UNIVERSITY *of York*

This is a repository copy of *Multi-objective digital circuit block optimisation based on cell mapping in an industrial electronic design automation flow*.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/id/eprint/202333/

Version: Published Version

Article:

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

White Rose
university consortium
Universities of Leeds, Sheffield & York

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk/

IET Computers & Digital Techniques

The Institution of Engineering and Technology WILEY

ORIGINAL RESEARCH

Multi-objective digital circuit block optimisation based on cell mapping in an industrial electronic design automation flow

Linan Cao [ID] | **Simon J. Bale** | **Martin A. Trefzer** [ID]

School of Physics, Engineering, and Technology, University of York, York, UK

Correspondence

Linan Cao.
Email: lc1492@york.ac.uk

Abstract

Modern electronic design automation (EDA) tools can handle the complexity of state-of-the-art electronic systems by decomposing them into smaller blocks or cells, introducing different levels of abstraction and staged design flows. However, throughout each independently optimised design step, overheads and inefficiencies can accumulate in the resulting overall design. Performing design-specific optimisation from a more global viewpoint requires more time due to the larger search space but has the potential to provide solutions with improved performanc. In this work, a fully-automated, multi-objective (MO) EDA flow is introduced to address this issue. It specifically tunes drive strength mapping, prior to physical implementation, through MO population-based search algorithms. Designs are evaluated with respect to their power, performance and area (PPA). The proposed approach is aimed at digital circuit optimisation at the block level, where it is capable of expanding the design space and offers a set of trade-off solutions for different case-specific utilisation. We have applied the proposed multi-objective electronic design automation flow (MOEDA) framework to ISCAS-85 and EPFL benchmark circuits by using a commercial 65 nm 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.

KEYWORDS

circuit optimisation, electronic design automation, integrated circuit design, VLSI

1 | INTRODUCTION

The process of building a digital integrated circuit (IC) using blocks or cells from a foundry is a common and mature approach in modern digital VLSI design. Comprehensive industry-standard electronic design automation (EDA) flows are available to tape out digital chips. Technology down-scaling enables high-density integrated circuits, and the EDA tools, therefore, need to handle a large quantity of cells during the flow. To find possible optimal trade-off solutions in regard to power, performance and area (PPA) using appropriate library cells while consuming less turnaround time is the challenge of design optimisation [1].

Standard cell libraries typically contain a large number of functions, and each function has multiple cells differing in drive strength. This enables numerous possible combinations of logic functions or drive strengths depending on the design specifications and the required loads in circuit paths. The possible design space is thus huge and complex because a circuit might be composed of millions of gates. Different combinations of gates (drive strengths) thus can directly determine the PPA metrics of a circuit.

In addition, the parameter search space when building and optimising digital ICs 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 objectives simultaneously while satisfying all rules and constraints at the layout level, which might be beyond what experienced engineers can manually handle. Automatic efficient design space exploration approaches promise to balance multiple design objectives. Researchers both from academia and industry have focussed on investigating design

space in the synthesis, place and route flow or up to the system level and applying optimisation in the flow. Several techniques have been adopted such as heuristic techniques [2], machine learning (ML) [3, 4] and design-parameter tuning [5–7].

Population-based metaheuristic optimisation algorithms like multi-objective evolutionary algorithms (MOEAs) are widely-used existing techniques that can efficiently perform design space exploration and ultimately find a set of Pareto-optimised solutions. Many publications exist on applying EAs or genetic algorithms (GAs) to the VLSI design process from the system level down to the physical level, which also includes optimisation and design space exploration on individual design levels, such as standard cell library depletion [8], macro-cell placement optimisation [9], gate-sizing-based soft error optimisation [10], netlist partitioning [11], circuit equivalence checking [12] and system-on-chip (SoC) design space optimisation [13, 14].

However, limited research investigates how multi-objective (MO) optimisation techniques can fully integrate into industrial synthesis, place and route flows, and how well MOEAs can work in optimising designs down to physical layouts. An automated MO optimisation flow crossing different design levels from a global perspective is required to recover performance which may otherwise be lost in generic overheads spread across the hierarchical design process.

This paper proposes a population-based evolutionary search to maintain the optimisation in multiple objectives through refining drive strengths of logic gates and applies it to a standard digital flow to enhance the design solution in the loop. Due to the scaling behaviour of the problem domain and the optimisation algorithm, the proposed optimisation approach is best-suited for the MO design of IP/block-level circuits. The main contributions of this work are summarised as follows: (1) A 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 circuit netlists. (3) Enhanced trade-off design solutions with improved PPA metrics. (4) A methodology to seed the MOEA with a solution population across different circuit topologies for MO design space optimisation. (5) Improved coverage of the feasible design space providing a set of Pareto-optimised solutions.

The paper is structured as follows: Section 2 gives an overview of related work. Section 3 introduces the proposed multi-objective electronic design automation flow (MOEDA). An experimental setup is described in Section 4. Section 5 presents the MO optimisation results of each benchmark used. Section 6 presents the analysis of tool-generated design space and the MO design space exploration based on it. Section 7 provides conclusions.

2 | RELATED WORK

2.1 | Design flow modifications

Modern digital IC design flow is a mature EDA process including various steps from register-transfer level (RTL)

design, logic synthesis to physical implementation. As each step introduces its own level of abstraction (e.g., from cells to functions, from functions to blocks), any margin or error introduced will therefore 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. In addition, the abstraction introduced in each step may speed-up evaluation at the cost of optimal performance. Furthermore, standard cell libraries from a foundry do not allow transistor resizing or cell layout modifications when they are used in the digital flow. These limits may prevent EDA tools from making full use of the capability of a process technology.

In previous work [15], we introduced a customised MO automatic design flow to adjust parameterised circuit layouts. This optimisation flow exploited a method to tune cell drive strengths using a scripted layout template, which aimed to achieve improved solutions in delay, energy and area.

Chinnery stated that there is a gap between full-custom design and standard digital flow in terms of speed and power [16, 17] in the 2000s. 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. But designers in industry still focus on synthesis-centred methodology to save design efficiency due to the today's time-to-market pressure.

Implementing extra custom design and optimisation techniques compensating to the standard digital flow can achieve better results [18]. Dally proposed to selectively apply a set of custom design methods in the digital flow, including custom floor-planning, place and route critical signals, to achieve the most compact layout structure [19]. To accelerate custom design, [20] introduced an ASIC design methodology with on-demand library generation in the digital flow producing cells with tailored drive strengths from a set of symbolic layouts.

2.2 | Design space exploration using standard digital flow

Optimisation using steps of an industrial EDA flow in the loop can be viewed as black-box design space exploration. While many of the algorithms used in 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 or ML approach to fully utilise the optimisation potential that the tools are capable of.

Kahng presented in Ref. [7] that there is unpredictable 'noisiness' in tool-generated solutions causing variability in the resulting PPA metrics, and 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 Ref. [6], an automated method to explore the search space via tuning parameters at the synthesis step for MO optimisation in a rank-based iterative process is proposed.

Running through the whole design flow leads to more computing resource consumption. In Ref. [5], an automated selection mechanism based on searching the design space in parallel while pruning non-competitive solutions at early stage is exploited, rather than propagating through the entire design flow. In Ref. [3], ML approaches were employed to bridge the synthesis solution space to the physical solution space using a weighted sum cost function for solution evaluation, which aims to enable Pareto-driven exploration for high speed and power efficient adder designs. In Ref. [21], the authors propose ML-based methodologies to predict the actual wirelength of designs for a better early-stage performance analysis. ML-based approaches are efficient to search a complex design space. However, ML techniques heavily rely on huge amounts of training data, which is not always readily available, possibly due to confidentiality in the IC design area. Such issues are not present in the proposed MOEA. In EDA, the training process of MLs, which is compute and time-intensive, also cannot be overlooked and puts requirements of other optimisation approaches in perspective.

2.3 | Discrete gate sizing for PPA optimisation

Gate sizing is a crucial step for achieving timing closure and power minimisation of ICs. It originally refers to determining transistor widths inside of logic gates to make designs meet constraints. Modern digital flows synthesise designs using a set of pre-defined cells. The optimisation problem thus is shifted to focussing on cell selection, regarding drive strengths and threshold voltage assignment, from discretised gate libraries.

A typical goal of gate sizing is to minimise power consumption while meeting timing requirements [22]. Lagrangian Relaxation (LR) is a recently adapted theory for gate sizing optimisation [23–25], which moves the timing constraints to the objective function weighted by multipliers to penalise the overall results of the objective function. The problem is thus simplified to find the solution of weight factors.

In regard to the optimisation objectives in these LR-based approaches, [23] derived LR associated with finding trade-offs between leakage power and circuit timing [24] expanded the primary objective function (power minimisation) by adding the area objective using an extra weight factor. More recently, the authors in Ref. [25] also considered maximum load capacitance and maximum input slew constraints for simultaneous gate sizing and clock skew scheduling. Although LR has been successfully applied to discrete gate-sizing, it is typically formulated for continuous problems, where it performs more optimally [26, 27].

There are alternative MO gate sizing frameworks, like geometric programming [28, 29], simulated annealing [30], which have been investigated using weighted sum objective functions which is a common scalarising method in MO problems similar to the LR.

In Ref. [26], 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.

However, limited work completed the gate sizing with simultaneously handling all PPA metrics through industrial design flows and libraries, all the way from synthesis and physical implementation, to investigate how beneficial these methods can be in practice [27]. The work in Ref. [31] stated that significant changes in cell sizes, 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 work, typical heuristic techniques like GAs were applied to solving gate sizing problems. The methods for MO optimisation in Refs. [32, 33] both are still based on scalarised cost functions. More recently, gate-sizing-based soft error optimisation using MOEAs is proposed in Ref. [10], but its objectives are soft error rate, critical path delay and area.

2.4 | Summary

VLSI design is MO in nature, often with a need to compromise between several conflicting design goals. A range of methods are developed including design flow revamping with custom-design techniques, intelligent approaches for design space exploration or dedicated design steps in EDA flows. In addition, the weighted sum function is often used in existing gate-sizing work as stated in previous literature. It is a popular linear scalarising approach to decompose the complexity of MO problems since its high search efficiency, and it is inherently used for convex problems [34]. However, the physical characteristics of IC devices imply non-convexities and non-linearity [35], so the weighted sum method is not sufficient to search for feasible Pareto-optimal solutions [34].

Since solving the discrete gate-sizing problem still lacks theoretical guarantees [22] and still has been actively investigating, it is worthwhile to apply global search methods to optimise such a problem. Deterministic algorithms often used in EDA tools can always deliver the same solution for a given input with one execution but might be limited to reach the possible global optimum. The MOEA, handling multiple design parameters and objectives inter-independently is well-suited to perform the global search, particularly regarding a large, complex design space.

3 | MULTI-OBJECTIVE ELECTRONIC DESIGN AUTOMATION FRAMEWORK

3.1 | Preliminaries: Evolutionary algorithms

Evolutionary optimisation algorithms are a class of population-based metaheuristics using mechanisms inspired by biological evolution like reproduction, genetics and natural selection. An initial *population*, which consists of *N individuals* (candidate solutions), is allowed to age with *M* evolutionary *generations*.

The N is referred to the *population size*. The initial population can be either initialised randomly or seeded with a set of specific configurations. During each generation, individuals can be modified through *mutation* or *crossover* (i.e., recombination with each other) variations on their *chromosomes*. All individuals are evaluated using a *fitness* function at the end of each generation. Only the fittest individuals survive the selection process for the subsequent generation. Termination of the evolution process is triggered when specific criteria are met, like sufficient quality of solutions or maximum number of generations.

Applying an EA needs three main preparatory steps:

1) *Definition of representation*. This is the data structure that the EA manipulates. It represents individuals as a set of genes (i.e., a *chromosome*) comprising all variables and parameters necessary to describe it.
2) *Implementation of genetic operations*. Mutation and crossover are commonly applied in the evolution process. Mutation modifies genes of individuals, and crossover combines subsets of genes of multiple individuals to produce new ones.
3) *Definition of a fitness function*. This is used to calculate a fitness score for each individual based on its performance regarding design objectives. The fitness scores are used during the ranking and selection process to determine which individuals survive to form the population for the next generation.

NSGA-II [36], one of the most popular MOEAs, has been adapted as the search tool in this and our related work [37]. The fast non-dominated sorting approach and diversity preservation strategies used ensure convergence while achieving a uniform spread of Pareto-optimal solutions [38]. A conceptual example is shown in Figure 1.

Non-dominated sorting. 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, and the second is a set of solutions that p dominates. The individuals are grouped based on their domination counts into multiple fronts $\mathbf{F} = (\mathbf{F}_1, \ldots, \mathbf{F}_i)$ as shown in Figure 1a. The non-dominated individuals which have the lowest domination counts (i.e., zero) form 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. 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 (*DescendSort*) 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. So as an example shown in Figure 1b, individual p will be preserved.
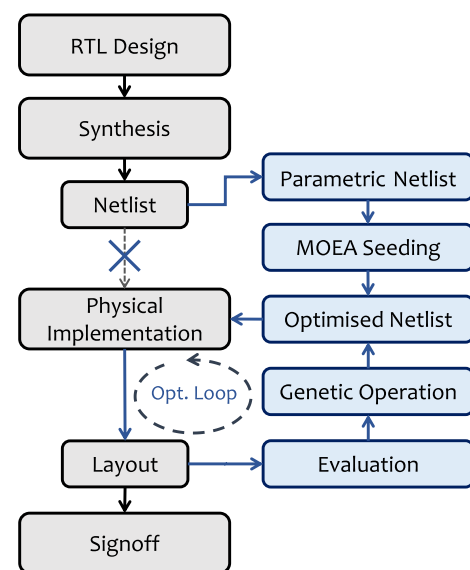
3.2 | Multi-objective (MO) electronic design automation digital flow

The MOEDA digital flow, illustrated in Figure 2, is a fully-automated MO design framework 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

1) *Parametric netlist*. A synthesised netlist is composed of technology-specified logic gates and their connectivity. The MOEA representation encodes the drive strengths of gates
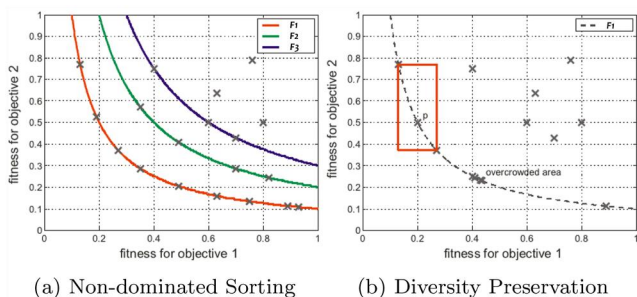
FIGURE 1 The non-dominated sorting technique is illustrated in (a) showing the first three non-dominated frontiers. The diversity preservation strategy is shown in (b).

FIGURE 2 Multi-objective electronic design automation flow (MOEDA) digital flow. The flowchart on the left side is the standard digital flow, and on the right side the multi-objective (MO) extension is shown.

into a set of genes, a string vector **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.

2) *MOEA seeding*. In this work, initial populations are seeded from the 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 optimise them.

3) *Genetic operations*. Only a mutation operator is used in this work. The mutation operation modifies the drive strength of components based on a given probability ρ (i.e., determining how many components out of all will be modified). 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. Crossover is not used because if two circuits with different topology are selected, it is impossible to guarantee the preservation of circuit topology or function, with the direct mapping used here. In this work, focussing on modifying the drive strengths of logic gates using mutation only ensures that the overall function is unchanged. Exclusively applying mutation operation here does not limit the MOEA technique to a random search because all elite individuals are kept and evolved to the following generations using the non-dominated sorting approach working on trade-off solutions across multiple objective dimensions.

4) *Evaluation*. This calculates the fitness scores of each individual. MOEA-optimised netlists are propagated to 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 the post-route stage from the place and route tool. Fitness scores are then fed back to the MOEA for ranking and selection using non-dominated sorting and diversity preservation strategy of NSGA-II in this work.

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

$$f(\mathbf{g}) = \min \quad [D_{wc}(\mathbf{g}), \quad P_{total}(\mathbf{g}), \quad A_{gate}(\mathbf{g})]$$
$$\text{s.t.} \quad \mathbf{g} = (g_1, g_2, ..., g_i), \quad 3 \le g_i \le 11, \quad \forall g_i \in \mathbb{G} \quad (1)$$

where the chromosome vector **g** is the input variables to the fitness function, which are the drive strengths of gates (g_i) selected from a standard cell library ($\mathbb{G}$). There are between three and 11 drive options available for each logic gate in $\mathbb{G}$ according to the commercial library used in this work. Hence, the average synthesised design space size of, for example, the log2 circuit (used in the experiment) is between $3^{10,000}$ and $11^{10,000}$.

Figure 3 demonstrates a population example where $\mathbf{P}_t$ consists of N layout individuals (L_1, L_2,…, L_n). Each L has a
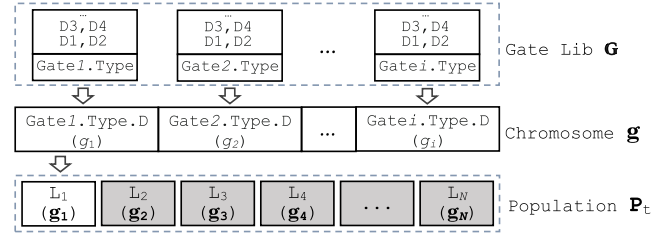
FIGURE 3 A chromosome example of an individual in a population and how each gene is mutated using a logic gate library. 'Gate.Type' represents the logic function of a gate, and 'D' represents its drive strength.

chromosome **g** comprising a set of genes (g_1, g_2, …, g_i). The chromosome overall represents the all logic gates of a netlist. Each single g (Gate.Type.D) represents a logic gate (Gate) including its properties: function type (Type) and drive strength size (D). 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 the all drive strength options (D) of this gate function from $\mathbb{G}$ and finally choose one from them to replace the previous one.

The overall optimisation process, presented in Algorithm 1, is continuously producing different circuit layout instances by adjusting the netlists and keeping improved solutions generation-by-generation.

Algorithm 1 Adapted NSGA-II for MOEDA [36]

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

1: Initialise parent population $\mathbf{P}_t$ in size N ▷ Seed with synthesis-optimised solutions 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:     $\mathbf{P}_{t+1} \leftarrow \varnothing$
8:     $i \leftarrow 1$
9:     **while** $|\mathbf{P}_{t+1}| + |\mathbf{F}_i| \le N$ **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**

4 | EXPERIMENT SETUP

We implement the proposed algorithm in C++ and conduct the MOEDA design flow experiments on a 2.2 GHz Xeon E5-2650 CPU. The benchmark circuits from ISCAS-85 [39] and EPFL [40] 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). These netlists are then optimised using the proposed flow in tandem with the physical implementation tool Innovus™ (v17.11) 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 65 nm 9-track low-power core cell library (TCBN65LP) in standard threshold voltage.

4.1 | Tool environment setup

The MOEDA flow is applied to further enhance designs which are already well-optimised by the Cadence® 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, we create an ideal general clock for all inputs and outputs, which means all paths are clocked with two ideal flip-flops at the beginning and the end of each path shown in Figure 4. The benchmarks used are all combinational circuits so that the ideal clock was not applied with any uncertainties or transition delays.

To tighten the timing constraint, shown in Figure 4, the output delay constraint (T_{od}) is gradually increased for a given clock period (T_c). The required time (T_r) is calculated through Equation (2) in EDA tools, so the required time (T_r) that solutions need to meet is gradually becoming strict.

$$T_r = T_c - T_{od} \qquad (2)$$
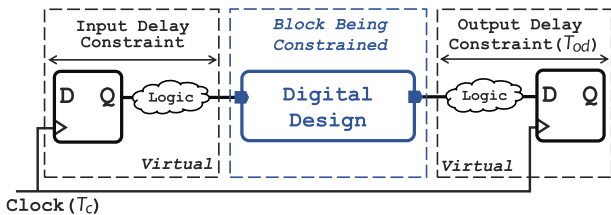
FIGURE 4 Conceptual testbench to define the timing constraints in the electronic design automation tools. Input/output delay constraints are set by using virtual logic parts and flip-flops, allowing users to specify clocks and timing requirements. The *Digital Design* in the middle is constrained.

The circuit path arrival time (T_a) should be less than the required time (T_r) to meet the timing constraint. The settings of both the synthesis step and physical implementation step are summarised in Table 1.

The output load capacitance (set_load) is also specified in part of the following 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-driven routing are enabled for better performance.

4.2 | Objective evaluation in tools

In this work, the evaluation regarding three objectives takes place after place-and-route with Innovus™ as follows:

D_{wc}, worst case T_a which is the value of T_r minus the worst negative slack amongst all path delays. This is achieved by performing static timing analysis at the post-route stage.

P_{total}, which is the results reported in Innovus™ by power analysis. It includes switching power, internal power (short-circuit power) and leakage power. Both internal and leakage power are calculated based on power tables provided in the Liberty (.lib) file, which contains the specifications and characterisations of the standard cells. Switching power is calculated based on the equation $P = 0.5*C_L V^2 F*A$, 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™).

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

All evaluations above are performed on a single mode under typical corner conditions (PVT: TT, 1.2 V, 25°C). The reason why we only focus on a single corner is that the power-sensitive ICs often only apply the standard cell library with one threshold voltage (*Vth*) (usually high *Vth*) to control the overall power dissipation. Therefore, performing the optimisation with just one *Vth* cell library will be certainly more difficult than with multiple *Vth* libraries since the design space of using single *Vth* is limited.

4.3 | Multi-threads running and runtime

According to the computing resources and licences, we run all experiments in this work parallly using 24 threads in an MOEDA run for evaluating individuals.

TABLE 1 Tool settings for synthesis and physical implementation.

Synthesis setup	Place & route setup
syn_generic_effort = high iopt_ultra_optimisation = true	Die aspect ratio = 1.0
	Core utilisation = 0.7
	Timing-driven placement = true
	Timing-driven routing = true
	SI-driven routing = true

The MO 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 in this case. This aims to achieve accurate metrics as close as possible to sign-off. Therefore, the runtime is not the primary concern here.

However, due to the inherent parallelism of the population-based approach, this can be overcome using a larger number of licences and high-performance computing resources. So if we run the algorithm in embarrasingly parallel, twice as many cores, half the time, scales linearly with compute resources.

In cases where a design is fabricated in very large numbers, the longer time invested in the optimisation of, for example, power consumption, will also be justified.

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. In Ref. [21], the authors proposed ML techniques to improve the speed and accuracy of MO design space exploration problems. We also see the most powerful solutions in the future when combining different methods appropriately at different levels and stages of the design hierarchy, that is, ML + EA, but this is not the focus of this work.

5 | MULTI-OBJECTIVE OPTIMISATION EXPERIMENTS

5.1 | Experiments with a full foundry library

In this set of experiments, 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 16 × 16 multiplier (C6288). One large circuit used from the EPFL benchmark suite is an arithmetic function for log2 calculation. The statistics of benchmarks are summarised in Table 2. The reason that we only use combinational circuits is that 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 [40]. The longest path of a combinational part is thus the most critical path of a circuit. In addition, although most gate-sizing related research optimises sequential circuits, they still only manipulate on combinational components [23, 25–27].

T A B L E 2 Statistics of benchmarks.

Test case	No. Inputs	No. Outputs	No. Gates
C1908	33	25	880
C5315	178	123	2307
C6288	32	32	2406
log2	32	32	32,060

The proposed MOEDA flow can handle the drive strength optimisation for all types of cells available from the TSMC TCBN65LP library. 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 later) is the tightest constraint that can just be met, resulting in a solution with the best delay. In the second case (named 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 (named b) is chosen in the middle of the first and second. The three different solutions obtained will be used as seeds to perform 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.

In this work, we have set EA parameters that are widely-used in the MOEA literature [41]. In this set of experiments, all circuits are optimised with running $M = 200$ generations with a population size N of 200 individuals, and output load constraints have not been applied. We have run preliminary experiments with NSGA-II to confirm that reliably converges to similar performance when run multiple times with the same evaluation budget. Since the focus here is not on statistical analysis of the MOEA, we run the algorithm once for each benchmark to manage runtime. The number of synthesised gates and the number of genes are the same shown in Table 3 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 2. It is the reason that 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. So this makes the synthesis tool to automatically merge the simple gates into complex ones for the total transistor count and physical area reduction, to finally reduce the number of gates.

This may compact the design space and reduce the search complexity but still increase the difficulty of PPA extra optimisation. In real-world libraries, complex logic cells have less options of drive strengths (normally no more than five) due to the layout 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 MOEDA-optimised results are still promising compared to the Syn-Opt. Solutions which are obtained by running the synthesis tool with 'try hard' mode, Multi-objective electronic design automation flow 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 Table 3. The reported improvement of an objective does not (or slightly) sacrifice the metrics of other objectives. Although
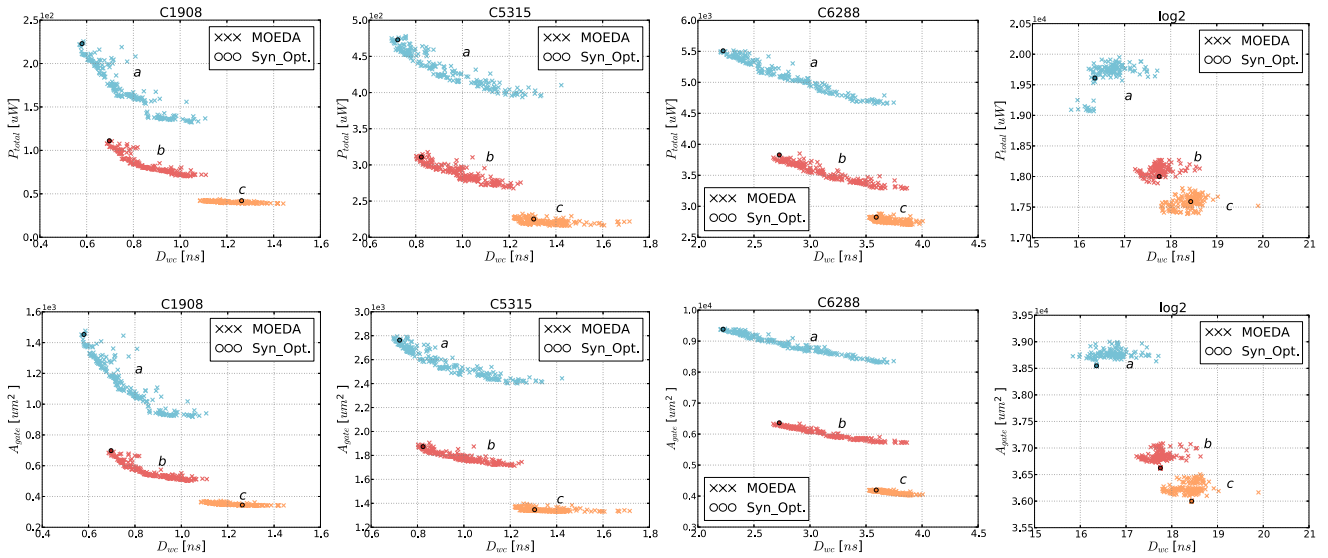
TABLE 3 Multi-objective electronic design automation flow design flow with using the full commercial library.

Units: D_{wc} [ns] P_{total} [uW] A_{gate} [um²]					$N = 200, M = 200, \rho = 1\%$, set_load = 0		
					MOEDA solution		
Test case	Clock (T_c)	(No.) T_r	No. Syn gates No. Genes	Syn-opt. Solution	Best D_{wc} (Δ%)	Best P_{total} (Δ%)	Best A_{gate} (Δ%)
C1908	250 MHz	(a) 0.60 ns	299	D_{wc}: 0.580	0.569 (1.9%)	0.580	0.580
			299	P_{total}: 222.9	221.9	211.0 (5.3%)	211.0
				A_{gate}: 1452.96	1451.88	1388.16	1388.16 (4.5%)
		(b) 0.76 ns	178	D_{wc}: 0.697	0.687 (1.4%)	0.688	0.696
			178	P_{total}: 111.1	107.9	107.5 (3.2%)	107.8
				A_{gate}: 698.04	682.92	682.2	678.96 (2.7%)
		(c) 1.50 ns	105	D_{wc}: 1.263	1.234 (2.3%)	1.249	1.251
			105	P_{total}: 42.1	39.69	39.32 (6.6%)	39.51
				A_{gate}: 344.52	344.52	343.08	342.72 (0.5%)
C5315	250 MHz	(a) 0.74 ns	750	D_{wc}: 0.723	0.706 (2.4%)	0.715	0.72
			750	P_{total}: 472.9	470.5	458.9 (3.0%)	461.2
				A_{gate}: 2762.64	2755.44	2729.16	2724.48 (1.4%)
		(b) 0.88 ns	516	D_{wc}: 0.824	0.805 (2.3%)	0.819	0.823
			516	P_{total}: 310.9	309.0	304.6 (2.0%)	305.7
				A_{gate}: 1873.44	1869.48	1859.76	1852.56 (1.1%)
		(c) 1.50 ns	400	D_{wc}: 1.305	1.241 (4.9%)	1.289	1.302
			400	P_{total}: 225.2	222.3	217.4 (3.5%)	220
				A_{gate}: 1346.76	1343.52	1343.16	1336.68 (0.8%)
C6288	250 MHz	(a) 2.34 ns	2178	D_{wc}: 2.225	2.204 (0.9%)	2.206	2.204
			2178	P_{total}: 5509	5495	5481 (0.5%)	5495
				A_{gate}: 9382.32	9364.68	9377.28	9364.68 (0.2%)
		(b) 2.90 ns	1555	D_{wc}: 2.726	2.673 (1.9%)	2.708	2.708
			1555	P_{total}: 3829	3785	3732 (2.5%)	3732
				A_{gate}: 6363.00	6331.32	6278.76	6278.76 (1.3%)
		(c) 4.00 ns	1140	D_{wc}: 3.591	3.528 (1.8%)	3.59	3.585
			1140	P_{total}: 2824	2821	2754 (2.5%)	2777
				A_{gate}: 4194.00	4191.84	4183.92	4137.48 (1.3%)
log2	40 MHz	(a) 16.4 ns	11,838	D_{wc}: 16.355	15.839 (3.2%)	16.24	16.355
			11,838	P_{total}: 19610	19,090	19070 (2.8%)	19,610
				A_{gate}: 38547.0	38728.1 (−0.5%)	38797.6 (−0.6%)	38547.0 (0.0%)
		(b) 17.9 ns	11,272	D_{wc}: 17.751	17.364(2.2%)	17.72	17.751
			11,272	P_{total}: 18000	18,000	17890 (0.6%)	18,000
				A_{gate}: 36623.9	36840.6 (−0.6%)	36726.8 (−0.3%)	36623.9 (0.0%)
		(c) 18.8 ns	11,119	D_{wc}: 18.435	17.795 (3.5%)	18.357	18.435
			11,119	P_{total}: 17590	17,510	17390 (1.1%)	17,590
				A_{gate}: 35999.6	36227.5 (−0.6%)	36203.0 (−0.5%)	35999.6 (0.0%)

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 [27].

In Figure 5, the final generation of each circuit with three independent seeding runs is shown, plotting 'D_{wc} versus P_{total}' (upper row), 'D_{wc} versus A_{gate}' (lower row) and the

FIGURE 5 Multi-objective electronic design automation flow (MOEDA) flow optimisation results using full commercial standard cell library for C1908, C5315, C6288 and log2 circuits.

corresponding Syn-Opt. Reference solutions. The three clusters (*a*, *b* and *c*) correspond to the three seed timing constraints, listed in Table 3. In all cases, the MOEDA produces a wide range of useful trade-off solutions, with improved delay, reduced power consumption or area, within the boundaries of the given seed (Syn_Opt. solution) 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 used 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-optimised 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 the 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 a 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. This can conclude that the MOEDA flow demonstrates the

capability of balancing these three objectives to a greater extend while the 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 (reducing gate count) instead of only focussing on drive strength refinement. But it is still worthwhile to take the area as one of the objectives in the optimisation, which otherwise may have much degradation on the area when optimising other objectives.

5.2 | Comparative analysis with stochastic search

To demonstrate the optimisation efficiency of the MOEA used in the proposed MOEDA flow, we performed a comparative study between the MOEDA search and stochastic search. The selected test case is C5315-(a) from Section 5.1, a 9-bit ALU with a tight timing constraint (a). For the MOEDA search, we initially run the NSGA-II using a 1% mutation rate with a 200-individual population size for 200 generations, so 40,000 evaluations are generated in total. The MOEDA optimisation results (red cluster) shown in Figure 6 are seeded with the tool-optimised Syn_Opt solution (the black circle).

Two stochastic search experiments are then run here for comparison. The first one, referred to a local stochastic search (grey cluster shown in Figure 6), is to randomly produce 40,000 individuals seeding with the same Syn_Opt solution, and each of them is achieved by randomly mutating the chromosome using the same probability (1%). The second one is completely randomised results referred to a global stochastic search, which produces 40,000 individuals seeding with the same Syn_Opt solution, but all genes (i.e., drive strengths of logic gates) of each individual are modified (100% mutation rate). The results

of global stochastic search are the blue cluster shown in Figure 6.

Based on the observations made from these plots, it demonstrates that the NSGA-II algorithm used in the MOEDA flow has superior optimisation performance when compared to the stochastic search. Since the focus of this work is not on investigating which MOEA is the best to achieve the optimum results in VLSI design optimisation, only NSGA-II is used here for experiments.
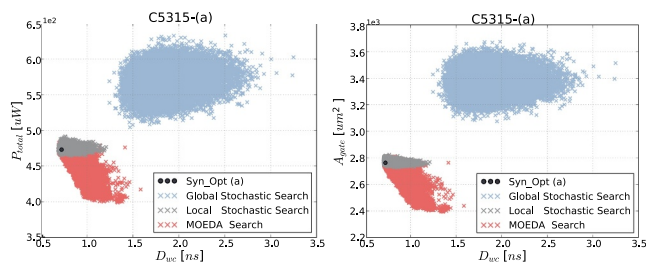
5.3 | Discussion

The runtime for the largest case optimisation (log2. a) needs 138 h. 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.

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 section will investigate how running synthesis multiple times can be harnessed to expand, access and explore the design space with respect to different circuit topologies.

6 | MULTI-OBJECTIVE DESIGN SPACE EXPLORATION

6.1 | Optimisation using multiple seed designs

Instead of seeding the initial population by using a single synthesis-optimised solution for a separate MOEDA run, this section investigates how the proposed algorithm can explore the design space simultaneously by using a set of multiple different seeds. The seeds are a range of different solutions generated using the standard digital flow under a number of different timing constraints.

The methodology to obtain different seeds from the standard design tools is the same as that in Section 5.1. However, 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 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. Table 4 summarises timing constraint settings of each test case, including the number of synthesised gate from minimum to maximum. Different output load scenarios, including loading with drive strength D1 and D8, are applied to the outputs of 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 cell library. The reason of 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 first column of Figures 7 and 8 illustrates the standard tool's design space for each benchmark circuit under D1 and D8 output load scenarios. Their respective optimised design space from the MOEDA flow is shown in the second and third columns of Figures 7 and 8. From 'Standard Flow' columns, all cross markers represent tool-generated solutions in 'D_{wc} versus 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 Syn-Opt. 'elite' solution front, which is calculated using the non-dominated sorting approach in three dimensions in 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-Frontiers 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 even the log2 circuit can be synthesised and optimised well by the tool, as the set of solutions forms a

FIGURE 6 multi-objective electronic design automation flow search compared to stochastic search, demonstrating the superior optimisation efficiency of evolutionary algorithms.

TABLE 4 Timing constraints of each benchmark.

| Test case	Clock (T_c)	T_r (Increment factor)	Set load	# Syn gates # genes
C1908	250 MHz	1.50 − 0.51 ns	D1	105–445
	(4 ns)	(0.01 ns)	D8	105–468
C5315	250 MHz	1.50 − 0.51 ns	D1	396–1323
	(4 ns)	(0.01 ns)	D8	401–1287
C6288	250 MHz	4.00 − 2.02 ns	D1	1105–3208
	(4 ns)	(0.02 ns)	D8	1123–3222
log2	40 MHz	25.00 − 15.10 ns	D1	10801–12561
	(25 ns)	(0.10 ns)	D8	10797–12555
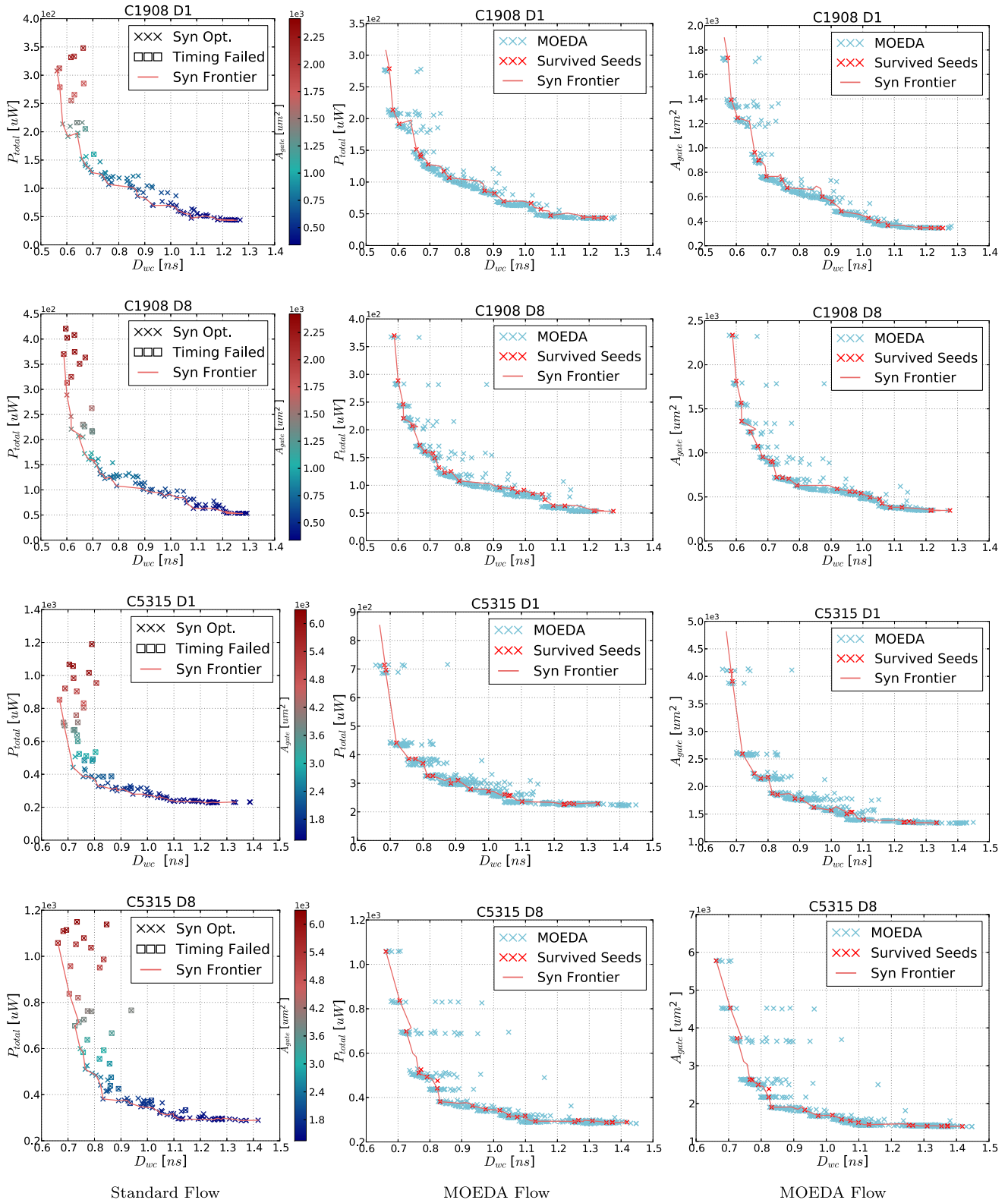
FIGURE 7 Design space optimisation results under the drive strength D1 and D8 output load scenarios for C1908 16-bit error detector/corrector and C5315 9-bit ALU. $N = 500$, $M = 100$, $\rho = 1\%$, set_load information is labelled in the title at the top of each plot.

smooth Pareto frontiers. However, the 16×16 multiplier (C6288), which is a highly structured circuit using a number of adders, yields a less regular frontier with more clustered solutions. This indicates that the synthesis tool struggles to effectively trade off multiple objectives when optimising a complex design with a relatively fixed circuit topology.
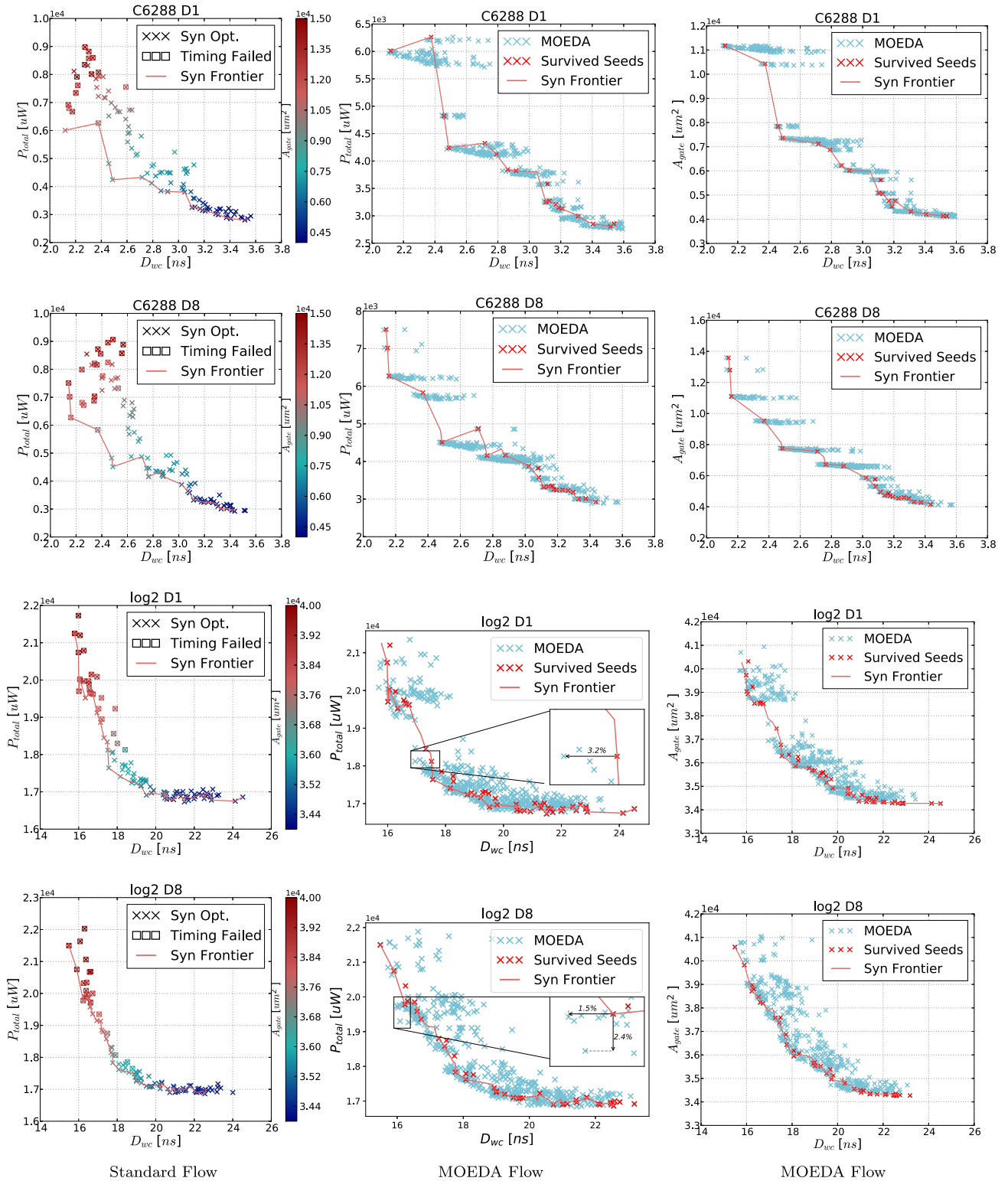
FIGURE 8 Design space optimisation results under the drive strength D1 and D8 output load scenarios for C6288 16 × 16 multiplier and log2 calculation circuit. $N = 500$, $M = 100$, $\rho = 1\%$, set_load information is labelled in the title at the top of each plot. The optimised design space of log2 with D1 and D8 loads is shown with zoom-in views to present the improvements clearly.

6.2 | Squeeze design space for PPA

The design space comprising the 100 seed solutions, in different circuit topologies, is the baseline for the MOEDA to

perform optimisation on. All 100 seed solutions are loaded into the initial population of the MOEDA flow and optimised generation by generation. All test cases are optimised over 100 generations using a population size of 500, that is, the

initial population comprises five copies of each seed circuit. The plots in 'MOEDA Flow' columns of Figures 7 and 8 show the improved solution space, plotting 'D_{wc} versus P_{total}' and 'D_{wc} versus A_{gate}'. The red line shows the Syn-Frontiers from the baseline design space. 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 MOEDA flow comprising of all individuals of the final generation.

The results confirm that the MOEDA flow can push the baseline frontier further to extend the design space of all test cases in all three objectives, through 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 form a smooth Pareto frontier, whereas there are some gaps in the optimised design space of C5315, log2 and particularly of C6288. 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. Most of the surviving seeds are positioned on the Syn-Frontier, while others have been discarded in the evolution process. This indicates that there is 'noisiness' inside of standard flows/tools and not all solutions generated by tools are presumably optimised, which might lose some well trade-off solutions. This normally requires iterations with applying modifications in the design flow achieved by engineers with custom design efforts. The MOEDA can auto-iterate designs throughout the whole flow for better trade-offs in PPA metrics.

To demonstrate the search efficiency of the MOEDA, Figures 7 and 8 also include the MOEDA-optimised non-dominated solutions covering all three objectives to show the relative position of the optimised and the initial tool-generated ones. The results clearly show that the optimised front dominates the initial tool-generated one. Table 5 further summarises the quantified quality of the Pareto solution sets compared to the total number of designs explored. The number of Pareto solutions presented in the table is the total non-dominated solutions of the final MOEDA generation in each test case. The total number of evaluations is 50,000 which is the same for all cases. The search efficiency is then obtained by calculating the ratio of the Pareto solutions in the final generation to the total number of evaluations. In log2 circuit, the efficiency is slightly lower than for other designs due to its larger size.

TABLE 5 Search efficiency of multi-objective electronic design automation flow.

Test case	Set load	No. Pareto solutions in final generation	Total no. evaluations	Search efficiency (%)
C1908	D1	304	50,000	0.61%
	D8	307		0.61%
C5315	D1	254	50,000	0.51%
	D8	250		0.50%
C6288	D1	257	50,000	0.51%
	D8	259		0.52%
log2	D1	150	50,000	0.30%
	D8	166		0.33%

6.3 | Discussion

The runtime of largest case (log2.D8) is 162 h. The MOEDA flow needs more computing resources due to the continuous generation of design layouts. This aims for accurate and real-world evaluation. It is possible to speed up the flow through not updating the physical layout at every iteration of the MOEA or straightforwardly making design evaluations at earlier design stage without place and route, but what we are investigating in this work is whether the proposed MOEDA flow has generic optimisation capability in an industrial environment, and the MOEDA flow has feasibly improved the performance of block circuit instances used in this work. With regard to scalability, in terms of design size, an iterative critical path optimisation for extreme-large designs (e.g., millions of gates) using MOEDA flow is also our work in progress, with the potential aim to solve timing violations faster and still without increasing the power or area.

The MOEDA flow achieves significant improvements on PPA over the standard design tool's solutions across the entire design space with different circuit topologies. However, although the proposed method is capable of exploiting design opportunities to refine technology mapping by adjusting drive strengths at the gate level, circuit topology optimisation is currently not yet included. This current limitation is likely the reason that design space gaps cannot be fully closed, which would provide the best trade-off design choices. This is particularly visible in the results for C6288, due to its fixed topology. From these results it can be envisaged that including topology modification in our approach could enable further design optimisation opportunities, particularly in the case of complex circuits with rigid structure.

7 | CONCLUSIONS

This paper proposes a fully-automated MO EDA flow (MOEDA) extension to enhance the current industry-standard synthesis and physical implementation flow, primarily suited

for IP/block level designs. 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 designs from ISCAS-85 and EPFL benchmark suite using the TSMC 65 nm low power standard cell library.

Experimental results show that the proposed MOEDA flow has operated design optimisation gaining significant improvements on PPA over the standard tool's solutions. It can be concluded that optimising technology mapping to refine drive strength selection of cells is beneficial to improving PPA of circuits. This has not only been shown for a single solution but across the entire design space with various circuit topologies.

From a designer's point of view, the MO optimisation approach has the added benefit of producing a set of best trade-off solutions which are as uniformly as possible distributed. This provides designers with a choice and allows to select designs with the most appropriate objective trade-off for different applications.

AUTHOR CONTRIBUTIONS

Linan Cao: Conceptualization; Data curation; Formal analysis; Investigation; Methodology; Software; Validation; Visualization and Writing—original draft. **Simon J. Bale**: Conceptualization; Investigation; Methodology; Supervision and Writing—review & editing. **Martin A. Trefzer**: Conceptualization; Investigation; Methodology; Project administration; Resources; Supervision and Writing—review & editing.

ACKNOWLEDGEMENTS

CONFLICT OF INTEREST STATEMENT

We do not have a conflict of interest.

DATA AVAILABILITY STATEMENT

The results data that support the findings of this study are openly available in [MOEDA Drive Strength Optimisation Physical Layout] at 10.15,124/7260ff92-4eaf-4cf9-bb85-add74e01b7ff. The benchmark data are openly available from [ISCAS-85 Ciruits] at [http://web.eecs.umich.edu/~jhayes/iscas.restore/benchmark.html], reference number [39], and [EPFL Circuits] at [https://www.epfl.ch/labs/lsi/page-102566-en-html/benchmarks/], reference number [40]. Restrictions apply to the semiconductor process data which were used under license for this study.

ORCID

Linan Cao https://orcid.org/0000-0002-0439-7979
Martin A. Trefzer https://orcid.org/0000-0002-6196-6832

REFERENCES

1. Kahng, A.B., et al.: VLSI Physical Design: From Graph Partitioning to Timing Closure. Springer Science & Business Media (2011)
2. Rao, D.S., Kurdahi, F.J.: Hierarchical design space exploration for a class of digital systems. IEEE Trans on VLSI Systems 1(3), 282–295 (1993). https://doi.org/10.1109/92.238442
3. Ma, Y., et al.: Cross-layer optimization for high speed adders: a pareto driven machine learning approach. IEEE Trans on CAD of Integrated Circuits and Systems 38(12), 2298–2311 (2019). https://doi.org/10.1109/tcad.2018.2878129
4. Kwon, J., Ziegler, M.M., Carloni, L.P.: A learning-based recommender system for autotuning design flows of industrial high-performance processors. In: Proc. 56th DAC., pp. 1–6. IEEE (2019)
5. Anwar, M., et al.: Early scenario pruning for efficient design space exploration in physical synthesis. In: 2016 29th Int. Conf. on VLSI Design and 2016 15th Int. Conf. on Embedded Systems (VLSID), pp. 116–121. IEEE (2016)
6. Ziegler, M.M., et al.: A synthesis-parameter tuning system for autonomous design-space exploration. In: Proc. DATE., pp. 1148–1151. IEEE (2016)
7. Kahng, A.B., Kumar, S., Shah, T.: A no-human-in-the-loop methodology toward optimal utilization of eda tools and flows. In: Proc 55th DAC. WIP Track (2018)
8. Ricci, A., De.Munari, I., Ciampolini, P.: An evolutionary approach for standard-cell library reduction. In: Proc. 17th ACM Great Lakes Symp. on VLSI., pp. 305–310. ACM (2007)
9. Rahim, H.A., et al.: The performance study of two genetic algorithm approaches for vlsi macro-cell layout area optimization. In: 2008 2nd Asia Int. Conf. on Modelling & Simulation (AMS), pp. 207–212. IEEE (2008)
10. Sheng, W., Xiao, L., Mao, Z.: Soft error optimization of standard cell circuits based on gate sizing and multi-objective genetic algorithm. In: Proc. 46th DAC., pp. 502–507 (2009)
11. Sait, S.M., El Maleh, A.H., Al Abaji, R.H.: Evolutionary algorithms for vlsi multi-objective netlist partitioning. Eng. Appl. Artif. Intell. 19(3), 257–268 (2006). https://doi.org/10.1016/j.engappai.2005.09.008
12. Vasicek, Z., Sekanina, L.: A global postsynthesis optimization method for combinational circuits. In: Proc. DATE., pp. 1–4. IEEE (2011)
13. Palesi, M., Givargis, T.: Multi-objective design space exploration using genetic algorithms. In: Proc. 10th Int. Symp. on Hardware/software Codesign., pp. 67–72 (2002)
14. Ascia, G., Catania, V., Palesi, M.: A framework for design space exploration of parameterized vlsi systems. In: Proc. 7th ASP-DAC and 15th Int. Conf. on VLSI Design, pp. 245–250. IEEE (2002)
15. Cao, L., Bale, S.J., Trefzer, M.A.: Instrumenting parametric physical layout for multi-objective optimisation. In: 2018 IEEE Symp. Series on Computational Intelligence (SSCI), pp. 1339–1345. IEEE (2018)
16. Chinnery, D.G., Keutzer, K.: Closing the power gap between asic and custom: an asic perspective. In: Proc. 42nd DAC., pp. 275–280 (2005)
17. Chinnery, D.G., Keutzer, K.: Closing the gap between asic and custom: an asic perspective. In: Proc. 37th DAC., pp. 637–642 (2000)
18. Chinnery, D.G., Keutzer, K.: High performance and low power design techniques for asic and custom in nanometer technologies. In: Proc. 2013 ACM Int. Symp. on Physical Design, pp. 25–32 (2013)
19. Dally, W.J., Chang, A.: The role of custom design in asic chips. In: Proc. 37th DAC., pp. 643–647 (2000)
20. Onodera, H., Hashimoto, M., Hashimoto, T.: Asic design methodology with on-demand library generation. In: 2001 Symp. on VLSI Circuits. Digest of Technical Papers, pp. 57–60. IEEE (2001)
21. Hyun, D., Fan, Y., Shin, Y.: Accurate wirelength prediction for placement-aware synthesis through machine learning. In: 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 324–327. IEEE (2019)
22. Lavagno, L., et al.: Electronic Design Automation for IC Implementation, Circuit Design, and Process Technology: Circuit Design, and Process Technology. CRC Press (2016)
23. Flach, G., et al.: Effective method for simultaneous gate sizing and v th assignment using Lagrangian relaxation. IEEE Trans on CAD of Integrated Circuits and Systems 33(4), 546–557 (2014). https://doi.org/10.1109/tcad.2014.2305847

24. Reimann, T.J., Sze, C.C., Reis, R.: Cell selection for high-performance designs in an industrial design flow. In: Proc. 2016 ACM Int. Symp. on Physical Design, pp. 65–72 (2016)

25. Sharma, A., Chinnery, D., Chu, C.: Lagrangian relaxation based gate sizing with clock skew scheduling-a fast and effective approach. In: Proc. 2019 Int. Symp. on Physical Design, pp. 129–137 (2019)

26. Hu, J., et al.: Sensitivity-guided metaheuristics for accurate discrete gate sizing. In: 2012 IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD), pp. 233–239 (2012)

27. Fatemi, H., et al.: Enhancing sensitivity-based power reduction for an industry ic design context. Integration 66, 96–111 (2019). https://doi.org/10.1016/j.vlsi.2019.01.008

28. Farshidi, A., et al.: A self-tuning multi-objective optimization framework for geometric programming with gate sizing applications. In: Proc. 23rd ACM Great Lakes Symp. on VLSI., pp. 305–310 (2013)

29. Farshidi, A., et al.: Optimal gate sizing using a self-tuning multi-objective framework. Integration 47(3), 347–355 (2014). https://doi.org/10.1016/j.vlsi.2013.10.008

30. Reimann, T., et al.: Simultaneous gate sizing and vt assignment using fanin/fanout ratio and simulated annealing. In: 2013 IEEE Int. Symp. on Circuits and Systems (ISCAS), pp. 2549–2552. IEEE (2013)

31. Yella, A.K., Srivatsa, G., Sechen, C.: Are standalone gate size and v t optimization tools useful? In: 2017 IEEE 30th Canadian Conf. on Electrical and Computer Engineering (CCECE), pp. 1–6. IEEE (2017)

32. Wang, X.D., Chen, T.: Performance and area optimization of vlsi systems using genetic algorithms. VLSI Des. 3(1), 43–51 (1995). https://doi.org/10.1155/1995/26912

33. Benkhider, S., Boumghar, F., Babaali, A.: A parallel genetic approach to the gate sizing problem of vlsi integrated circuits. In: Proc. of the 12th Int. Conf. on Microelectronics., pp. 169–173. IEEE (2000)

34. Wang, R., et al.: Localized weighted sum method for many-objective optimization. IEEE Trans. Evol. Comput. 22(1), 3–18 (2016). https://doi.org/10.1109/tevc.2016.2611642

35. Kahng, A.B., et al.: High-performance gate sizing with a signoff timer. In: 2013 IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD), pp. 450–457. IEEE (2013)

36. Deb, K., et al.: A fast and elitist multiobjective genetic algorithm: nsga-ii. IEEE Trans. Evol. Comput. 6(2), 182–197 (2002). https://doi.org/10.1109/4235.996017

37. Cao, L., Bale, S.J., Trefzer, M.A.: Multi-objective digital design optimization via improved drive granularity standard cells. IEEE Transactions on Circuits and Systems I: Regular Papers 68(11), 4660–4671 (2021). https://doi.org/10.1109/tcsi.2021.3109239

38. Verma, S., Pant, M., Snasel, V.: A comprehensive review on nsga-ii for multi-objective combinatorial optimization problems. IEEE Access 9, 57757–57791 (2021). https://doi.org/10.1109/access.2021.3070634

39. Brglez, F., Fujiwara, H.: A neutral netlist of 10 combinational benchmark circuits and a target translator in fortran. In: 1985 IEEE Int. Symp. on Circuits and Systems (ISCAS), pp. 677–692. IEEE (1985)

40. Amarú, L., Gaillardon, P.E., De.Micheli, G.: The EPFL combinational benchmark suite. In: Proc. of the 24th Int. Workshop on Logic & Synthesis (IWLS) (2015)

41. Trefzer, M.A., Tyrrell, A.M.: 'Evolvable Hardware', from Practice To Application. Springer (2015)