Multi-objective Digital Design Optimisation via Improved Drive Granularity Standard Cells

Linan Cao, *Student Member, IEEE,* Simon J. Bale, and Martin A. Trefzer, *Senior Member, IEEE*

arXiv:2105.11248v2 [cs.OH] 6 Sep 2021

***Abstract*—To tackle the complexity of state-of-the-art electronic systems, silicon foundries continuously shrink the technology nodes and electronic design automation (EDA) vendors offer hierarchical design flows to decompose systems into smaller blocks. However, such a staged design methodology consists of various levels of abstraction, where margins will be accumulated and result in degradation of the overall design quality. This limits the full use of capabilities of both the process technology and EDA tools. In this work, a study of drive granularity of standard cells is performed and an interpolation method is proposed for drive option expansion within original cell libraries. These aim to investigate how industrial synthesis tools deal with the drive strength selection using different granularity sets. In addition, a fully-automated, multi-objective (MO) EDA digital flow is introduced for power, performance, area (PPA) optimisation based on drive strength refinement. This population-based search method better handles the increased difficulty of cell selection when using larger logic libraries, producing better optimised solutions than standard tool flow in this case. The achieved experimental results demonstrate how the improved drive granularity cells overall enhance the quality of designs and how a significant improvement in trading off PPA is achieved by the MOEDA flow.**

***Index Terms*—Multi-objective Optimisation, Drive Strength, Standard Cells, Digital Flow, EDA.**

I. Introduction

STANDARD cells, the basic building blocks of digital integrated circuits (ICs), implement basic logic functions. Any large and complex logic function is composed of standard cells from a library providing multiple drive strength options for each cell to meet design specifications. Cells with different drive strengths are realised through different transistor sizes, whereby larger transistors provide increased current drive capabilities and smaller ones consume less area or power. Commercial digital IC design flows commonly use standard cell libraries from foundries, which are predefined for generic design requirements to tape out chips. Well-optimised libraries have therefore become crucial as they determine the overall achievable quality of results (QoR).

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 (typically introducing additional smaller sizes) can potentially improve designs particularly for lowering power [1] [2] [3].

Technology down-scaling leads to high-density in standard cell layouts that need to accommodate restrictive physical design rules. This incurs long turnaround time with significant human effort in transistor-level placement and routing when creating cell libraries. Automated transistor-sizing tools have been introduced for the provision of fine-tuned drive strength options, mainly focusing on low power design solutions [4]–[7]. Furthermore, on-demand transistor sizers [8] [9] have been developed for real-time library generation working in the digital EDA flow from logic synthesis to physical design. Such continuously-sized logic cells extend the solution space, but significantly increase the implementation and analysis time [10]. A synthesis-centered design approach, using discrete gate libraries, is still the most commonly used technique for producing new generations of chips in response to the rapid time-to-market process.

Seeking to achieve richer cell libraries, implementing logic designs using mixed-height (i.e., routing tracks like 9-track and 12-track) or double-row-height standard cells are recently proposed [11]–[14]. Smaller-height cells feature compact area and lower power dissipation, but are weaker in drive strength. Cells with larger heights 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.

However, richer standard cell libraries lead to increasingly difficult logic synthesis when aiming at producing well-optimised technology-mapped netlists. This makes design margins or even errors propagate through the entire flow and ultimately may lead to performance loss due to generic overheads. To resolve these issues, developing a method that can perform optimisation from a more global viewpoint requires more computing time but can provide solutions with considerable improvements.

Population-based optimisation techniques like evolutionary algorithms (EAs) are widely-adapted approaches to perform efficient design space search and provide globally-optimised solutions. Researchers are looking at evolutionary design space

The authors are with the Intelligent Systems and Nanoscience Research Group, Department of Electronic Engineering, University of York, York, North Yorkshire, YO10 5DD, United Kingdom. e-mail: (lc1492, martin.trefzer, simon.bale)@york.ac.uk

optimisation at the system or architecture level [15] [16], intellectual property (IP) blocks [17], standard cell library composition [18], etc., but limited research is investigating the application of evolutionary optimisation to the full standard digital flow. An automated multi-objective optimisation flow is needed which can compatibly work across different abstraction levels of existing flow to recover overall performance of designs.

Novel contributions made in this work are: 1) A methodology is introduced to interpolate fine-grained drive options into the original granularity of standard cell libraries. This helps to further exploit a technology node, enhancing design quality of digital circuits with better PPA metrics; 2) An MOEDA flow is proposed which applies multi-objective evolutionary algorithms (MOEAs) to a standard digital flow performing multi-objective optimisation and enlarging design solution space based on drive strength mapping. This optimisation approach is used to assist tools to further improve and better trade-off solutions in PPA when synthesising designs with an improved drive granularity library.

The remaining parts of the paper are structured as follows: Section II illustrates the design of fine-grained drive strength library. Section III provides an introduction of MOEDA framework. Experiment setup is described in Section IV. Section V demonstrates experimental results and Section VI concludes and outlines future work.

II. Drive Strength Design of Standard cells

A. Logic Design using Multiple Drive 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 is able to drive larger loads or increase clock frequency.

Synthesis tools are mapping drive strength from a finite set of discrete drive options for a generic functional gate, and usually need 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 [2]. 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 significantly 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 when creating cell libraries, and it is too expensive and time consuming to make libraries even larger than they are already (typically 600-1000 cells).

In prior work, [3] 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.

B. Improved Drive Granularity Library Design

Our 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 I. To focus the investigation on the drive strength selection and simplify the problem, we only consider drive strength expansion of inverters in this case.

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 $\mathrm{C_{unit_load}}$ in a period time $\mathrm{T_{unit_load}}$ (i.e., circuit delay), the X2 needs to be designed through iterative transistor-sizing until it can drive double the unit load capacitance $2 \times \mathrm{C_{unit_load}}$ taking a near-exact same time $\mathrm{T_{unit_load}}$. So the definition of drive strength:

$$\mathrm{X} = \frac{\mathrm{C_{X_load}}}{\mathrm{C_{unit_load}}} \quad \text{s.t.} \quad \mathrm{T_X} = \mathrm{T_{unit_load}} \tag{1}$$

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

TABLE I
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

The inverter drive strength X1 is defined by $\mathrm{PMOS_{size}} = 230\mathrm{nm}$ and $\mathrm{NMOS_{size}} = 165\mathrm{nm}$, so the $\mathrm{P/N}$ ratio adapted in this work is 1.39 for all cells. The X0 cell is defined by $\mathrm{NMOS_{min}}$ and $\mathrm{PMOS_{size}} = 1.39 \times \mathrm{NMOS_{min}}$ 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 $\mathrm{T_{unit_load}}$. All created cells keep the same transistor length $60\mathrm{nm}$.

In this work, both custom-designed "MINI_ORIG" and "MINI_FINE" libraries are implemented including schematics and layouts using Cadence® Virtuoso®. 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 using Cadence® Liberate™ and Abstract Generator™ tools. The standard cell design flow is illustrated in Fig. 1.

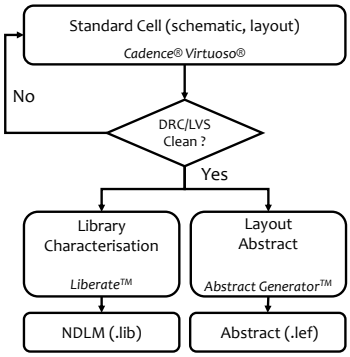

Fig. 1. Standard cell design flow including library characterisation and layout abstract.

The Non-Linear Delay Model (NLDM) is a 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 one contains the technology library including all environment descriptions like operating conditions, wire load mode, etc., and the second one 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 the 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 $\mathrm{uW/MHz}$ 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/min/max leakage power values are provided in nanowatts (nW) for immediate use.

C. 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 II shows the transistor count, cell width and leakage power (i.e., specifically contains average/min/max 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. All cells are created at the same height $1.8\mathrm{um}$, 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

TABLE II
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

rules. Hence, in this case, swapping the three smallest cells provides a gain in power reduction and not in area.

The characterised cell propagation delays confirm that fine-grained drive inverters are interpolating into the original drive granularity 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, all fine-grained drive inverters are also positioned in the middle of adjacent original granularity inverters.

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.

III. MODEA OPTIMISATION FRAMEWORK

A. Standard Digital Flow

Modern digital IC design flow, a solid and mature process, consists of various steps including register-transfer-level (RTL) design, logic synthesis, physical implementation (Place and Route) and sign-off (pre-fabrication testing and verification) [10] [19]. Although the commercial design kit is indeed powerful enough to tackle complex systems, there is still a need for significant human effort involved in the design process. If design violations cannot be resolved at the physical design stage through engineer-change-order (ECO) optimisation, engineers turn back to tuning synthesis, or even design adjustments in components or constraints at the system level, to achieve design closure. This is an extremely time-consuming cycle, with the overall design optimisation challenge to find possible optimal trade-off solutions in regard to multiple design requirements using appropriate library cells while reducing turnaround time [20].

B. Discrete Gate-sizing

Selecting the appropriate size for a logic gate to implement circuits down to physical level from discrete libraries is the crucial step to achieve an efficient design and timing closure. The optimisation goal is to minimise power consumption while meeting all timing constraints.

Such constrained optimisation problems have been researched for decades and many approaches to solve them have been proposed. Lagrangian Relaxation (LR) formulation, a mathematical theory, has been established in gate sizing problem with low runtime [21] [22]. The problem is then simplified by weighted factors (Lagrangian multipliers) that moves the constraints into the primal objective function. It is common to simplify circuit models and solve an abstract, so that a continuous version of the gate-sizing can facilitate convex optimisation problems [23]. This approach does not quite generalise in practice, because device physics often imply non-convex delay functions, causing non-convexity in SPICE results and nonlinear delay model (NLDM) tables [24].

Practical approaches like [23] used sensitivity guided greedy metaheuristic to reduce timing violations and then minimise the leakage power. In earlier works, typical heuristic techniques like genetic algorithms were applied to solving gate sizing problems [25] [26] based on weighted sum functions.

Gate sizing problem is multi-objective in nature. Most introduced methods are scalarising based to decompose the optimisation complexity, that combines objectives in one function (e.g., typical weighted sum method). This makes searching highly-efficient but limits achieving feasible Pareto-optimal solutions [27].

More recently, gate-sizing-based soft error optimisation using MOEAs is proposed [28] but its multi-objectives are soft error rate, critical path delay and area. In this work we apply MOEAs in a state-of-the-art digital EDA flow to perform drive-strength-mapping-based design space exploration offering a wide range of Pareto-optimised solutions. The optimisation is enhancing already well-optimised solutions generated by tools.

C. Evolutionary Algorithms

Evolutionary algorithms (EAs) are a class of population-based metaheuristic optimisation algorithms inspired by biological mechanisms like evolution, reproduction, genetics and natural selection. An EA normally starts with an initial *population*, consisting of N *individuals* (candidate solutions), which is allowed to breed with each evolutionary cycle (*generation*). The initial population can be either randomly initialised or seeded with a set of specific configurations. During each generation, individuals can be modified based on their *chromosomes* through genetic operations such as *mutation* or *crossover* (recombination with each other). All individuals are

evaluated using a *fitness* function and ranked according to their fitness score at the end of each generation. Only the fittest individuals survive the selection process forming the subsequent generation. Termination of the evolution process is triggered when specific criteria are met, e.g., sufficient quality of solution or maximum number of generations.

Implementation of EAs requires:

(1) Definition of representation. This is the data structure that the EA manipulates. It represents individuals as a set of genes, the *chromosome*, comprising all variables and parameters necessary to describe it.

(2) Implementation of genetic operators. Mutation and crossover are commonly applied during 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 function is used to calculate a fitness score for each individual based on its performance in design objectives. Fitness scores are used during the ranking and selection process to determine which individuals survive to form the population for the next generation.

D. Multi-objective (MO) EDA Digital Flow

Technology cell mapping is a sub-step in logic synthesis to specify which drive strength would be selected and mapped to each generic functional gate. Gate-level netlists are then generated after the completion of synthesis, where our proposed MOEDA flow focuses on.

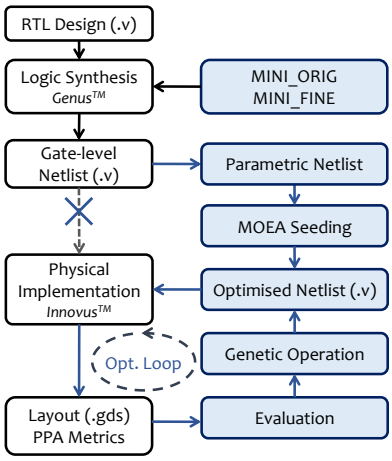

Fig. 2. MOEDA digital flow. The flowchart on the left side illustrates a standard digital flow. The MO evolutionary optimisation engine is shown on the right. The blue cross indicates the position where we break the standard flow. The custom-designed cell libraries are used in this flow instead of using the foundry libraries.

Fig. 2 presents the proposed flow. A multi-objective evolutionary optimisation loop is tapped between logic synthesis and physical implementation. The MOEDA optimiser automatically performs fine-tuning on drive strength selection of logic gates (i.e., inverters in this case) based on produced synthesised netlists. Logic gates are then placed and routed on physical layouts where the evaluation is performed. The proposed flow specifically involves few steps:

First step is to produce the *Parametric Netlist* of a synthesised gate-level design. This parameterisation process specifically encodes the drive strengths of inverters into a set of representations, chromosome $\mathbf{g}$, allowing the MOEA to modify them. In this case, a list of integer parameters define each inverter's drive strength to form a parametric netlist.

Second step is to install an initial population, called *MOEA seeding*. In this work, a solution optimised from the synthesis tool is chosen as the seed. So, seeding a population with a specific solution means all individuals are directly assigned with that solution. This is also the starting point of the MO optimisation process.

After initialising a population, the *Genetic Operation* only performs the modifications on inverters' drive strengths. Only mutation operator is implemented in this work based on a variation probability ρ to indicate how many inverters out of all will be modified. A mutation results in a modified new netlist ready for physical layout implementation. The evolutionary optimisation loop is then iteratively (over a number of generations) updating netlists for increasingly-optimised solutions.

Evaluation is performed using the multi-objective fitness function from equation (2). Metrics (worst case delay D_{wc}, total power consumption P_{total} and all gate area A_{gate}) are calculated by the EDA tool based on the circuit layout instance.

$$\begin{aligned} f(\mathbf{g}) = \min \quad & [D_{wc}(\mathbf{g}), \quad P_{total}(\mathbf{g}), \quad A_{gate}(\mathbf{g})] \\ \text{s.t.} \quad & \mathbf{g} = (g_1, ..., g_i), \quad \forall g_i \in \mathbf{G} \end{aligned} \tag{2}$$

The chromosome vector $\mathbf{g}$ 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 ($\mathbf{G}$). Fig. 3 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.

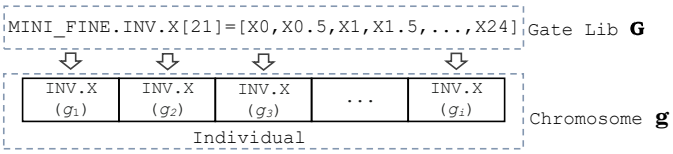

Fig. 3. A chromosome example of an individual (i.e., layout instance in this case). All individuals comprise a population.

In this work, NSGA-II, one of most popular MOEAs [29], has been adapted as the searching tool. The *Non-Dominated-Sorting* and *Diversity Preservation* are introduced to ensure convergence while achieving a uniform spread of Pareto-optimised solutions.

Non-Dominated-Sorting. This is a ranking scheme to evaluate individuals according to their domination level. If an individual p performs better than another q in at least one objective while no degradation in any other objectives, p is said to dominate q. In non-dominated-sorting, each individual (e.g., p) has a domination count, the number of solutions that dominate p. 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 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.

Algorithm 1 Evolutionary Optimisation in the MOEDA Flow

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

Initialize parent population $\mathbf{P}_t$ in size N ▷ Seed with a synthesis-optimised solution generated by the tool.
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 crowned individuals from the 1st 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

Diversity Preservation. This strategy estimates the solution density in the vicinity of each individual based on the Euclidean distance to their nearest neighbours. This needs to firstly calculate the distance of each individual to others and make *Crowding-Distance-Assignment* to each individual, then *Descend-Sort* the $\mathbf{F}$ according to the distance values. 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 overall optimisation process aging with a population in NSGA-II. The MOEDA flow is continuously producing different circuit instances and keeping elitist ones generation-by-generation, then ultimately achieve a set of wide spread of optimised trade-offs in all objectives.

IV. Experiment Setup

We implement the proposed algorithm in C++ and all experiments are running on a 2.2GHz Xeon E5-2650 CPU. Test circuits (RTL designs), from ISCAS85 benchmark suite [30], are synthesised into gate-level netlists using Cadence® Genus™ (v17.11). Cadence® Innovus™ (v17.11) tool completes the physical implementation, producing layout instances.

A. Tool Environment Setup

In order to take full advantage of the built-in optimisations of current EDA tools, it is necessary to push the limits of what tools can achieve with end user-accessible design options. Thus, the *synthesis-compile-effort* is set to high and *ultra-optimisation* is enabled in all experiments presented. In addition, in the timing constraint setup in the Genus™ tool, an ideal clock is created running at 250MHz for all inputs and outputs, which means all circuit paths are clocked with two virtual flip-flops from the beginning to the end of each path. The timing constraint is the required time that designs need to meet so the worst path arrival time should be less than the required time (i.e., 4ns in this case).

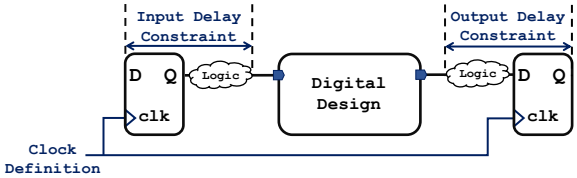

Fig. 4. Conceptual testbench to define timing constraints in EDA tools. Virtual logic parts and flip-flops allow the user to specify delays and clocks in the testbench. The design under test is *Digital Design* in the middle.

To tighten the timing constraint, the output delay is gradually increased as shown in Fig. 4. In this work the output delay constraint is increased in increments of 0.05ns, starting from 0ns. The *Required Timing* is obtained by *Clock Period Time - Output Delay Constraint*. Input delay constraints are not applied in this work. The test 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 loads. The specific values chosen correspond to the respective inverter X1 and X4's input pin capacitance from "MINI_FINE" library.

TABLE III
Design Constraint and 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 noPrePlaceOpt = true timing-driven placement = true timing-driven routing = true
Design Constraint	
set_load = X1/X4 create_clock = 250MHz	

All setup parameters of design constraint, synthesis, place and route are summarised in Table III. The die shape ratio is set to 1.0 and the core utilisation is set to 70%. 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. Because we are investigating how the tools select cells, it is worthwhile to keep netlists consistent during both synthesis and physical design steps. Both timing-driven placement and routing are enabled to make designs achieve the best timing that the tools can achieve automatically.

B. Objective Evaluation in EDA Tools

The fitness function is set up to simultaneously minimise all objectives (D_{wc}, P_{total} and A_{gate}), which aims to make a solution perform better in at least one objective without making performance in others worse. All evaluations take

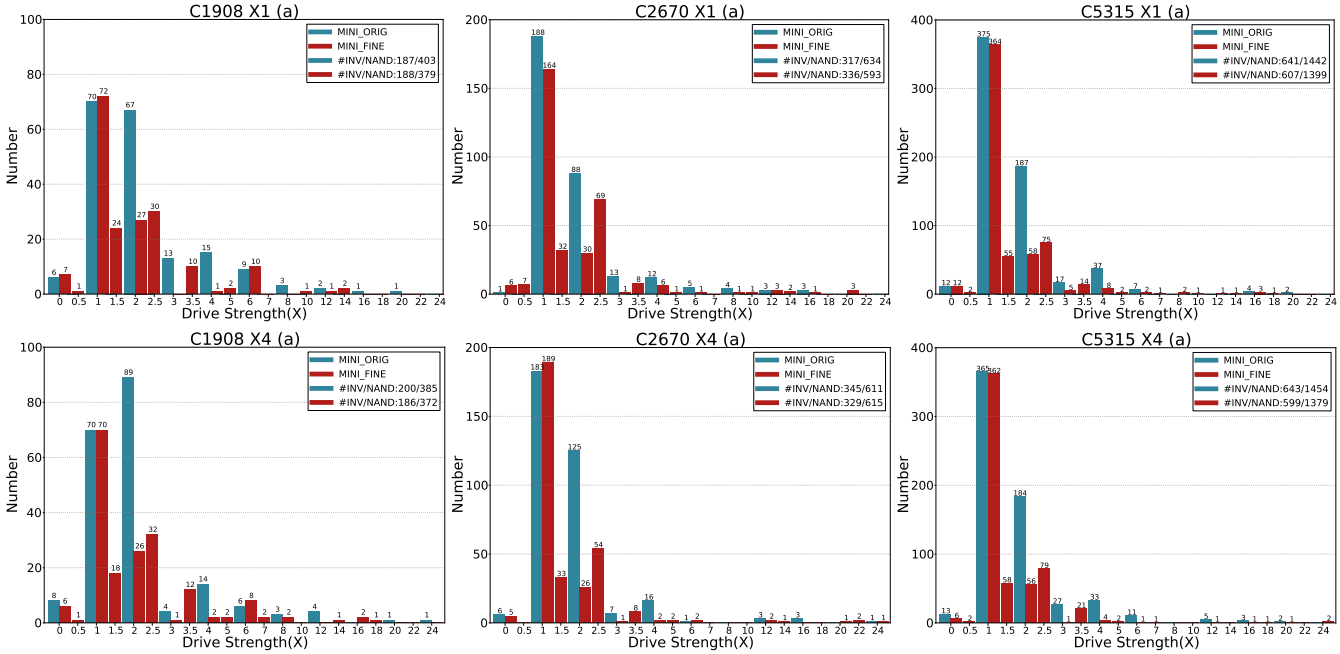

Fig. 5. Each plot shows the number of tool-selected inverters' drive strengths of each case. The blue bars are the inverters in original granularity from "MINI_ORIG" and red bars are the fine-grained inverters from "MINI_FINE". The number of synthesised inverters and nand gates are reported in the legends.

place after place-and-route with Innovus™ analysis based on typical corner conditions.

D_{wc}: This is the signal propagation time of the critical path, which is equal to the required time minus the worst negative slack amongst all path delays. It is calculated by static timing analysis at the post-route stage.

P_{total}: The results from the power analysis in Innovus™. This is the sum of leakage power, internal power and switching power consumption. The leakage and internal power are summarised in the Liberty (.lib) file as stated in Section II-B. Switching power consumed in the charging and discharging of interconnect and load capacitance is calculated based on the equation $\mathrm{P_{switching}} = 0.5 * \mathrm{C_L V^2 F} * \mathrm{A}$, where $\mathrm{C_L}$ is the output capacitive load (pin capacitance tables are available in Liberty (.lib) file for computing output loading), V is the supply voltage, F is frequency, and A is the average switching activity (the value 0.2 used here is the default from the tool).

A_{gate}: The sum of areas of all logic gates. This is directly reported by the Innovus™, based on the layout size of the cells used.

C. Multi-threads Running and Runtime

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

Evolutionary optimisation is population based and requires large numbers of evaluations. In this work, all circuits are placed and routed to achieve accurate metrics as close as possible to sign-off. Such an evaluation needs to be performed for each instance and represents the majority of the overall runtime of the optimisation loop. This can be overcome (sped up) through evaluating instances in the population in parallel using high-performance computing resources. However, each evaluation requires its own license when running solvers in parallel, which makes the achievable degree of parallelism (speedup) dependent on the number of EDA tool licenses available. In addition, the MOEDA flow delivers an entire set of trade-off solutions spanning the feasible design space in one go, rather than just a single, case-specific solution. So the runtime is not the key focus in this work.

V. Experimental Results

A. Original vs. Fine-grained Cells in Standard Digital Flow

We firstly load both "MINI_ORIG" and "MINI_FINE" libraries 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 ISCAS85 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. 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 IV.

TABLE IV
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

Fig. 5 presents 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 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.

Fig. 5 shows that the most-used fine-grained inverters are X1.5, X2.5, X3.5 when "MINI_FINE" library is used. 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. Table V summarises the PPA metrics (D_{wc}, P_{total}, A_{gate}) for each test case. 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 and 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 presented in Table V, 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, although degradation occurred in one of 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.

It explicitly shows 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 gates are also reported in Table V 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.

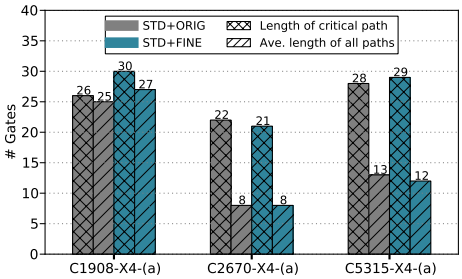

Fig. 6. This shows the changes of circuit paths of each circuit after applying "MINI_FINE" library under the standard flow. The path length is achieved by calculating the gate count of a path.

Synthesising designs with different granularity libraries may produce solutions with different circuit structures. Fig. 6 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.

B. 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. 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" achieve 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 regarding PPA, rather than starting from scratch.

Fig. 7 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 individuals 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. Fig. 8 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. This histogram shows similar drive strength distribution compared to the one of C1908-X1-(a) from Fig. 5, 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

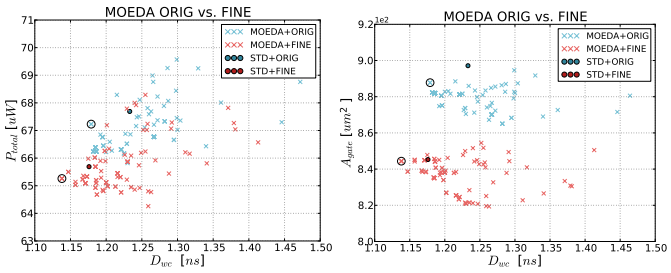

Fig. 7. The MOEDA flow optimisation results comparison between seeding with "STD+ORIG" (blue) and "STD+FINE" (red). The circled solutions are the best delay solution of each cluster.

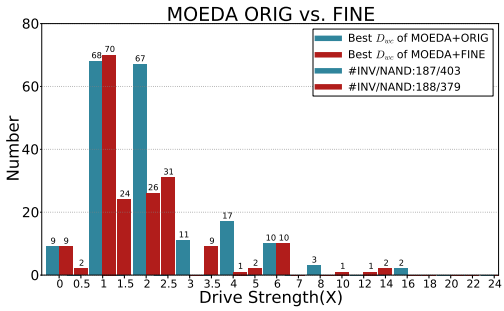

Fig. 8. The inverter histogram of the best delay solution of "MOEDA+ORIG" solution space and "MOEDA+FINE" solution space from Fig. 7.

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 Table V are the best trade-off solutions from the entire final solution space. The best trade-off solution 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 test cases marked with stars represent that "STD+ORIG" solutions have already failed timing requirements. Most of these failed cases have been improved with better D_{wc} in "STD+FINE" solutions, and all of them have been optimised by the MOEDA flow without compromising on other objectives to the point that they achieve timing closure.

Fig. 9 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

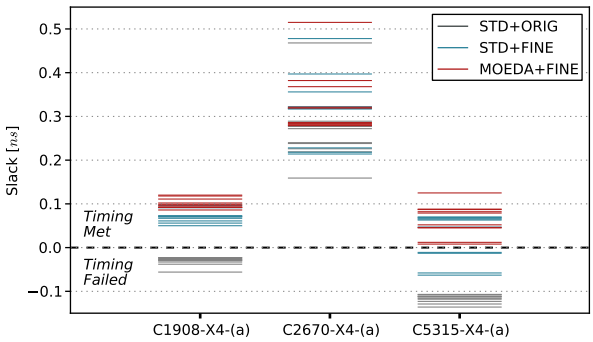

Fig. 9. Ten worst timing paths of test circuits for each corresponding tight timing constraint case X4-(a). All paths above the dash line have positive slacks which meet the timing.

the 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, Fig. 10 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.

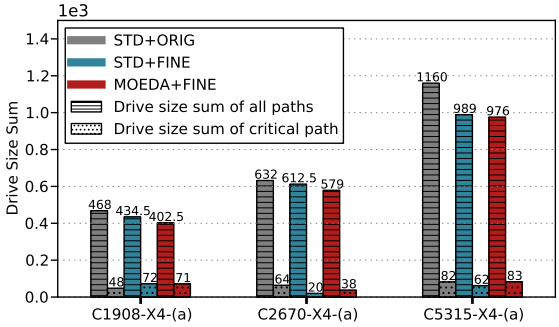

Fig. 10. The changes of drive strengths of critical paths and overall circuits when applying "STD+ORIG", "STD+FINE" and "MOEDA+FINE". The sum of drive strengths are reported from the X4-(a) case of each benchmark.

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 "STD+ORIG", but the total size of selected drive strengths is not always increased. This indicates that for each timing path drive strengths need to be optimised rather than simply scaling up the gate sizes. In C2670 circuit, the drive strength sum of the critical path in "STD+FINE" solution is greatly reduced while all paths are meeting the timing constraint and the worst slack is getting improved. The MOEDA flow then selects more larger cells to push the timing

TABLE V
RESULTS COMPARISON

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

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)**
	(c) $1.55ns$	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)**
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)**
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)**

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 with 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 all trade-off solutions and optimised solution space, Fig. 11 presents the final generation of MOEDA results of the tight timing constraint case X4-(a) of the C5315 benchmark. The "STD+ORIG" and "STD+FINE" solutions are also plotted here comparison. The MOEDA flow has successfully enlarged the feasible solution space while simultaneously achieve significant improvements in all PPA metrics. 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 Table V. The runtime of the largest and most complex case C5315-X4-(a) is 5.5 hours.

VI. CONCLUSION AND FUTURE WORK

We have shown that digital synthesis and implementation tools produced solutions can be improved when provided with fine-grained drive strength cell libraries. The industrial flow exploited the finer drive strengths to improve PPA metrics of all benchmarks used. The results indicate that providing finer

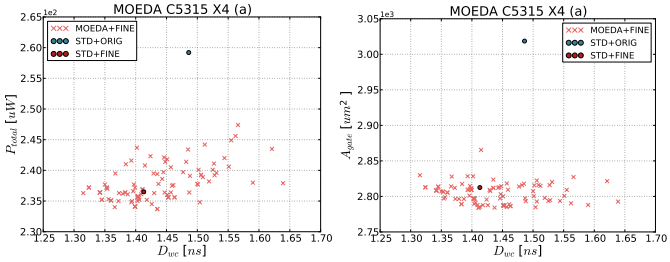

Fig. 11. The left plots in "D_{wc} vs. P_{total}" and "D_{wc} vs. A_{gate}" is on the right. There are two individuals in the round shape are the "STD+ORIG" and "STD+FINE" solutions. All other individuals in the shape of cross are the final generation of MOEDA optimised results based on the "STD+FINE" solution (MOEA seed).

drive resolution around predominantly-selected drive strengths (typically between X0 and X4) is particularly useful. This suggests that enriching drive options of functions of a standard cell library around predominantly selected drive strengths 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. Hence, EDA tools need to make a greater effort during drive strength mapping, due to the increased computational complexity, and may not always arrive at an optimum solution in a given time frame. The proposed MOEDA digital design flow can overcome these issues as it is capable of further balancing PPA metrics 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.

Based on these observations, we will expand fine drive strength granularity to more logic functions and investigate whether further performance, power, area (PPA) benefits can be achieved when handling very large circuits or when fitting designs into constrained floor plans and pin layouts. In addition, exploring how different drive strength combinations will affect the PPA metrics, and how to determine an optimum cell sets for possible best design results, will be investigated.

References

[1] 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 *Proc. ASP-DAC*, IEEE, 2003, pp. 589-590.

[2] 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.

[3] A. M. Islam, S. Nishizawa, Y. Matsui, and Y. Ichida, "Drive-strength selection for synthesis of leakage-dominant circuits," in *20th Int. Symp. on Quality Electronic Design (ISQED)*, IEEE, 2019, pp. 298-303.

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

[5] M. Vujkovic and C. Sechen, "Optimized power-delay curve generation for standard cell ics," in *IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*, 2002, pp. 387-394.

[6] R. Afonso, M. Rahman, H. Tennakoon, and C. Sechen, "Power efficient standard cell library design," in *IEEE Dallas Circuits and Systems Workshop (DCAS)*, IEEE, 2009, pp. 1-4.

[7] M. Rahman, R. Afonso, H. Tennakoon, and C. Sechen, "Design automation tools and libraries for low power digital design," in *IEEE Dallas Circuits and Systems Workshop (DCAS)*, IEEE, 2010, pp. 1-4.

[8] H. Onodera, M. Hashimoto, and T. Hashimoto, "Asic design methodology with on-demand library generation," in *2001 Symp. on VLSI Circuits. Digest of Technical Papers (IEEE Cat. No. 01CH37185)*, IEEE, 2001, pp. 57-60.

[9] G. A. Northrop and P.-F. Lu, "A semi-custom design flow in high-performance microprocessor design," in *Proc. 38th DAC*, IEEE, 2001, pp. 426-431.

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

[12] 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 Symp. on VLSI (ISVLSI)*, IEEE, 2018, pp. 64-69.

[13] 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 *Proc. DATE*, IEEE, 2019, pp. 328-331.

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

[15] M. Palesi and T. Givargis, "Multi-objective design space exploration using genetic algorithms," in *Proc. 10th Int. Symp. on Hardware/software codesign*, 2002, pp. 67-72.

[16] G. Ascia, V. Catania, and M. Palesi, "A framework for design space exploration of parameterized vlsi systems," in *Proc. 7th ASP-DAC and 15th Int. Conf. on VLSI Design*, IEEE, 2002, pp. 245-250.

[17] M. K. Papamichael, P. Milder, and J. C. Hoe, "Nautilus: Fast automated ip design space search using guided genetic algorithms," in *Proc. 52nd DAC*, 2015, pp. 1-6.

[18] A. Ricci, I. De Munari, and P. Ciampolini, "An evolutionary approach for standard-cell library reduction," in *Proc. 17th ACM Great Lakes Symp. on VLSI*, ACM, 2007, pp. 305-310.

[19] 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.

[20] 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.

[21] 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 Trans. on Computer-aided Design of Integrated Circuits and Systems*, vol. 33, no. 4, pp. 546-557, 2014.

[22] A. Sharma, D. Chinnery, and C. Chu, "Lagrangian relaxation based gate sizing with clock skew scheduling-a fast and effective approach," in *Proc. Int. Symp. on Physical Design*, 2019, pp. 129-137.

[23] 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 Int. Conf. on Computer-Aided Design (ICCAD)*, 2012, pp. 233-239.

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

[25] 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.

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

[27] R. Wang, Z. Zhou, H. Ishibuchi, T. Liao, and T. Zhang, "Localized weighted sum method for many-objective optimization," *IEEE Trans. on Evolutionary Computation*, vol. 22, no. 1, pp. 3-18, 2016.

[28] W. Sheng, L. Xiao, and Z. Mao, "Soft error optimization of standard cell circuits based on gate sizing and multi-objective genetic algorithm," in *Proc. 46th DAC*, 2009, pp. 502-507.

[29] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Trans. on Evolutionary Computation*, vol. 6, no. 2, pp. 182-197, 2002.

[30] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS 85)*, IEEE Press, Piscataway, N.J., 1985, pp. 677-692.