NP-Separate: A New VLSI Design Methodology for Area, Power, and Performance Optimization

Monzurul Islam Dewan and Dae Hyun Kim, *Member, IEEE*

***Abstract*—Use of standard cells in the very-large-scale integration (VLSI) design enables very short time to market even for complex microprocessors. Thus, most VLSI layouts are designed using standard cells. In this article, we propose a new design methodology, namely, *NP-Separate*, to reduce the power consumption and area and increase the performance of a VLSI layout more effectively than the standard-cell-based design methodology. NP-Separate uses *N cells* and *P cells* composed of NFETs and PFETs only, respectively, thereby providing a higher degree of flexibility than using standard cells. Our simulation results for several benchmark circuits show that NP-Separate reduces the layout area by 9%, power consumption by 10%, power-delay product by 18%, and energy-delay product by 26% on average while satisfying given timing constraints compared to standard-cell-based designs.**

***Index Terms*—Physical layout design, standard cells, very-large-scale integration (VLSI).**

I. Introduction

STANDARD-CELL-BASED design methodologies provide numerous advantages in the design of very-large-scale integration (VLSI) layouts. For example, drawing long horizontal lines at the top and bottom of the standard cell rows in a layout connects all the power and ground pins of all the standard cell instances in the rows to the main power/ground rings drawn around the core area of the layout. Thus, power/ground network design is greatly simplified [1], [2]. Standard cell placement easily optimizes the locations of all the transistors in the layout by optimizing the locations of the standard cell instances [3]–[5]. Timing and power optimization satisfies given timing constraints and minimizes the power consumption of the design by manipulating (inserting, removing, and relocating) repeaters, upsizing and downsizing standard cell instances, and replacing a set of instances by a different set of instances [6]–[8]. For these reasons, most VLSI layouts are designed using standard cells.

Each standard cell design is optimized to minimize the area of the cell and satisfy constraints such as a target output resistance. For example, if the smallest inverter is designed, the width of the NFET of the inverter is set to the minimum transistor width and that of the PFET is optimized so that the rise and fall times of the inverter are equal. Library characterization performs SPICE simulations to characterize the standard cells and generate a standard cell library. All the synthesis, placement, and routing software use the standard cell library to design VLSI layouts.

Manuscript received February 25, 2019; revised June 11, 2019 and October 1, 2019; accepted December 12, 2019. Date of publication January 13, 2020; date of current version November 20, 2020. This article was recommended by Associate Editor C. Zhuo. *(Corresponding author: Dae Hyun Kim.)*

The authors are with the School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164 USA (e-mail: monzurulislam.dewan@wsu.edu; daehyun@eecs.wsu.edu).

Digital Object Identifier 10.1109/TCAD.2020.2966551

TABLE I
NOTATIONS AND TERMINOLOGIES USED IN THIS ARTICLE

μ_n	Electron mobility
μ_p	Hole mobility
k_μ	μ_n/μ_p
$w_{\min}$	Minimum transistor width
R_{n}	Resistance of an NFET whose width is $w_{\min}$
N cell	A cell composed of NFETs only
P cell	A cell composed of PFETs only
NP cell	A fully-functional cell composed of an N and a P cells

One of the issues the standard-cell-based VLSI design methodology has is that all the design and optimization steps are based on standard cells, so it is impossible to fine-control the size of each transistor for further optimization. For example, suppose an optimization algorithm inserts an inverter into a net. Assuming the optimal sizes of the NFET and the PFET of the inverter are $3w_{\min}$ and $8k_\mu w_{\min}$, respectively, where $w_{\min}$ and k_μ are defined in Table I, the algorithm will likely insert an $8\times$ inverter whose NFET and PFET widths are $8w_{\min}$ and $8k_\mu w_{\min}$, respectively. In this case, the NFET is unnecessarily upsized, which leads to a larger area and higher power consumption. Although the standard cell library might have an inverter cell having the optimal NFET and PFET widths in this case, it would be practically impossible to design and use standard cells having many different combinations of NFET and PFET widths. In general, we cannot avoid overoptimizing some parts of a layout unless we can fine control the sizes of the transistors in the standard cells.

In this article, we propose a new VLSI design methodology, namely, *NP-Separate*, to optimize area, power, and performance of a layout by fine-tuning transistor sizes. NP-Separate designs a layout with *N cells* and *P cells* composed of NFETs and PFETs only, respectively, thereby providing a higher degree of flexibility. We design several layouts using NP-Separate and show that it reduces the layout area by 9%, power consumption by 10%, power-delay product by 18%, and energy-delay product by 26% on average with shorter critical path delays compared to standard-cell-based designs.

The rest of this article is organized as follows. In Section II, we review the conventional standard-cell-based physical VLSI layout design and transistor sizing. In Section III, we present the motivation leading to the NP-Separate design methodology.

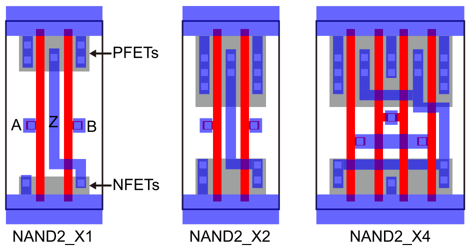

Fig. 1. Three layouts for two-input NAND cells.

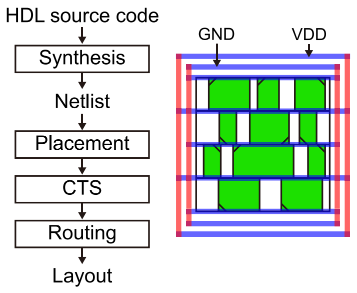

Fig. 2. Simplified standard-cell-based VLSI design process.

Section IV explains the details of NP-Separate. In Section V, we show case studies and compare the two design methodologies. We discuss future work to improve NP-Separate in Section VI, then we conclude in Section VII.

II. Standard-Cell-Based VLSI Design

In this section, we review the conventional standard-cell-based physical VLSI layout design and optimization process, transistor sizing, and multiheight standard cells.

A. Standard Cell Libraries and Standard Cells

A standard cell library for automatic placement and routing generally consists of physical libraries such as library exchange format (LEF) files, timing and power libraries such as Liberty format files, and interconnect technology files. The physical libraries contain physical information of the standard cells (such as the pin locations of a cell) and interconnect layers (such as the minimum width of a metal layer) in the standard cell library. The timing and power libraries contain timing, power, and functional information of the standard cells (such as the delay of a cell for various input slews and output loads). The interconnect technology files contain detailed information for interconnect resistance and capacitance (RC) extraction. In general, PFETs and NFETs of a standard cell are placed in the upper and lower regions of the cell layout, respectively, as shown in Fig. 1. Placing all the NFETs (or PFETs) in the same area enables the transistors to share their diffusion regions, which helps reduce the cell area.

B. Physical Design of VLSI Layout

Fig. 2 shows a simplified standard-cell-based physical VLSI layout design process. Physical design (including netlist synthesis) of a VLSI layout synthesizes a netlist from given hardware description language (HDL) source codes, places the standard cell instances in the netlist on a layout, performs clock-tree synthesis (CTS), and routes the instances. Timing and power optimization is performed before placement, CTS, routing, and after routing. Design-rule violations, such as max. capacitance violations, are also fixed during the physical design. All of these steps use standard cells. For example, gate sizing upsizes or downsizes instances for area, power, and performance optimization. Upsizing or downsizing an instance replaces it by a new standard cell instance having the same function with a different size (e.g., a NAND2_X4 instance is replaced by a NAND2_X1 instance). Similarly, repeater insertion inserts repeater instances for delay minimization.

C. Transistor Sizing

The sizes of the transistors in a standard cell are properly optimized for various purposes, such as delay minimization and fall/rise time matching. Since different transistor sizes have different input capacitances and output resistances, standard cell libraries generally have multiple cell sizes for each cell. For instance, a two-input NAND cell has three definitions, NAND2_X1, NAND2_X2, and NAND2_X4 in Fig. 1. The sizes of the transistors in the NAND2_X2 and NAND2_X4 cells are two and four times as large as those in the NAND2_X1 cell. Thus, NAND2_X2 and NAND2_X4 have lower output resistance and larger input and output capacitance than NAND2_X1. However, they might not occupy a larger area than NAND2_X1 in terms of the cell area because increasing the sizes of the transistors does not necessarily lead to a larger cell area as shown in Fig. 1.

Suppose the resistance of an NFET whose width is $w_{\min}$ is R_n. In this case, the resistance of a PFET whose width is $w_{\min}$ is $k_\mu \cdot R_n$. For an inverter, if the load capacitance is C_L and a given timing constraint is $\tau = R_n C_L$, the widths of the NFET and the PFET of the inverter are set to $w_{\min}$ and $k_\mu \cdot w_{\min}$, respectively. Similarly, if the timing constraint is $\tau = R_n C_L / r$, the widths of the NFET and the PFET of the inverter are set to $r \cdot w_{\min}$ and $r \cdot k_\mu \cdot w_{\min}$, respectively.

D. Multiheight Standard Cells

Recently, Baek *et al.* [9] proposed designing VLSI layouts using multiheight standard cells. The multiheight-standard-cell-based design methodology (MHSC) uses single-height standard cells for simple cells, such as inverters and multiheight standard cells for complex cells such as flip-flops. MHSC minimizes the layout area by reducing the height of the single-height cells and designing complex cells in double-height cells. The restriction of using the metal 1 layer only in the standard cell design unnecessarily increases the standard cell height and the area of complex cells. Thus, designing complex cells across two rows and using the metal 2 layer for power and ground routing helps reduce the layout area [9], [10]. To support the placement of mixed-height standard cells, several placement legalization algorithms have been proposed [11], [12]. In this article, we reduce the layout area by using NFETs and PFETs whose sizes are optimized separately. We also demonstrate how to incorporate optimal

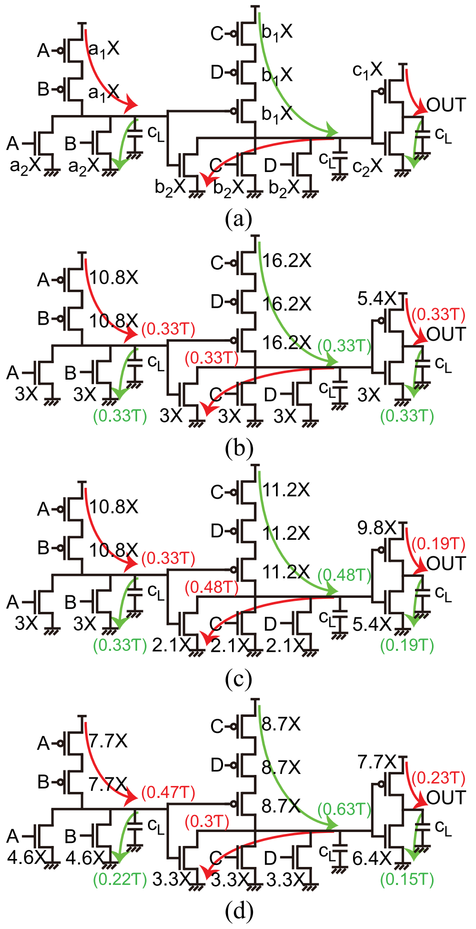

Fig. 3. (a) Single path transistor sizing example for a NOR2-NOR3-INV configuration. (b) Brute-force. (c) Heuristic. (d) Optimal.

transistor sizes in the automatic layout generation to minimize area, power, and performance. Thus, we can apply NP-Separate to MHSC to reduce the layout area further.

III. Motivation

This section shows the motivation of this article with an example. Fig. 3(a) shows a signal path composed of a two-input NOR instance (NOR2), a three-input NOR instance (NOR3), an inverter instance (INV), and some load capacitors. Some parasitic capacitances are not shown in the figure. We assume that all the NFETs (or PFETs) of each cell have the same width as shown in the figure. For example, the width of all the PFETs in the NOR2 instance is $a_1\times$, which is $a_1 \cdot w_{\min}$. We also assume that the load capacitance of each instance is C_L just for simplification. $k_\mu = \mu_n/\mu_p$ is set to 1.8. The red arrows show the signal flow of NOR2 = 1, NOR3 = 0, and INV = 1, which means that the outputs of the NOR2, NOR3, and INV instances are 1, 0, and 1, respectively. Similarly, the green arrows show the signal flow of NOR2 = 0, NOR3 = 1, and INV = 0. A given timing constraint is $\tau = R_n C_L$.

Fig. 3(b) shows the result of a brute-force transistor sizing algorithm by which each instance is upsized to $3\times$ so that the path delay is evenly distributed throughout the three instances. Thus, the delay of each instance is $(1/3)\tau$ and the total transistor width is $93.6w_{\min}$. Fig. 3(c) shows the result of a heuristic transistor sizing algorithm by which NOR2, NOR3, and INV are upsized to $a\times$, $b\times$, and $c\times$, respectively. The algorithm minimizes the total transistor width as follows:

$$\text{Minimize} \quad W = \left(2+4k_\mu\right)a + \left(3+9k_\mu\right)b + \left(1+k_\mu\right)c \tag{1}$$

$$\text{Subject to} \quad \text{Rising: } \frac{2k_\mu \cdot R_n}{2k_\mu a} C_L + \frac{R_n}{b} C_L + \frac{k_\mu \cdot R_n}{c} C_L \le \tau \tag{2}$$

$$\text{Falling: } \frac{R_n}{a} C_L + \frac{3k_\mu \cdot R_n}{3k_\mu b} C_L + \frac{R_n}{c} C_L \le \tau. \tag{3}$$

Solving the above problem leads to $(a, b, c) = (3\times, 2.1\times, 5.4\times)$. The delays of the NOR2, NOR3, and INV instances are 0.33τ, 0.48τ, and 0.19τ, respectively. The total transistor width is $82.6w_{\min}$, which is approximately 11.8% smaller than that of the brute-force algorithm.

Fig. 3(d) shows the result of an optimal algorithm by which the PFETs and NFETs of NOR2, NOR3, and INV are upsized to $(a_1\times, a_2\times)$, $(b_1\times, b_2\times)$, and $(c_1\times, c_2\times)$, respectively. The following formulates the problem:

$$\text{Minimize} \quad W = 2(a_1 + a_2) + 3(b_1 + b_2) + (c_1 + c_2) \tag{4}$$

$$\text{Subject to} \quad \text{Rising: } \frac{2k_\mu \cdot R_n}{a_1} C_L + \frac{R_n}{b_2} C_L + \frac{k_\mu \cdot R_n}{c_1} C_L \le \tau \tag{5}$$

$$\text{Falling: } \frac{R_n}{a_2} C_L + \frac{3k_\mu \cdot R_n}{b_1} C_L + \frac{R_n}{c_2} C_L \le \tau. \tag{6}$$

Solving the above problem leads to $(a_1, a_2) = (7.7\times, 4.6\times)$, $(b_1, b_2) = (8.7\times, 3.3\times)$, and $(c_1, c_2) = (7.7\times, 6.4\times)$. The total transistor width is $74.7w_{\min}$, which is 20.2% and 9.6% smaller than the sizes of the transistors optimized by the brute-force and heuristic algorithms, respectively. For the rising path, the delays of the NOR2, NOR3, and INV instances are 0.47τ, 0.3τ, and 0.23τ, respectively. For the falling path, the delays are 0.22τ, 0.63τ, and 0.15τ, respectively. The delays are unevenly distributed among the three instances as shown above and even the PFETs and NFETs of an instance have different delay values.

Table II also compares the three transistor sizing algorithms for various paths. For example, the total transistor width of the NOR4-NOR4-NOR4-NOR4 path optimized by the brute-force and heuristic algorithms is $524.80w_{\min}$, whereas that optimized by the optimal sizing is $434.13w_{\min}$. Thus, the optimal sizing algorithm achieves 17.28% smaller transistor width than the other two algorithms. Note that the optimal transistor sizing has been proposed in [13]–[18], some of which used more complicated but accurate delay models such as the Elmore delay model. In addition, they also minimized the total area or power consumption and we can also take the internal capacitances into account [15], [18].

IV. NP-Separate: New VLSI Design Methodology

In this section, we propose a new VLSI design methodology, namely, *NP-Separate*, to minimize the layout area and power consumption of a given design.

TABLE II
TOTAL AREAS (UNIT: $w_{\min}$) OF SINGLE PATH CIRCUITS OPTIMIZED BY THE BRUTE-FORCE, HEURISTIC, AND OPTIMAL SIZING ALGORITHMS FOR DIFFERENT LOGIC GATE COMBINATIONS

# Stages	Logic gate combinations of single path circuits	Area			%-Improvement	
		Brute-force (A)	Heuristic (B)	Optimal (C)	$\frac{\lvert C-A\rvert\times 100}{A}$	$\frac{\lvert C-B\rvert\times 100}{B}$
4	NOR4-NOR4-NOR4-NOR4	524.80	524.80	434.13	17.28%	17.28%
	NOR3-NOR2-NOR3-NOR4	321.60	307.08	264.00	17.91%	14.30%
	INV-NOR4-NAND3-NAND4	292.80	256.38	238.62	18.50%	6.93%
	INV-NAND2-NOR3-NOR4	249.60	211.38	187.65	24.82%	11.23%
8	INV-NOR4-NOR4-INV-NAND2-INV-NOR4-NOR4	1177.60	941.59	815.72	30.73%	13.37%
	NAND2-NAND2-NOR3-NOR4-INV-NOR4-NOR4-XOR2	1264.00	1121.11	996.74	21.14%	11.09%
	NAND2-NOR2-NOR4-NOR4-INV-NAND2-INV-INV	787.20	626.05	566.60	28.02%	9.50%
	INV-NAND2-NAND4-NOR4-XOR2-XOR2-INV-NOR4	1174.4	1013.79	961.38	18.14%	5.17%

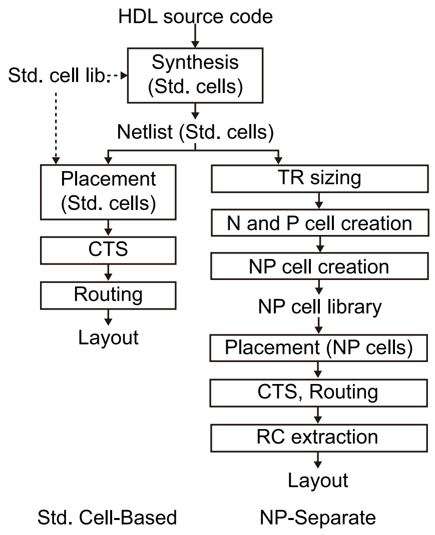

Fig. 4. Standard-cell-based and NP-Separate VLSI design flows.

A. Overview

Fig. 4 shows an overview of the NP-Separate design methodology we propose. First, we begin the design from the synthesis of a given HDL source code using a plain standard cell library. The synthesis generates a netlist composed of standard cell instances. Then, we size the transistors of the instances using the optimal transistor sizing explained in the previous section. When we optimally size the transistors, we estimate the load capacitance of each instance using the standard cell library and add the capacitance to the timing constraints. The optimal transistor sizing gives us the size of each transistor of each standard cell instance. Then, we create N and P cells having the sizes found by the optimal sizing. The creation of the N and P cells includes layout drawing, design-rule check (DRC), and layout-versus-schematic (LVS). Then, we create an NP cell corresponding to each standard cell instance by merging the N and P cells. The creation of an NP cell includes physically placing an N and a P cell in a layout editor and routing the input and output ports. The next step is to replace the standard cell instances with the NP cell instances in the layout. We also prepare a physical library in LEF for the NP cells and use the library and a commercial router to perform CTS and route the NP cells. The following sections show more details of each step.

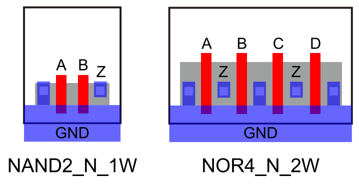

Fig. 5. NAND2_N_1W and NOR4_N_2W cells.

B. N Cells and P Cells

N and P cells are similar to standard cells. However, N cells have NFETs only and P cells have PFETs only. Although the transistors in an N or P cell can have different widths, *we apply the same width to all the transistors in an N or P cell* for the following reasons. First, applying different widths to the transistors in an N or P cell leads to too many N or P cell designs, which will increase the overall design time significantly. Second, timing constraints are greatly simplified if all the transistors in a cell have the same width.

An N cell is named *Func*_N_*s*W, where *Func* is the function of the cell such as NAND2, N denotes the type of the cell (N cell), and *s*W is the size of each NFET in the cell. For example, NOR4_N_2W is a four-input NOR N cell and the width of each NFET in the cell is $2 \cdot w_{\min}$. Fig. 5 shows our layouts for NAND2_N_1W and NOR4_N_2W. Notice that the two output ports in the NOR4 cell are not routed yet, although they could be prerouted in the N cell. In our methodology, they are routed after the creation of an NP cell. A P cell is named similarly like *Func*_P_*s*W. Once we design N and P cells, we can reuse them to create NP cells. Thus, creating N and P cells in Fig. 4 will create only the N and P cells missing in the NP cell library.

C. NP Cell Creation

Once the size of each transistor is optimized in the transistor sizing step, we create all the required N and P cells. Then, we create NP cells by merging and routing the N and P cells as follows. First, each standard cell instance in the synthesized netlist is replaced by an N and a P cell instances as shown in Fig. 6. For example, the optimal sizes of the NFETs and PFETs of the NAND2_X1 instance in Fig. 6(a) are $2w_{\min}$ and $2w_{\min}$, respectively. Thus, we create an NP cell NAND2_N_2W_P_2W by combining a NAND2_N_2W instance and a NAND2_P_2W instance as shown in Fig. 6(b).

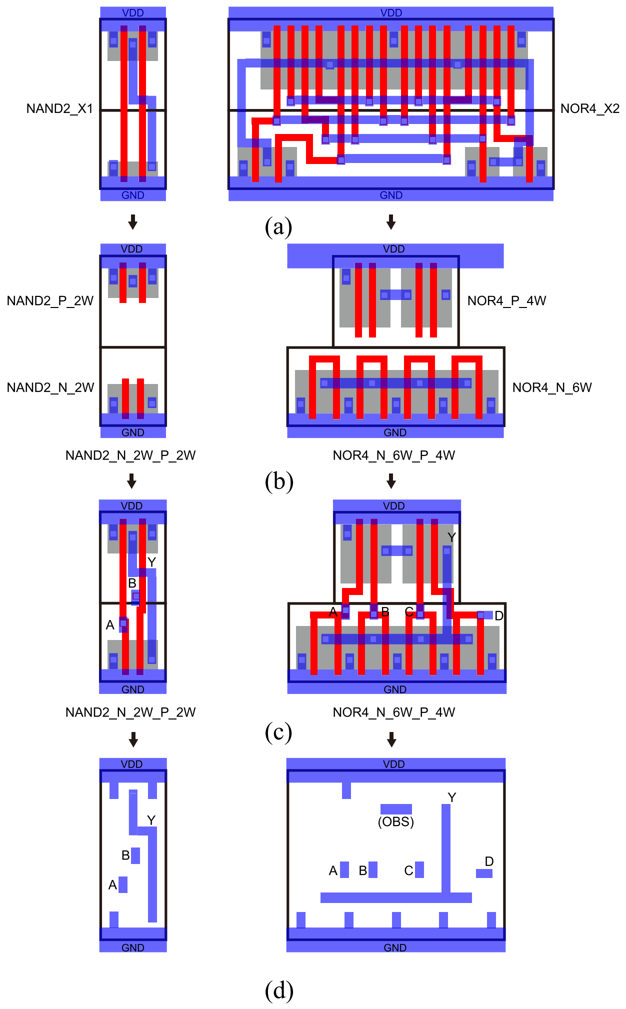

Fig. 6. (a) Two standard cells NAND2_X1 and NOR4_X2. (b) NP cell creation. (c) Input and output port routing. (d) Abstraction.

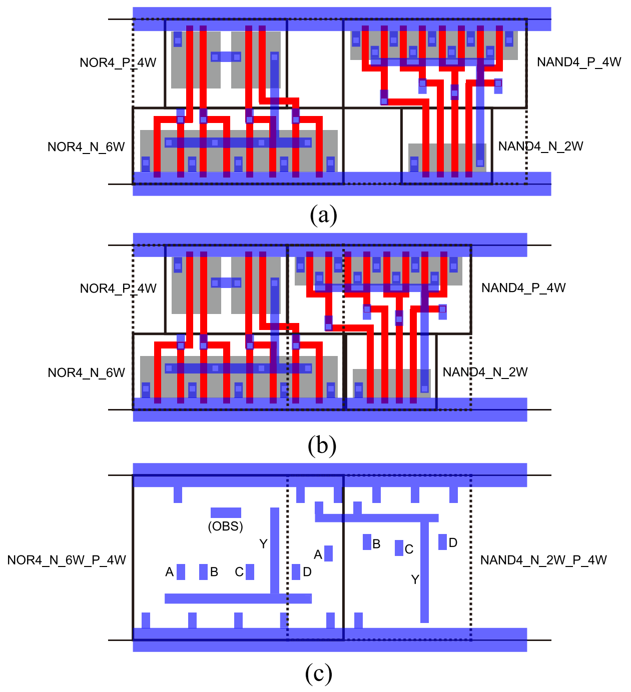

Fig. 7. Optimization in an NP-Separate design. (a) Two NP cell instances placed by a placement software. (b) Area minimization by instance overlap. (c) Abstracted view of the instances.

Similarly, the optimal sizes of the NFETs and PFETs of the NOR4_X2 instance are $6w_{\min}$ and $4w_{\min}$, respectively. Thus, we create an NP cell NOR4_N_6W_P_4W by combining a NOR4_N_6W instance and a NOR4_P_4W instance. The PFETs in the NOR4_P_4W instance are separated into two diffusion regions to reduce the complexity of input/output port routing. Notice that the centers of the NOR4_N_6W and NOR4_P_4W instances are aligned, but they do not need to be aligned.

Once we merge an N cell and a P cell into an NP cell, we route the input and output ports using the poly and metal 1 layers as shown in Fig. 6(c) so that the NP cell becomes a fully functional cell. We manually route them in this article, but we can route them automatically using commercial tools. We also create input and output pins like standard cell pins. All the processes in the routing and pin creation follow all the design rules, so the final NP cell layout is DRC-clean. We also perform LVS for the NP cell to verify its function.

DRC and LVS are followed by abstraction as shown in Fig. 6(d). The abstraction process for NP cells is exactly the same as that for standard cells. One thing to notice is that the boundary (width and height) of an NP cell is determined by the maximum width of the N and P cells in the NP cell and the sum of the heights of the N and P cells. Thus, the shape of an NP cell is always a rectangle like standard cells. The abstraction of the NP cells creates a physical library in LEF format so that the NP cell instances can be routed automatically using commercial tools.

D. Placement, CTS, and Routing

Once we create all the NP cells and an NP cell library for the cells, we place all the NP cell instances in the layout. We use a commercial tool to place the NP cell instances. Notice that the physical library of the NP cells looks similar to that of standard cells. Thus, placing the NP cell instances does not differ from placing standard cell instances. Similarly, CTS and signal routing can be performed using commercial tools.

However, there is a fundamental difference between the two design methodologies. In NP-Separate, we can optimize the locations of the transistors further after placement so that we can reduce the area further. For example, a NOR4_N_6W_P_4W and a NAND4_N_2W_P_4W NP cell instances are placed in Fig. 7(a). Notice that the two instances cannot overlap because placement software will avoid overlapping the instance boundaries. In this case, however, we can shift the NAND4_N_2W_P_4W instance to the left as shown in Fig. 7(b) so that we can reduce the total area further. Fig. 7(b) shows that the two instances overlap, but it is allowed in NP-Separate as long as all the layout objects such as wires do not violate the design rules. Fig. 7(c) shows that the instances overlap in the abstracted design level, but they can be routed by commercial tools.

V. Case Study

In this section, we design five 8-bit adders, three 32-bit adders, a 4-bit multiplier, a 64-bit pipelined multiplier, and a data encryption standard (DES) core using the conventional standard-cell-based and NP-Separate design methodologies and compare the quality of the designs. The five 8-bit adders are ripple-carry adder (RCA), carry-lookahead adder (CLA), Brent–Kung adder (BKA), Kogge–Stone adder (KSA), and binary coded decimal adder (BCD). The three 32-bit adders are 32-bit version of the CLA, BKA, and KSA. The 4-bit multiplier is a Wallace-tree-based parallel multiplier (WT). The pipelined multiplier, MUL_B64 is a high-throughput 64-bit multiplier with nine pipelined stages [19]. The DES core, DES_PERF is a five-stage DES encryption core optimized for performance. We chose these benchmarks because they have different connectivity and routing patterns. For example, the RCA is slow, but has very low routing complexity. On the other hand, the CLA is fast, but has forward and backward paths among its submodules. Both the BKA and KSA are high-speed prefix adders, but the former has a large logic depth with low routing complexity, whereas the latter has the minimum logic depth with very high routing complexity. The BCD adder and the multiplier also have unique logic and routing characteristics. The MUL_B64 and DES_PERF are hierarchically designed, partitioned into several pipeline stages, and much larger than the other benchmarks.

A. Layout Design and Analysis

We first synthesized HDL source codes for the benchmarks using a 22-nm standard cell library and Cadence Genus. We turned on all the optimization options in Genus to optimize the critical path delay, power, and area. For the standard-cell-based layouts, which we will call *S-designs* in this article, we used Cadence Innovus for placement and routing. We will call the NP-Separate layouts *NP-designs*. For both the S- and NP-designs, we set the initial core area utilization to almost 100% so that the layouts have no white space. After we obtained the S- and NP-designs, we imported the layouts into Cadence Virtuoso, performed parasitic RC extraction, simulated the final netlists with the parasitic RC using Synopsys HSpice, and obtained critical path delays and power consumption. We used ALGLIB [20] to solve the nonlinear optimization for transistor sizing. For the NP-designs of MUL_B64 and DES_PERF, we designed them hierarchically starting from the bottommost-level modules.

B. Simulation Results

1) Transistor Width and Layout Area: Table III compares the S- and NP-designs. First, the average transistor width of the NP-designs is 20% less than that of the S-designs. In addition, the average transistor width reduction by NP-Separate for the five large benchmarks (BKA32, CLA32, KSA32, MUL_B64, and DES_PERF) is 22%. This is because NP-Separate can freely size the transistors, whereas the conventional VLSI design tools cannot. However, the actual cell area reduction ratio is less than the transistor width reduction ratio. The table shows that the NP-designs have 4%–13% smaller area than the S-designs and the average area of the NP-designs is 9% less than that of the S-designs. For the five large benchmarks, the NP-designs occupy 8% less area on average. Since we set the core utilization of the layouts to almost 100%, this area benefit is the actual benefit we can obtain from the NP-Separate design methodology.

2) Wirelength: The NP-designs also have shorter wirelength than the S-designs. For all the benchmarks except CLA32 and DES_PERF, the NP-designs have up to 12% shorter wirelength than the S-designs. For CLA32, the wirelength reduction ratio is 27%, which is because the S-design has 11 rows for placement, whereas the NP-design has ten rows. As a result, the CLA32 NP-design reduced the wirelength dramatically. On the other hand, the S-designs and NP-designs have the same number of rows for most of the other benchmarks, so the wirelength reduction ratio of the designs is relatively smaller than that of CLA32 design. DES_PERF has a different reason for the huge wirelength reduction ratio (26%). Both the S- and NP-designs use the same number of rows, so the area difference between them is only 4%. However, we found that optimizing the timing of the S-design was much harder than optimizing the NP-design. As a result, the S-design used much more upper metal layers and had many routing detours. At the same time, the coupling capacitance of the S-design is much larger than that of the NP-design due to the detours as shown in the table. However, the S- and NP-designs have almost equal parasitic resistance. In summary, optimizing the total transistor size by NP-Separate helps reduce not only the area and wirelength but also the parasitic capacitance.

3) Critical Path Delay and Power Consumption: Table III also compares the critical path delay, power consumption, power-delay product (PDP), and energy-delay product (EDP) of the S- and NP-designs. When we solved the nonlinear optimization problems for the NP-designs, we applied slightly tighter timing constraints to them than the S-designs because we did not use an accurate delay model for the sizing. Thus, the NP-designs have a 9% shorter critical path delay on average than the S-designs. However, this does not mean that the comparison is not fair. Rather, it means that the NP-designs can have even smaller area than the area shown in Table III with the same critical path delays as the S-designs if we slightly loosen the timing constraints for the NP-designs.

The NP-designs also have 10% lower power consumption on average than the S-designs because the NP-designs have a smaller capacitance than the S-designs. Since all the NP-designs have shorter critical path delays and lower power consumption than the S-designs, the NP-designs have smaller PDP and EDP (18% and 26% on average, respectively). In addition, for the five large benchmarks, NP-designs have 10% lower power consumption with 16% smaller PDP and 22% smaller EDP on average than the S-designs.

4) Temperature: All the NP-designs consume less power than the S-designs, but the NP-designs have smaller area than the S-designs. Thus, the power density of an NP-design could be smaller or larger than its S-design counterpart depending on their power and area reduction ratios. As Table III shows, the NP-designs of RCA08, BKA08, CLA32, MUL_B64, and

TABLE III

COMPARISON OF THE LAYOUTS BUILT BY THE CONVENTIONAL STANDARD-CELL-BASED (DENOTED BY S-DESIGN) AND NP-SEPARATE (DENOTED BY NP-DESIGN) DESIGN METHODOLOGIES FOR THE CRITICAL PATH DELAY (CPD), POWER CONSUMPTION, POWER-DELAY PRODUCT (PDP), ENERGY-DELAY PRODUCT (EDP), POWER DENSITY, AND MAXIMUM AND MINIMUM TEMPERATURE. "# INSTS" DENOTES THE NUMBER OF INSTANCES. TR WIDTH IS THE TOTAL TRANSISTOR WIDTH (UNIT: $w_{\min}$). WL IS THE TOTAL WIRELENGTH. R IS THE TOTAL PARASITIC RESISTANCE AND C AND Cc ARE THE TOTAL PARASITIC GROUND AND COUPLING CAPACITANCES, RESPECTIVELY, EXTRACTED FROM THE LAYOUT

Benchmark	Design type	# Insts	# Nets	TR width	Area (μm^2)	WL (μm)	R (kΩ)	C (fF)	Cc (fF)
RCA08	S-design	64	81	1334 (1.00)	21.60 (1.00)	224 (1.00)	273 (1.00)	21 (1.00)	118 (1.00)
	NP-design			767 (0.57)	19.80 (0.92)	221 (0.99)	265 (0.97)	20 (0.95)	93 (0.79)
CLA08	S-design	79	96	1650 (1.00)	28.35 (1.00)	281 (1.00)	351 (1.00)	24 (1.00)	135 (1.00)
	NP-design			1520 (0.92)	26.10 (0.92)	257 (0.91)	344 (0.98)	23 (0.96)	124 (0.92)
BKA08	S-design	79	95	1488 (1.00)	25.20 (1.00)	247 (1.00)	304 (1.00)	23 (1.00)	128 (1.00)
	NP-design			1368 (0.92)	22.50 (0.89)	230 (0.93)	300 (0.99)	21 (0.91)	109 (0.85)
KSA08	S-design	112	129	2179 (1.00)	35.64 (1.00)	328 (1.00)	434 (1.00)	28 (1.00)	191 (1.00)
	NP-design			1788 (0.82)	31.05 (0.87)	314 (0.96)	440 (1.01)	26 (0.93)	157 (0.82)
BCD08	S-design	124	141	2610 (1.00)	42.84 (1.00)	355 (1.00)	520 (1.00)	31 (1.00)	224 (1.00)
	NP-design			2371 (0.91)	37.80 (0.88)	338 (0.95)	504 (0.97)	29 (0.94)	202 (0.90)
WT04	S-design	100	108	2036 (1.00)	32.85 (1.00)	279 (1.00)	434 (1.00)	26 (1.00)	156 (1.00)
	NP-design			1717 (0.84)	29.00 (0.88)	258 (0.92)	426 (0.98)	24 (0.92)	147 (0.94)
BKA32	S-design	260	324	5193 (1.00)	88.20 (1.00)	1082 (1.00)	1138 (1.00)	55 (1.00)	475 (1.00)
	NP-design			3837 (0.74)	81.00 (0.92)	1012 (0.94)	1126 (0.99)	52 (0.95)	505 (1.06)
CLA32	S-design	271	336	6142 (1.00)	102.96 (1.00)	1447 (1.00)	1334 (1.00)	62 (1.00)	630 (1.00)
	NP-design			4555 (0.74)	90.90 (0.88)	1061 (0.73)	1245 (0.93)	57 (0.92)	487 (0.77)
KSA32	S-design	305	370	6001 (1.00)	99.99 (1.00)	1309 (1.00)	1304 (1.00)	61 (1.00)	585 (1.00)
	NP-design			4331 (0.72)	91.80 (0.92)	1151 (0.88)	1292 (0.99)	58 (0.95)	549 (0.94)
MUL_B64	S-design	73303	90076	1090740 (1.00)	44428 (1.00)	202384 (1.00)	406084 (1.00)	17444 (1.00)	185808 (1.00)
	NP-design			967260 (0.89)	41807 (0.94)	197484 (0.98)	411768 (1.01)	16884 (0.97)	170380 (0.92)
DES_PERF	S-design	112075	122605	1455680 (1.00)	47580 (1.00)	990325 (1.00)	584245 (1.00)	18725 (1.00)	364015 (1.00)
	NP-design			1214640 (0.83)	45455 (0.96)	728510 (0.74)	584045 (1.00)	17588 (0.94)	278515 (0.77)
			Geo. mean (NP / S)	0.80	0.91	0.90	0.98	0.94	0.88

Benchmark	Design type	CPD (ps)	Power (uW)	PDP (fJ)	EDP (yJ·s)	Power density (W/um)	Min. temp. (K)	Max. temp. (K)
RCA08	S-design	470 (1.00)	56 (1.00)	26 (1.00)	12 (1.00)	2.59 (1.00)	301.39 (1.00)	301.45 (1.00)
	NP-design	380 (0.81)	44 (0.79)	17 (0.65)	6 (0.50)	2.22 (0.86)	301.15 (1.00)	301.20 (1.00)
CLA08	S-design	310 (1.00)	68 (1.00)	21 (1.00)	7 (1.00)	2.40 (1.00)	301.65 (1.00)	301.71 (1.00)
	NP-design	270 (0.87)	66 (0.97)	18 (0.86)	5 (0.71)	2.53 (1.05)	301.62 (1.00)	301.68 (1.00)
BKA08	S-design	270 (1.00)	66 (1.00)	18 (1.00)	5 (1.00)	2.62 (1.00)	301.58 (1.00)	301.64 (1.00)
	NP-design	265 (0.98)	58 (0.88)	15 (0.83)	4 (0.80)	2.58 (0.98)	301.42 (1.00)	301.48 (1.00)
KSA08	S-design	360 (1.00)	91 (1.00)	33 (1.00)	12 (1.00)	2.55 (1.00)	302.21 (1.00)	302.28 (1.00)
	NP-design	290 (0.81)	80 (0.88)	23 (0.70)	7 (0.58)	2.58 (1.01)	301.93 (1.00)	302.00 (1.00)
BCD08	S-design	455 (1.00)	127 (1.00)	58 (1.00)	26 (1.00)	2.96 (1.00)	303.10 (1.00)	303.19 (1.00)
	NP-design	430 (0.95)	125 (0.98)	54 (0.93)	23 (0.88)	3.31 (1.12)	303.01 (1.00)	303.11 (1.00)
WT04	S-design	325 (1.00)	78 (1.00)	25 (1.00)	8 (1.00)	2.37 (1.00)	301.87 (1.00)	301.93 (1.00)
	NP-design	315 (0.97)	74 (0.95)	23 (0.92)	7 (0.88)	2.55 (1.08)	301.78 (1.00)	301.85 (1.00)
BKA32	S-design	635 (1.00)	141 (1.00)	90 (1.00)	57 (1.00)	1.60 (1.00)	303.56 (1.00)	303.62 (1.00)
	NP-design	540 (0.85)	132 (0.94)	71 (0.79)	38 (0.67)	1.63 (1.02)	303.39 (1.00)	303.45 (1.00)
CLA32	S-design	710 (1.00)	140 (1.00)	99 (1.00)	70 (1.00)	1.36 (1.00)	303.49 (1.00)	303.54 (1.00)
	NP-design	640 (0.90)	116 (0.83)	74 (0.75)	47 (0.67)	1.28 (0.94)	303.01 (1.00)	303.07 (1.00)
KSA32	S-design	695 (1.00)	153 (1.00)	106 (1.00)	74 (1.00)	1.53 (1.00)	303.81 (1.00)	303.87 (1.00)
	NP-design	690 (0.99)	142 (0.93)	98 (0.92)	68 (0.92)	1.55 (1.01)	303.69 (1.00)	303.76 (1.00)
MUL_B64	S-design	973 (1.00)	75404 (1.00)	73368 (1.00)	71387 (1.00)	1.70 (1.00)	321.39 (1.00)	323.23 (1.00)
	NP-design	966 (0.99)	70392 (0.93)	67999 (0.93)	65687 (0.92)	1.68 (0.99)	320.83 (1.00)	322.74 (1.00)
DES_PERF	S-design	350 (1.00)	57075 (1.00)	19976 (1.00)	6992 (1.00)	1.20 (1.00)	317.28 (1.00)	318.39 (1.00)
	NP-design	335 (0.96)	48905 (0.86)	16383 (0.82)	5488 (0.78)	1.08 (0.90)	316.56 (1.00)	317.58 (1.00)
	Geo. mean (NP / S)	0.91	0.90	0.82	0.74	0.99	1.00	1.00

DES_PERF have 1%–14% smaller power densities than their S-design counterparts. However, the NP-designs of the other benchmarks have 1%–12% higher power densities than their S-design counterparts. On average, the S- and NP-designs have almost the same power density. In addition to the power density computation, we also performed thermal simulation using 3D-ICE [21] to show the minimum and maximum temperature values of the designs. We observe in Table III that the differences between the minimum and maximum temperatures of all the designs are less than 2 °C. We also find that the maximum temperature difference between an S-design and its NP-design counterpart is less than 1 °C.

5) In-Depth Analysis: Since the S- and NP-designs use the same netlist for each benchmark, we also compare the widths of the NFETs and PFETs of the NP cell instances in the NP-designs and their corresponding standard cell instances in the S-designs for CLA08 and WT04 in Fig. 8. The x-axis shows the instance indices in the netlists. The y-axis shows the differences of the transistor widths of the NP and standard cell instances corresponding to the instance index. If the difference is positive, the transistor width in the NP-design is larger than that in the S-design. We observe from the figure that if the NFETs of a standard cell instance are upsized (or downsized) in its corresponding NP-cell instance, the PFETs are also upsized (or downsized) in most cases. This is counterintuitive because the example in Fig. 3 shows that the PFETs connected in series are generally downsized while the NFETs connected in parallel are generally upsized in the NP-designs compared to the S-designs.[1] Since the PFET network of a cell

[1]The heuristic optimization algorithm is similar to what is used for the standard-cell-based design methodologies.

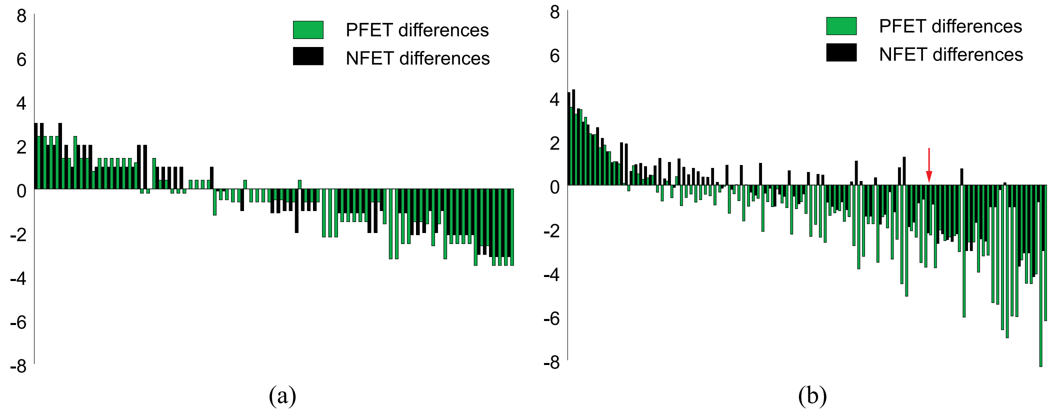

Fig. 8. Differences of the widths of the NFETs and PFETs of the NP cell instances in the NP-designs and the corresponding standard cell instances in the S-designs for (a) CLA08 and (b) WT04. x-axis: The instance indices. y-axis: The differences of the transistor widths (unit: $w_{\min}$).

is the dual network of the NFET network of the cell, the PFETs are connected in series if the NFETs are connected in parallel and *vice versa*. Thus, if the NFETs of an instance are downsized (or upsized), the PFETs of the instance are expected to be upsized (or downsized). In Fig. 8, however, the NFETs and PFETs of an instance are either upsized or downsized together. This is mainly because both the NFETs and PFETs of the instances in the critical paths are upsized, whereas both the NFETs and PFETs of the instances in the noncritical paths are downsized.

In addition, the NFETs and PFETs of an instance have different sizing ratios. In some NP cell instances, the PFETs are significantly downsized while the NFETs are not sized at all. For example, the instance marked by a red arrow in Fig. 8(b) is a two-input NOR instance. In the S-design, the instance is upsized to 2×, so the widths of the NFETs and PFETs become $2w_{\min}$ and $7.2w_{\min}$, respectively. In the NP-design, on the other hand, the NFETs are not upsized (so their widths are $1w_{\min}$) and the PFETs are upsized to $1.2w_{\min}$. Thus, the sizing ratios between the S- and NP-designs for the NFETs and PFETs of the instance are 2 and 6, respectively.

6) Design Time: Table IV shows the design time for the S- and NP-designs. We used an Intel Xeon CPU E5-2650 v3 (2.30 GHz) computer for all the simulations. The netlist synthesis took just a few seconds for all the designs. However, the transistor sizing step spent a few seconds for small designs such as RCA08 to 40 h for large designs such as DES_PERF. Although the MUL_B64 and DES_PERF designs are much larger than the other designs, they have several pipeline stages and each pipeline is hierarchically organized. Thus, we applied NP-Separate to each subdesign with proper timing budgeting, which reduced the runtime significantly. Creating (drawing a layout and performing DRC and LVS) an N or P cell took approximately 20 min on average. Similarly, creating (drawing input and output pins and performing DRC and LVS) an NP cell took about 15 min on average. However, after we created the N and P cells for a design, we reused them for the other designs, so the N- and P-cell creation time is a one-time design cost. For the NP-cell creation time, if we can automatically route the input and output pins of an NP cell, the NP-cell creation time will be reduced significantly. We discuss the automatic input/output pin routing in Section VI-D. Innovus spent only a few seconds to perform placement and routing.

TABLE IV
COMPARISON OF THE RUNTIME OF EACH STAGE. "P&R" DENOTES PLACEMENT AND ROUTING

Benchmark	Design type	Synthesis	TR sizing	N/P creation	NP creation	P&R
RCA08	S NP	1s	- 7s	20m	15m	4s
CLA08	S NP	1s	- 6s			
BKA08	S NP	1s	- 17s			
KSA08	S NP	1s	- 45s			
BCD08	S NP	2s	- 1h39m56s			
WT04	S NP	2s	- 1m55s			
BKA32	S NP	4s	- 1h11m55s			
CLA32	S NP	5s	- 2h15m22s			
KSA32	S NP	5s	- 13h51m53s			
MUL_B64	S NP	10s	- 37h 20m			20s
DES_PERF	S NP	15s	- 42h 40m			

7) Iterative Transistor Sizing: In the transistor sizing step, the load capacitance of each instance is unknown. Thus, we estimate the load capacitance of an instance using a given standard cell library as follows. Suppose the output of an instance drives a two-input NAND and a two-input NOR instances. Then, we obtain the input capacitance (c_1) of a two-input NAND gate of a certain size and that (c_2) of a two-input NOR gate of a certain size and set the load capacitance of the target instance to the sum ($c_1 + c_2$) of the two capacitances. Once we set the load capacitances for all the instances, we perform transistor sizing and obtain the size of each transistor. Then, we modify the load capacitances using the result of the transistor sizing and perform transistor sizing again.

TABLE V
MINIMUM, MAXIMUM, AND AVERAGE OF THE DIFFERENCES (IN fF) OF THE LOAD CAP. OF TWO SUCCESSIVE ITERATIONS FOR THE NP-DESIGNS

Benchmark	Iteration-1			Iteration-2			Iteration-3		
	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg
RCA08	-0.84	0.34	-0.21	-0.15	0.00	-0.02	-0.09	0.00	-0.01
CLA08	-0.79	0.32	-0.08	-0.10	0.05	-0.01	0.00	0.00	0.00
BKA08	-0.81	0.53	-0.04	-0.07	0.11	0.00	0.00	0.05	0.00
KSA08	-0.75	0.34	-0.14	-0.11	0.11	-0.01	-0.08	0.05	0.00
BCD08	-1.61	0.85	-0.10	-0.24	0.21	-0.02	-0.05	0.06	0.00
WT04	-1.29	0.34	-0.22	-0.22	0.05	-0.02	-0.10	0.00	-0.01
BKA32	-1.27	0.62	-0.22	-0.20	0.19	-0.03	-0.10	0.09	0.00
CLA32	-1.72	0.95	-0.33	-0.25	0.29	-0.01	-0.09	0.09	0.00
KSA32	-1.21	0.34	-0.21	-0.24	0.11	-0.02	-0.22	0.03	-0.01
MUL_B64	-0.81	0.40	-0.03	-0.22	0.20	-0.01	0.00	0.00	0.00
DES_PERF	-1.88	0.50	-0.22	-0.21	0.09	-0.03	-0.07	0.01	0.00

Table V shows the minimum, maximum, and average differences between estimated and actual load capacitances at each iteration step. As the table shows, the absolute value of the average difference after the first transistor sizing (Iteration-1) is approximately 0.03 fF to 0.33 fF. After the second transistor sizing (Iteration-2), the average difference becomes almost zero.

VI. DISCUSSION

In this section, we discuss several issues existing in NP-Separate and future work for the adoption of NP-Separate in the VLSI layout design.

A. NP Cell Library

In Fig. 4, the NP cell library has only a physical library, but not a timing and power library. The reason is that we allow overlapping N and P instances in the placement. Thus, if we characterize the NP cells and create a timing and power library, we can use the timing and power library without allowing instance overlaps. If we use the library and allow instance overlaps, however, the accuracy of the library will go down, so we will have to perform a full-design RC extraction again in a later design stage. We can also build NP-designs without the timing and power library as we showed in the case study. In this case, however, the timing and power estimation is postponed to the end of the layout design, so timing closure and optimization will need several iterations from the placement to the routing steps. We think the second method would be the best among them, but better methodologies should be developed for accurate timing and power estimation for overlapped NP cell instances.

B. Transistor Sizing

An issue in NP-Separate is that the number of timing constraints has a great impact on the runtime of the transistor sizing. Therefore, if the circuit size increases, the runtime for transistor sizing will go up significantly. In our simulation, we were able to successfully optimize all the transistors in the 11 benchmarks. However, we failed to solve the nonlinear optimization problem for more complex benchmarks because the solver could not find optimal solutions in a reasonable amount of time. We can resolve the issue by using more efficient algorithms [22], [23], fine-grained design partitioning, multilevel hierarchical design, and progressive sizing.

C. Algorithms—Placement

Since we can treat NP cells as standard cells, we can use any placement algorithm to place NP cell instances. However, overlapping NP cell instances requires additional algorithms, especially new detailed placement algorithms. We formulate the detailed placement problem for NP-Separate as follows. The objective would be minimizing the total layout area. There could be several constraints, but at least the following three constraints should be imposed. First, the wirelength after the detailed placement (overlapping NP cell instances) should be within a certain range. For example, the wirelength overhead should be less than 5%, otherwise, the layout area minimization might lead to performance degradation. Second, the maximum displacement of each NP cell instance should also be within a certain range. Third, the N and P cell instances belonging to an NP cell instance should be aligned. For instance, suppose a wide N cell instance whose width is w_N and a narrow P cell instance whose width is w_P are combined into an NP cell instance. If the x-coordinate of the bottom-left corner of the NP cell instance is x_{NP}, the following should be satisfied:

$$x_N = x_{NP} \tag{7}$$

$$x_{NP} \leq x_P \leq x_{NP} + (w_N - w_P) \tag{8}$$

where x_N and x_P are the x-coordinates of the N and P cell instances, respectively. Then, the detailed placement problem is to adjust the locations of the N and P cell instances (i.e., overlap NP cell instances) under the constraints so that the objective function is minimized.

Algorithm 1 shows a stochastic algorithm based on simulated annealing for the detailed placement. The input consists of a global placement result S_o generated from a commercial tool, the initial temperature T_o and the stopping temperature T_f for the annealing process, and the initial iteration number M_o. α is the cooling rate, β controls the number of iterations for each specific temperature, and maxTime and maxClimb are the maximum limits for the iterations and the number of hill climbing, respectively. First, we set the current temperature T to T_o, the current layout S and the best layout S_{best} to S_o, the maximum number of moves M to M_o, the current time t to zero, and the number of rejects r to zero (line 1). If t is less than the time limit, T is greater than T_f, and r is less than a certain number (line 2), we call the metropolis function in which we perturb the current layout to obtain better layouts and adjust r, t, T, and M (lines 3 and 4).

In the metropolis function, we first set the number of rejected moves to zero (line 7) and repeat perturbing the current layout M times (line 8). We obtain a new layout S_{new} from the current layout S (line 9) by the solution_perturbation function. S_{new} might be NULL in which case we just reduce M and repeat the solution perturbation (lines 10–13). If S_{new} is not NULL, we compute the difference of the costs between the new and the current layouts (line 14). The cost function could be the layout area for area minimization. If the cost difference is less than zero (i.e., the new layout is better than the current layout) or a randomly generated number is less than $e^{-k \cdot \Delta\text{cost}/T}$, we replace the current layout by the new layout (line 16) and adjust the number of hill climbing (line 18) and the best layout (line 21).

The solution perturbation is 1) moving a randomly chosen NP cell instance from its current row to one of its adjacent rows or to a different location in the same row or 2) swapping two randomly chosen NP cell instances. To maximize the effect of moving NP cell instances, however, we precompute the difference d between the widths of the N and P cells for each NP cell instance. If d is positive, the N cell has a larger width than the P cell. Thus, if two horizontally adjacent NP cell instances have positive (or negative) d values, we cannot overlap the instances. However, if one of them has a positive d value, whereas the other has a negative d value, we might be able to overlap them to minimize the total area. Thus, we choose a positive-d and a negative-d NP cell instances for the solution perturbation.

In line 29 in Algorithm 1, we randomly choose a row (case 1) or two adjacent rows (case 2). For case 1, the row becomes the p- and the n-rows for the perturbation. For case 2, we randomly choose one of the rows for its p-row and the other becomes its n-row. Then, we insert all the instances having positive d values in the p-row to set P and all the instances having negative d values in the n-row to set N (lines 30 and 31). If one of the sets is empty, we return NULL (line 33). Otherwise, we randomly choose an instance I_P from P and an instance I_N from N (lines 35 and 36). Then, we move I_P (or I_N) to a location adjacent to I_N (or I_P) or swap I_P and I_N (line 37). If the wirelength of the new layout is larger than that of the current layout by at least 5% after the perturbation, we reject the move (line 39). Otherwise, we return the new layout (line 41). Notice that moving NP cell instances requires adjustment of the locations of some or all of the NP cell instances in the rows affected by the move. Thus, the move in line 37 should include the adjustment of the locations of the affected instances. Once the detailed placement finishes, we can overlap adjacent NP cell instances by checking the boundaries of the N and P cells of the NP cell instances as shown in Fig. 7.

Algorithm 1 A Simulated-Annealing-Based Detailed Placement Algorithm

```
Function: detailed_placement ($S_o$, $T_o$, $T_f$, $M_o$, $\alpha$, $\beta$,
    maxTime, maxClimb)
 1: $T = T_o$, $S = S_o$, $S_{best} = S_o$, $M = M_o$, $t = 0$, $r = 0$;
 2: while $t \leq$ maxTime and $T > T_f$ and $\frac{r}{t} < 0.95$ do
 3:     $r = r +$ metropolis ($S, T, M$);
 4:     $t = t + M$; $T = \alpha \cdot T$; $M = \beta \cdot M$;
 5: end while
 6: Return $S_{best}$;
Function: metropolis ($S, T, M$)
 7: reject = 0; climb = 0;
 8: while $M > 0$ and climb $\leq$ maxClimb do
 9:     $S_{new}$ = solution_perturbation ($S$);
10:     if $S_{new}$ == NULL then
11:         $M = M - 1$;
12:         continue;
13:     end if
14:     $\Delta cost$ = cost($S_{new}$) - cost($S$);
15:     if $\Delta cost \leq 0$ or random() $< e^{\frac{-k \cdot \Delta cost}{T}}$ then
16:         $S = S_{new}$;
17:         if $\Delta cost > 0$ then
18:             climb = climb + 1;
19:         end if
20:         if cost($S$) < cost($S_{best}$) then
21:             $S_{best} = S$;
22:         end if
23:     else
24:         reject = reject + 1;
25:     end if
26:     $M = M - 1$;
27: end while
28: Return reject;
Function: solution_perturbation ($S$)
29: Randomly pick two rows from $S$, same or adjacent.
30: Set $P$ = {all the $+d$-value instances in the $p$-row};
31: Set $N$ = {all the $-d$-value instances in the $n$-row};
32: if $P$.size == 0 or $N$.size == 0 then
33:     Return NULL;
34: end if
35: $I_P$ = Randomly pick an instance from $P$.
36: $I_N$ = Randomly pick an instance from $N$.
37: Move $I_P$ (or $I_N$) to an adjacent location of $I_N$ (or $I_P$) or
    swap $I_P$ and $I_N$.
38: if $\frac{\textbf{hpwl}(S_{new})}{\textbf{hpwl}(S)} > 1.05$ then
39:     Return NULL;
40: end if
41: Return $S_{new}$;
```

D. Algorithms—Routing (Input and Output Pins)

After the placement, we should route the input and output pins of the N and P cell instances. We routed them manually in this article, but automatic algorithms for the routing of input and output pins would help reduce the design time. One way to automatically route them is to use a signal net router. Suppose an NP cell instance has two inputs A and B and an output Y. Then, the N and P cell instances have their own input and output pins. In this case, we can create a physical library for the N and P cell instances and a netlist having the two instances. The netlist has two nets for the input pins A and B and a net for the output pin. Then, any router can route the three nets using the netlist and only one metal layer. However, more efficient algorithms dedicated for the routing of the inputs and outputs of N and P cell instances would help minimize the parasitic RC of the instances. For example, if we allow only I-, L-, or Z-shaped routing topologies for the input and output pins, we can generate all possible I-, L-, and Z-shaped routing topologies for each pin and choose best shapes for all the pins concurrently by linear programming. If some of the pins need detours, we can route the inner pins first and the rest of them by maze routing.

E. Synthesis and Layout Optimization

In Fig. 4, we synthesized a netlist using a standard cell library, then converted the standard cell instances into NP cell instances. However, synthesis software should be able to use NP cell libraries to directly synthesize netlists of NP cell instances. Timing and power optimization algorithms will also have to handle N and P cell instances. For example, sizing an NP cell instance should be able to size the N and P cell instances of the NP cell instance separately, which will also require rerouting of the input and output pins of the NP cell instance. Basically, all the optimization algorithms such as repeater insertion should be able to optimize a given layout using N and P cells.

F. Issues in NP-Separate

NP-Separate minimizes the layout area by incorporating optimal transistor sizes and overlapping NP cell instances. As a result, the pin density of an NP-design is higher than its S-design counterpart, which might lead to an unroutable layout. In this case, NP-Separate can mitigate the routability issue by inserting white space into the congested area. Overlapping NP cell instances as shown in Fig. 7 will also increase the parasitic coupling capacitance between them. Thus, it might degrade the performance of the instances, which could also be mitigated by inserting white space between the instances critically affected by the overlaps. In addition, we perform RC extraction and characterization for the whole design after placement and routing for accurate timing and power analysis of the design. Thus, if the design is too large, it would be better to characterize the NP cells and generate a timing and power library after creating the cells and use the library during placement and routing without cell overlapping.

In this article, we sized the transistors of inverters, NAND, NOR, XOR, and XNOR cells only. A main reason that we did not apply NP-Separate to more complex combinational and sequential cells is as follows. First, the simple cells have very regular, symmetric layout patterns. Thus, all the NFETs of an N cell (or PFETs of a P cell) can have the same width, thereby reducing the runtime for transistor sizing. In addition, since the transistors of an N or P cell have the same width, the number of N or P cells to create for a simple cell such as NAND2 is not too many. For example, the smallest and largest widths of the two-input NAND cells used in the 11 benchmarks are $1\times$ and $16\times$, respectively. Thus, there are total 16 different N cells for two-input NAND. However, suppose the N cell of a complex cell has n different widths for the NFETs in the cell. If each of the widths can be $1\times$ to $m\times$, the total number of N cells that should be generated for the complex cell is $m \cdot n$ in the worst case. In addition, the layout of the N cell might become very irregular.

VII. Conclusion

In this article, we have proposed a new VLSI design methodology, NP-Separate, to optimize area, power, and performance of a VLSI layout. NP-Separate uses N and P cells to incorporate optimal transistor sizes in the VLSI layout generation. The simulation results for all the benchmarks show that NP-Separate reduces the layout area by 9%, power consumption by 10%, PDP by 18%, and EDP by 26%. We also discussed future work to apply the NP-Separate design methodology to larger circuits. We believe that this article will initiate research and development of new, effective algorithms for NP-Separate to optimize area, power, and performance further.

References

[1] T. Mitsuhashi and E. S. Kuh, "Power and ground network topology optimization for cell based VLSIs," in *Proc. ACM Design Autom. Conf.*, Anaheim, CA, USA, 1992, pp. 524–529.

[2] X. Wu, C. Qiao, and X. Hong, "Design and optimization of power/ground network for cell-based VLSIs with macro cells," in *Proc. Asia South Pac. Design Autom. Conf.*, Hong Kong, 1999, pp. 21–24.

[3] G.-J. Nam and J. J. Cong, *Modern Circuit Placement: Best Practices and Results*. New York, NY, USA: Springer, 2007.

[4] J. Lu *et al.*, "ePlace: Electrostatics-based placement using fast Fourier transform and Nesterov's method," *ACM Trans. Design Autom. Electron. Syst.*, vol. 20, no. 2, pp. 1–34, Mar. 2015.

[5] Z. Zhu, J. Chen, Z. Peng, W. Zhu, and Y.-W. Chang, "Generalized augmented Lagrangian and its applications to VLSI global placement," in *Proc. ACM Design Autom. Conf.*, Jun. 2018, pp. 1–6.

[6] L. P. P. P. van Ginneken, "Buffer placement in distributed RC-tree networks for minimal Elmore delay," in *Proc. IEEE Int. Symp. Circuits Syst.*, Vancouver, BC, Canada, May 1990, pp. 865–868.

[7] J. Cong, J. Lee, and L. Vandenberghe, "Robust gate sizing via mean excess delay minimization," in *Proc. Int. Symp. Phys. Design*, Apr. 2008, pp. 10–14.

[8] Y.-M. Jiang, A. Krstic, K.-T. Cheng, and M. Marek-Sadowska, "Post-layout logic restructuring for performance optimization," in *Proc. ACM Design Autom. Conf.*, Jun. 1997, pp. 662–665.

[9] S.-H. Baek, H.-Y. Kim, Y.-K. Lee, D.-Y. Jin, S.-C. Park, and J.-D. Cho, "Ultra-high density standard cell library using multi-height cell structure," in *Proc. SPIE*, vol. 7268, 2008, pp. 1–8.

[10] S. Dobre, A. B. Kahng, and J. Li, "Mixed cell-height implementation for improved design quality in advanced nodes," in *Proc. IEEE Int. Conf. Comput.-Aided Design*, Austin, TX, USA, Nov. 2015, pp. 854–860.

[11] C.-H. Wang *et al.*, "An effective legalization algorithm for mixed-cell-height standard cells," in *Proc. Asia South Pac. Design Autom. Conf.*, 2017, pp. 450–455.

[12] C.-Y. Hung, P.-Y. Chou, and W.-K. Mak, "Mixed-cell-height standard cell placement legalization," in *Proc. Great Lakes Symp. VLSI*, May 2017, pp. 149–154.

[13] J. P. Fishburn and A. E. Dunlop, "TILOS: A posynomial programming approach to transistor sizing," in *Proc. IEEE Int. Conf. Comput.-Aided Design*, 1985, pp. 326–328.

[14] M. A. Cirit, "Transistor sizing in CMOS circuits," in *Proc. ACM Design Autom. Conf.*, 1987, pp. 121–124.

[15] J.-M. Shyu, A. Sangiovanni-Vincentelli, J. P. Fishburn, and A. E. Dunlop, "Optimization-based transistor sizing," *IEEE J. Solid-State Circuits*, vol. JSSC-23, no. 2, pp. 400–409, Apr. 1988.

[16] B. A. Richman, J. E. Hansen, and K. Cameron, "A deterministic algorithm for automatic CMOS transistor sizing," *IEEE J. Solid-State Circuits*, vol. JSSC-23, no. 2, pp. 522–526, Apr. 1988.

[17] A. R. Conn, P. K. Coulman, R. A. Haring, G. L. Morrill, and C. Visweswariah, "Optimization of custom MOS circuits by transistor sizing," in *Proc. IEEE Int. Conf. Comput.-Aided Design*, San Jose, CA, USA, 1996, pp. 174–180.

[18] V. Sundararajan, S. S. Sapatnekar, and K. K. Parhi, "Fast and exact transistor sizing based on iterative relaxation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 21, no. 5, pp. 568–581, May 2002.

[19] J. Seo and D. H. Kim, "High-throughput multiplier architectures enabled by intra-unit fast forwarding," in *Proc. IEEE Int. Symp. Comput. Arithmetic*, Jun. 2019, pp. 143–150.

[20] B. S. Anatolyevich, (2020). *ALGLIB*. [Online]. Available: http://www.alglib.net

[21] A. Sridhar, A. Vincenzi, D. Atienza, and T. Brunschwiler, "3D-ICE: A compact thermal model for early-stage design of liquid-cooled ICs," *IEEE Trans. Comput.*, vol. 63, no. 10, pp. 2576–2589, Oct. 2014.

[22] S. Joshi and S. Boyd, "An efficient method for large-scale gate sizing," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 9, pp. 2760–2773, Oct. 2008.

[23] S. Daboul, N. Hähnle, S. Held, and U. Schorr, "Provably fast and near-optimum gate sizing," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 12, pp. 3163–3176, Dec. 2018.

Monzurul Islam Dewan received the B.Sc. degree in electrical and electronic engineering from the Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in 2012. He is currently pursuing the Ph.D. degree with the Department of Electrical Engineering and Computer Science, Washington State University, Pullman, WA, USA.

He served as a Lecturer with the Department of Electrical and Electronic Engineering, Ahsanullah University of Science and Technology, Dhaka, from 2013 to 2015. His research interests include circuit optimization, design automation and computer-aided design algorithms for very large-scale integration, and architecture for high-performance computing.

Dae Hyun Kim (Member, IEEE) received the B.S. degree in electrical engineering from Seoul National University, Seoul, South Korea, in 2002, and the M.S. and Ph.D. degrees in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 2007 and 2012, respectively.

He is an Assistant Professor with the School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA, USA. He worked on physical layout optimization with Cadence Design Systems, Inc., San Jose, CA, USA, from 2012 to 2014. His research interests include electronic design automation and computer-aided design for VLSI, high-performance and/or low-power VLSI and computer systems, and 3-D integrated circuits and systems.

Dr. Kim received the Defense Advanced Research Projects Agency Young Faculty Award in 2016.