Design Automation Algorithms for the NP-Separate VLSI Design Methodology

MONZURUL ISLAM DEWAN and DAE HYUN KIM, Washington State University

The NP-Separate design methodology for **very-large-scale integration (VLSI)** design fine-controls the sizes of transistors, thereby achieving significant power, performance, and area improvement compared to the conventional standard-cell-based design methodology. NP-Separate uses NP cells formed by merging and routing N and P cells having only NFETs and PFETs, respectively. The NP cell formation, however, should be automated to design large circuits using the NP-Separate design methodology. In this paper, we propose design automation algorithms to create NP cells automatically. Simulation results show that the automated NP-Separate reduces the design time significantly, decreases the coupling capacitance by 13%, the critical path delay by 6%, and the power consumption by 10% on average compared to the manual NP-Separate designs. We also propose a detailed placement algorithm to generate more compact VLSI layouts with a little wirelength overhead. The combined effect reduces the coupling capacitance by 10%, the critical path delay by 5%, and the power consumption by 10% on average compared to the manual NP-Separate designs.

CCS Concepts: • **Hardware** → *Standard cell libraries*; **Physical design (EDA);**

Additional Key Words and Phrases: NP-Separate, N cell, P cell, design automation algorithms, cell overlapping, detailed placement, cell-level routing

ACM Reference format:
Monzurul Islam Dewan and Dae Hyun Kim. 2022. Design Automation Algorithms for the NP-Separate VLSI Design Methodology. *ACM Trans. Des. Autom. Electron. Syst.* 27, 5, Article 53 (June 2022), 20 pages.
https://doi.org/10.1145/3508375

1 INTRODUCTION

In the **very-large-scale integration (VLSI)** design world, the standard-cell-based design methodology has vast popularity for efficient layout design [6]. The standard-cell-based design methodology uses standard cells and standard cell libraries for logic synthesis and physical layout design to optimize power, performance, and area with shorter design time and less human effort [7, 11–13, 23, 26]. Standard cells are optimized to minimize the cell area and satisfy given constraints such as equal rise and fall delays. However, a standard cell can be optimized further by designing its NFETs and PFETs separately in an *N cell* and a *P cell*, respectively, and merging them into a single cell. This introduced a new VLSI design methodology, *NP-Separate*, which reduced the chip area further and showed significant power and performance improvement over the standard-cell-based design methodology [3]. Figure 1 shows a simplified NP-Separate based VLSI layout.

Authors' address: M. I. Dewan and D. H. Kim, Washington State University, School of Electrical Engineering and Computer Science, 355 NE Spokane St, Pullman, WA, 99164; emails: {monzurulislam.dewan, daehyun.kim}@wsu.edu.

1084-4309/2022/06-ART53 $15.00
https://doi.org/10.1145/3508375

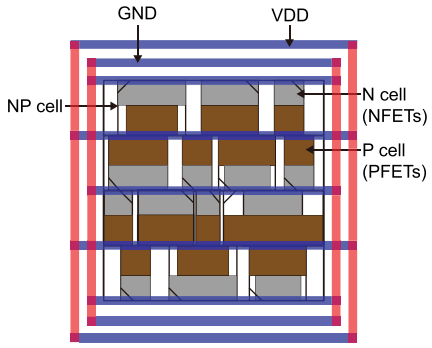

Fig. 1. A simplified NP-Separate based VLSI layout.

Briefly speaking, suppose the optimal sizes of the NFETs and PFETs of a standard cell instance are $n \cdot w_{\min}$ and $p \cdot w_{\min}$, respectively ($w_{\min}$ is the minimum transistor width). If the standard cell library does not have the cell having the optimal transistor sizes, however, physical design software will use a substitutive standard cell, which will over-optimize the paths going through the instance. NP-Separate, however, creates an *NP cell* by creating and merging an N cell having NFETs of the optimal size $n \cdot w_{\min}$ and a P cell having PFETs of the optimal size $p \cdot w_{\min}$, thereby avoiding the over-optimization of the design.

However, the NP-Separate design methodology is highly manual tangled with the following issues. First, manual NP-Separate creates NP cells from N and P cells, which requires placing N-P pairs, routing their input and output ports with poly and metal 1 (M1) layers, and creating input and output pins to create **design-rule check (DRC)** and **layout-versus-schematic (LVS)** clean NP cells. All these design steps are very time-consuming and error-prone. Second, the manual routing of the input and output ports and creation of the input and output pins cannot assure effective use of the given routing resources, which will increase the coupling capacitance. Finally, NP-Separate allows cell overlapping by which NP cells are overlapped to generate more compact designs. This necessitates design automation algorithms for both the NP cell placement and routing.

In this paper, we propose design automation algorithms, namely *Auto-NP-Separate*, to automate the manual NP-Separate design methodology. The algorithms automate all the design steps (routing of input and output ports, creation of input pins, and detailed placement) in NP-Separate. Noticeably, Auto-NP-Separate can generate more optimized design layouts in terms of **power, performance, and area (PPA)** compared to the Manual-NP designs. Moreover, since the conventional placement tools do not allow cell overlapping, we also propose a detailed placement algorithm, namely *Auto-NP-Separate-P*, to generate more compact layouts with a small wirelength overhead. This automation facilitates overlapping instances and routing overlapped cells in order to reduce the chip area further, which could not be performed manually for larger circuits. We use the same set of benchmarks reported in [3] for the evaluation of the algorithms and show that Auto-NP-Separate reduces the total coupling capacitance by 13%, the critical path delay by 6%, the power consumption by 10%, the power-delay product by 15%, and the energy-delay product by 20% on average compared to the manual NP-Separate design methodology. Notice that expert layout designers might be able to obtain similar PPA after spending an excessively high amount of time and effort, so we expect that they could also use Auto-NP-Separate to generate highly-optimized NP-Separate layouts by Auto-NP-Separate and then manually optimize them further. The target technology node is the conventional transistor technology such as 22nm and earlier (28nm, 45nm, etc.), but that could be extended to advanced technology with appropriate modifications in the future.

Table 1. Notations and Terminologies Used in This Paper

k_μ	Electron to hole mobility ratio
$w_{\min}$	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
N-P pair	An N cell merged to a P cell

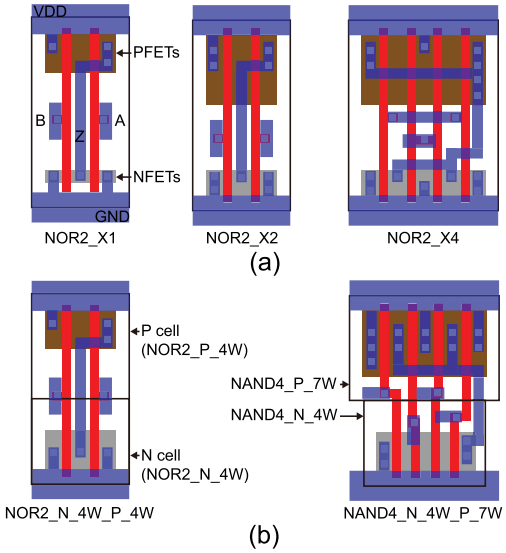

Fig. 2. (a) Three layouts for two-input NOR cells. (b) An example of NP cells: NOR2_N_4W_P_4W and NAND4_N_4W_P_7W.

The rest of the paper is arranged in the following sequence. In Section 2, we discuss and compare the conventional standard-cell-based design methodology with the manual NP-Separate design methodology and show the motivation for the design automation algorithms. In Section 3 and 4, we explain in detail the design automation and placement algorithms, respectively. Section 5 compares all the results of the Auto-NP-Separate and Auto-NP-Separate-P with the manual NP-Separate and standard-cell-based design methodologies. In Section 6, we discuss limitations of the NP-Separate design methodology and generalize the proposed methodology for advanced technology nodes. Finally, we summarize the proposed algorithms and conclude in Section 7.

2 THE NP-SEPARATE VLSI DESIGN METHODOLOGY

In this section, we briefly review the NP-Separate design methodology for physical VLSI layout design. Table 1 shows the notations and terminologies used in this paper.

2.1 Standard-Cell-Based Layout Design

The standard-cell-based design methodology uses standard cells and standard cell libraries containing physical, timing, and power information of the cells and the interconnect information of the technology. Figure 2(a) shows NOR2 standard cells having all the PFETs on the upper

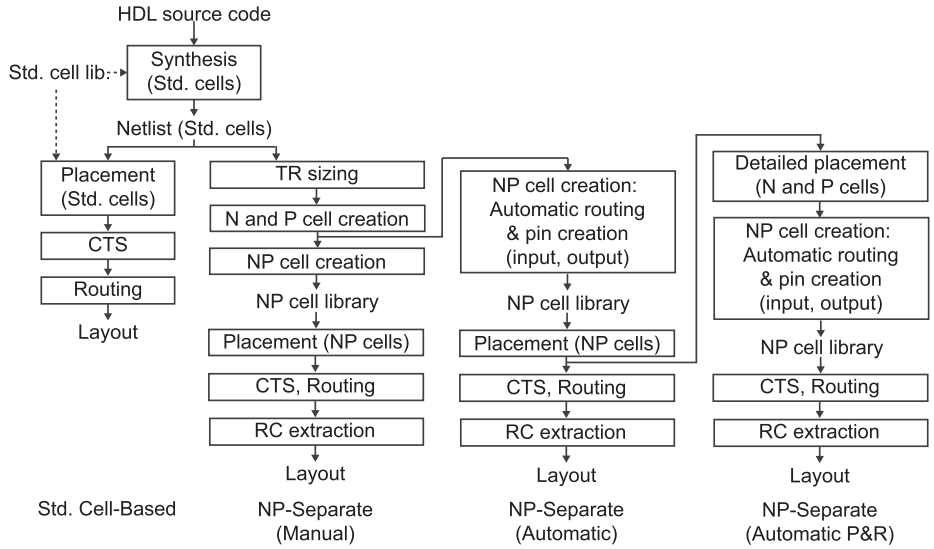

Fig. 3. Standard-cell-, manual NP-Separate, Auto-NP-Separate, and Auto-NP-Separate-P based VLSI design flows.

portion and all the NFETs on the lower portion of the cells. The sizes and shapes of the transistors are optimized for several objectives such as matching the rise and fall delays. A cell type (e.g., two-input NOR) has multiple cells having different input capacitance and output resistance values. For example, Figure 2(a) shows three candidates of the NOR2 type. NOR2_X2 and NOR2_X4 have transistors 2× and 4× as large as those of NOR2_X1, thereby having 2× and 4× lower output resistances, respectively. Physical layout design of the standard-cell-based design methodology consists of netlist synthesis and **placement and routing (P&R)** in which various steps such as placement, clock-tree synthesis, routing, and layout optimization are performed. Figure 3 shows a simplified standard-cell-based design methodology.

2.2 NP-Cell-Based Layout Design

Figure 3 also shows a design flow for NP-Separate. The design process starts with synthesizing a standard-cell-based netlist. At this stage, the netlist consists of standard cell instances optimized for given constraints and objectives. Then, NP-Separate re-optimizes the sizes of the NFETs and PFETs of the instances separately (denoted by "TR sizing" in the figure). For example, when an instance in the netlist is INV_X8 whose NFET and PFET widths are 8× and $8k_\mu$×, respectively, the NFET could be upsized to 16× while the PFET is downsized to $2k_\mu$×. The goal of the re-optimization is to reduce the power consumption and the area further. Then, N and P cells are created and merged to form NP cells, and all the standard cell instances in the netlist are replaced by NP cell instances.

An N cell has only NFETs, a P cell has only PFETs, and an NP cell is composed of an N cell and a P cell. Figure 2(b) shows an example for a NOR2 cell. The NOR2_N_4W_P_4W NP cell is formed by combining a NOR2_N_4W N cell and a NOR2_P_4W P cell. NOR2_X_YW (X is N or P and Y is an integer) means that the cell is an X-type cell and the width of the transistors is $Y \cdot w_{\min}$. The "N and P cell creation" step in Figure 3 creates all the N and P cells required in the re-optimized netlist. The "NP cell creation" step creates all the NP cells used in the netlist by merging the layouts of the N and P cells and routing their input and output ports. Once all the NP cells have been created, a new NP cell library is created for the NP cells. The NP cells are very similar to the standard cells

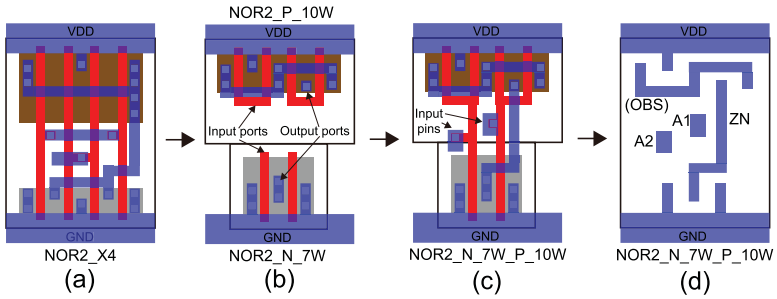

Fig. 4. Cell creation, routing (input and output ports), and abstraction of an NP cell. (a) Standard cell NOR2_X4. (b) NP cell creation of NOR2_N_7W_P_10W. (c) The input and output ports are routed and then input pins are created. (d) The pins and obstructions are abstracted.

from an abstract point of view, so layout designers can use the NP cells as if they are standard cells. The following explains the design steps used in NP-Separate in more detail.

2.2.1 Transistor Sizing. NP-Separate can use any algorithms for the transistor sizing. For example, the formula used in [3] minimizes the total transistor width and satisfies given timing constraints, which is solved by a nonlinearly-constrained optimization solver [2]. The layout constraint used in the paper is that all the NFETs (or PFETs) in an N (or P) cell have the same width. Although the size of each transistor can be optimized separately, this layout constraint provides various benefits such as the regularity of the cell layouts and the reduction of the number of cell layouts.

2.2.2 NP Cells. The transistor sizing step generates a new netlist from a given synthesized netlist. The instances in the new netlist are NP cell instances such as NOR2_N_4W_P_2W. Thus, the next step is to create all N and P cells used in the new netlist. Although this step is executed for each design, we can design the N and P cells once and reuse them for other designs if necessary. Designing (drawing a layout and performing DRC, LVS, and RC extraction) an N or P cell is performed manually in [3]. Once all the N and P cells have been designed, NP cells are created by merging the N and P cells in the NP cell creation step. The creation of an NP cell requires routing the input and output ports of the N and P cells merged for the NP cell and creating input pins. We use ***ports*** for the input poly and output M1 wires and ***pins*** for the actual wires accessible by routing tools. The routing was performed manually in [3], so it was a serious design bottleneck in NP-Separate. The routing of the input and output ports is followed by DRC and LVS to generate DRC- and LVS-clean NP cells. Then, all the pin and obstruction information is abstracted from the NP cell layouts to create a new physical library for the NP cells.

Figure 4 shows an example of the NP cell creation step. Figure 4(a) shows a layout of a NOR2_X4. If this is to be replaced by NOR2_N_7W_P_10W after transistor sizing, we manually design NOR2_N_7W and NOR2_P_10W in Figure 4(b), the input and output ports of the N and P cells are routed in Figure 4(c), and the input and output ports and obstructions are abstracted in Figure 4(d).

2.3 Motivation

The manual creation of NP cells is very time-consuming and error-prone. For the designs used in [3], for example, almost 140 different NP cells were created manually and we predict that larger designs will require much more various NP cells. Thus, the NP cell creation step should be automated for the adoption of the NP-Separate design methodology for VLSI design. Coupling

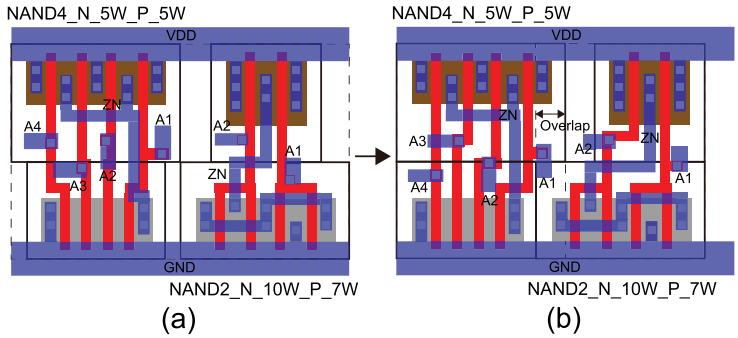

Fig. 5. Demonstration of cell overlapping in the NP-Separate design methodology. (a) Two NP cell instances placed without overlapping. (b) Cell overlapping.

capacitance minimization is also an important factor for the automation of the NP cell creation step. The internal capacitance of an NP cell is strongly dependent on the routing pattern of the input and output ports of the N and P cells constituting the NP cell. In addition, input and output pin creation is also an important step because non-optimized input and output pin locations would lead to routing failure during P&R. Thus, the internal capacitance and I/O pin locations should be optimized simultaneously. In this paper, we aim to automate the NP cell creation step optimizing given objectives. Obviously, the N and P cell creation step could also be automated. However, we design N and P cells manually because it is much easier than creating all NP cells as shown in Figure 4 and we can reuse the N and P cells.

We also propose a detailed placement algorithm to overlap NP cell instances to reduce the chip area further. For example, Figure 5(a) shows two NP cell instances NAND4_N_5W_P_5W and NAND2_N_10W_P_7W placed side by side. The dotted lines show the cell boundaries and placement tools do not allow the instances to overlap. In Figure 5(b), however, NAND2_N_10W_P_7W is shifted to the left to make the design more compact. Figure 5(b) also shows how the top-level routing is performed without violating design rules. These necessitate a detailed placement algorithm for the instance overlapping and an automatic routing algorithm for cell-level routing after detailed placement.

3 DESIGN AUTOMATION ALGORITHMS FOR NP-SEPARATE

In this section, we propose design automation algorithms for NP-separate, namely *Auto-NP-Separate*. We also propose a detailed placement algorithm along with the design automation algorithms, namely *Auto-NP-Separate-P*, for further area reduction.

3.1 Overview

Figure 3 shows the design flows of the manual NP-Separate, Auto-NP-Separate, and Auto-NP-Separate-P design methodologies. All the design processes begin with netlist synthesis. After the transistor sizing step (netlist re-optimization), the N and P cells used in the re-optimized netlist are created manually. Notice again that the N and P cells can be reused once they have been created. Then, NP cells are automatically created in both Auto-NP-Separate and Auto-NP-Separate-P and an NP cell library is created from the NP cells. A commercial tool is used for the placement of the NP cell instances in a given design. Auto-NP-Separate-P performs an additional detailed placement for further area optimization. The new placement result requires the NP cell creation step one more time, which is followed by the **clock tree synthesis (CTS)**, routing, and RC extraction steps.

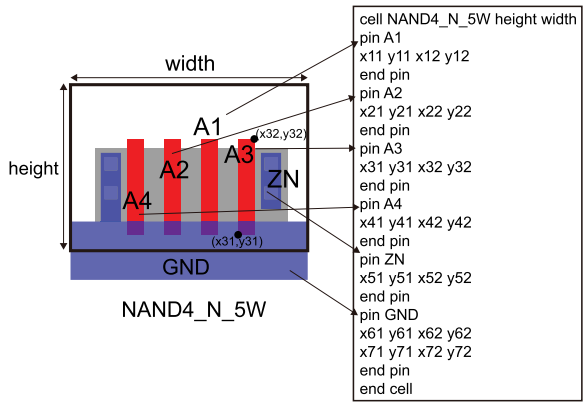

Fig. 6. A portion of SLEF.

3.2 Physical Libraries for N and P Cells

In order to automate the routing and generation of the LEF file for NP cells, we create a special type of LEF file, namely *SLEF*, having the physical cell information of all the N and P cells. Each cell represents either an N cell or a P cell and includes its power and ground ports, input and output ports, and obstacle geometries. The input and output ***ports*** of an N or P cell are the poly and M1 wire pieces that will be connected to the input and output ***pins*** of the NP cell formed from the cell. Figure 6 shows an example of an SLEF file.

3.3 Input and Output Ports Routing

An NP cell is formed by merging an N and a P cells. Each N (or P) cell could be simple or folded. A *simple* N or P cell has only one input port for an input pin, whereas a *folded* cell has multiple input ports for an input pin. In Figure 7(a), for example, the NAND4_N_5W cell has one input port for each input pin, so it is a simple N cell. On the contrary, the NAND3_N_10W cell in Figure 7(f) has two input ports (denoted by A2 and A2_1) for input pin A2, so it is a folded N cell. The multiple input ports for the same pin are routed during the NP cell creation. An NP cell is simple (or folded) if both of its N and P cells are simple (or folded). Otherwise, the cell is *mixed*. We assume that the locations of the N and P cells are given (the center lines of the cells are aligned in general).[1]

For a given NP cell (an N cell, a P cell, and their relative locations), we first route the input ports using the poly layer only. Then, we route the output ports using the M1 layer and finally create input pins. Notice that the output ports are on the M1 layer and routed using the same layer, so we do not need to create output pins. The order between the routing of the output port and the creation of the input pins is closely related to the routability of the input and output ports and the feasibility of the creation of input pins. In our algorithm, we find all minimum-cost routing results for input and output ports, then find the best routing solution as detailed below.

Notice that we should consider all the design rules in the routing. In this work, we use all the design rules related to the active, contact, poly, and M1 layers since those layers are related to the cell-level routing. The rules include the minimum width, minimum spacing, extension and overlap in between layers or the same layer, and minimum area. However, we do not use the parallel run length, step height, and end-of-line spacing rules. There are options to include these advanced rules if necessary since our algorithms enumerate all combinations, it will be able to find

[1]We observed that center-aligned NP cells have better routability than left- or right-aligned NP cells in general.

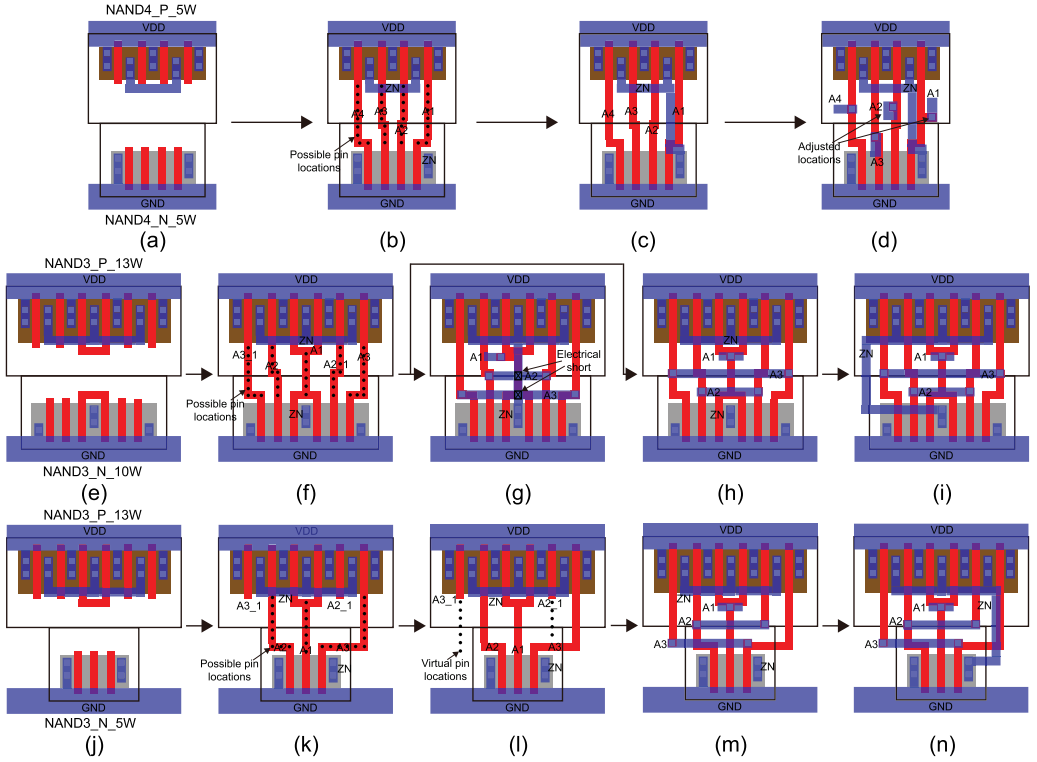

Fig. 7. Automatic cell-level routing of simple, folded, and mixed NP cells. (a) Placement result of simple N cell and P cell. (b) Input poly routing. (c) Output M1 routing before input pin creation. (d) Input pin creation. (e) Placement result of folded N cell and P cell. (f) Input poly routing. (g) Output M1 routing before input pin creation indicates circuit violation. (h) Input pin creation before output M1 routing. (i) Output M1 routing. (j) Placement result of simple N cell and folded P cell, introducing mixed N-P pair. (k) Input poly routing. (l) Virtual pin candidates of unrouted poly ports. (m) Input pin creation before output M1 routing. (n) Output M1 routing.

a solution if any. However, the standard-cell-based and Manual-NP-based designs in [3, 8] did not use the parallel run length, step height, or end-of-line spacing rules and since the outcome as well as the routability depend on the design rules, we do not use those rules either for the Auto-NP and the Auto-NP-P designs for a fair comparison.

3.3.1 Input Port Routing. The **route_input** function in Algorithm 1 shows the algorithm for the routing of the input ports of a given NP cell. The algorithm finds multiple paths using maze routing [9] for each input port pair of the N and P cells and selects the best combination of the paths that has the minimum coupling capacitance and no design rule violation. In Line 1, we initialize $routedPaths$ and $finalPaths$ (the former will have routing paths for each input and the latter will have final routing paths for all inputs). Then, we perform maze routing for each input i (Line 2 to 12). First, we insert all input ports of i in the N and P cells of the given NP cell into pN and pP, respectively (Line 3, 4). For example, $pN = \{A1, A2, A3\}$ and $pP = \{A1, A2, A2\_1, A3, A3\_1\}$ in Figure 7(k). Then, we obtain $pPairs$ from pN and pP (Line 5). Notice that some input ports are not paired even if they belong to the same input. For example, input $A2$ in Figure 7(k) has one port $A2$ in the N cell and two ports $A2$ and $A2\_1$ in the P cell. However, only the $A2$ ports are paired and $A2\_1$

ALGORITHM 1: Input port routing

```
Require: route_input (c: NP cell)
1:  routedPaths = {{}}; finalPaths = {};
2:  for each input i of c do
3:      pN = {All input ports of i in the N cell of c};
4:      pP = {All input ports of i in the P cell of c};
5:      pPairs = pairs of the input ports from pN and pP;
6:      for each p ∈ pPairs do
7:          Initialize a maze-routing grid G for p.
8:          maze_filling (p[N], p[P], G);
9:          Paths = trace_parent (p[P], G);
10:         Insert Paths into routedPaths[i].
11:     end for
12: end for
13: for each input i of c that has unrouted secondary ports do
14:     pP = {All pin locations on the primary port of i};
15:     for each unrouted port p do
16:         pm = Extend p vertically and maximally.
17:         pS = {All pin locations in pm};
18:         comPins = {(r, s) : r ∈ pP, s ∈ pS};
19:         for each cp ∈ comPins do
20:             Initialize a maze-routing grid G for cp.
21:             maze_filling (cp[0], cp[1], G);
22:             Paths = trace_parent (cp[1], G);
23:             Insert Paths into routedPaths[i].
24:         end for
25:     end for
26: end for
27: allPaths = All combinations of the paths in routedPaths.
28: for each path combination p ∈ allPaths do
29:     Perform DRC and estimate the cost.
30:     if DRC-clean and the cost < threshold then
31:         Insert p into finalPaths.
32:     end if
33: end for
```

is left unpaired and will be routed later. On the contrary, both the N and P cells have $A2$ and $A2\_1$ ports in Figure 7(f). Thus, the $A2$ ports are paired, so are the $A2\_1$ ports. The next step is to perform maze routing for each port pair p (Line 6 to 11). The **maze_filling** function is the filling function from one of the ports to the other and the **trace_parent** function is the back-propagation function. We find all low-cost paths from the maze routing and insert them into $routedPaths[i]$ for input i (Line 10). The cost is a weighted sum of the total wirelength and the number of bends. Notice that a path for input i and a path for input j might have design rule violations. Nonetheless, we find all minimum-cost paths for each input because we want to find all the DRC-clean combinations of the paths of all the inputs.

If the given NP cell is mixed or folded, it might have unrouted secondary ports as shown in Figure 7(k). In this case, we route each of the unrouted secondary ports to its primary port (Line 13 to 26). We first insert all pin locations on the primary port of input i into pP (Line 14). Some examples of the pin locations are shown in Figure 7(k). Then, we extend each unrouted port p vertically and maximally and assign the geometry to pm as shown in Figure 7(l). The extension

could be more sophisticated to include various shapes such as L and Z shapes, but we found that the vertical extension was enough for successful routing. Then, we insert all pin locations in *pm* into *pS* (Line 17) and find all combinations of the pin locations of *pP* and *pS* (Line 18). For each pin location combination, we perform maze routing and find all low-cost paths (Line 19 to 24). Notice that this routing strategy allows detours, which sacrifices the routing cost for routability in case we need some detours later to route output ports. This process completes the routing of all unrouted secondary ports, so now all the inputs have completely routed paths in *routedPaths*.

The next step is to find all DRC-clean, low-cost combinations of the paths. For this, we enumerate all combinations of the paths and insert them into *allPaths* (Line 27). For each combination, we perform DRC and estimate the cost (Line 29 to 32). At this time, the cost function is a weighted sum of the total coupling capacitance and the number of bends. We estimate the coupling capacitance between two adjacent wires by the reciprocal of their distance. If the combination is DRC-clean and its cost is less than a pre-determined threshold value, we insert it into *finalPaths* (Line 31). Finally, *finalPaths* has all the combinations of the routing paths of the inputs.

Notice that maze routing has also been used for fully automatic cell-level routing for standard cell libraries [4]. The Boolean satisfiability (SAT) solver along with the maze routing has been used in [18] to generate all possible legal routing solutions for standard cell libraries. However, both the articles did not select the best routing solutions considering the intra-cell coupling capacitance. Moreover, the **satisfiability modulo theory (SMT)** and the **integer linear programming (ILP)** based simultaneous placement and intra-cell routing of the FETs using multiple metal layers with conditional design rules for advanced technology nodes like sub-10nm were performed in [10, 15, 16]. However, fully automatic cell-level placement and routing lacks both in performance and optimality compared to the placement and routing obtained from the skillful manual effort [17]. In the Auto-NP-Separate design methodology, therefore, we try to balance the scale by placing NFETs (or PFETs) in N (or P) cells, routing N (or P) cells manually and elaborately, and merging the N and P cells to complete the cell-level routing automatically for better-optimized cell layouts.

3.3.2 Output Port Routing and Input Pin Creation. Algorithm 2 shows the proposed algorithm to route the output port of a given NP cell and create its input pins. *inPaths* in Line 1 is the set *finalPaths* of all routed paths of the inputs of *c* found in Algorithm 1. *Pins* in Line 2 is the set of potential locations of the input pins of *c*. Then, we find possible pin locations on the path *p* in *inPaths* of each input *i* (Line 3 to 7). The black dots in Figure 7(f) show an example of the possible pin locations. When we find possible pin locations on path *p*, we use a pre-determined step distance (e.g., 30nm) so that we can find a proper number of possible locations to check. If we fail to create input pins for *c*, we reduce the step distance and try the input pin creation process again. Then, we create a set *comPins* for all the DRC-clean combinations of the pin locations of all the pins of *c* (Line 8). Since an NP cell is small, the total number of elements of *comPins* is not many, so we try all of them. Before we create input pins, we route the output port because the input pin locations have a greater degree of flexibility than the feasible paths of the output port. For this, we first prepare two sets *yN* and *yP* of connection points on the output port of the N and P cells of *c*, respectively (Line 9, 10). The connection points are similar to the possible pin locations for the inputs, but they are created on the output ports of the N and P cells. Then, for each combination of the elements of *yN* and *yP*, we perform maze routing and find all low-cost paths (Line 14 to 16). These routing paths might overlap with the input pin locations in *comPins*, so we check overlaps among them. If there is any violation, we try to slightly adjust the locations of the pins violating the design rules to remove the violations (Line 19). Function **adjust_locations** checks the design rules and returns *cp* if there is no violation or adjusted pin locations if *cp* and *p* have violations. The adjustment is creating horizontal exit paths from the original pin location

ALGORITHM 2: Output port routing and input pin creation

```
Require: route_out (c: NP cell)
1:  inPaths = {All routed paths of the inputs of c};
2:  Pins = {{}};
3:  for each input i of c do
4:      for each path p of i do
5:          Pins[i] = {Possible pin locations on p};
6:      end for
7:  end for
8:  comPins = {All DRC-clean combinations of the pin locations in Pins};
9:  yN = {Connection points on the output port of the N cell of c};
10: yP = {Connection points on the output port of the P cell of c};
11: yNP = {(n, p) : n ∈ yN, p ∈ yP};
12: best_cost = ∞;
13: for each y ∈ yNP do
14:     Initialize a maze-routing grid G for y.
15:     maze_filling (y[0], y[1], G);
16:     Paths = trace_parent (y[1], G);
17:     for each p ∈ Paths do
18:         for each cp ∈ comPins do
19:             tp = adjust_locations (cp, p);
20:             if tp == {} then
21:                 continue;
22:             end if
23:             cost = estimate_cost (tp, p);
24:             if cost < best_cost then
25:                 best_cost = cost;
26:             end if
27:         end for
28:     end for
29: end for
30: route_extension(c);
31: Return the best layout.
```

as shown in Figure 7(d) where pin A1 has an extended poly wire to avoid the minimum spacing rule in the M1 layer. However, the adjustment might fail if there is not enough space. In this case, **adjust_locations** returns an empty object and we discard the combination of p and cp (Line 21). If tp is not empty, we compute the cost of the layout, which is a weighted sum of the total capacitance and the number of bends of tp (Line 23). The **route_extension(**c**)** function routes any unrouted secondary ports of the inputs of c, which is actually extending them vertically to the selected virtual pin locations. Finally, the algorithm returns the best layout minimizing the cost function.

The proposed algorithm does not guarantee that it will always be able to find a DRC-clean layout. For example, if an NP cell requires complex detours for the routing of its input and output ports, the algorithm will fail to find a DRC-clean layout. In that case, we can manually layout the NP cell. In our simulation, however, the algorithm successfully found DRC-clean layouts for all the NP cells although there were only two to four horizontal M1 routing tracks in the cells.

3.3.3 Complexity Analysis. In Algorithm 1, the complexity of the maze routing is $O(w \cdot h)$ where w and h are the width and height of a given cell, respectively. Notice that the number of grid points is proportional to $w \cdot h$. The complexity of the first FOR statement (Line 2 to 12) is $O(p^2 \cdot w \cdot h)$

where p is the number of input signal pins of the cell. Notice that p is generally small (maximum four) and h is a constant because all the cells have the same height. The complexity of the second FOR statement (Line 13 to 26) is $O(h^2 \cdot w \cdot h) = O(w \cdot h^3)$. The complexity of the third FOR statement (Line 28 to 33) is dependent on the number of paths found and the number of design rules. In fact, both numbers are small. In summary, the complexity of Algorithm 1 is $O(w \cdot h^3)$.

In Algorithm 2, we can consider the number of paths found for each input as a constant c as stated above. In addition, the paths have no or small detour, so the number of possible pin locations on each path is $O(c \cdot w \cdot h)$. The complexity of the second FOR statement (Line 13 to 29) is $O(h^2 \cdot (w \cdot h + c \cdot w \cdot h)) = O(w \cdot h^3)$. Thus, the complexity of the overall routing algorithm is $O(w \cdot h^3)$.

4 DETAILED PLACEMENT

In this section, we propose a detailed placement algorithm for area compaction for Auto-NP-Separate-P.

4.1 Overview

Auto-NP-Separate-P follows the steps similar to the Auto-NP-Separate methodology until the placement step as shown in Figure 3. After that, we perform a detailed placement of the N and P cells using the proposed placement algorithm to reduce the chip area further as shown in Figure 5. The placement algorithm strictly satisfies two constraints. First, the wirelength overhead caused by the detailed placement is less than a pre-determined threshold value. Second, the new locations of the N and P cells of each NP cell will not change the width of the NP cell, otherwise the top-level routing will be much more complicated or even the design might become unroutable. Once the locations of the N and P cells are adjusted by the detailed placement, we route them to create NP cells.

4.2 Placement Algorithm

Simulated annealing has been used for detailed placement in several papers [5, 19, 25]. Thus, we use simulated annealing for the detailed placement of NP cells in this paper as mentioned in [3]. The input consists of a placement result and simulated annealing parameters such as initial and final temperature values and a cooling rate. The cost function is the layout area. For the solution perturbation, we use instance swapping as shown in Algorithm 3. We first compute the d value, which is the difference between the widths of the N and P cell instances, for each NP cell instance (Line 1). If d is positive, the N cell has a larger width than the P cell. The rationale for the use of d is that we can reduce the total area by placing two instances side by side and overlapping them as shown in Figure 5 if they have opposite signs of d. Then, we randomly pick two rows in the layout L (Line 2). The rows should be the same row or adjacent rows. The first and the second rows are called the p-row and the n-row, respectively. Then, we insert all NP cell instances in the p-row and the n-row whose d values are positive and negative into set P and N, respectively (Line 3, 4). If P or N is empty, the perturbation returns the current layout (Line 6). Otherwise, we randomly pick an instance I_P from P and an instance I_N from N (Line 8, 9). Then, we move I_P (or I_N) to the left or right of I_N (or I_P) or swap I_P and I_N (Line 10). Then, we compute the wirelength and revert the perturbation if the wirelength overhead is greater than the threshold value $maxW$ (Line 12). Finally, we return L (Line 14).

Once the simulated annealing finishes, we adjust the locations of N and P cells as follows. For the non-overlapping instances, we center-align the N and P cells inside the instance boundary (the NAND4 and NAND2 instances in Figure 5(a)). For the overlapped instances, for each overlapped-instance pair, we left-align the N and P cells of the left instance (the NAND4 instance in Figure 5(b)) and right-align the N and P cells of the right instance (the NAND2 instance in Figure 5(b)) to ensure

Table 2. The Number of Instances Used in the Benchmarks

Instance type	RCA08	CLA08	BKA08	KSA08	BCD08	WT04	BKA32	CLA32	KSA32	MUL_B64	DES_PERF
DFF	-	-	-	-	-	-	-	-	-	15623	9920
INV	10	15	22	30	25	16	42	31	58	12796	15705
NAND2	21	27	33	47	41	33	120	92	127	21868	37515
NAND3	-	7	3	5	9	-	7	16	19	532	6000
NAND4	-	-	-	1	-	-	1	2	4	112	2800
NOR2	15	8	10	17	25	32	21	35	36	8176	24800
NOR3	1	-	-	1	4	1	3	7	4	112	6640
NOR4	1	-	-	-	-	-	3	1	-	-	1520
XOR2	1	6	3	1	3	3	31	34	30	2072	2970
XNOR2	15	16	8	10	17	15	32	53	27	12012	4205

ALGORITHM 3: Solution perturbation algorithm for simulated annealing

Require: **solution_perturbation** (L: layout, $maxW$)
(Compute d for each NP cell instance.)
Randomly pick two rows from L, same or adjacent.
P = {all the $+d$-value instances in the p-row};
N = {all the $-d$-value instances in the n-row};
if P.size == 0 or N.size == 0 **then**
    Return L;
end if
I_P = Randomly pick an instance from P.
I_N = Randomly pick an instance from N.
Randomly pick an operation, either move I_P to adjacent location of I_N or I_N to adjacent location of I_P or swap I_P with I_N.
if $\frac{\mathbf{hpwl}(S_{new})}{\mathbf{hpwl}(S)} > maxW$ **then**
    Revert the perturbation.
end if
Return L;

the maximum spacing between the overlapped-instances to avoid routing congestion in their input and output ports.

5 SIMULATION RESULTS

In this section, we use the same set of benchmarks used in [3], which were picked based on different characteristics including design complexity, logic depth, and routing pattern. We designed five 8-bit adders, three 32-bit adders, a 4-bit multiplier, a 64-bit pipelined multiplier [20], and a **data encryption standard (DES)** core using Auto-NP-Separate and Auto-NP-Separate-P and compare the results with the **standard-cell-based (Std)** and **manual NP-Separate (Manual-NP)** design methodologies from [3] to show the effectiveness of the proposed algorithms. Table 2 shows the number of instances used in the benchmarks. The details of the benchmark circuits can be found in [3]

We used the same netlists used in [3] for a fair comparison. The netlists were synthesized using a 22nm standard cell library and Cadence Genus. The 22nm standard cell library used an aspect ratio of 1.8, placement site width of 0.1um, and used 0.9um single-height cells for all the instances (0.4um and 0.5um for the NFET and PFET regions, respectively). Therefore, we also used the same

Table 3. Average Routing Time Per Each Instance Type

Instance type	Routing time
AND(OR): 2/3/4	0.34s/2.10s/15.40s
AOI(OAI): 21/22/211/221/222/33	0.74s/7.23s/7.78s/10.73s/1m10s/1m13s
BUF/INV	0.04s/0.01s
HA	1m6s
MUX2	19.72s
NAND(NOR): 2/3/4	0.12s/1.73s/7.30s
XOR(XNOR): 2	0.4s

dimensions (0.9um, 0.4um, and 0.5um for the heights of the NP cells, N cells, and P cells, respectively) for the NP cells. Table III of [8] lists the existing 22nm standard cells with the corresponding cell sizes. The minimum width and pitch of poly are 26nm and 52nm, respectively, and those of M1 wires defined in the LEF file are 36nm and 90nm, respectively. The details of the 22nm standard cell library used in this paper can be found in [8]. We used Cadence Innovus for the placement and signal routing, Mentor Calibre for parasitic RC extraction, and Synopsys HSpice for post-layout simulation with parasitic RC.

For the layout generation, we used the maximum initial core utilization, which was almost 100% for most of the designs. The reason is as follows. One of the main benefits of NP-Separate is to reduce the layout area (more precisely speaking, the area occupied by transistors). However, the layout area of a design is determined at the beginning of the design process. Therefore, we need to (1) use the maximum initial core utilization for each design or (2) use a certain initial core utilization (e.g., 60%), but compare the actual area occupied by transistors. Since the actual transistor areas of the Auto-NP designs are always less than or equal to those of the Manual-NP designs, we decided to use the former approach (using the maximum initial core utilization) in this paper.

5.1 Design Time

Table 3 shows the average cell-level routing time for each instance type routed by the routing algorithm we proposed. We used an Intel Xeon CPU E5-2650 v3 (2.30GHz) server for all the simulations. We were able to route all the combinational logic instances except a sequential logic (DFF) and a full adder logic in Table III of [8]. The factors governing the routing time include the number of input and output ports to be routed per cell (as we consider all possible combinations of the routing paths), the available routing space in between the two diffusion regions along with the cell width, and the N cell to P cell alignment of each instance. Note that we can decompose some instances into smaller instances to reduce the routing time. However, it might compromise the optimality of the routing. For example, decomposing a half adder into an XOR2 and an AND2 cell reduces the routing time from 66s to 0.74s (0.4s and 0.34s for the XOR2 and AND2 cells, respectively) on average, but it might increase the power consumption slightly since the routing of the half adder circuit optimizes the entire instance as a whole. Thus, instances with the higher number of input and output ports can always be routed within a reasonable amount of time by the instance decomposition at a cost of the reduced cell-level routing optimality.

Next, we compare the design times of Auto-NP-Separate and Auto-NP-Separate-P. Table 4 shows the runtime for the NP cell creation (routing of the input and output ports) and detailed placement. The NP cell instances were routed sequentially. Although some of the instances were identical (i.e., they were of the same type and the relative locations of their N and P cells were the same), we routed all the instances individually assuming the worst case scenario. As the table shows, the average routing time per instance varies from 0.1 second to 2.6 seconds. The DES_PERF design has

Table 4. Comparison of the Design Time for the NP Cell Routing and Detailed Placement

Benchmark	Insts	Design type	Routing	Routing per instance	Detailed placement	P&R
RCA08	64	Std	-	-	-	4s
		Auto-NP	26s	0.4s	-	
		Auto-NP-P	-	-	-	
CLA08	79	Std	-	-	-	
		Auto-NP	30s	0.4s	-	
		Auto-NP-P	34s	0.4s	45s	
BKA08	79	Std	-	-	-	
		Auto-NP	14s	0.2s	-	
		Auto-NP-P	19s	0.2s	47s	
KSA08	112	Std	-	-	-	
		Auto-NP	40s	0.4s	-	
		Auto-NP-P	1m4s	0.6s	47s	
BCD08	124	Std	-	-	-	
		Auto-NP	56s	0.5s	-	
		Auto-NP-P	1m11s	0.6s	55s	
WT04	100	Std	-	-	-	
		Auto-NP	14s	0.1s	-	
		Auto-NP-P	-	-	-	
BKA32	260	Std	-	-	-	
		Auto-NP	1m55s	0.4s	-	
		Auto-NP-P	2m22s	0.5s	1m5s	
CLA32	271	Std	-	-	-	
		Auto-NP	2m29s	0.5s	-	
		Auto-NP-P	3m10s	0.7s	1m17s	
KSA32	305	Std	-	-	-	
		Auto-NP	2m46s	0.5s	-	
		Auto-NP-P	3m35s	0.7s	1m17s	
MUL_B64	73303	Std	-	-	-	20s
		Auto-NP	2h24m	0.1s	-	
		Auto-NP-P	-	-	-	
DES_PERF	112075	Std	-	-	-	
		Auto-NP	66h40m	2.3s	-	
		Auto-NP-P	72h30m	2.6s	11h20m	

"P&R" denotes placement and routing of the standard-cell-based designs from [3].

many high fan-in instances such as NAND4 and NOR4 as shown in Table 2, so it has a much longer routing time than the other designs. Notice that most of the instances can be routed in parallel, so the actual routing time can be reduced significantly by parallel routing. In addition, the instances can be reused for other designs, which will reduce the design time further. In fact, we were able to route more than 800 different N-P cell pairs of the instance types reported in Table 2 within 12 minutes. Notice that routing a group of overlapped instances in the Auto-NP-Separate-P layouts takes more time than routing a single instance. However, the groups of the overlapped instances are independent of each other, so we can route the groups in parallel to reduce the routing time.

The runtime for detailed placement is approximately a minute for small designs and more than 11 hours for the largest design. Notice that the detailed placement is an optional step for further area reduction. Thus, if area reduction is crucial for a certain design, this runtime overhead is sufficiently acceptable for trying the detailed placement. For RCA08, WT04, and MUL_B64, we also designed them using Auto-NP-P. However, their Auto-NP designs were sufficiently compact, so their Auto-NP-P designs did not obtain enough area reduction. Thus, we did not show the design time for the routing and detailed placement of the designs in Table 4.

Table 5. Comparison of the Area, Wirelength (WL), Parasitic Resistance (R), Parasitic Ground (C) and Coupling Capacitances (Cc), Critical Path Delay (CPD), and Power Consumption of the Layouts Built by the Conventional Standard-cell-based (denoted by Std), Manual NP-Separate (denoted by Manual-NP), Auto-NP-Separate (denoted by Auto-NP), and Auto-NP-Separate-P (denoted by Auto-NP-P) Design Methodologies

Benchmark	Design type	# Nets	Area (μm^2)	WL (μm)	R (kΩ)	C (fF)	Cc (fF)	CPD (ps)	Power (uW)
RCA08	Std	81	21.60 (1.09)	224 (1.01)	273 (1.03)	21 (1.05)	118 (1.27)	470 (1.24)	56 (1.27)
	Manual-NP		19.80 (1.00)	221 (1.00)	265 (1.00)	20 (1.00)	93 (1.00)	380 (1.00)	44 (1.00)
	Auto-NP		19.80 (1.00)	224 (1.01)	271 (1.02)	20 (1.00)	74 (0.80)	350 (0.92)	39 (0.89)
	Auto-NP-P		19.80 (1.00)	224 (1.01)	271 (1.02)	20 (1.00)	74 (0.80)	350 (0.92)	39 (0.89)
CLA08	Std	96	28.35 (1.09)	281 (1.09)	351 (1.02)	24 (1.04)	135 (1.09)	310 (1.15)	68 (1.03)
	Manual-NP		26.10 (1.00)	257 (1.00)	344 (1.00)	23 (1.00)	124 (1.00)	270 (1.00)	66 (1.00)
	Auto-NP		26.10 (1.00)	257 (1.00)	351 (1.02)	23 (1.00)	111 (0.90)	275 (1.02)	60 (0.91)
	Auto-NP-P		25.65 (0.98)	262 (1.02)	346 (1.01)	23 (1.00)	113 (0.91)	275 (1.02)	60 (0.91)
BKA08	Std	95	25.20 (1.12)	247 (1.07)	304 (1.01)	23 (1.10)	128 (1.17)	270 (1.02)	66 (1.14)
	Manual-NP		22.50 (1.00)	230 (1.00)	300 (1.00)	21 (1.00)	109 (1.00)	265 (1.00)	58 (1.00)
	Auto-NP		22.50 (1.00)	230 (1.00)	287 (0.96)	21 (1.00)	88 (0.81)	255 (0.96)	52 (0.90)
	Auto-NP-P		22.05 (0.98)	235 (1.02)	290 (0.97)	21 (1.00)	100 (0.92)	265 (1.00)	54 (0.93)
KSA08	Std	129	35.64 (1.15)	328 (1.04)	434 (0.99)	28 (1.08)	191 (1.22)	360 (1.24)	91 (1.14)
	Manual-NP		31.05 (1.00)	314 (1.00)	440 (1.00)	26 (1.00)	157 (1.00)	290 (1.00)	80 (1.00)
	Auto-NP		31.05 (1.00)	316 (1.01)	430 (0.98)	26 (1.00)	139 (0.89)	295 (1.02)	73 (0.91)
	Auto-NP-P		30.15 (0.97)	324 (1.03)	427 (0.97)	26 (1.00)	141 (0.90)	290 (1.00)	74 (0.93)
BCD08	Std	141	42.84 (1.13)	355 (1.05)	520 (1.03)	31 (1.07)	224 (1.11)	455 (1.06)	127 (1.02)
	Manual-NP		37.80 (1.00)	338 (1.00)	504 (1.00)	29 (1.00)	202 (1.00)	430 (1.00)	125 (1.00)
	Auto-NP		37.80 (1.00)	340 (1.01)	520 (1.03)	29 (1.00)	165 (0.82)	375 (0.87)	101 (0.81)
	Auto-NP-P		36.54 (0.97)	349 (1.03)	511 (1.01)	29 (1.00)	175 (0.87)	385 (0.90)	104 (0.83)
WT04	Std	108	32.85 (1.13)	279 (1.08)	434 (1.02)	26 (1.08)	156 (1.06)	325 (1.03)	78 (1.05)
	Manual-NP		29.00 (1.00)	258 (1.00)	426 (1.00)	24 (1.00)	147 (1.00)	315 (1.00)	74 (1.00)
	Auto-NP		29.00 (1.00)	257 (1.00)	412 (0.97)	24 (1.00)	123 (0.84)	295 (0.94)	63 (0.85)
	Auto-NP-P		29.00 (1.00)	257 (1.00)	412 (0.97)	24 (1.00)	123 (0.84)	295 (0.94)	63 (0.85)
BKA32	Std	324	88.20 (1.09)	1082 (1.07)	1138 (1.01)	55 (1.06)	475 (0.94)	635 (1.18)	141 (1.07)
	Manual-NP		81.00 (1.00)	1012 (1.00)	1126 (1.00)	52 (1.00)	505 (1.00)	540 (1.00)	132 (1.00)
	Auto-NP		81.00 (1.00)	1031 (1.02)	1124 (1.00)	52 (1.00)	433 (0.86)	545 (1.01)	119 (0.90)
	Auto-NP-P		80.10 (0.99)	1040 (1.03)	1143 (1.02)	52 (1.00)	452 (0.90)	540 (1.00)	124 (0.94)
CLA32	Std	336	102.96 (1.13)	1447 (1.36)	1334 (1.07)	62 (1.09)	630 (1.29)	710 (1.11)	140 (1.21)
	Manual-NP		90.90 (1.00)	1061 (1.00)	1245 (1.00)	57 (1.00)	487 (1.00)	640 (1.00)	116 (1.00)
	Auto-NP		90.90 (1.00)	1061 (1.00)	1250 (1.00)	57 (1.00)	449 (0.92)	585 (0.91)	110 (0.95)
	Auto-NP-P		90.00 (0.99)	1092 (1.03)	1276 (1.02)	57 (1.00)	453 (0.93)	630 (0.98)	107 (0.92)
KSA32	Std	370	99.99 (1.09)	1309 (1.14)	1304 (1.01)	61 (1.05)	585 (1.07)	695 (1.01)	153 (1.08)
	Manual-NP		91.80 (1.00)	1151 (1.00)	1292 (1.00)	58 (1.00)	549 (1.00)	690 (1.00)	142 (1.00)
	Auto-NP		91.80 (1.00)	1170 (1.02)	1298 (1.00)	58 (1.00)	488 (0.89)	580 (0.84)	134 (0.94)
	Auto-NP-P		90.90 (0.99)	1184 (1.03)	1289 (1.00)	58 (1.00)	521 (0.95)	655 (0.95)	131 (0.92)
MUL_B64	Std	90076	44428 (1.06)	202384 (1.02)	406084 (0.99)	17444 (1.03)	185808 (1.09)	973 (1.01)	75404 (1.07)
	Manual-NP		41807 (1.00)	197484 (1.00)	411768 (1.00)	16884 (1.00)	170380 (1.00)	966 (1.00)	70392 (1.00)
	Auto-NP		41807 (1.00)	200452 (1.02)	412300 (1.00)	16912 (1.00)	161784 (0.95)	812 (0.84)	61068 (0.87)
	Auto-NP-P		41807 (1.00)	200452 (1.02)	412300 (1.00)	16912 (1.00)	161784 (0.95)	812 (0.84)	61068 (0.87)
DES_PERF	Std	122605	47580 (1.05)	990325 (1.36)	584245 (1.00)	18725 (1.06)	364015 (1.31)	350 (1.04)	57075 (1.17)
	Manual-NP		45455 (1.00)	728510 (1.00)	584045 (1.00)	17588 (1.00)	278515 (1.00)	335 (1.00)	48905 (1.00)
	Auto-NP		45455 (1.00)	739115 (1.01)	586540 (1.00)	17675 (1.00)	250995 (0.90)	340 (1.01)	45440 (0.93)
	Auto-NP-P		45360 (1.00)	757215 (1.04)	598271 (1.02)	17604 (1.00)	264047 (0.95)	325 (0.97)	47120 (0.96)
Geo. mean (Std/Manual-NP)			1.10	1.11	1.02	1.06	1.14	1.10	1.11
Geo. mean (Auto-NP/Manual-NP)			1.00	1.01	1.00	1.00	0.87	0.94	0.90
Geo. mean (Auto-NP-P/Manual-NP)			0.99	1.02	1.00	1.00	0.90	0.95	0.90

The RC values include both the internal RC values of all instances and the RC values of the global interconnects.

5.2 Layout Area and Wirelength

Table 5 shows the layout area of all the benchmarks designed by the four design methodologies. Notice that the layouts of the standard-cell- and manual-NP-based designs are from [3]. Since Auto-NP-Separate automates the routing of the input and output ports, but does not alter the locations of the NP cell instances, the Manual-NP and Auto-NP layouts have the same area and very similar wirelength. The maximum wirelength overhead of Auto-NP compared to Manual-NP is only 2%.

The Auto-NP-P designs show area improvement up to 3% for KSA08 and BCD08 benchmarks and 1% to 2% for most of the other benchmarks with 2% wirelength overhead on average

compared to the Manual-NP designs. Table 5 also shows that for RCA08, WT04, and MUL_B64, the detailed placement has no area improvement because the Auto-NP layouts are already compact enough. Furthermore, for some benchmarks, we can obtain more area improvement than the reported cases with larger wirelength overhead. However, it will increase routing congestion and the parasitic resistance and capacitance of the layouts, thereby degrading the overall performance of the Auto-NP-P designs. Note that we implemented simulated annealing for the detailed placement in this paper, but any other detailed placement algorithms such as linear programming, instance-swapping, or linear placement-based algorithms [14, 24] can be applied to the Auto-NP-P standalone or in addition to the simulated annealing.

5.3 Parasitic RC, Critical Path Delay, and Power Consumption

The Manual-NP, Auto-NP, and Auto-NP-P designs show similar parasitic resistance and ground capacitance values because they are determined only by the routed length and all the designs have similar routing paths. However, the Auto-NP designs reduce the coupling capacitance of the layouts by 5% to 20% (13% on average) compared to the Manual-NP designs. The reason is that the routing algorithm performs an exhaustive search for the routing paths of the input and output ports to minimize the coupling capacitance. Table 5 also shows that the Auto-NP-P designs reduce the coupling capacitance of the layouts by 5% to 20% (10% on average) compared to the Manual-NP designs. Since they have more compact layouts than the Auto-NP designs, they have slightly larger capacitance, but still 10% lower coupling capacitance on average than the Manual-NP designs.

The Auto-NP designs have 6% less critical path delay on average than the Manual-NP designs. Especially, the Auto-NP designs of KSA32 and MUL_B64 show 16% critical path delay reduction. However, the Auto-NP designs of CLA08, KSA08, BKA32, and DES_PERF show a 1% to 2% increase in the critical path delay compared to the Manual-NP designs. This is because the resistance and coupling capacitance in the critical paths of the Auto-NP designs are larger than those in the Manual-NP designs although the former have less total resistance and coupling capacitance than the latter. The Auto-NP-P designs have 5% less critical path delay on average compared to the Manual-NP designs.

Table 5 shows 10% power improvement on average for both the Auto-NP and Auto-NP-P designs compared to the Manual-NP designs. The BCD08 benchmark shows a maximum 19% (or 17%) power improvement due to the significantly reduced coupling capacitance in the Auto-NP (or Auto-NP-P) design. For all the benchmarks except CLA32 and KSA32, the Auto-NP designs show slightly lower or similar power consumption than that of the Auto-NP-P designs. Although for the CLA32 and KSA32 benchmarks the Auto-NP-P designs have larger total coupling capacitance than the Auto-NP designs, the latter consume slightly higher power. Moreover, the lower critical path delay and power consumption of the Auto-NP and Auto-NP-P designs lead to improved power-delay product (15% and 13%, respectively) and energy-delay product (20% and 14%, respectively) compared to the Manual-NP designs.

6 DISCUSSION

In this section, we discuss several topics related to the NP-Separate design methodology.

6.1 Library Design and Reuse for NP-Separate

In all the NP-Separate methodologies shown in Figure 3, we perform RC extraction at the end of the design flow. In fact, once NP cells are created in the NP cell creation step, we can build not only a physical library but also a timing and power library for the NP cells. Then, we can use the timing and power library for the rest of the steps for Manual-NP and Auto-NP designs, so we will not need the full RC extraction step. Notice that library characterization is a highly-parallel

task, so runtime for the library characterization will have a small impact on the total design time. Notice also that the library for the characterized NP cells could be reused for future designs. For Auto-NP-P designs, however, we cannot characterize an NP cell instance alone if it is overlapped with another NP cell instance. In this case, we can group the overlapped instances, create a super cell instance containing the instances, and characterize the super cell instance. This approach will enable the traditional static timing analysis for all Manual-NP, Auto-NP, and Auto-NP-P designs.

6.2 Manual-NP vs. Auto-NP

Although the routing algorithm developed for Auto-NP-Separate in this paper was able to route most of the NP cells, it does not guarantee 100% routability, which is why it could not route complex NP cells such as D flip-flops and full adders. In this case, skillful layout engineers would be able to route the cells manually. In addition, they would also be able to design more optimal NP cell layouts than Auto-NP-Separate. However, the design time for Manual-NP designs is prohibitively longer than that for Auto-NP. Thus, Auto-NP routes simple NP cells while layout engineers route complex NP cells to generate more optimal NP-Separate layouts if highly-optimized layouts are to be generated.

Additionally, the average routing time per each instance type increases with the higher fan-in/out instances. In some cases, therefore, it may not be possible to enumerate all the combinations of possible routing paths to route such instances within a reasonable amount of time. In this case, we can group a portion of the NFETs and PFETs (decompose them into multiple sub-groups) and route each group separately by the proposed routing algorithm. The time-consuming cell-level routing of the high fan-in/out cells could be regulated this way at a cost of reduced optimality.

Notice also that the results shown in Table 5 do not mean that the routing algorithm of Auto-NP can almost always generate better NP cell layouts than skillful layout engineers. Rather, it means that the NP cell layouts obtained by the Auto-NP design methodology and algorithms could be as good as those obtained by skillful layout engineers. In conclusion, the primary objective of this work is to propose and implement algorithms to automate the design process and reduce the runtime of the NP-Separate design methodology for the generation of highly-optimized designs with minimal human effort.

6.3 Auto-NP-Separate to Advanced Technology Nodes

Our methodology and algorithms are proposed for 22nm technology node and earlier. However, advanced technology nodes such as sub-10nm process nodes require conditional design rules and sophisticated manufacturing-aware design challenges which can be included in our algorithms since we are enumerating all possible cases. As mentioned earlier, the proposed algorithms successfully generated DRC-clean layouts for most of the NP cells even with the limited routing tracks (only two to four horizontal M1 routing tracks in the cells). For the advanced nodes with restricted manufacturing-aware design challenges, we will also have to explore ILP- and SMT-based cell-level routing for sub-10nm process nodes as mentioned in [10, 15, 16]. However, for all these, a design automation framework is inevitable, which in turn motivates this work even for earlier technology nodes. Moreover, we can merge the concept of **simultaneous place-and-route (SP&R)** using SMT from [10] with NP-Separate in the future to automate the N/P-cell formation along with NP cell formation stage for advanced technology nodes.

6.4 NP-Separate to Other Transistor Architectures

FinFETs and **gate-all-around (GAA)** transistors have very high current density, faster switching speed, minimized short channel effects, and controlled leakage current compared to the conventional MOSFETs. The NP-Separate design methodology can be applied to FinFET based circuits as

well. The FinFET widths can only be integer multiples of the minimum quantized fin-width. A FinFET with a single fin has the minimum channel width of w_{fmin}. Thus, the width will be $N \cdot w_{\text{fmin}}$ for N-fins. We can discretize the resulting transistor widths in the transistor sizing step to the nearest number of fins without significantly affecting the overall optimality [1, 22]. In the N/P-cell formation step, optimized N and P cells can be constructed by skilled manual effort for the multi-patterning and significantly constrained FinFET layouts. Moreover, SAT-based placement adjustment of the FinFETs inside the unroutable NP cells can be explored by [21] if necessary. For the NP cell formation step, we believe that a modified version of Auto-NP-Separate would be able to generate better NP cells since our algorithms enumerate all possible routes and port locations. However, as discussed in the previous subsection, advanced technology nodes require various routing layers and multiple pattern-supporting design rules for cell-level routing that can be explored using ILP and SMT [10, 15, 16] or genetic algorithm and reinforcement learning [17] in the future.

7 CONCLUSION

In this paper, we proposed design automation algorithms to automate the creation of NP cells and cell libraries, which reduced the design time and the coupling capacitance significantly. Along with design automation, we also proposed a detailed placement algorithm to reduce the chip area with a little wirelength overhead without compromising power and performance compared to the manual NP-Separate designs. The Auto-NP-Separate methodology decreases the coupling capacitance by 13%, critical path delay by 6%, power consumption by 10%, power-delay product by 15%, and energy-delay product by 20% on average compared to the manual NP-Separate methodology. The combined effect in Auto-NP-Separate-P shows an area improvement of 1% with 2% wirelength overhead on average, reduces the coupling capacitance by 10%, critical path delay by 5%, power consumption by 10%, power-delay product by 13%, and energy-delay product by 14% on average compared to the manual NP-Separate methodology.

REFERENCES

[1] Massimo Alioto. 2011. Comparative evaluation of layout density in 3t, 4t, and MT FinFET standard cells. *IEEE Trans. Very Large Scale Integr. Syst.* 19, 5 (May 2011), 751–762. https://doi.org/10.1109/tvlsi.2010.2040094

[2] Bochkanov Sergey Anatolyevich. 2018. ALGLIB. http://www.alglib.net.

[3] Monzurul Islam Dewan and Dae Hyun Kim. 2020. NP-Separate: A new VLSI design methodology for area, power, and performance optimization. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* 39, 12 (Dec. 2020), 5111–5122. https://doi.org/10.1109/TCAD.2020.2966551

[4] Mohan Guruswamy, Robert L. Maziasz, Daniel Dulitz, Srilata Raman, Venkat Chiluvuri, Andrea Fernandez, and Larry G. Jones. 1997. CELLERITY: A fully automatic layout synthesis system for standard cell libraries. In *Proc. ACM Design Automation Conf.* Association for Computing Machinery, New York, NY, USA, 327–332. https://doi.org/10.1145/266021.266126

[5] Renato Hentschke, Guilherme Flach, Felipe Pinto, and Ricardo Reis. 2006. Quadratic placement for 3D circuits using Z-Cell shifting, 3D iterative refinement and simulated annealing. In *Proc. Annual Symposium on Integrated Circuits and Systems Design.* Association for Computing Machinery, New York, NY, USA, 220–225. https://doi.org/10.1145/1150343.1150399

[6] Andy J. Kessler and A. Ganesan. 1985. An introduction to standard-cell VLSI design: Very large scale integration (VLSI) is becoming an important means of producing electronic circuits at low cost, on tight schedules, and with protection for proprietary designs. *IEEE Potentials* 4, 3 (1985), 33–36. https://doi.org/10.1109/MP.1985.6500265

[7] Bernhard Kick, Ulrich Baur, Juergen Koehl, Thomas Ludwig, and Thomas Pflueger. 1997. Standard-cell-based design methodology for high-performance support chips. *IBM Journal of Research and Development* 41, 4.5 (1997), 505–514. https://doi.org/10.1147/rd.414.0505

[8] Dae Hyun Kim and Sung Kyu Lim. 2012. Design quality trade-off studies for 3-D ICs built with sub-micron TSVs and future devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 2, 2 (June 2012), 240–248. https://doi.org/10.1109/JETCAS.2012.2193840

[9] C. Y. Lee. 1961. An algorithm for path connections and its applications. *IRE Transactions on Electronic Computers* EC-10, 3 (1961), 346–365. https://doi.org/10.1109/TEC.1961.5219222

[10] Daeyeal Lee, Dongwon Park, Chia-Tung Ho, Ilgweon Kang, Hayoung Kim, Sicun Gao, Bill Lin, and Chung-Kuan Cheng. 2020. SP&R: SMT-based simultaneous place-&-route for standard cell synthesis of advanced nodes. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* (Nov. 2020), 1–1. https://doi.org/10.1109/TCAD.2020.3037885

[11] Jingwei Lu, Pengwen Chen, Chin-Chih Chang, Lu Sha, Dennis Jen-Hsin Huang, Chin-Chi Teng, and Chung-Kuan Cheng. 2015. ePlace: Electrostatics-based placement using fast fourier transform and Nesterov's method. *ACM Trans. on Design Automation of Electronics Systems* 20, 2 (Feb. 2015), 17:1–17:34. https://doi.org/10.1145/2699873

[12] Takashi Mitsuhashi and Ernest S. Kuh. 1992. Power and ground network topology optimization for cell based VLSIs. In *Proc. ACM Design Automation Conf.* IEEE Computer Society Press, Washington, DC, USA, 524–529. https://doi.org/10.1109/DAC.1992.227748

[13] Gi-Joon Nam and Jingsheng Jason Cong. 2007. *Modern Circuit Placement: Best Practices and Results*. Springer, New York, NY.

[14] Min Pan, Natarajan Viswanathan, and Chris Chu. 2005. An efficient and effective detailed placement algorithm. In *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers (ICCAD'05)*. 48–55. https://doi.org/10.1109/ICCAD.2005.1560039

[15] Dongwon Park, Daeyeal Lee, Ilgweon Kang, Sicun Gao, Bill Lin, and Chung-Kuan Cheng. 2020. SP&R: Simultaneous placement and routing framework for standard cell synthesis in sub-7nm. In *Proc. Asia and South Pacific Design Automation Conf.* Association for Computing Machinery, New York, NY, USA, 345–350. https://doi.org/10.1109/ASP-DAC47756.2020.9045729

[16] Dongwon Park, Daeyeal Lee, Ilgweon Kang, Chester Holtz, Sicun Gao, Bill Lin, and Chung-Kuan Cheng. 2020. Grid-based framework for routability analysis and diagnosis with conditional design rules. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* 39, 12 (Dec. 2020), 5097–5110. https://doi.org/10.1109/TCAD.2020.2977066

[17] Haoxing Ren and Matthew Fojtik. 2021. Standard cell routing with reinforcement learning and genetic algorithm in advanced technology nodes. In *Proceedings of the 26th Asia and South Pacific Design Automation Conference (ASPDAC'21)*. Association for Computing Machinery, New York, NY, USA, 684–689. https://doi.org/10.1145/3394885.3431569

[18] Nikolai Ryzhenko and Steven Burns. 2012. Standard cell routing via Boolean satisfiability. In *Proc. ACM Design Automation Conf.* Association for Computing Machinery, New York, NY, USA, 603–612. https://doi.org/10.1145/2228360.2228470

[19] Carl Sechen. 1988. Chip-planning, placement, and global routing of macro/custom cell integrated circuits using simulated annealing. In *Proc. ACM Design Automation Conf.* IEEE, Washington, DC, USA, 73–81. https://doi.org/10.1109/DAC.1988.14737

[20] Jihee Seo and Dae Hyun Kim. 2019. High-throughput multiplier architectures enabled by intra-unit fast forwarding. In *Proc. IEEE Int. Symp. on Computer Arithmetic*. IEEE Computer Society, Los Alamitos, CA, USA, 143–150. https://doi.org/10.1109/ARITH.2019.00036

[21] Anton Sorokin and Nikolay Ryzhenko. 2019. SAT-based placement adjustment of FinFETs inside unroutable standard cells targeting feasible DRC-clean routing. In *Proceedings of the 2019 on Great Lakes Symposium on VLSI (GLSVLSI'19)*. Association for Computing Machinery, New York, NY, USA, 159–164. https://doi.org/10.1145/3299874.3317965

[22] Brian Swahn and Soha Hassoun. 2006. Gate sizing: FinFETs vs 32nm bulk MOSFETs. In *Proceedings of the 43rd Annual Design Automation Conference (DAC'06)*. Association for Computing Machinery, New York, NY, USA, 528–531. https://doi.org/10.1145/1146909.1147047

[23] Xiaohai Wu, Changge Qiao, and Xianlong Hong. 1999. Design and optimization of power/ground network for cell-based VLSIs with macro cells. In *Proc. Asia and South Pacific Design Automation Conf.* IEEE Computer Society, Los Alamitos, CA, USA, 21–24. https://doi.org/10.1109/ASPDAC.1999.759700

[24] A. Zelikovsky, P. Tucker, and A. B. Kahng. 1999. Optimization of linear placements for wirelength minimization with free sites. In *Asia and South Pacific Design Automation Conference*. IEEE Computer Society, Los Alamitos, CA, USA, 241. https://doi.org/10.1109/ASPDAC.1999.760005

[25] Lihong Zhang and Yingtao Jiang. 2005. Global-routing driven placement strategy in analog VLSI physical designs. In *Proc. IEEE Int. Midwest Symp. on Circuits and Systems*. IEEE, Washington, DC, USA, 1239–1242. https://doi.org/10.1109/MWSCAS.2005.1594332

[26] Ziran Zhu, Jianli Chen, Zheng Peng, Wenxing Zhu, and Yao-Wen Chang. 2018. Generalized augmented Lagrangian and its applications to VLSI global placement. In *Proc. ACM Design Automation Conf.* Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/3195970.3196057

Received March 2021; revised December 2021; accepted December 2021