTL designs) are synthesised into gate-level netlists using Cadence® Genus™ [116]. Cadence® Innovus™ [117] tool completes the physical implementation, producing layout instances. The evaluation is also performed in Innovus™, reporting all objective metrics.

7.4.1 Tool Environment Setup

Most of tool settings in both synthesis and physical implementation steps, summarised in Table 7.3, are the same as used in Chapter 5 and Chapter 6. However, the pre-place optimisation, "*PrePlaceOpt*", that is to delete buffers or inverters on the gate-level netlists before the placement is disabled in this case. This is because we are investigating how the tools select cells, it is worthwhile to keep netlists consistent during both synthesis and physical implementation steps.

In terms of design constraint for synthesis, an ideal clock is created running at 250MHz, so the worst path arrival time should be less than the required time (i.e., 4ns in this case) for timing closure. The testing cases used in this work are combinational circuits, thus, the clock is ideal without any uncertainties or transition delays.

The environment electrical constraint is applied by setting drive strength X1 and X4 output capacitive loads. The specific values chosen correspond to the respective inverter X1 and X4's input pin capacitance from "MINI_FINE" library.

Table 7.3 Design Constraint and Tool Settings in Digital Flow

<table>
<tr><th>Synthesis Setup</th><th>Place & Route Setup</th></tr>
<tr><td>syn_generic_effort = high
iopt_ultra_optimisation = true</td><td rowspan="3">aspect ratio = 1.0
core utilisation = 0.7
noPrePlaceOpt = true
timing-driven placement = true
timing-driven routing = true
SI-driven routing = true</td></tr>
<tr><td>Design Constraint</td></tr>
<tr><td>set_load = X1/X4
create_clock = 250MHz</td></tr>
</table>

7.4.2 Objective Evaluation in EDA Tools

All evaluations of D_{wc}, P_{total} and A_{gate} take place after place-and-route with Innovus^{TM} based on typical corner conditions.

(1) D_{wc}: This is the signal propagation time of the critical path calculated by static timing analysis in Innovus^{TM}.

(2) P_{total}: It is the sum of leakage power, internal power and switching power, which is from the average power analysis in Innovus^{TM}.

(3) A_{gate}: The sum area of all logic gates and it is directly reported by the Innovus^{TM}.

All experiments in this work are running 24 MOEDA evaluation threads in parallel and the runtime is still not the key focus in this work.

7.5 Experimental Results

7.5.1 Original vs. Fine-grained Cells in the Standard Flow

Firstly both "MINI_ORIG" and "MINI_FINE" libraries are loaded into the standard digital flow to investigate how the tools deal with different drive-granularity libraries and which drive strengths that the tool prefers. Three benchmarks with different circuit structures and functions from ISCAS-85 benchmark suite are synthesised and implemented in physical layouts. They are: a 16-bit error detector/corrector (C1908), a 12-bit ALU and controller (C2670) and a 9-bit ALU (C5315). Each circuit is implemented under three different timing constraints and two different output load constraints, resulting in 6 test cases per circuit and 18 in total. This aims to verify that the improved drive-granularity library can demonstrate generic benefits for designs when applying different timing goals (stringent or relaxed) and load capacitance (nominal or larger). The experiment information is summarised in Table 7.4.

Table 7.4 Test Case Summary

Design	Lib	Load	(#) Required Timing $[ns]$
C1908	ORIG/FINE	X1/X4	(a)1.25 (b)1.40 (c)1.55
C2670	ORIG/FINE	X1/X4	(a)1.20 (b)1.35 (c)1.50
C5315	ORIG/FINE	X1/X4	(a)1.35 (b)1.50 (c)1.65

Since the synthesise tool will automatically use more logic gates when timing constraints are stringent, the test cases with tightest timing constraint are selected as representatives for inverter drive strength selection analysis. Figures 7.7, 7.8 and 7.9 present histograms of inverters used in each tool-synthesised benchmark circuit when applying tightest timing requirements from case (a). The blue bars represent the histogram of the "MINI_ORIG" library and the red ones show the histogram of the "MINI_FINE" library. The number of all synthesised inverters and nand gates are reported in the legends to show the change of logic gates after "MINI_FINE" library is applied.

From these plots, the drive strengths X1 and X2 are the most commonly used cells. They are most dominant in the histograms of all test cases using the "MINI_ORIG" library. A number of fine-grained drive strength inverters are selected by the synthesis tool when using the "MINI_FINE" library. The peak around drive strength X2 is significantly flatter when fine-grained inverters are selected, although the number of drive strength X1 is still high. The likely reason for this is that, for many circuit paths, drive strength X1 is capable of driving the load at the endpoint, which is often a single gate. In addition, the improved drive strength resolution around X1 is exploited, although it may still not be fine enough to reduce the dominant X1 peak in the histograms of inverters used.

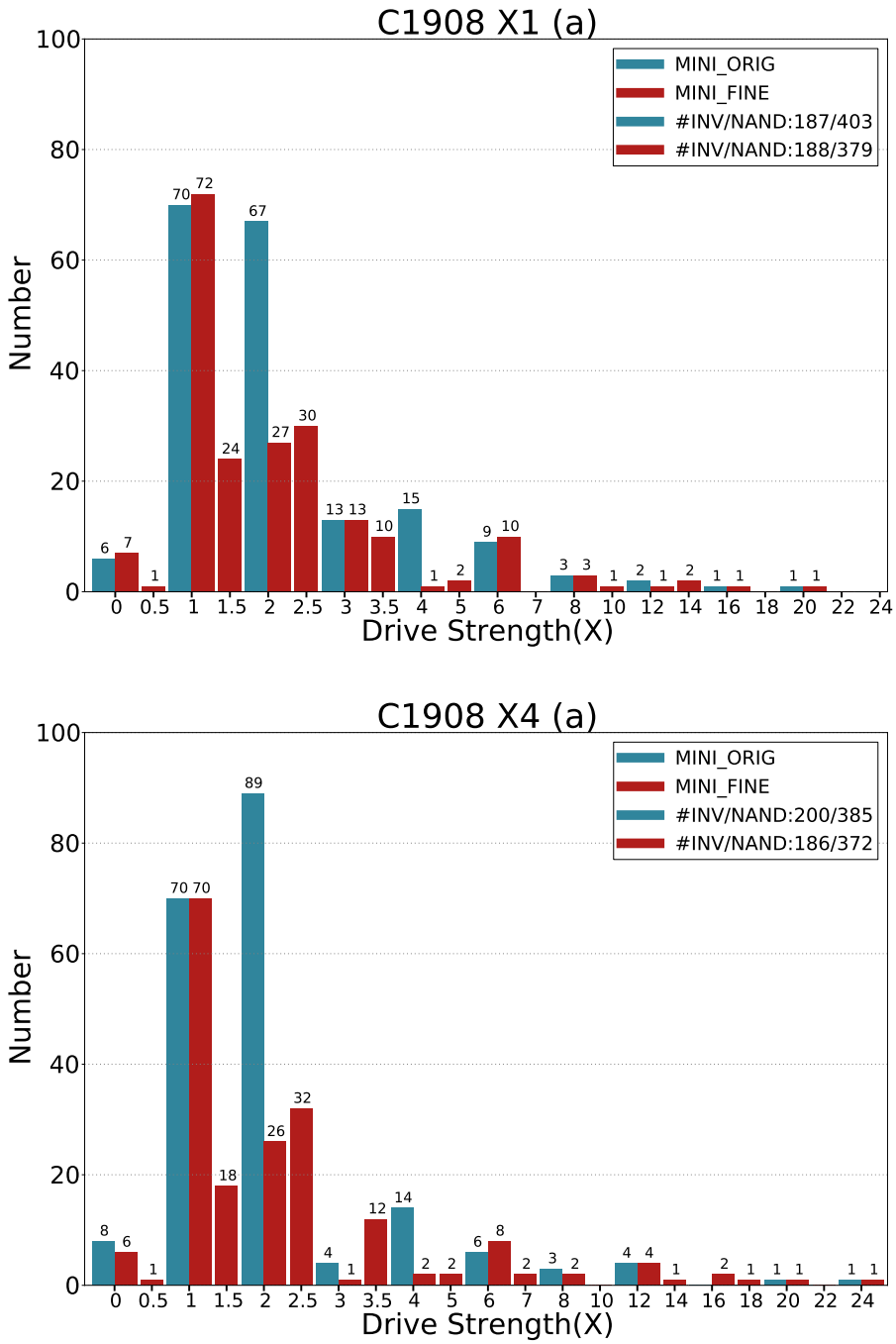


Figure 7.7 The histogram of tool-selected inverters' drive strengths of C1908. The blue bars are the inverters in original granularity from "MINI_ORIG" and red bars are the fine-grained inverters from "MINI_FINE".

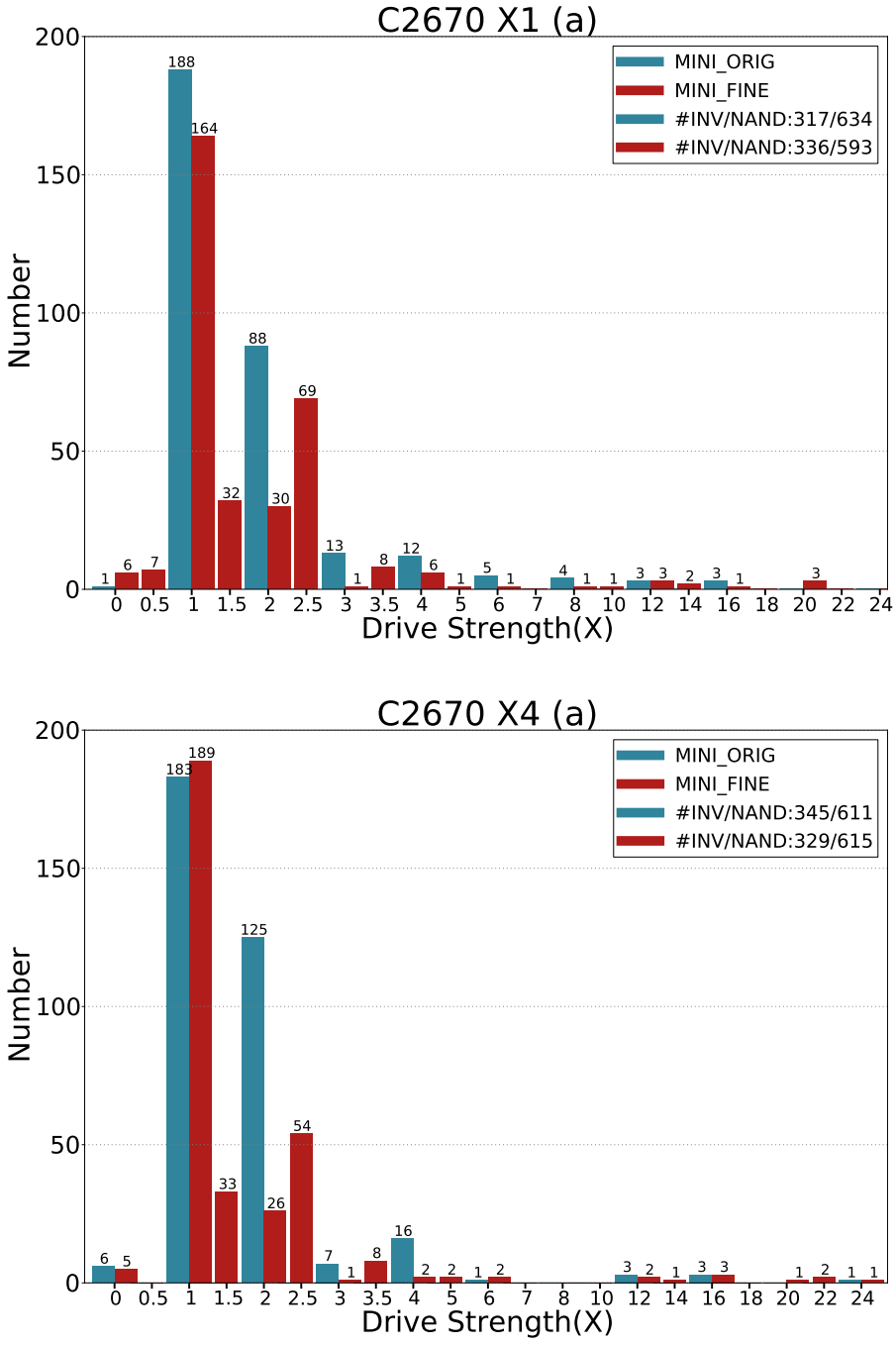


Figure 7.8 The histogram of tool-selected inverters' drive strengths of C2670.

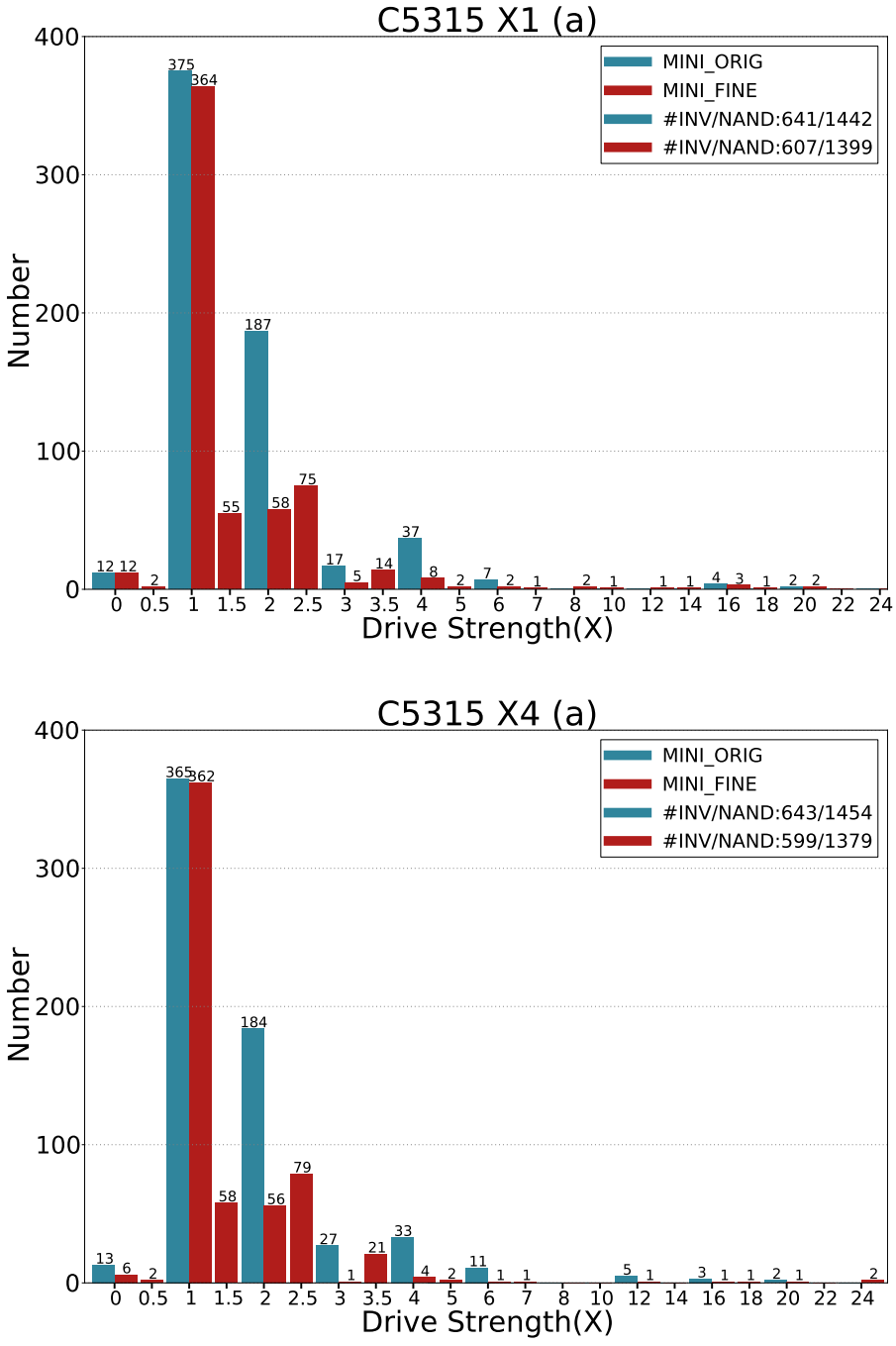


Figure 7.9 The histogram of tool-selected inverters' drive strengths of C5315.

These histograms of using "MINI_FINE" library also show that the most-selected fine-grained inverters of the synthesis tools are X1.5, X2.5, X3.5. This indicates which fine-grained gate sizes will be most useful and show significant benefits to designs particularly when applying this interpolation method to more common logic functions, e.g., NAND, NOR, AND, etc. Therefore, adding non-integer gate sizes between X0 and X4 (i.e., predominantly-selected by the tool) will be promising for better PPA metrics during synthesis of real-world chip design process, whereas the provided drive options (i.e., normally integer sizes) in foundry libraries are relatively coarse-grained.

All circuit evaluations in terms of PPA are performed based on the physical layouts. Tables 7.5, 7.6 and 7.7 summarise the PPA metrics for each test case, including worst case delay D_{wc}, total power consumption P_{total} and area sum of all gates A_{gate}. The normalised ($N.$) results are shown for easier improvement comparison. Each test case has three sets of results that are (1) "STD+ORIG": synthesising and implementing designs using the standard flow with the "MINI_ORIG" library; (2) "STD+FINE": synthesising and implementing designs using the standard flow with the "MINI_FINE" library; (3) "MOEDA+FINE": optimising designs using the MOEDA flow with the "MINI_FINE" library starting from the "STD+FINE" results. The results of "STD+ORIG" and "STD+FINE" are discussed first, followed by an illustration of the "MOEDA+FINE" results in the next section.

Based on the results shown in Tables 7.5, 7.6 and 7.7, using the "MINI_FINE" library can generate designs that achieve better trade-off solutions in PPA compared with using "MINI_ORIG" library running in the standard digital flow (with improvements up to 10% in D_{wc} of C2670-X1-(c), 14% in P_{total} and 13% in A_{gate} of C1908-X4-(b)), although degradation occurred in one of the objectives in some cases, e.g., C1908-X1-(b), C2670-X4-(a) and C5315-X4-(b). This may be due to the richer library leading to a larger design search space and the therefore increased computational complexity increasing synthesis and implementation effort.

The result tables also explicitly show that the synthesis tools can take advantage of the full capabilities of the fine-grained library "MINI_FINE" evidenced by the large amount of fine-grained inverters used, as shown in column "FINE INV UI." (i.e., fine-grained inverters utilisation).

The number of inverters, NANDs and total number of gates are also reported in Tables 7.5, 7.6 and 7.7 to show how their utilisation changes when synthesising designs using different drive granularity libraries. The total number of gates has decreased in most cases, and up to 6% (119 gates) reduction in the case of C5315-X4-(a), directly saving circuit area.

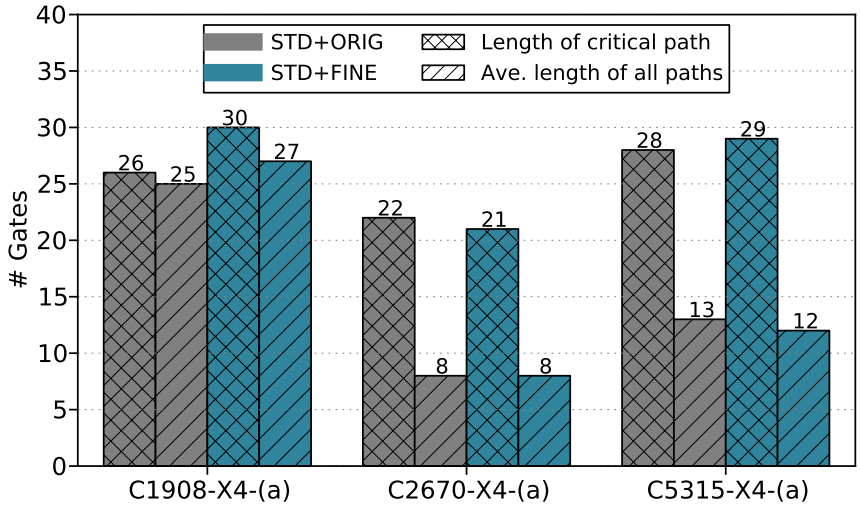


Figure 7.10 This shows the changes of circuit paths of each circuit after applying "MINI_FINE" library under the standard flow. The path length is achieved by calculating the gate count of a path.

Synthesising designs with different granularity libraries may produce solutions with different circuit structures. Figure 7.10 investigates whether applying fine-grained drive strength cells will change the circuit structure when using the standard flow. So a comparison is made here between the results of "STD+ORIG" and "STD+FINE" in the X4-(a) case of each circuit. The length of the critical path and average length of all paths are plotted here. Slight difference are shown between "STD+ORIG" and "STD+FINE" in terms of the circuit paths. This confirms when applying fine-grained

drive cells in the standard flow, the synthesis tool chose more suitable cells (fine-grained ones in a significant amount) from a wider available set to meet timing of each path, but the whole circuit structure did not change too much.

7.5.2 Fine-grained Cells in MOEDA Flow

To further improve solutions while balancing multiple objectives, which the standard digital flow is not capable of and instead prioritises timing alone, the MOEDA flow is used to enlarge the solution space, offering a wide range of Pareto-optimised solutions. The subsequent optimisation performed by the MOEDA flow is starting from a set of solutions obtained by the standard digital flow with the "MINI_FINE" library ("STD+FINE"). This is because the results of "STD+FINE" has achieved better circuit evaluation metrics than the solution of using the "MINI_ORIG" library initially, so that the MOEDA's optimisation efforts are focused on finding better trade-off solutions in regard of PPA, rather than starting from scratch.

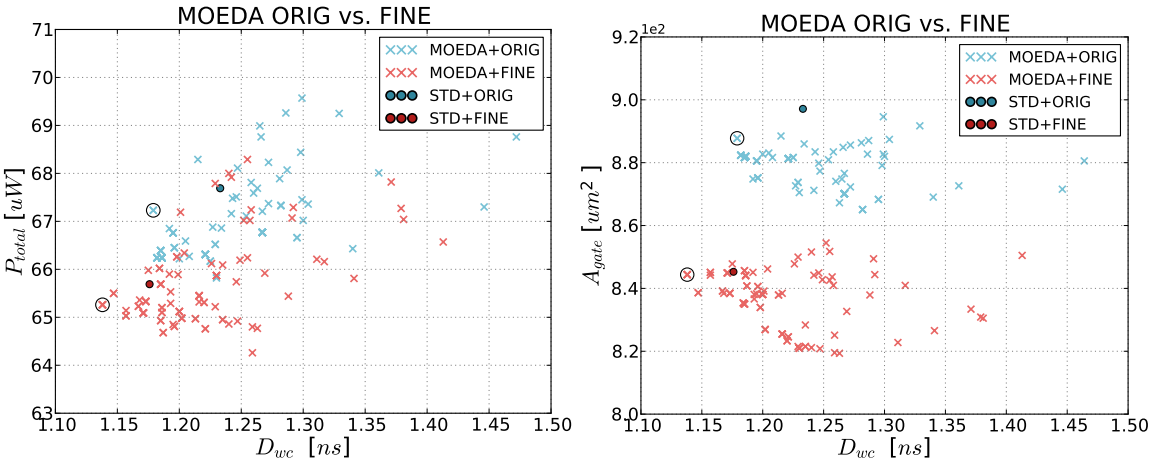


Figure 7.11 The MOEDA flow optimisation results comparison between seeding with "STD+ORIG" (blue) and "STD+FINE" (red). The circled solutions are the best delay solution of each cluster.

Figure 7.11 compares the optimisation results of MOEDA flow using the "MINI_ORIG" and the "MINI_FINE" libraries in the C1908-X1-(a) case. Both run with N=100

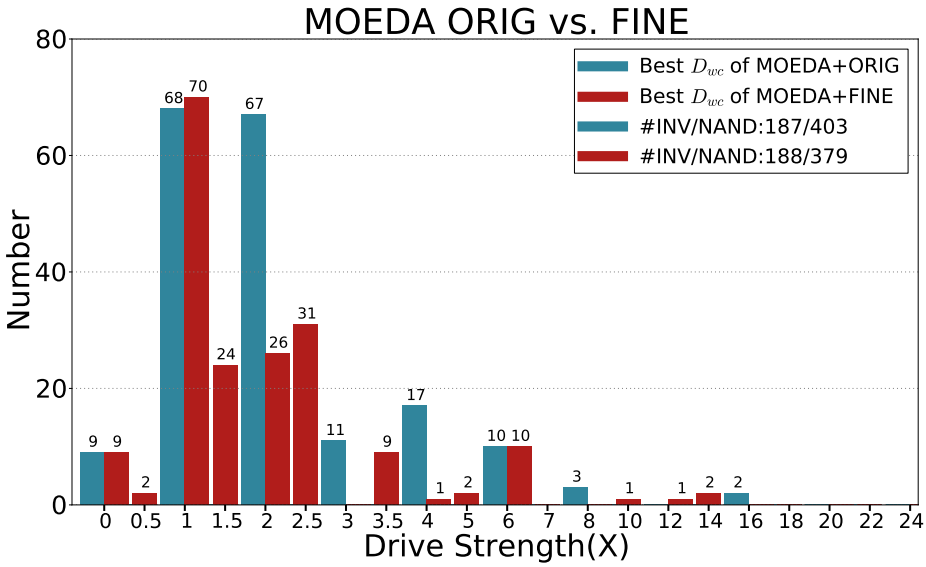


Figure 7.12 The inverter histogram of the best delay solution of "MOEDA+ORIG" solution space and "MOEDA+FINE" solution space from Figure 7.11. EA run settings: $N = 100$, $M = 100$ and $\rho = 0.5\%$.

individual of a population for M=100 generations using mutation rate ρ=0.5%. The optimisation run seeded with "STD+FINE" solutions can achieve a wider coverage of the design space featuring solutions with better PPA metrics than those based on "STD+ORIG" alone. Figure 7.12 shows the inverter histogram of each best-delay solution of the "MOEDA+ORIG" solution space (in blue bars) and the "MODEA+FINE" solution space (in red bars). Both are circled as shown in the plots (Figure 7.11). This histogram shows similar drive strength distribution compared to the one of C1908-X1-(a) from Figure 7.7, which shows the synthesis results without the MOEDA flow. But the drive strength selection results from the tool still has been refined after applying the optimisation of MOEDA flow. In both cases of "MOEDA+ORIG" and "MOEDA+FINE", drive strengths smaller than X1 are selected more often and drive strengths larger than X2.5 are used less, resulting in power saving. Comparing the solution space of the "MOEDA+ORIG" with the "MOEDA+FINE" 's, the "MOEDA+FINE" 's results further reduce the use of drive strengths larger than X2.5, so that the power and area of the best D_{wc} solution of "MOEDA+FINE" is much lower than the "MOEDA+ORIG" 's. This confirms that the MOEDA flow can balance

multiple objectives through selection of more appropriate drive strengths for digital circuits. Also, the MOEDA flow can efficiently deal with the richer drive granularity library in trading off solutions.

Due to the previous findings, MOEDA flow optimisation is carried out for the next experiments, only initialised with "STD+FINE" seed solutions from the standard flow. All the rest of experiments also run with 100 individuals for 100 generations using 0.5% mutation rate. The MOEDA optimisation results, highlighted in Tables 7.5, 7.6 and 7.7, are the best trade-off solutions from the entire final solution space. The best trade-off solution is taking all objectives into account simultaneously, which is defined here as an individual from the final generation that is positioned at the shortest Euclidean distance from the origin. These trade-off solutions demonstrate the optimisation capability of achieving improvements in all objectives simultaneously. Four testing cases marked with stars represent that "STD+ORIG" solutions have already failed timing requirements. Most of these failed cases have been improved in D_{wc} in "STD+FINE" solutions, and all of them have been improved by the MOEDA flow without compromising on other objectives to the point that they achieve timing closure.

Table 7.5 Results Comparison of C1908

$N = 100$, $M = 100$, $\rho = 0.5\%$

Design (set_load)	(#) Required Timing	Flow	Lib (MINI)	# INV/NAND	Total # Gates	FINE INV UT.[%]	D_{wc} $(N.)$ $[ns]$	P_{total} $(N.)$ $[uW]$	A_{gate} $(N.)$ $[um^2]$
C1908 (X1)	(a) $1.25ns$	STD	ORIG	187/403	590	0%	1.233 (1.00)	67.69 (1.00)	897.12 (1.00)
		STD	FINE	188/379	567	37.2%	1.176 (0.95)	65.69 (0.97)	845.28 (0.94)
		MOEDA	FINE	188/379	567	37.8%	**1.138 (0.92)**	**65.26 (0.96)**	**844.38 (0.94)**
	(b) $1.40ns$	STD	ORIG	192/396	588	0%	1.211 (1.00)	67.59 (1.00)	893.16 (1.00)
		STD	FINE	173/362	535	37.0%	1.244 (1.03)	61.65 (0.91)	797.04 (0.89)
		MOEDA	FINE	173/362	535	35.8%	**1.186 (0.98)**	**60.23 (0.89)**	**790.92 (0.88)**
	(c) $1.55ns$	STD	ORIG	189/386	575	0%	1.218 (1.00)	67.31 (1.00)	881.64 (1.00)
		STD	FINE	183/363	546	35.5%	1.212 (0.99)	62.46 (0.93)	819.36 (0.93)
		MOEDA	FINE	183/363	546	34.4%	**1.164 (0.95)**	**60.96 (0.90)**	**815.76 (0.92)**
C1908 (X4)	(a) $1.25ns$*	STD	ORIG	200/385	585	0%	1.306 (1.00)	71.56 (1.00)	889.56 (1.00)
		STD	FINE	186/372	558	37.1%	1.200 (0.92)	67.56 (0.94)	852.12 (0.96)
		MOEDA	FINE	186/372	558	37.1%	**1.164 (0.89)**	**66.75 (0.93)**	**838.80 (0.94)**
	(b) $1.40ns$	STD	ORIG	205/421	626	0%	1.255 (1.00)	78.08 (1.00)	970.56 (1.00)
		STD	FINE	183/372	555	38.8%	1.210 (0.96)	67.20 (0.86)	842.76 (0.87)
		MOEDA	FINE	183/372	555	36.1%	**1.191 (0.95)**	**65.99 (0.84)**	**824.94 (0.85)**
	(a) $1.55ns$	STD	ORIG	205/407	612	0%	1.247 (1.00)	75.05 (1.00)	938.88 (1.00)
		STD	FINE	192/387	579	39.6%	1.255 (1.01)	72.31 (0.96)	887.94 (0.95)
		MOEDA	FINE	192/387	579	41.1%	**1.180 (0.94)**	**70.41 (0.94)**	**870.12 (0.92)**

Table 7.6 Results Comparison of C2670

$N = 100$, $M = 100$, $\rho = 0.5\%$

Design (set_load)	(#) Required Timing	Flow	Lib (MINI)	# INV/NAND	Total # Gates	FINE INV UT.[%]	D_{wc} $(N.)$ $[ns]$	P_{total} $(N.)$ $[uW]$	A_{gate} $(N.)$ $[um^2]$
C2670 (X1)	(a) 1.20ns	STD	ORIG	317/634	951	0%	0.982 (1.00)	98.23 (1.00)	1377.72 (1.00)
		STD	FINE	336/593	929	35.7%	0.952 (0.97)	95.43 (0.97)	1375.56 (0.99)
		MOEDA	FINE	336/593	929	36.6%	**0.902 (0.92)**	**94.75 (0.96)**	**1368.36 (0.99)**
	(b) 1.35ns	STD	ORIG	360/633	993	0%	1.013 (1.00)	103.5 (1.00)	1485.00 (1.00)
		STD	FINE	359/627	986	34.5%	0.957 (0.94)	103.1 (0.99)	1464.48 (0.99)
		MOEDA	FINE	359/627	986	35.1%	**0.895 (0.88)**	**102.2 (0.98)**	**1463.76 (0.98)**
	(c) 1.50ns	STD	ORIG	332/601	933	0%	1.116 (1.00)	98.29 (1.00)	1430.28 (1.00)
		STD	FINE	336/601	937	28.9%	1.007 (0.90)	92.85 (0.94)	1355.22 (0.95)
		MOEDA	FINE	336/601	937	29.2%	**0.972 (0.87)**	**92.13 (0.93)**	**1355.22 (0.95)**
C2670 (X4)	(a) 1.20ns	STD	ORIG	345/611	956	0%	1.041 (1.00)	100.8 (1.00)	1376.64 (1.00)
		STD	FINE	329/615	944	30.4%	0.986 (0.95)	101.9 (1.01)	1373.22 (0.99)
		MOEDA	FINE	329/615	944	31.3%	**0.921 (0.88)**	**101.8 (1.009)**	**1358.10 (0.98)**
	(b) 1.35ns	STD	ORIG	387/662	1049	0%	1.073 (1.00)	115.4 (1.00)	1559.16 (1.00)
		STD	FINE	347/616	963	30.5%	1.046 (0.97)	101.0 (0.88)	1403.28 (0.90)
		MOEDA	FINE	347/616	963	32.0%	**0.937 (0.87)**	**100.0 (0.86)**	**1402.56 (0.90)**
	(c) 1.50ns	STD	ORIG	338/644	982	0%	1.083 (1.00)	107.0 (1.00)	1458.72 (1.00)
		STD	FINE	331/580	911	37.5%	1.081 (0.99)	99.07 (0.93)	1400.58 (0.96)
		MOEDA	FINE	331/580	911	38.7%	**0.981 (0.90)**	**98.64 (0.92)**	**1379.34 (0.94)**

Table 7.7 Results Comparison of C5315

$N = 100$, $M = 100$, $\rho = 0.5\%$

Design (set_load)	(#) Required Timing	Flow	Lib (MINI)	# INV/NAND	Total # Gates	FINE INV UT.[%]	D_{wc} $(N.)$ $[ns]$	P_{total} $(N.)$ $[uW]$	A_{gate} $(N.)$ $[um^2]$
C5315 (X1)	(a) 1.35ns *	STD	ORIG	641/1442	2083	0%	1.405 (1.00)	249.4 (1.00)	2968.20 (1.00)
		STD	FINE	607/1399	2006	25.0%	1.417 (1.01)	234.9 (0.94)	2850.66 (0.96)
		MOEDA	FINE	607/1399	2006	24.9%	**1.291 (0.92)**	**232.0 (0.93)**	**2841.12 (0.95)**
	(b) 1.50ns	STD	ORIG	624/1416	2040	0%	1.433 (1.00)	238.9 (1.00)	2875.32 (1.00)
		STD	FINE	620/1378	1998	23.4%	1.408 (0.98)	234.6 (0.98)	2820.60 (0.98)
		MOEDA	FINE	620/1378	1998	23.9%	**1.332 (0.93)**	**229.5 (0.96)**	**2814.48 (0.97)**
	(c) 1.65ns	STD	ORIG	624/1374	1998	0%	1.437 (1.00)	235.9 (1.00)	2821.68 (1.00)
		STD	FINE	622/1371	1993	22.8%	1.469 (1.02)	231.5 (0.98)	2801.16 (0.99)
		MOEDA	FINE	622/1371	1993	24.0%	**1.325 (0.92)**	**226.5 (0.96)**	**2798.46 (0.99)**
C5315 (X4)	(a) 1.35ns *	STD	ORIG	643/1454	2097	0%	1.486 (1.00)	259.2 (1.00)	3018.60 (1.00)
		STD	FINE	599/1379	1978	27.4%	1.413 (0.95)	236.5 (0.91)	2812.50 (0.93)
		MOEDA	FINE	599/1379	1978	27.7%	**1.342 (0.90)**	**236.4 (0.91)**	**2808.36 (0.93)**
	(b) 1.50ns *	STD	ORIG	605/1357	1962	0%	1.592 (1.00)	244.3 (1.00)	2784.24 (1.00)
		STD	FINE	604/1372	1976	25.7%	1.543 (0.97)	237.6 (0.97)	2798.28 (1.005)
		MOEDA	FINE	604/1372	1976	25.5%	**1.356 (0.85)**	**231.6 (0.95)**	**2796.30 (1.004)**
	(c) 1.65ns	STD	ORIG	652/1457	2109	0%	1.343 (1.00)	255.9 (1.00)	2989.08 (1.00)
		STD	FINE	619/1397	2016	24.1%	1.365 (1.02)	242.2 (0.95)	2870.82 (0.96)
		MOEDA	FINE	619/1397	2016	24.7%	**1.268 (0.94)**	**239.7 (0.93)**	**2863.26 (0.95)**

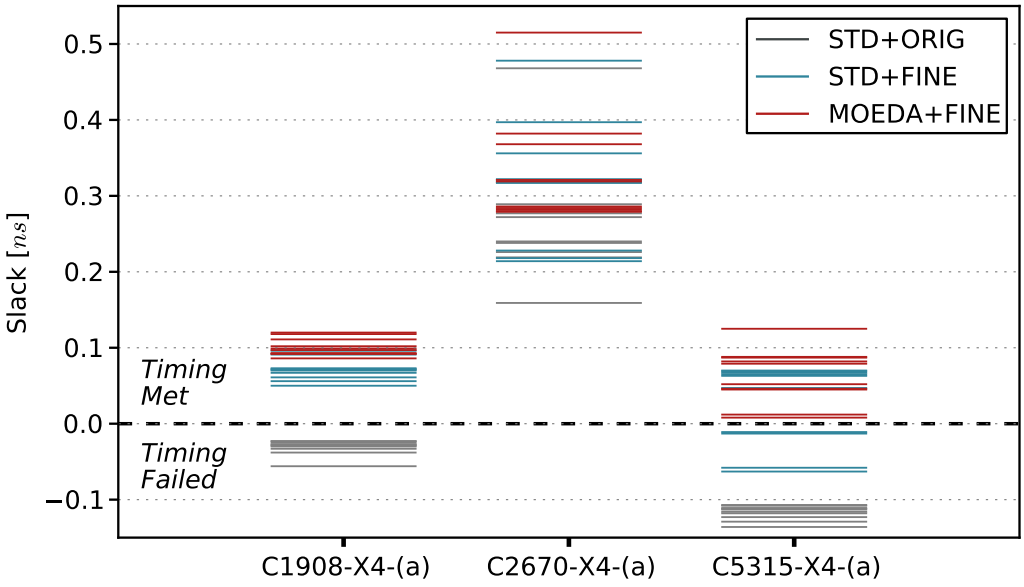


Figure 7.13 Ten worst timing paths of test circuits for each corresponding tight timing constraint case X4-(a). All paths above the dash line have positive slacks which meet the timing. The slack is higher the circuit timing is better.

Figure 7.13 plots the ten worst timing paths of the X4-(a) case (tightest timing constraint in this work) of each test circuit. This shows how the timing of paths, particularly the critical path, has been optimised. The results of "MOEDA+FINE" (red lines) recovered the all timing failed paths and performed the hill-climbing on the slack of critical paths, where only applying "MINI_FINE" library in the standard (STD) flow ("STD+FINE" in blue lines) is not capable of. In addition, the results of applying "MINI_ORIG" in the STD flow ("STD+ORIG" in gray lines) explicitly show inferior timing performance, especially in C1908 and C5315 circuits with negative slacks.

To investigate the changes of drive strength selection when using different libraries and flows, Figure 7.14 presents the sum of drive strength sizes of each whole circuit and their corresponding critical paths. The tight timing case X4-(a) of each benchmark is still used for analysis here. Based on the observation of this plot, the overall drive size sum of all circuit paths has decreased after applying "STD+FINE" and has further

been optimised by the MOEDA flow. This straightforwardly saves the resulting power and area of designs.

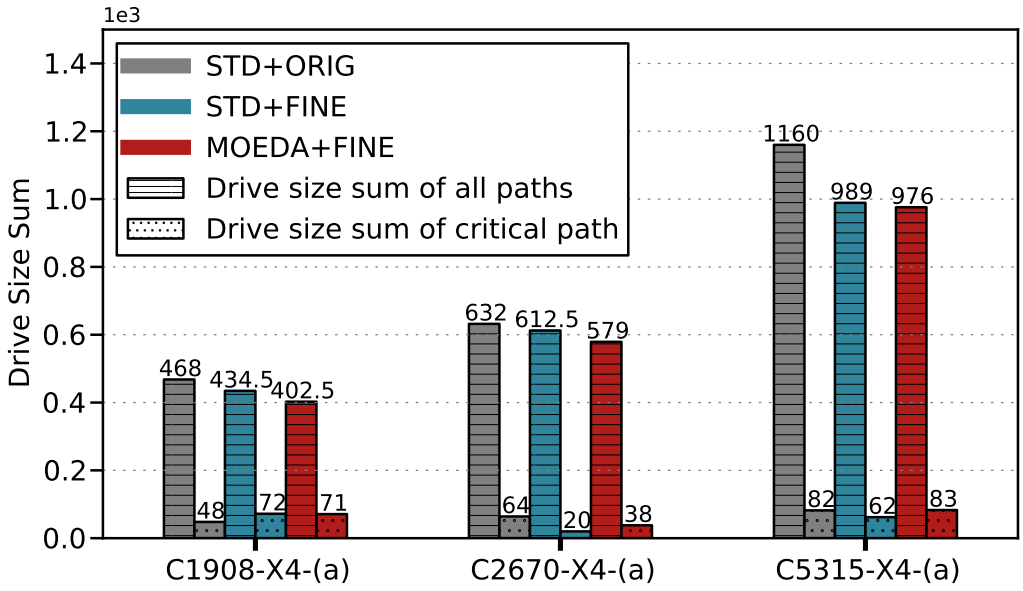


Figure 7.14 The changes of drive strengths of critical paths and overall circuits when applying "STD+ORIG", "STD+FINE" and "MOEDA+FINE". The sum of drive strengths are reported from the X4-(a) case of each benchmark.

In terms of critical paths relating to the circuit worst slack, more larger drive cells from "MINI_FINE" library are selected by MOEDA flow to solve timing violations in C1908 and C5315, since they had timing failed paths in the initial solution generated by "STD+ORIG". The critical path delay of "STD+FINE" solution in C1908 and C5315 was improved over the "STD+ORIG", but the total size of selected drive strengths is not always increased. This indicates that to optimise the path timing it needs to choose drive strengths in a proper way instead of significantly scaling up the gate sizes. In C2670 circuit, the drive strength sum of "STD+FINE" solution is reduced greatly while all paths are meeting the timing constraint and the worst slack is improved. The MOEDA flow then selects more larger cells to push the timing performance but the used drive strengths is still less than the "STD+ORIG" one.

In addition, since margins shown in the EDA tools for drive strength mapping, that redundant larger cells are selected by the standard flow using the coarse-grained library,

the performance of EDA tools is variable and might be not capable of getting an optimum solution, particularly when handling enlarged search space.

To show the complete trade-off solutions and optimised solution space, Figures 7.15 and 7.16 respectively present the final generation of MOEDA optimisation of the tight timing constraint case, X1-(a) and X4-(a), of each benchmark circuit. The "STD+ORIG" and "STD+FINE" solutions of each corresponding case are plotted for comparison. The MOEDA flow has successfully enlarged the feasible solution space while simultaneously achieve significant improvements in respond to PPA. If designers focus on one or two of these objectives, the available solutions from MODEA flow can obtain greater objective improvements than the trade-off solutions' reported in the result Tables 7.5, 7.6 and 7.7. The runtime of the largest and most complex case C5315-X4-(a) is 5.5 hours.

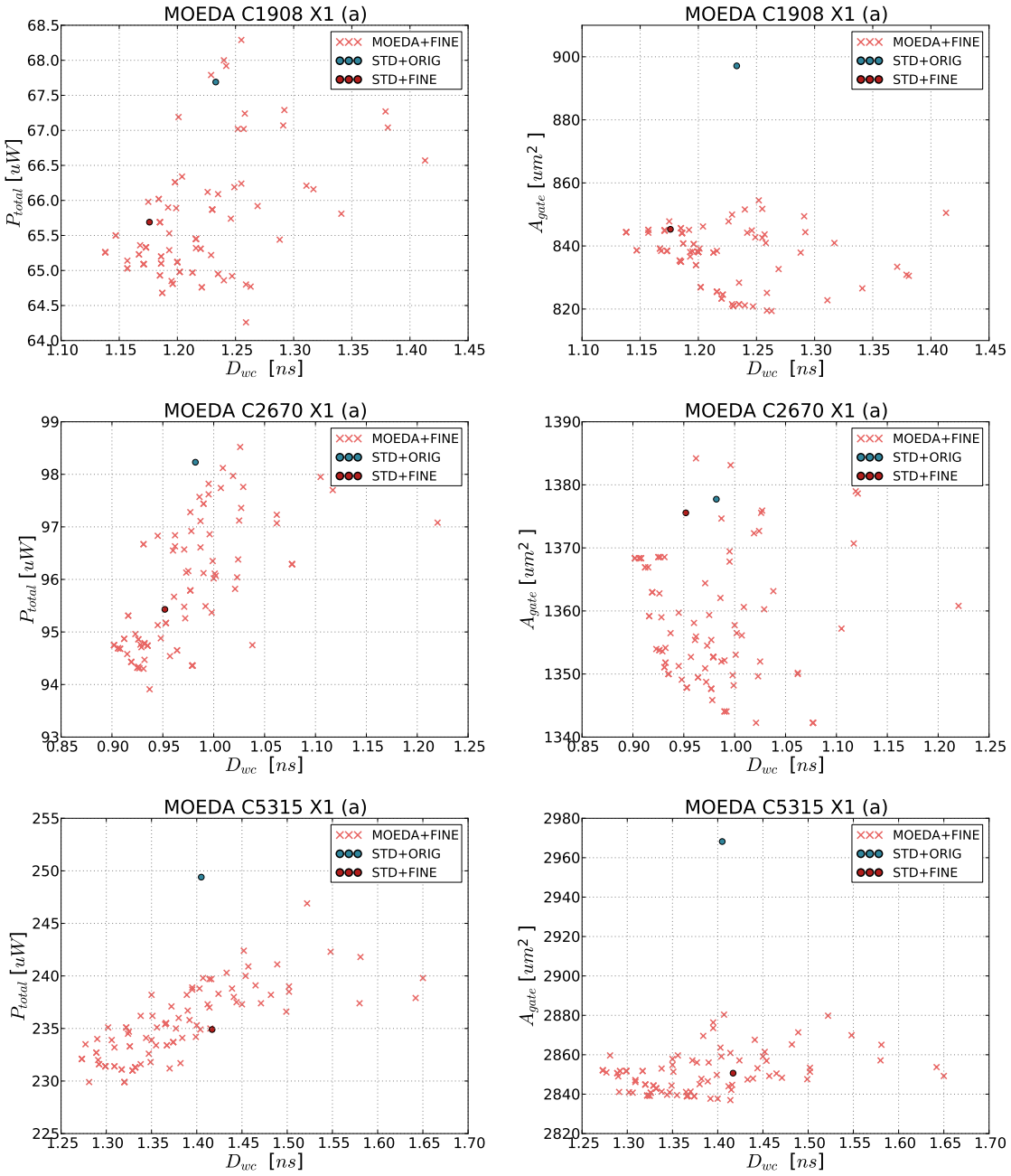


Figure 7.15 For the X1-(a) case of each circuit, the left column plots in "D_{wc} vs. P_{total}" and "D_{wc} vs. A_{gate}" is on the right column. There are two individuals in the round shape are the "STD+ORIG" and "STD+FINE" solutions. All other individuals in the shape of cross are the final generation of MOEDA optimised results based on the "STD+FINE" solution (i.e., MOEA seed).

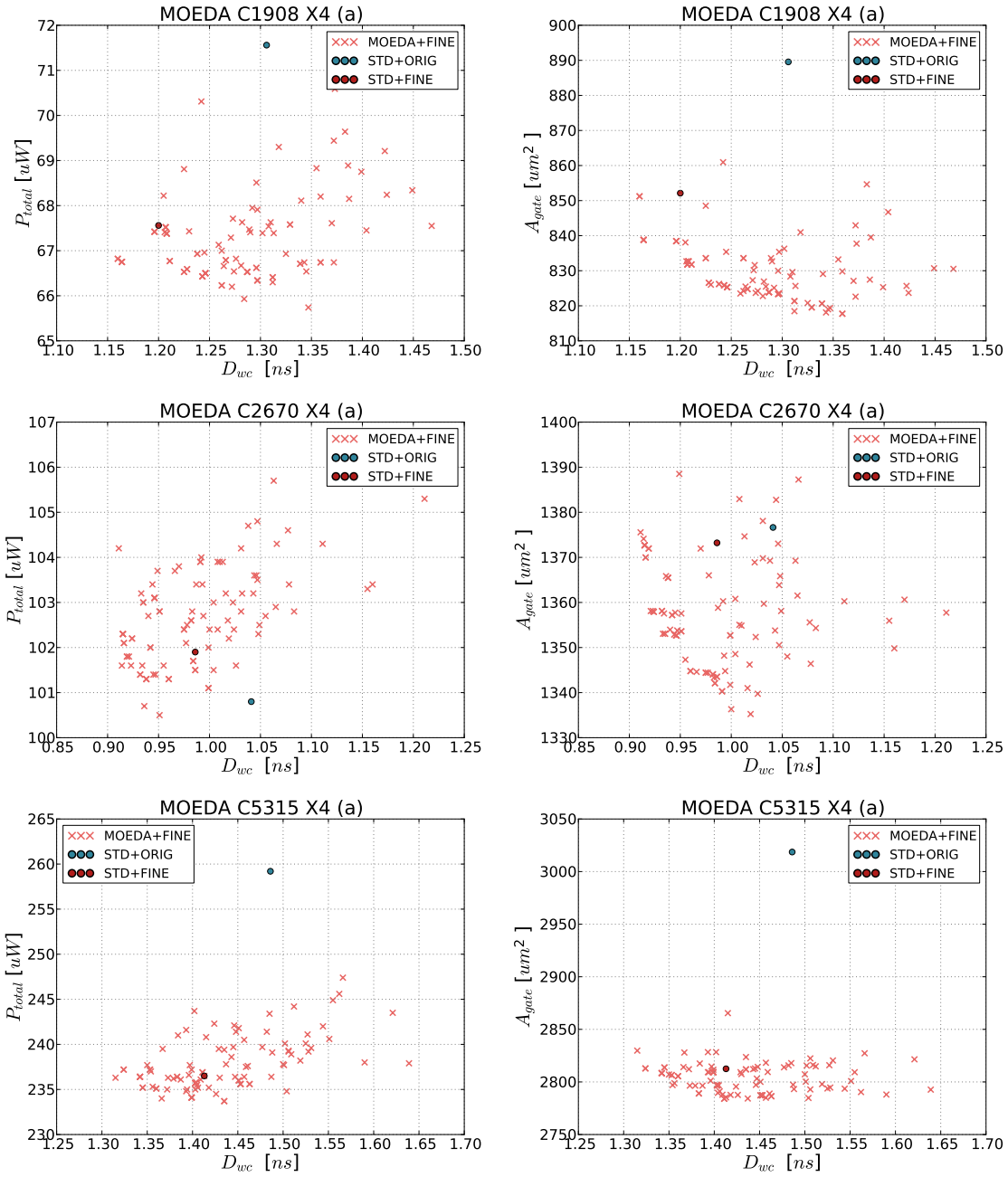


Figure 7.16 For the X4-(a) case of each circuit, the left column plots in "D_{wc} vs. P_{total}" and "D_{wc} vs. A_{gate}" is on the right column. There are two individuals in the round shape are the "STD+ORIG" and "STD+FINE" solutions. All other individuals in the shape of cross are the final generation of MOEDA optimised results based on the "STD+FINE" solution (i.e., MOEA seed).

7.6 Summary

This chapter has shown that digital synthesis and implementation tools produced solutions can be improved when provided with fine-grained drive strength cell libraries. The industrial tool flow exploited the finer drive strengths to improve PPA of all benchmarks used. The results indicate that providing finer drive resolution around predominantly-selected drive strengths is particularly useful. This suggests that enriching drive options of functions of a standard cell library around predominantly selected drive strengths (typically between X0 and X4) is a promising method to get better performance for large-scale designs out of the standard EDA tools.

The main challenge of the proposed fine-grained cells approach is that enlarged standard cell libraries result in a larger design search space. The EDA tools have to make a greater effort during drive strength mapping, due to the increased computational complexity, and may not always arrive at an optimum solution (PPA metrics cannot be improved simultaneously) in a given time frame. The proposed MOEDA digital design flow can overcome these issues as it is capable of further balancing PPA trade-offs and provide a range of design solutions where standard EDA tool performance is quite variable and cannot trade-off PPA well.

The capability of the proposed MOEDA flow to offer a set of well-balanced, and often improved, trade-off solutions with regard to PPA also opens up opportunities for designers to choose the most appropriate solution for different applications.

Chapter 8

Conclusions and Further Work

8.1 Conclusions

This thesis has developed an automated optimisation framework combining multi-objective evolutionary algorithms (MOEAs) with industry-standard digital VLSI design processes down to the physical layout level. The goals of this thesis are two-fold: first, to deliver solutions with improved overall performance in PPA. Second, to provide a flexible global optimisation framework that seamlessly fits with existing EDA design tools to achieve this. Multi-objective approaches have been applied to more efficiently run industrial EDA tool flows to produce Pareto-optimised solutions, expanding solution space across various circuit topologies, and dealing with complex floorplan constraints. In addition, combining this framework with transistor sizing, an essential part of standard cell design, a methodology for improving the drive granularity of a standard cell library has been proposed. This opens up opportunities to further take advantage of an existing technology node by reducing design margins and pushing its performance.

This work starts with exploring the research background, in Chapter 2 and Chapter 3, regarding silicon technology devices, standard digital VLSI design process, multi-objective problems (MOPs) and MOEAs. The demand identified for modern digital VLSI design and optimisation is how to find good trade-off solutions through efficiently running standard EDA flows to deal with the intrinsic limitations and scalability challenges of advanced silicon devices, and the growing size and complexity of the designs themselves. The capability of MOEAs in solving MOPs can fit in such a context involving multiple design objectives, or goals. The existing research showed that limited work focused on using evolutionary-inspired techniques to aid real-world chip design for solution quality enhancement regarding power, performance, and area down to physical circuit layout.

Chapter 4 developed an automated multi-objective physical design framework to adjust standard cell (drive strength) selection in physical layouts for a CMOS VLSI design. The experimental results of this feasibility study confirmed that the MOEA was able to successfully apply Pareto-driven search for a set of solutions optimised in

three objectives (i.e, delay, energy and area) allowing designers to choose appropriate solutions for use of different case scenarios.

A multi-objective (MO) EDA framework was developed in Chapter 5 to cooperate with an industry-standard RTL-to-GDSII flow in the optimisation process. The MOEDA flow has successfully optimised a series of benchmark circuits using a foundry library through refining drive strength mapping on gate-level netlists. In such a way, designs were improved with better PPA (power, performance, area) metrics in subsequent physical implementation step. The optimised quality of results (QoRs) demonstrated the significantly enhanced performance over the standard EDA tool flow.

Given the contributions made in Chapter 4 and Chapter 5, the **objective 1**, "Develop an automated multi-objective VLSI design optimisation framework allowing the manipulation of digital circuit building blocks down to physical layout level", has been completed.

In Chapter 6, a methodology was proposed for seeding MOEAs with various circuit topologies to expand the standard-flow-generated design space. The performance variation has been revealed inside commercial synthesis tools in producing trade-off solutions, particularly when timing constraints are tight. The proposed methodology performed multi-objective design space exploration (MODSE) on the basis of an initially tool-generated design space. The optimised design space showed better coverage with significant PPA improvements over the entire baseline. This has not only been demonstrated in nominal floorplan settings but also been empirically studied with practical cases when applying complex floorplan constraints (e.g., polygon die shape, restricted pin placement).

The experimental results achieved shown in Chapter 5 and the entire work of Chapter 6 correspond to **objective 2** that is "Demonstrate that framework capable of improving performance of VLSI designs (including complex physical corner cases) offering a range of Pareto-optimised solutions and better design space coverage over industrial-flow-

generated ones". These contributions can reasonably support marking **objective 2** as successfully completed.

Chapter 7 proposed a methodology to enrich a standard cell library through appropriate interpolation of fine-grained drive strength options into the original drive granularity present in the cell library. This process aimed at making better use of a process technology, thereby avoiding over-design of solutions with regards to power and area. The empirical study was performed by using fine-grained drive cells in an industrial digital flow. A number of fine-grained cells (around 30% out of all gates for a design) were selected by commercial tools while improving the PPA of generated solutions. However, not all objectives (PPA) of designs always got simultaneously improved and some of them were slightly degrade. Therefore, the proposed MOEDA framework was used here to adjust drive strength mapping from expanded libraries in order to further balance the PPA objectives.

Chapter 7 corresponds to **objective 3** that is "Investigate the application to library level optimisation via improving drive strength granularity of standard cells for better use of foundry technology nodes and better resulting quality of VLSI design solutions". Contributions made in this chapter fully support the related objective and it can be considered as successfully completed.

Hypothesis Review:

Based on the conducted work and achieved objectives, a review of hypotheses will be discussed. At the beginning of the thesis, a main hypothesis was stated:

"Combining multi-objective evolutionary algorithms with digital VLSI design process can achieve performance-improved solutions down to physical layout level, expand feasible design space, and handle complex physical layout constraints more efficiently via refining standard cell mapping and improving standard cell granularity."

with sub-hypotheses:

Sub-hypothesis 1.1: "Multi-objective evolutionary algorithms can optimise the drive strength mapping of logic gates in digital VLSI designs for superior performance with a wide spread of feasible trade-off solutions than standard tools."

Sub-hypothesis 1.2: "Multi-objective evolutionary algorithms in conjunction with an industrial digital IC flow can achieve better Pareto-driven search space coverage across various circuit topologies than standard tools alone."

Sub-hypothesis 1.3: "Multi-objective evolutionary algorithms can explore a larger feasible solution (objective) space to deal with complex physical floorplan constraints efficiently."

Sub-hypothesis 1.4: "Fine-grained drive strength resolution of standard cells can optimise digital VLSI designs for over-design mitigation and pushing performance of silicon technologies."

Two multi-objective optimisation frameworks were proposed in Chapters 4 and 5, which combined the MOEAs with digital VLSI design process. One is an automated place and route flow for a custom layout design, and another is the MOEDA flow compatible with industry-standard synthesis, place and route tools. Both successfully performed the drive strength refinement down to physical layouts for accurate evaluations close to post-fabrication scenario. In case of the MOEDA flow, it showed the superiority in generating trade-off solutions than standard tools can. The evidence provided can fully support the **sub-hypothesis 1.1**.

The methodology proposed in Chapter 6 for seeding MOEAs with many circuit topologies expanded the industrial-flow-generated design space for a more uniform spread of trade-off solutions. This has also been experimented with different physical floorplan settings for efficiently exploring lager feasible design space. The resultant evidence from Chapter 6 can reasonably support **sub-hypothesis 1.2** and **1.3**.

The interpolation methodology proposed in Chapter 7 for improving drive strength resolution of standard cells offered more drive options in order to push performance

of silicon technologies. This improved the achievable performance (delay, slack) of digital VLSI designs while mitigating over-design (power, area) to a significant extent. The results and evidence from Chapter 7 can conclude that **sub-hypothesis 1.4** is completely supported.

Critical Comments:

This PhD work has successfully pushed the research boundary in general that MOEAs are promising techniques to solve MOPs of digital VLSI designs in the context of industrial practice and design environment. The thesis mainly provides knowledge contribution to the VLSI community where using MOEAs for VLSI design and optimisation lacked practicability [72].

Although this cross-disciplinary field was investigated around 20 years ago, both areas of evolutionary computation and digital VLSI design have significantly changed from what they used to be. It is therefore worthwhile to combine the existing popular evolutionary-inspired techniques with state-of-the-art digital IC design methodologies, practice, tools/flows regarding nowadays's chip design common challenges.

However, a few limitations of this PhD work are listed below, which can be potential research topics for the future.

- The proposed evolutionary multi-objective optimisation framework requires a long runtime.
- The proposed MOEDA flow and the MOEA seeding methodology for MODSE do not support circuit topology optimisation beyond the topologies present in the seed population, also presented in Chapters 5 and 6.
- The proposed methodology for standard cell drive strength expansion shows benefits for improving performance of VLSI designs, but it still needs custom design efforts from engineers.

8.2 Future Work

Based on the current findings of this thesis, few directions can be explored in the future.

Runtime of MOEA Optimisation Approaches:

Due to the inherent mechanism (i.e., population-based method) of MOEAs, the evolutionary optimisation approach requires numerous evaluations and computing resources. This will increase the runtime of the algorithm and limit the optimisation efficiency. Particularly when dealing with large designs, it often needs exponentially increased population size and the number of generations to achieve better results. Investigating how to efficiently run MOEAs for fast optimisation convergence will be a potential research avenue for the future. In detail, a number of opportunities within MOEA setup configurations, e.g. tuning mutation rate and crossover operations, can be explored.

The mutation rate used in this work is a constant during the proposed optimisation frameworks. However, applying different mutation rates to run EAs will deliver different results for a given MOP. Selecting an appropriate mutation rate of EAs to solve a specific problem, getting the possible optimum through running with fewer generations and smaller population size, is hard but is worthwhile to investigate. Determining mutation rates is case-specific and often based on designers' experience, which can be automatically decided by machine learning techniques in the future. In addition, exploring varying mutation rate during the evolutionary process will be an interesting and promising area for saving EA runtime.

The crossover operator has not been applied in this work. The main reason mentioned earlier is that the crossover operator can not generally handle the function-preserved variation when different circuit topologies are present. For a single circuit topology optimisation, it is possible to implement the crossover operator, but researchers need to define the crossover points (including the number of points and the position of each)

within a chromosome. In this work, a chromosome represents all logic gates of a circuit. Defining the number of crossover points and their corresponding cutting positions is similar to the partitioning step of physical design which will lead to another research domain. Manipulating evolutionary optimisation using crossover variation combining with physical design partitioning will be an interesting research avenue for MOEA running efficiency.

The used MOEA in this work is NSGA-II. It can produce a set of trade-off solutions close to the Pareto-optimum scenario. Alternative MOEAs such as evolution strategy (ES) instead excel in searching for a possible best solution with balanced objectives in an efficient way. This will also be one of the future work fields for runtime efficiency.

The significant runtime in proposed optimisation frameworks is mainly owing to the large search space of the entire design (all logic gates considered). For instance, if a design has 10000 gates (i.e., relative small-scale in real-world chip design) and each gate has five drive strengths on average. The solution space complexity is 5^{10000} alternative options, and this will be further complicated with multiple threshold voltage (V_{th}) versions (typically SVT, HVT and LVT), so the solution space will then exponentially grow to 15^{10000}. Therefore, focusing on critical paths of a design in the evolutionary process will be useful to improve the optimisation efficiency particularly for extreme-large modern designs (millions of gates). This will also provide multi-objective methods to fix and optimise timing-violated paths.

Standard Cell Design Efficiency:

The methodology introduced in Chapter 7 investigates how to enrich the drive strength granularity on an existing standard cell library. To ensure the design feasibility and manufacturability, the proposed methodology still manually handles standard cell design complying with all foundry design rules. However, the logic cells used in the synthesis and physical implementation flow are abstracted models. Creating fake cells to optimise a library in drive granularity based on abstracted models can efficiently

co-design and co-optimise ICs close to exhaustive transistor sizing, making most of the potential capabilities of a technology node. The most selected fake cells can be designed in real physical layouts later by engineers who can try to make the real-world cells' performance close to the ideal fake ones as much as possible.

Circuit Topology Co-optimisation:

As shown in Chapters 5 and 6, circuit topologies can significantly influence the optimisation results. Highly structured circuits are hard to optimise for high-performance only through tuning drive strength mapping. Although circuit topologies often lead to system-level design, gate-level structure optimisation is crucial to achieving design closures. During the standard digital flow, if design violations appeared in physical design, engineers need to perform minor modifications on gate-level designs or even have to re-synthesis the design with perturbations of constraints and initial conditions to resolve the violations. Co-tuning gate-level design topologies with standard cell mapping will further explore the solution space that has already been achieved in this work. Therefore, a multi-loop multi-objective optimisation approach will be a promising and viable direction. For instance, one loop can take charge of circuit topology adjustment to minimise the number of gates and length of critical paths. Another loop selects promising circuit candidates from the first loop to optimise standard cell mapping for the solutions with better PPA achievable at physical implementation step.

Analog Circuit Optimisation: For a larger picture of the future work, the optimisation methodology proposed in the thesis is mainly focusing on tuning the size of transistors. Such an approach can be migrated to the analog circuit design process. Determining appropriate transistor sizes using MOEAs for different analog circuits can be interesting research avenue.

Appendix A

SKILL Scripts for Creating Parametric Layouts:

```
procedure (StripeLayoutCreate(G0 G1 G2 G3 G4 G5 G6 G7 G8 G9 G10 G11 G12 G13 G14 G15 G16 G17)
Inv_Lib = '(list("Inv0" 1.8 0.135 1.665)
list("Inv1" 1.62 0.135 1.485)
list("Inv2" 1.55 0.135 1.415)
list("Inv3" 1.5 0.135 1.365)
list("Inv4" 1.9 0.135 1.765)
list("Inv5" 1.85 0.135 1.715)
list("Inv6" 1.81 0.135 1.675)
list("Inv7" 1.8 0.135 1.665)
list("Inv8" 1.76 0.135 1.625)
list("Inv9" 2.1 0.135 1.965))

;inverters library '("name"; width; input_pos; output_pos)
nand_width = 2.2
nand_A = 0.135 ;nand input A position
nand_B = 1.895 ;nand input B position
nand_Y = 1.625 ;nand output Y position

;channel for routing metal2
channel_1 = 1.46
channel_2 = 1.18
channel_3 = 0.9
channel_4 = 0.62
channel_5 = 0.34

cursor = 0 ;drawing cursor
Inv_count = 0 ; inverter names counter
Path_count = 0 ; path names counter
nand_count = 0
```

```
;start layout
cv = dbOpenCellViewByType("design" "f_adder_auto" "layout" "maskLayout" "w")


tf = techGetTechFile(cv)
viaDef=techFindViaDefByName(tf "M2_M1")


nand_id = dbOpenCellViewByType("std_cell" "nand2" "layout")


;————————————————nand and buffer————————————————;
FuncBuffer(G0 G1)
FuncBuffer(G2 G3)
FuncBuffer(G4 G5)
FuncBuffer(G6 G7)
FuncBuffer(G8 G9)
FuncBuffer(G10 G11)
FuncBuffer(G12 G13)
FuncBuffer(G14 G15)
FuncBuffer(G16 G17)

;————————————————Channel 5 Routing ————————————————;
Path_start = 0
Path_end = 0

;————————————————Path 1————————————————————————-;
inst = dbFindAnyInstByName(cv "Inv1")
Path_start = car(inst >xy) + nth(4 nth(G1 Inv_Lib))
ViaPlacement(Path_start channel_5)

inst = dbFindAnyInstByName(cv "nand1")
Path_end = car(inst >xy) + nand_B
ViaPlacement(Path_end channel_5)
PathCreate(Path_start Path_end channel_5)

Path_start = Path_end
inst = dbFindAnyInstByName(cv "nand2")
Path_end = car(inst >xy) + nand_A
ViaPlacement(Path_end channel_5)
PathCreate(Path_start Path_end channel_5)

Path_start = Path_end
inst = dbFindAnyInstByName(cv "nand6")
Path_end = car(inst >xy) + nand_B
ViaPlacement(Path_end channel_5)
```

```
PathCreate(Path_start Path_end channel_5)

;————————————————Path 2————————————————-;
inst = dbFindAnyInstByName(cv "Inv15")
Path_start = car(inst >xy) + nth(4 nth(G15 Inv_Lib))
ViaPlacement(Path_start channel_5)

inst = dbFindAnyInstByName(cv "nand8")
Path_end = car(inst >xy) + nand_B
ViaPlacement(Path_end channel_5)
PathCreate(Path_start Path_end channel_5)
;————————————————————————————————————————————————————————————————-;

;————————————————Channel 4 Routing ————————————————;
Path_start = 0
Path_end = 0

;————————————————Path 1————————————————-;
inst = dbFindAnyInstByName(cv "Inv5")
Path_start = car(inst >xy) + nth(4 nth(G5 Inv_Lib))
ViaPlacement(Path_start channel_4)

inst = dbFindAnyInstByName(cv "nand3")
Path_end = car(inst >xy) + nand_B
ViaPlacement(Path_end channel_4)
PathCreate(Path_start Path_end channel_4)

;————————————————Path 2————————————————-;
inst = dbFindAnyInstByName(cv "nand4")
Path_start = car(inst >xy) + nand_B
ViaPlacement(Path_start channel_4)

inst = dbFindAnyInstByName(cv "nand7")
Path_end = car(inst >xy) + nand_B
ViaPlacement(Path_end channel_4)
PathCreate(Path_start Path_end channel_4)
;————————————————————————————————————————————————————————————————-;

;————————————————Channel 3 Routing ————————————————;
Path_start = 0
Path_end = 0

;————————————————Carry out signal ————————————————;
inst = dbFindAnyInstByName(cv "Inv13")
```

```
CoutVia = car(inst >xy) + nth(4 nth(G13 Inv_Lib))
ViaPlacement(CoutVia channel_3)
Path_start = CoutVia - 0.3
Path_end = CoutVia + 0.3
PathCreate(Path_start Path_end channel_3)
;————————————————————————————————————-;

;———————————————Channel 2 Routing ————————————;
Path_start = 0
Path_end = 0
;Path 1
inst = dbFindAnyInstByName(cv "nand0")
Path_start = car(inst >xy) + nand_B
ViaPlacement(Path_start channel_2)

inst = dbFindAnyInstByName(cv "nand2")
Path_end = car(inst >xy) + nand_B
ViaPlacement(Path_end channel_2)

PathCreate(Path_start Path_end channel_2)

;Path2
inst = dbFindAnyInstByName(cv "Inv9")
Path_start = car(inst >xy) + nth(4 nth(G9 Inv_Lib))
ViaPlacement(Path_start channel_2)

inst = dbFindAnyInstByName(cv "nand5")
Path_end = car(inst >xy) + nand_B
ViaPlacement(Path_end channel_2)

PathCreate(Path_start Path_end channel_2)

Path_start = Path_end
inst = dbFindAnyInstByName(cv "nand6")
Path_end = car(inst >xy) + nand_A
ViaPlacement(Path_end channel_2)

PathCreate(Path_start Path_end channel_2)

Path_start = Path_end
inst = dbFindAnyInstByName(cv "nand7")
Path_end = car(inst >xy) + nand_A
ViaPlacement(Path_end channel_2)
```

```
PathCreate(Path_start Path_end channel_2)
;------------------------------------------------------------------------;

;------------------------Channel 1 Routing ------------------------;
Path_start = 0
Path_end = 0

;Path 1
inst = dbFindAnyInstByName(cv "nand0")
Path_start = car(inst >xy) + nand_A
ViaPlacement(Path_start channel_1)

inst = dbFindAnyInstByName(cv "nand1")
Path_end = car(inst >xy) + nand_A
ViaPlacement(Path_end channel_1)

PathCreate(Path_start Path_end channel_1)

;Path2
inst = dbFindAnyInstByName(cv "Inv3")
Path_start = car(inst >xy) + nth(4 nth(G3 Inv_Lib))
ViaPlacement(Path_start channel_1)

inst = dbFindAnyInstByName(cv "nand3")
Path_end = car(inst >xy) + nand_A
ViaPlacement(Path_end channel_1)

PathCreate(Path_start Path_end channel_1)

;Path3
inst = dbFindAnyInstByName(cv "Inv7")
Path_start = car(inst >xy) + nth(4 nth(G7 Inv_Lib))
ViaPlacement(Path_start channel_1)

inst = dbFindAnyInstByName(cv "nand4")
Path_end = car(inst >xy) + nand_A
ViaPlacement(Path_end channel_1)

PathCreate(Path_start Path_end channel_1)

Path_start = Path_end
inst = dbFindAnyInstByName(cv "nand5")
Path_end = car(inst >xy) + nand_A
ViaPlacement(Path_end channel_1)
```

```
PathCreate(Path_start Path_end channel_1)

;Path4
inst = dbFindAnyInstByName(cv "Inv11")
Path_start = car(inst >xy) + nth(4 nth(G11 Inv_Lib))
ViaPlacement(Path_start channel_1)

inst = dbFindAnyInstByName(cv "nand8")
Path_end = car(inst >xy) + nand_A
ViaPlacement(Path_end channel_1)

PathCreate(Path_start Path_end channel_1)

;Path5
inst = dbFindAnyInstByName(cv "Inv17")
S_Via = car(inst >xy) + nth(4 nth(G17 Inv_Lib))
ViaPlacement(S_Via channel_1)
Path_start = S_Via - 0.5
Path_end = S_Via + 0.1
PathCreate(Path_start Path_end channel_1)

;-------------------------------------------------------------------------------;

;-------------------------------Pin Create -------------------------------;
Pin_pos = 0

;-------------------vdd-----------------------;
rodCreateRect(

?name "vdd!"
?layer list("M1" "pin")
?width 0.09
?length 0.09
?origin 1:1.755
?cvId cv

?netName "vdd!"
?termName "vdd!"
?pin t
?pinLabel t
?pinLabelHeight 0.09
?pinLabelLayer list("M1" "pin")
)
```

```
;————————————gnd————————————;
rodCreateRect(

?name "gnd!"
?layer list("M1" "pin")
?width 0.09
?length 0.09
?origin 1: -0.045
?cvId cv

?netName "gnd!"
?termName "gnd!"
?pin t
?pinLabel t
?pinLabelHeight 0.09
?pinLabelLayer list("M1" "pin")
)

;————————————Input A ————————————;
Pin_pos = Pin_pos + nand_A
rodCreateRect(

?name "A"
?layer list("M2" "pin")
?width 0.1
?length 0.1
?origin (Pin_pos - 0.05):(channel_1- 0.05)
?cvId cv

?netName "A"
?termName "A"
?termIOType "input"
?pin t
?pinLabel t
?pinLabelHeight 0.1
?pinLabelLayer list("M2" "pin")
)

;————————————Input B ————————————;
inst = dbFindAnyInstByName(cv "nand0")
Pin_pos = car(inst >xy) + nand_B
rodCreateRect(
```

```
?name "B"
?layer list("M2" "pin")
?width 0.1
?length 0.1
?origin (Pin_pos - 0.05):(channel_2- 0.05)
?cvId cv

?netName "B"
?termName "B"
?termIOType "input"
?pin t
?pinLabel t
?pinLabelHeight 0.1
?pinLabelLayer list("M2" "pin")
)

;————————————Input Cin ————————————;
inst = dbFindAnyInstByName(cv "nand7")
Pin_pos = car(inst >xy) + nand_B
rodCreateRect(
?name "Cin"
?layer list("M2" "pin")
?width 0.1
?length 0.1
?origin (Pin_pos - 0.05):(channel_4- 0.05)
?cvId cv

?netName "Cin"
?termName "Cin"
?termIOType "input"
?pin t
?pinLabel t
?pinLabelHeight 0.1
?pinLabelLayer list("M2" "pin")
)

;————————————Output Cout ————————————;
inst = dbFindAnyInstByName(cv "Inv13")
Pin_pos = car(inst >xy) + nth(4 nth(G13 Inv_Lib))
rodCreateRect(

?name "Cout"
?layer list("M2" "pin")
?width 0.1
```

```
?length 0.1
?origin (Pin_pos - 0.05):(channel_3- 0.05)
?cvId cv

?netName "Cout"
?termName "Cout"
?termIOType "output"
?pin t
?pinLabel t
?pinLabelHeight 0.1
?pinLabelLayer list("M2" "pin")
)
;————————————Output S ————————————;
inst = dbFindAnyInstByName(cv "Inv17")
Pin_pos = car(inst >xy) + nth(4 nth(G17 Inv_Lib))
rodCreateRect(

?name "S"
?layer list("M2" "pin")
?width 0.1
?length 0.1
?origin (Pin_pos - 0.05):(channel_1- 0.05)
?cvId cv

?netName "S"
?termName "S"
?termIOType "output"
?pin t
?pinLabel t
?pinLabelHeight 0.1
?pinLabelLayer list("M2" "pin")
)

dbSave(cv)
dbClose(cv)
)

let((G0 G1 G2 G3 G4 G5 G6 G7 G8 G9 G10 G11 G12 G13 G14 G15 G16 G17)
G0 = 9
G1 = 9
G2 = 1
G3 = 2
G4 = 0
G5 = 1
```

G6 = 0
G7 = 1
G8 = 2
G9 = 0
G10 = 0
G11 = 0
G12 = 1
G13 = 0
G14 = 3
G15 = 0
G16 = 4
G17 = 0
StripeLayoutCreate(G0 G1 G2 G3 G4 G5 G6 G7 G8 G9 G10 G11 G12 G13 G14 G15 G16 G17)
)

Abbreviations

2D Two Dimensions

3D Three Dimensions

ALU Arithmetic Logic Unit

ASIC Application Specific Integrated Circuit

CAD Computer-aid Design

CCEA Cooperative Co-evolutionary Algorithm

CMOS Complementary Metal-Oxide-Semiconductor

DRC Design Rule Checking

DSE Design Space Exploration

EA Evolutionary Algorithm

ECO Engineering Change Order

EDA Electronic Design Automation

FET Field-effect Transistor

GAA All-Gate-Around

GDSII Graphic Design System II

HDL Hardware Description Language

HLS High-level Synthesis

HVT High Voltage Threshold

I/O Input/Output

IC Integrated Circuit

ID Identity

IEEE Institute of Electrical and Electronics Engineers

IP Intellectual Property

IRDS International Roadmap for Devices and Systems

ISCAS International Symposium on Circuits and Systems

ITRS International Technology Roadmap for Semiconductors

LR Lagrangian Relaxation

LVS Layout Versus Schematic

LVT Low Voltage Threshold

MAB Multi-armed Bandit

MODSE Multi-objective Design Space Exploration

MOEA Multi-objective Evolutionary Algorithm

MOEDA Multi-objective Electronic Design Automation

MOGA Multi-objective Genetic Algorithm

MOP Multi-objective Problem

MOSFET Metal-Oxide-Semiconductor Field-Effect Transistor

MOS Metal-Oxide-Semiconductor

MO Multi-objective

NLDM Non-linear Delay Model

NMOS N-type Metal-Oxide-Semiconductor

NSGA-II Non-dominated Sorting Genetic Algorithm-II

NSGA Non-dominated Sorting Genetic Algorithm

NoC Network-on-chip

PDK Process Design Kit

PEX Parasitic Extraction

PMOS P-type Metal-Oxide-Semiconductor

PPA Power, Performance and Area

PVT Process Voltage Temperature

QoR Quality of Results

RTL Register-Transfer Level

SI Signal Integrity

SPEA2 Strength Pareto Evolutionary Algorithm 2

SPEA Strength Pareto Evolutionary Algorithm

SPICE Simulation Program with Integrated Circuit Emphasis

STA Static Timing Analysis

SVT Standard Voltage Threshold

SoC System-on-Chip

TNS Total Negative Slack

TSMC Taiwan Semiconductor Manufacturing Company

VEGA Vector Evaluated Genetic Algorithm

VHDL VHSIC Hardware Description Language

VHSIC Very High Speed Integrated Circuit

VLSI Very Large Scale Integration

WNS Worst Negative Slack

References

[1] Wikipedia, "Moore's law — Wikipedia, The Free Encyclopedia," http://en.wikipedia.org/wiki/Moore%27s_law, [Online; accessed 15-May-2021].

[2] P. McLellan, "Imec's plan for continued scaling," https://semiengineering.com/imecs-plan-for-continued-scaling/, 2021, online; Accessed 2021-01-01.

[3] L. Cao, S. J. Bale, and M. A. Trefzer, "Instrumenting parametric physical layout for multi-objective optimisation," in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2018, pp. 1339–1345.

[4] L. Cao, S. J. Bale, and M. A. Trefzer, "Multi-objective optimisation of digital circuits based on cell mapping in an industrial eda flow," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2021, First Revision.

[5] L. Cao, S. J. Bale, and M. A. Trefzer, "Multi-objective digital design optimisation via improved drive granularity standard cells," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2021, in Press/Accepted.

[6] L. Cao, S. J. Bale, and M. A. Trefzer, "Multi-objective design optimisation for handling complex floorplan constraints," *Proc. Design, Automation and Test in Europe (DATE)*, 2022, Under Review.

[7] "International technology roadmap for semiconductors (itrs)," http://www.itrs2.net/, 2001, online; Accessed 2021-01-01.

[8] "International roadmap for devices and systems (irds)," https://irds.ieee.org/, 2020, online; Accessed 2021-01-01.

[9] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu, *VLSI physical design: from graph partitioning to timing closure*. Springer Science & Business Media, 2011.

[10] L. Lavagno, I. L. Markov, G. Martin, and L. K. Scheffer, *Electronic Design Automation for IC Implementation, Circuit Design, and Process Technology: Circuit Design, and Process Technology*. CRC Press, 2016.

[11] A. Sengupta, "Design flow of a digital ic: The role of digital ic\/soc design in ce products," *IEEE Consumer Electronics Magazine*, vol. 5, no. 2, pp. 58–62, 2016.

[12] G. E. Moore *et al.*, "Cramming more components onto integrated circuits," 1965.

[13] N. H. Weste and D. Harris, *CMOS VLSI design: a circuits and systems perspective*. Pearson Education India, 2015.

[14] R. H. Dennard, F. H. Gaensslen, H.-N. Yu, V. L. Rideout, E. Bassous, and A. R. 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, 1974.

[15] M. T. Bohr and I. A. Young, "Cmos scaling trends and beyond," *IEEE Micro*, vol. 37, no. 6, pp. 20–29, 2017.

[16] E. N. Shauly, "Cmos leakage and power reduction in transistors and circuits: process and layout considerations," *Journal of Low Power Electronics and Applications*, vol. 2, no. 1, pp. 1–29, 2012.

[17] "Wikichip," https://en.wikichip.org/, online; Accessed 2021-01-01.

[18] "Cello library creation, migration and optimization," https://silvaco.com/, online; Accessed 2020-01-10.

[19] G. A. Northrop and P.-F. Lu, "A semi-custom design flow in high-performance microprocessor design," in *Proceedings of the 38th Design Automation Conference*. IEEE, 2001, pp. 426–431.

[20] T. Hashimoto, "Layout generation of primitive cells with variable driving strength," *Proc. SASIMI, 2000*, 2000.

[21] M. Hashimoto, K. Fujimori, and H. Onodera, "Standard cell libraries with various driving strength cells for 0.13, 0.18 and 0.35/spl mu/m technologies," in *Proceedings of (ASP-DAC) Asia and South Pacific Design Automation Conference*. IEEE, 2003, pp. 589–590.

[22] H. Onodera, M. Hashimoto, and T. Hashimoto, "ASIC design methodology with on-demand library generation," in *2001 Symposium on VLSI Circuits. Digest of Technical Papers.* IEEE, 2001, pp. 57–60.

[23] M. Hashimoto and H. Onodera, "Post-layout transistor sizing for power reduction in cell-base design," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 84, no. 11, pp. 2769–2777, 2001.

[24] S. Nishizawa, T. Ishihara, and H. Onodera, "Layout generator with flexible grid assignment for area efficient standard cell," *IPSJ Transactions on System LSI Design Methodology*, vol. 8, pp. 131–135, 2015.

[25] J. Zhou, S. Jayapal, B. Busze, L. Huang, and J. Stuyt, "A 40 nm dual-width standard cell library for near/sub-threshold operation," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 11, pp. 2569–2577, 2012.

[26] M. Kondo, S. Nishizawa, T. Ishihara, and H. Onodera, "A standard cell optimization method for near-threshold voltage operations," in *International Workshop on Power and Timing Modeling, Optimization and Simulation.* Springer, 2012, pp. 32–41.

[27] J. Morris, P. Prabhat, J. Myers, and A. Yakovlev, "Unconventional layout techniques for a high performance, low variability subthreshold standard cell library," in *VLSI (ISVLSI), 2017 IEEE Computer Society Annual Symposium on.* IEEE, 2017, pp. 19–24.

[28] J. Jun, J. Song, and C. Kim, "A near-threshold voltage oriented digital cell library for high-energy efficiency and optimized performance in 65nm cmos process," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 5, pp. 1567–1580, 2017.

[29] A. M. Islam, S. Nishizawa, Y. Matsui, and Y. Ichida, "Drive-strength selection for synthesis of leakage-dominant circuits," in *2019 International Symposium on Quality Electronic Design (ISQED).* IEEE, 2019, pp. 298–303.

[30] N. A. Sherwani, *Algorithms for VLSI physical design automation.* Springer Science & Business Media, 2012.

[31] C. J. Alpert, D. P. Mehta, and S. S. Sapatnekar, *Handbook of algorithms for physical design automation.* CRC press, 2008.

[32] A. K. Yella, G. Srivatsa, and C. Sechen, "Are standalone gate size and VT optimization tools useful?" in *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*. IEEE, 2017, pp. 1–6.

[33] W. J. Dally and A. Chang, "The role of custom design in ASIC chips," in *Proceedings of 37th Design Automation Conference (DAC)*, 2000, pp. 643–647.

[34] D. G. Chinnery and K. Keutzer, "Closing the gap between ASIC and custom: an ASIC perspective," in *Proceedings of 37th Design Automation Conference (DAC)*, 2000, pp. 637–642.

[35] ——, "Closing the power gap between ASIC and custom: an ASIC perspective," in *Proceedings of 42nd Design Automation Conference (DAC)*, 2005, pp. 275–280.

[36] ——, "High performance and low power design techniques for ASIC and custom in nanometer technologies," in *Proceedings of 2013 ACM International Symposium on Physical Design*, 2013, pp. 25–32.

[37] J. Chen, I. H.-R. Jiang, J. Jung, A. B. Kahng, V. N. Kravets, Y.-L. Li, S.-T. Lin, and M. Woo, "Datc rdf-2019: Towards a complete academic reference design flow," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2019, pp. 1–6.

[38] T. Ajayi, D. Blaauw, T. Chan, C. Cheng, V. Chhabria, D. Choo, M. Coltella, S. Dobre, R. Dreslinski, M. Fogaça *et al.*, "Openroad: Toward a self-driving, open-source digital layout implementation tool chain," *Proceedings of GOMACTECH*, pp. 1105–1110, 2019.

[39] T. Ajayi, V. A. Chhabria, M. Fogaça, S. Hashemi, A. Hosny, A. B. Kahng, M. Kim, J. Lee, U. Mallappa, M. Neseem *et al.*, "Toward an open-source digital flow: First learnings from the openroad project," in *Proceedings of 56th Design Automation Conference (DAC)*, 2019, pp. 1–4.

[40] C. A. C. Coello, G. B. Lamont, D. A. Van Veldhuizen *et al.*, *Evolutionary algorithms for solving multi-objective problems*. Springer, 2007, vol. 5.

[41] C. A. C. Coello, "A short tutorial on evolutionary multiobjective optimization," in *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 2001, pp. 21–40.

[42] V. Pareto, *Cours d'économie politique*. Librairie Droz, 1964, vol. 1.

[43] K. Miettinen, *Nonlinear multiobjective optimization.* Springer Science & Business Media, 1999, vol. 12.

[44] R. T. Marler and J. S. Arora, "The weighted sum method for multi-objective optimization: new insights," *Structural and multidisciplinary optimization*, vol. 41, no. 6, pp. 853–862, 2010.

[45] L. Zadeh, "Optimality and non-scalar-valued performance criteria," *IEEE transactions on Automatic Control*, vol. 8, no. 1, pp. 59–60, 1963.

[46] A. Konak, D. W. Coit, and A. E. Smith, "Multi-objective optimization using genetic algorithms: A tutorial," *Reliability Engineering & System Safety*, vol. 91, no. 9, pp. 992–1007, 2006.

[47] M. Ehrgott, *Multicriteria optimization.* Springer Science & Business Media, 2005, vol. 491.

[48] G. Chiandussi, M. Codegone, S. Ferrero, and F. E. Varesio, "Comparison of multi-objective optimization methodologies for engineering applications," *Computers & Mathematics with Applications*, vol. 63, no. 5, pp. 912–942, 2012.

[49] I. Giagkiozis and P. J. Fleming, "Methods for multi-objective optimization: An analysis," *Information Sciences*, vol. 293, pp. 338–350, 2015.

[50] R. Wang, Z. Zhou, H. Ishibuchi, T. Liao, and T. Zhang, "Localized weighted sum method for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 3–18, 2018.

[51] D. G. Chinnery and K. Keutzer, "Linear programming for sizing, vth and vdd assignment," in *2005 International Symposium on Quality Electronic Design (ISQED)*, 2005, pp. 149–154.

[52] K. Jeong, A. B. Kahng, and H. Yao, "Revisiting the linear programming framework for leakage power vs. performance optimization," in *2009 International Symposium on Quality Electronic Design (ISQED).* IEEE, 2009, pp. 127–134.

[53] A. Farshidi, L. Rakai, L. Behjat, and D. Westwick, "A self-tuning multi-objective optimization framework for geometric programming with gate sizing applications," in *Proceedings of 23rd ACM Great Lakes Symposium on VLSI*, 2013, pp. 305–310.

[54] ——, "Optimal gate sizing using a self-tuning multi-objective framework," *Integration*, vol. 47, no. 3, pp. 347–355, 2014.

[55] A. B. Kahng, S. Kang, H. Lee, I. L. Markov, and P. Thapar, "High-performance gate sizing with a signoff timer," in *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2013, pp. 450–457.

[56] F. Kashfi, S. Hatami, and M. Pedram, "Multi-objective optimization techniques for vlsi circuits," in *2011 International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2011, pp. 1–8.

[57] C. C. Coello, "Evolutionary multi-objective optimization: a historical view of the field," *IEEE computational intelligence magazine*, vol. 1, no. 1, pp. 28–36, 2006.

[58] Y. Wang, "Circuit clustering for cluster-based fpgas using novel multiobjective genetic algorithms," *PhD Thesis, University of York*, 2015.

[59] M. A. Trefzer and A. M. Tyrrell, *Evolvable hardware: From practice to application*. Springer, 2015.

[60] A. E. Eiben, J. E. Smith *et al.*, *Introduction to evolutionary computing*. Springer, 2003, vol. 53.

[61] K. A. De Jong and W. M. Spears, "A formal analysis of the role of multi-point crossover in genetic algorithms," *Annals of mathematics and Artificial intelligence*, vol. 5, no. 1, pp. 1–26, 1992.

[62] C. A. C. Coello, S. G. Brambila, J. F. Gamboa, M. G. C. Tapia, and R. H. Gómez, "Evolutionary multiobjective optimization: open research areas and some challenges lying ahead," *Complex & Intelligent Systems*, vol. 6, no. 2, pp. 221–236, 2020.

[63] I. Das and J. E. Dennis, "A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems," *Structural optimization*, vol. 14, no. 1, pp. 63–69, 1997.

[64] H. R. Maier, S. Razavi, Z. Kapelan, L. S. Matott, J. Kasprzyk, and B. A. Tolson, "Introductory overview: Optimization using evolutionary algorithms and other metaheuristics," *Environmental Modelling & Software*, vol. 114, pp. 195–213, 2019.

[65] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proceedings of the First International Conference on Genetic*

Algorithms and Their Applications, 1985. Lawrence Erlbaum Associates. Inc., Publishers, 1985.

[66] C. M. Fonseca, P. J. Fleming *et al.*, "Genetic algorithms for multiobjective optimization: Formulation discussion and generalization." in *Icga*, vol. 93, no. July, 1993, pp. 416–423.

[67] N. Srinivas and K. Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary computation*, vol. 2, no. 3, pp. 221–248, 1994.

[68] E. Zitzler and L. Thiele, "An evolutionary algorithm for multiobjective optimization: The strength pareto approach," *TIK-report*, vol. 43, 1998.

[69] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm," *TIK-report*, vol. 103, 2001.

[70] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[71] R. Drechsler, "Evolutionary algorithms for vlsi cad [book review]," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 3, pp. 251–253, 1999.

[72] J. Cohoon, J. Kairo, and J. Lienig, "Evolutionary algorithms for the physical design of vlsi circuits," in *Advances in Evolutionary Computing*. Springer, 2003, pp. 683–711.

[73] B. Becker and R. Drechsler, "Ofdd based minimization of fixed polarity reed-muller expressions using hybrid genetic algorithms," in *Proceedings 1994 IEEE International Conference on Computer Design: VLSI in Computers and Processors*. IEEE, 1994, pp. 106–110.

[74] K. Ohmori and T. Kasai, "Logic synthesis using a genetic algorithm," in *1997 IEEE International Conference on Intelligent Processing Systems*, vol. 1. IEEE, 1997, pp. 137–142.

[75] P. Thomson and J. F. Miller, "Comparison of and-xor logic synthesis using a genetic algorithm against misii for implementation on fpgas," in *Second International Conference On Genetic Algorithms In Engineering Systems: Innovations And Applications*. IET, 1997.

[76] R. Drechsler and W. Günther, "Evolutionary synthesis of multiplexor circuits under hardware constraints," in *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation*, 2000, pp. 513–518.

[77] T. N. Bui and B. R. Moon, "A fast and stable hybrid genetic algorithm for the ratio-cut partitioning problem on hypergraphs," in *31st Design Automation Conference*. IEEE, 1994, pp. 664–669.

[78] S. M. Sait, A. H. El-Maleh, and R. H. Al-Abaji, "Evolutionary algorithms for vlsi multi-objective netlist partitioning," *Engineering Applications of Artificial Intelligence*, vol. 19, no. 3, pp. 257–268, 2006.

[79] M. Bhuvaneswari, *Application of evolutionary algorithms for multi-objective optimization in VLSI and embedded systems*. Springer, 2014.

[80] M. Rebaudengo and M. S. Reorda, "Gallo: A genetic algorithm for floorplan area optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 8, pp. 943–951, 1996.

[81] C. L. Valenzuela and P. Y. Wang, "Vlsi placement and area optimization using a genetic algorithm to breed normalized postfix expressions," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 4, pp. 390–401, 2002.

[82] M. Tang and X. Yao, "A memetic algorithm for vlsi floorplanning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 1, pp. 62–69, 2007.

[83] J. Chen and W. Zhu, "A hybrid genetic algorithm for vlsi floorplanning," in *2010 IEEE International Conference on Intelligent Computing and Intelligent Systems*, vol. 2. IEEE, 2010, pp. 128–132.

[84] R. M. Kling and P. Banerjee, "Esp: Placement by simulated evolution," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 8, no. 3, pp. 245–256, 1989.

[85] K. Shahookar and P. Mazumder, "A genetic approach to standard cell placement using meta-genetic parameter optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 9, no. 5, pp. 500–511, 1990.

[86] V. Schnecke and O. Vornberger, "An adaptive parallel genetic algorithm for vlsi-layout optimization," in *International Conference on Parallel Problem Solving from Nature*. Springer, 1996, pp. 859–868.

[87] H. A. Rahim, R. B. Ahmad, W. N. S. F. W. Ariffin, M. I. Ahmad *et al.*, "The performance study of two genetic algorithm approaches for vlsi macro-cell layout area optimization," in *2008 2nd Asia International Conference on Modelling & Simulation (AMS)*. IEEE, 2008, pp. 207–212.

[88] J. Lienig and K. Thulasiraman, "A genetic algorithm for channel routing in vlsi circuits," *Evolutionary Computation*, vol. 1, no. 4, pp. 293–311, 1993.

[89] J. Lienig, "A parallel genetic algorithm for performance-driven vlsi routing," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 29–39, 1997.

[90] A. Ricci, I. De Munari, and P. Ciampolini, "An evolutionary approach for standard-cell library reduction," in *Proceedings of the 17th ACM Great Lakes symposium on VLSI*. ACM, 2007, pp. 305–310.

[91] M. K. Papamichael, P. Milder, and J. C. Hoe, "Nautilus: Fast automated ip design space search using guided genetic algorithms," in *Proceedings of the 52nd Annual Design Automation Conference*, 2015, pp. 1–6.

[92] M. Palesi and T. Givargis, "Multi-objective design space exploration using genetic algorithms," in *Proceedings of 10th International Symposium on Hardware/software Codesign*, 2002, pp. 67–72.

[93] G. Ascia, V. Catania, and M. Palesi, "A framework for design space exploration of parameterized vlsi systems," in *Proceedings of ASP-DAC/VLSI Design 2002. 7th Asia and South Pacific Design Automation Conference and 15th International Conference on VLSI Design*. IEEE, 2002, pp. 245–250.

[94] C. Erbas, S. Cerav-Erbas, and A. D. Pimentel, "Multiobjective optimization and evolutionary algorithms for the application mapping problem in multiprocessor system-on-chip design," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 358–374, 2006.

[95] S. Dey, S. Nandi, and G. Trivedi, "Pgopt: Multi-objective design space exploration framework for large-scale on-chip power grid design in vlsi soc using evolutionary computing technique," *Microprocessors and Microsystems*, vol. 81, p. 103440, 2021.

[96] W. Sheng, L. Xiao, and Z. Mao, "Soft error optimization of standard cell circuits based on gate sizing and multi-objective genetic algorithm," in *Proceedings of 46th Design Automation Conference (DAC)*, 2009, pp. 502–507.

[97] B. Yuan, H. Chen, and X. Yao, "Toward efficient design space exploration for fault-tolerant multiprocessor systems," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 1, pp. 157–169, 2020.

[98] V. Krishnan and S. Katkoori, "A genetic algorithm for the design space exploration of datapaths during high-level synthesis," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 213–229, 2006.

[99] F. Ferrandi, P. L. Lanzi, D. Loiacono, C. Pilato, and D. Sciuto, "A multi-objective genetic algorithm for design space exploration in high-level synthesis," in *2008 IEEE Computer Society Annual Symposium on VLSI*. IEEE, 2008, pp. 417–422.

[100] M. Holzer, B. Knerr, and M. Rupp, "Design space exploration with evolutionary multi-objective optimisation," in *Industrial Embedded Systems, 2007. SIES'07. International Symposium on*. IEEE, 2007, pp. 126–133.

[101] "Virtuoso layout suite," https://www.cadence.com, Cadence Design Systems.

[102] "Calibre," https://eda.sw.siemens.com/, Mentor Graphics.

[103] "Eldo," https://eda.sw.siemens.com/, Mentor Graphics.

[104] W. H. Kao, C.-Y. Lo, M. Basel, and R. Singh, "Parasitic extraction: current state of the art and future trends," *Proceedings of the IEEE*, vol. 89, no. 5, pp. 729–739, 2001.

[105] R. Tayrani, J. E. Gerber, T. Daniel, R. S. Pengelly, and U. L. Rohde, "A new and reliable direct parasitic extraction method for mesfets and hemts," in *Microwave Conference, 1993. 23rd European*. IEEE, 1993, pp. 451–453.

[106] G. Flach, T. Reimann, G. Posser, M. Johann, and R. Reis, "Effective method for simultaneous gate sizing and v th assignment using lagrangian relaxation," *IEEE Transactions on computer-aided design of integrated circuits and systems*, vol. 33, no. 4, pp. 546–557, 2014.

[107] T. J. Reimann, C. C. Sze, and R. Reis, "Cell selection for high-performance designs in an industrial design flow," in *Proceedings of 2016 ACM International Symposium on Physical Design*, 2016, pp. 65–72.

[108] A. Sharma, D. Chinnery, and C. Chu, "Lagrangian relaxation based gate sizing with clock skew scheduling-a fast and effective approach," in *Proceedings of 2019 International Symposium on Physical Design*, 2019, pp. 129–137.

[109] J. Hu, A. B. Kahng, S. Kang, M.-C. Kim, and I. L. Markov, "Sensitivity-guided metaheuristics for accurate discrete gate sizing," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2012, pp. 233–239.

[110] H. Fatemi, A. B. Kahng, H. Lee, J. Li, and J. P. de Gyvez, "Enhancing sensitivity-based power reduction for an industry ic design context," *Integration*, vol. 66, pp. 96–111, 2019.

[111] T. Reimann, G. Posser, G. Flach, M. Johann, and R. Reis, "Simultaneous gate sizing and vt assignment using fanin/fanout ratio and simulated annealing," in *2013 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2013, pp. 2549–2552.

[112] X.-D. Wang and T. Chen, "Performance and area optimization of vlsi systems using genetic algorithms," *VLSI Design*, vol. 3, no. 1, pp. 43–51, 1995.

[113] S. Benkhider, F. Boumghar, and A. Baba-ali, "A parallel genetic approach to the gate sizing problem of vlsi integrated circuits," in *Proceedings of of the 12th International Conference on Microelectronics*. IEEE, 2000, pp. 169–173.

[114] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran," in *Proceedings of IEEE International Symposium Circuits and Systems (ISCAS 85)*. IEEE Press, Piscataway, N.J., 1985, pp. 677–692.

[115] L. Amarú, P.-E. Gaillardon, and G. De Micheli, "The epfl combinational benchmark suite," in *Proceedings of the 24th International Workshop on Logic & Synthesis (IWLS)*, 2015.

[116] "Genus synthesis solution," https://www.cadence.com, Cadence Design Systems.

[117] "Innovus implementation system," https://www.cadence.com, Cadence Design Systems.

[118] M. Anwar, S. Saha, M. M. Ziegler, and L. Reddy, "Early scenario pruning for efficient design space exploration in physical synthesis," in *2016 29th International*

Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID). IEEE, 2016, pp. 116–121.

[119] M. M. Ziegler, H.-Y. Liu, G. Gristede, B. Owens, R. Nigaglioni, and L. P. Carloni, "A synthesis-parameter tuning system for autonomous design-space exploration," in *2016 Design, Automation & Test in Europe, (DATE)*. IEEE, 2016, pp. 1148–1151.

[120] A. B. Kahng, S. Kumar, and T. Shah, "A no-human-in-the-loop methodology toward optimal utilization of eda tools and flows," *Proceedings of DAC, WIP Track*, 2018.

[121] S. Roy, Y. Ma, J. Miao, and B. Yu, "A learning bridge from architectural synthesis to physical design for exploring power efficient high-performance adders," in *2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, 2017, pp. 1–6.

[122] Y. Ma, S. Roy, J. Miao, J. Chen, and B. Yu, "Cross-layer optimization for high speed adders: A pareto driven machine learning approach," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 12, pp. 2298–2311, 2019.

[123] J. Kwon, M. M. Ziegler, and L. P. Carloni, "A learning-based recommender system for autotuning design fiows of industrial high-performance processors," in *Proceedings of 56th Design Automation Conference (DAC)*. IEEE, 2019, pp. 1–6.

[124] D. Hyun, Y. Fan, and Y. Shin, "Accurate wirelength prediction for placement-aware synthesis through machine learning," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 324–327.

[125] A. B. Kahng, "Mlcad today and tomorrow: Learning, optimization and scaling," in *Proceedings of the 2020 ACM/IEEE Workshop on Machine Learning for CAD*, 2020, pp. 1–1.

[126] X. Zhang and M. X. Wang, "Standard cell library having cell drive strengths selected according to delay," Sep. 16 2008, uS Patent 7,426,710.

[127] X. Xu, N. Shah, A. Evans, S. Sinha, B. Cline, and G. Yeric, "Standard cell library design and optimization methodology for asap7 pdk," in *2017 IEEE/ACM*

International Conference on Computer-Aided Design (ICCAD). IEEE, 2017, pp. 999–1004.

[128] S. Dobre, A. B. Kahng, and J. Li, "Mixed cell-height implementation for improved design quality in advanced nodes," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2015, pp. 854–860.

[129] Y.-X. Chiang, C.-W. Tai, S.-R. Fang, K.-C. Peng, Y.-D. Chung, J.-K. Yang, and R.-B. Lin, "Designing and benchmarking of double-row height standard cells," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2018, pp. 64–69.

[130] Y.-C. Zhao, Y.-C. Lin, T.-C. Wang, T.-H. Wang, Y.-R. Wu, H.-C. Lin, and S.-Y. Kao, "A mixed-height standard cell placement flow for digital circuit blocks," in *2019 Design, Automation & Test in Europe, (DATE)*. IEEE, 2019, pp. 328–331.

[131] Z. Zhu, J. Chen, W. Zhu, and Y.-W. Chang, "Mixed-cell-height legalization considering technology and region constraints," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 12, pp. 5128–5141, 2020.

[132] "Liberate characterization," https://www.cadence.com, Cadence Design Systems.

[133] "Virtuoso abstract generator," https://www.cadence.com, Cadence Design Systems.