IEICE
TRANSACTIONS

on Fundamentals of Electronics,
Communications
and Computer Sciences

DOI:10.1587/transfun.2024EAP1160

Publicized:2024/12/23

A PUBLICATION OF THE ENGINEERING SCIENCES SOCIETY
The Institute of Electronics, Information and Communication Engineers
Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3 chome, Minato-ku, TOKYO, 105-0011 JAPAN

PAPER *Special Section on VLSI Design and CAD Algorithms*

Standard Cell Structure and Diffusion Reordering for Block Area Reduction in Double Diffusion Break FinFET Process

Shinichi NISHIZAWA[†a)], ***Member and*** **Shinji KIMURA**[†b)], ***Fellow***

SUMMARY This paper proposes standard cell layout style to reduce the block area in double-diffusion break FinFET process. The first generation of FinFET process technology requires a double-diffusion break to shutdown the leakage current under the dummy gate. Double-diffusion break at the edge of the standard cell requires two additional unit cells for the dummy gates and it results in a large block area. We propose a FinFET cell layout style which VDD/VSS diffusions can be shared with adjacent cells. The proposed layout structure places the VDD/VSS-diffusions at the cell edge to place these nodes adjacently, and it eliminates the use of a double diffusion break. We also propose a diffusion reorder algorithm to improve the use of common potential node sharing. Experimental results show that the proposed cell library with a new layout style and reordering algorithm achieves an 8.39% area reduction in on average.

key words: *Standard Cell, FinFET, Double Diffusion Break*

1. Introduction

Recent state-of-the-art CMOS transistor technology uses FinFET structure for both high-performance processors and mobile System-on-Chips. FinFET has tiny three-dimensional (3-D) silicon fins surrounded by the gate for good channel controllability, thus it achieves both high drivability and low leakage current. However, these 3-D structures require several new challenges in fabrication [1]. Layout-dependent stress effect and its impact on device performance is one of the major challenges in scaled CMOS technologies. The dependence of the channel stress on different Fin lengths due to the Fin cut process has been widely concerned. The first generation of the commercial FinFET process requires a double diffusion break (DDB) to mitigate this stress effect in the Fin cut process at the boundaries of adjacent standard cells. DDB needs two dummy transistors at the cell boundary thus it needs a large area penalty in layout design. Beyond the 7-nm node, a single diffusion break (SDB) replaces DDB to improve the transistor density [2][3]. However, still, this SDB process has been reported to cause undesired stress relaxation in FinFET since it needs tiny and tall shallow trench isolation (STI) in SDB [4][5].

From the circuit designer's point of view, diffusion break can be avoided by dummy gate insertion [6], s.t., that is to isolate two adjacent diffusions by a dummy transistor. Dummy transistor insertion is an area-efficient isolation technique without introducing a thin STI trench. However dummy transistor has leakage from one side of diffusion to the other, and it will degrade signal quality and cause operation failure of the circuit. These dummy transistors can be turned off by applying gate-source voltage to 0 V, however, at least one source node for the dummy transistor is needed to supply VDD/VSS to the dummy transistor. In [6], the placement of standard cells is reordered to place at least one VDD/VSS diffusion to the neighbor of the dummy gate to be powered off. In [7], [8], the transistor placement inside the standard cells is reordered and optimized to place the diffusions connected VDD/VSS as the neighbor of the dummy transistor to be powered off. These techniques reduce the number of DDBs inside the circuit and successfully reduce the block area.

On the other hand, the leakage current is not a problem if both sides of the dummy transistor have the same voltage. For example, if two adjacent standard cells have diffusion connected to VDD/VSS, SDBs can be introduced at this cell boundary. This approach can reduce the area of circuit block replacing DDB to SDB under the DDB design rule.

This paper, an extension of [9], proposes the common-node-aware standard cell library using ASAP7 predictive 7-nm PDK [10] which assumes DDB placement in block design. We propose cell layout structure to adjacent places or overlap the common-node diffusions for adjacent cells. We also propose a simple diffusion reorder technique to enhance the usage of common-node adjacent placement or overlap placement to reduce the circuit area. The key extensions of this paper are,

- evaluation of the impact of common node sharing on delay performance by its relatively high diffusion resistance,
- increase the logic family of cell library, s.t., full adders, half adders Flip-Flops with asynchronous set and reset,
- add evaluation of the practical circuits from Synopsys DesignWare IPs and OpenCores.

The rest of this paper is organized as follows. Section 2 describes the related works for diffusion-sharing aware transistor placement. Section 3 describes the common-node sharing aware cell structure. Section 4 describes the simple transistor reorder algorithm to improve the usage of common-node sharing. Section 5 describes the experimental results. Section 6 concludes this paper.

[†]The author is with the Graduate School of Information, Production and Systems, Waseda University.
a) E-mail: nishizawa@aoni.waseda.jp
b) E-mail: shinji_kimura@waseda.jp

2. Related works of diffusion sharing and their limitations

This section describes the reason why this new technique is required for FinFET Process. Diffusion is an important component to form MOSFET. Diffusion needs some area, and its parasitic capacitance and resistance degrade both the operation speed and power performance of the circuit. There are several techniques to reduce the area of diffusion to improve parasitic capacitance and resistance. Gate jogging and folding (finger) are simple layout techniques to reduce the area of diffusion for the given transistor placement [11].

Changing the transistor placement can offer more gain to reduce diffusion area. Several papers propose method of transistor placement for diffusion sharing. An Euler Path is a path that uses every edge of the graph exactly once. This idea can be extended to circuit design to find a series of transistor placements with fewer breaks and improve diffusion sharing. References [12], [13] use Euler Path approach to improve the use of diffusion sharing. Reference [14] implements transistor placement with “Threshold Accepting” heuristic algorithm to maximize the diffusion sharing and minimizes the interconnection length. Reference [15] formulates the diffusion sharing problem as the objective function of SAT and solves it by SAT solver.

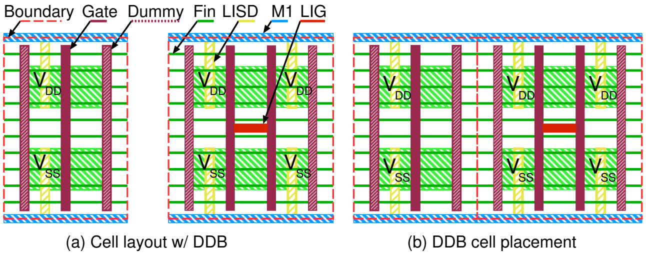
These papers optimize transistor placement inside one cell. These papers do not consider adjacent cells to share diffusions since the area penalty of diffusion break on Planer technology is often small, less than one unit cell. Also, diffusion sharing between adjacent cells in planer technology is not common due to the difficulty of predicting the diffusion capacitance and resistance of diffusion sharing between adjacent cells.

On the other hand, the diffusion break in DDB FinFET needs two unit cells, and this is negligible. Therefore, we propose a new layout style to share diffusion between adjacent cells. The proposed layout style places the VDD/VSS-diffusion at the cell boundary, which means the diffusion sharing is broken, so the idea of the Euler Path cannot be used simply. Diffusion sharing between adjacent cells in FinFET also makes it difficult to predict the impact of diffusion capacitance and resistance. We confirmed that the impact is small in the ASAP7 7-nm predictive PDK [10] as a result of simulation.

3. Layout structure with common node sharing

3.1 Single diffusion break design on double diffusion break rule

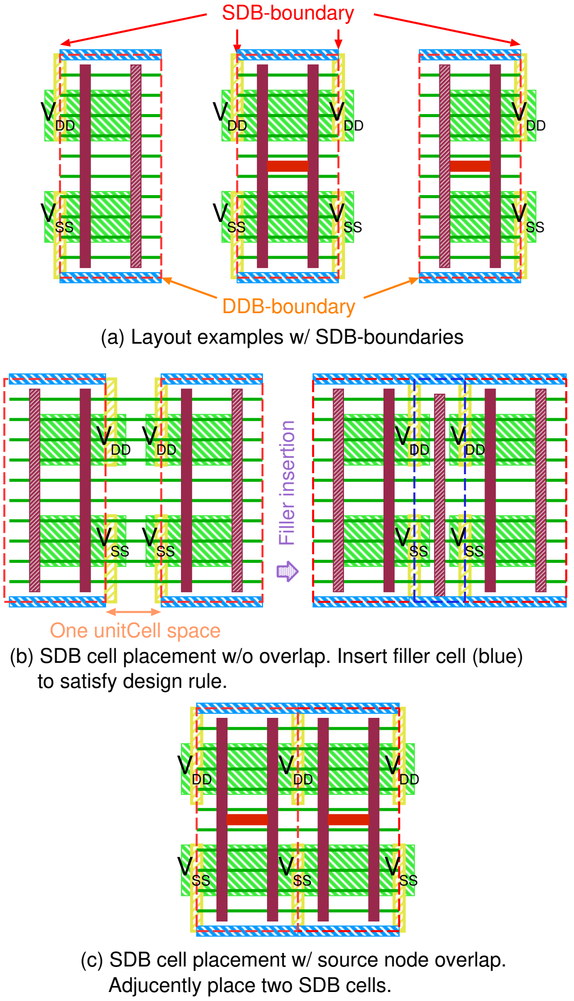
Figure 1 shows the conventional cell layout with the DDB layout rule. To satisfy the DDB design rule, dummy transistors are required in each cell boundary. It is clear that the DDB structure has an area overhead, and this area overhead is relatively high for the cells with a smaller number of transistors.



(a) Cell layout w/ DDB

(b) DDB cell placement

Fig. 1 DDB cell layouts and placement example.



(a) Layout examples w/ SDB-boundaries

(b) SDB cell placement w/o overlap. Insert filler cell (blue) to satisfy design rule.

(c) SDB cell placement w/ source node overlap. Adjucently place two SDB cells.

Fig. 2 SDB cell layouts, and these adjacent placement and overlap placement example.

However, even when using DDB design rules, diffusions with the same signal node can be placed adjacent each other. More aggressively, these nodes might be overlapped if their electrical performance is acceptable. To make the diffusions with the same signal nodes adjacent or overlapping, a common-potential-aware cell structure has been introduced and illustrated in Fig. 2. This structure itself is not new. A similar structure is available to improve the drivability in both digital and analog circuits [16][1]. However, the use of this structure to replace DDB to SDB in the DDB design rule is the key point of this paper.

It is clear that not all of the diffusions at cell boundaries are VDD/VSS. If the cell boundary has both p-diffusion with VDD connection and n-diffusion with VSS connection, the boundary can be treated as SDB or overlap, and denote this boundary as SDB-boundary. In other cases (where either p- or n-diffusion is not VDD/VSS), it is treated as a conventional DDB, and two dummy transistors are required at the cell boundary. This boundary is denoted as DDB-boundary.

These boundaries need some placement rules to satisfy design rule. Two DDB-boundaries can be adjacent to each other since these boundaries have two dummy transistors to form the DDB. The DDB-boundary and SDB-boundary cannot be adjacent since they need to form a DDB. In this case, one unit cell filler with the dummy transistor must be placed between the boundaries to form a DDB with two dummy transistors, and we do not have any area reduction. Similarly, two SDB-boundaries may need one unit cell filler with the dummy transistor placed between boundaries to satisfy the design rule. In this case, the boundary is SDB and it can gain the space of one unit cell. If we accept two SDB-boundaries to overlap, these boundaries can be placed adjacent to each other and no extra unit cell or dummy transistor is required. Thus, we can reduce the block area if two SDB-boundaries are adjusted or overlapped.

These cells need cell space control between different boundary conditions and cells. Modern Place-and-Route tools support these space controls to prevent design rule violation and conflict between several cells. Synopsys IC Compiler supports set_spacing_label_rule command to define the spacing rule and set_lib_cell_spacing_label command to name these rules to the right and/or left boundaries of target standard cells.

3.2 Selection of diffusion abutment or overlap

There are two options for SDB-boundaries: abutment placement or overlap placement. To determine the placement option, its electrical characteristics should be carefully considered.

It is widely known that FinFETs have relatively high diffusion resistance, which affects the long tail in voltage swing and results in a long delay. This is a serious problem for both the high performance digital and analog circuits. This high diffusion resistance is critical when two transistors share one VDD/VSS-diffusion and operate simultaneously: large current flow in the VDD/VSS-diffusions generate a voltage drop and it decreases the transistor performance. The most critical problem of VDD/VSS-diffusion overlap is that this effect cannot be predicted at the cell level characterization since cell characterization does not consider the adjacent cells.

To evaluate the effect of VDD/VSS-diffusion overlap on delay characteristics, we evaluate the delay of Inverter cells with different scenarios. Figure 3 shows the target circuits. We prepare four sets of circuits under test to evaluate the impact of diffusion resistance on circuit operation. Figure 3 (a) is the baseline circuit, only one Inverter switches and its charge/discharge current will flow into parasitic diffusion resistance. Figures 3 (b) to (d) are the two Inverter circuits with simultaneous input. Switching current through the parasitic diffusion resistance will generate a voltage drop and it will generate negative body bias and degrade transistor performance. An Inverter with 1× drive strength (Figure 3(b)) shares their VDD/VSS-diffusion with another Inverter. Inverters with 2× and 4× drive strength (Figures 3(c) and (d)) have its own VDD/VSS-diffusion so it has less effect on the charge/discharge by another Inverter. We assume Fan-out 4 (FO4) loading condition for all of the Inverters

Figure 4 shows the transient waveform of input, output, and voltage drop of parasitic diffusion resistance. Circuit simulation has been done by post-layout simulation with Synopsys HSPICE. Figure 4 (b) shows the voltage drop of parasitic diffusion resistor R_{SN1} in Fig. 3. The result shows the maximum value of voltage drop of two shared 1× inverters has twice larger than the single switch case. The maximum value of voltage drop decreases as the number of driveability increases. An Inverter with 1× drive strength shares its contacts with the other cell, and its contact has a larger voltage drop caused by the operation current of two Inverters. For Inverters with higher drive strength, they have their own contacts inside the cell thus the impact of shared contact on cell delay becomes small. However, this effect is almost negligible on the output waveform in ASAP7 PDK, as plotted in Fig. 4 (c).

In this paper, as a demonstration of cell space control, we assume that the transistor with more than two parallel connections can overlap. Otherwise, the VDD/VSS-diffusion cannot be shared, but they can be placed adjacent each other.

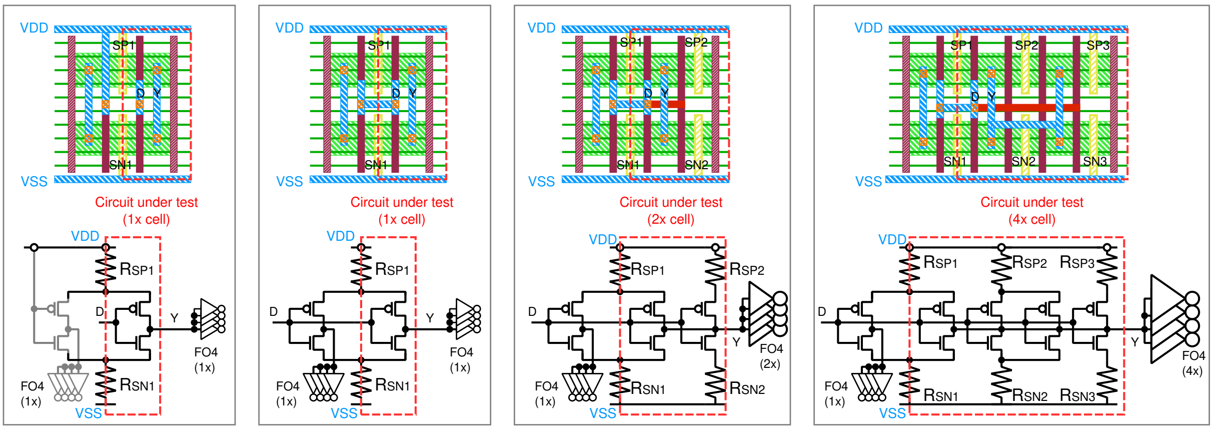
3.3 Impact of diffusion sharing

Diffusion sharing (or finger) is a common method to share the same diffusion nodes to reduce the diffusion capacitance. Since will we modify the standard cell layout to improve the number of SDBs, it is important to evaluate the impact of diffusion sharing. Figure 5 shows two Inverter cells with 2x drivabilities. Figure 5(a) is a two parallel Inverters without diffusion sharing, and figure 5 (b) is an Inverter cell with diffusion sharing. We assume FO4 loading condition.

Circuit simulation has been done by post-layout simulation with Synopsys HSPICE. The propagation delay of Inverters without diffusion sharing and Inverters with diffusion sharing are 13.7 ps, 13.3 ps, respectively. This difference is 3.0%. Simulation results show the diffusion sharing change delay performance of cells, and it requests re-characterization of the cell library for accurate synthesis and timing-driven place-and-route.

4. SDB-aware diffusion reorder algorithm

To improve the utilization rate of SDB, a simple diffusion reorder algorithm is introduced. This algorithm starts from the baseline transistor placement and swaps the diffusions to



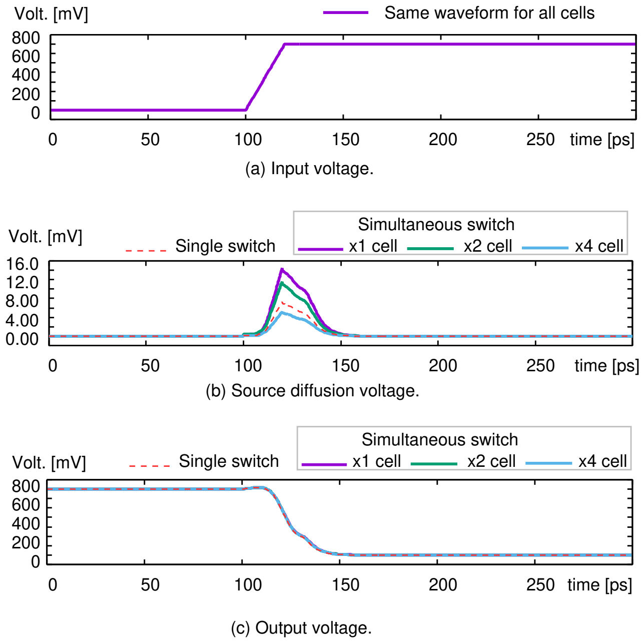
(a) Inv 1x (CUT), no simultaneous switching.

(b) Inv 1x and Inv 1x (CUT), simultaneous switching.

(c) Inv 1x and Inv 2x (CUT), simultaneous switching.

(d) Inv 1x and Inv 4x (CUT), simultaneous switching.

Fig. 3 Simultaneous switch circuit to evaluate the impact of parasitic diffusion resistance on circuit operation. (a) Stand alone switching. (b) Simultaneous switching with two 1x Inverters. (c) Simultaneous switching with 1x Inverter and 2x Inverter. (d) Simultaneous switching with 1x Inverter and 4x Inverter.



(a) Input voltage.

(b) Source diffusion voltage.

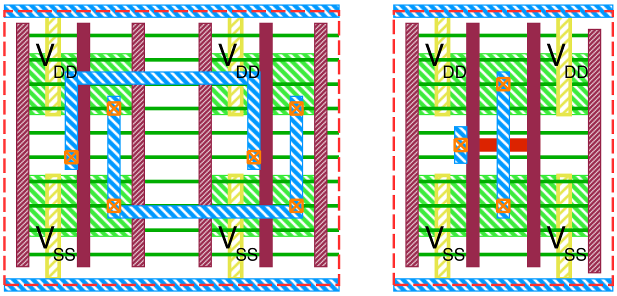
(c) Output voltage.

Fig. 4 Simulation waveform. (a) Input waveform. (b) Voltage drop of parasitic source diffusions (R_{SN1} in Fig.3). (c) Output waveform.

the cell boundary to achieve SDB placement.

The algorithm swaps the diffusions of the transistor from the left(right) side of the cell edge to the opposite side of the cell edge, and accepts the reordering result when that achieves the smallest cell width. Note that in the FinFET process, we cannot cut the active transistor gate; The PMOS and NMOS gates inside the same unit cell should be physically and electrically connected to each other. This is a design constraint of transistor placement in this process technology.

Algorithm 1 and 2 shows the main procedure and diffusion reorder procedure, respectively. Figure. 6 shows an example. Diffusion reordering starts from the initial placement. Before the reorder, two ”virtual source” nodes are placed on the boundary of each edge of the cell to help reorder (VV_{DD} and VV_{SS} in figure 6). The original cell layout starts from the DDB structure, and the number of transistors N includes both active transistors and dummy transistors (For Fig. 6, N=7).



(a) Cell layout w/o shared output diffusions

(b) Cell layout w/ shared output diffusions

Fig. 5 Evaluation of common diffusion sharing. (a) Layout without common diffusion sharing. (b) Layout with common diffusion sharing.

Here let’s reorder the transistor diffusions from left to right. The first operation is VDD/VSS node swap: if i-th transistor has a VDD/VSS node for both PMOS and NMOS transistors, swap its location of two diffusions. Every time the placement is modified, insert a dummy transistor to satisfy DDB rule, then remove the redundant dummy transistors.

The second operation is non VDD/VSS node swap, and this operation depends on the node of diffusion. If the diffusion of i-th transistor is the same node as the diffusion of a previous active transistor (case A in Fig. 6), the diffusion of i-th transistor can be swapped and merged to the same node diffusion of the previous active transistor. Another possible operation is location adjustment of VDD/VSS diffusion of PMOS and NMOS transistors. There are two possible way

Algorithm 1 Main procedure

```
function procReorderTransistors({Sequence of all transistors})
    $T_{\mathrm{all,org}} \Leftarrow$ {Sequence of all transistors}
    $T_{\mathrm{act,org}} \Leftarrow$extActiveTran({Sequence of all transistors})
    $T_{\mathrm{virtual,L}}$.pdiff.r $\Leftarrow$ {VDD}, $T_{\mathrm{virtual,L}}$.ndiff.r $\Leftarrow$ {VSS}
    $T_{\mathrm{virtual,R}}$.pdiff.l $\Leftarrow$ {VDD}, $T_{\mathrm{virtual,R}}$.ndiff.l $\Leftarrow$ {VSS}
    $T_{\mathrm{act,org}} \Leftarrow T_{\mathrm{virtual,l}}$, $T_{\mathrm{act,org}}$, $T_{\mathrm{virtual,R}}$
    $U_{\mathrm{org}} \Leftarrow$numUnitCell($T_{\mathrm{act,org}}$)

# (1) reorder transistors from left
    $T_{\mathrm{reord,L}}$ = procReorderFromLeft($T_{\mathrm{act,org}}$)
# (2) reorder transistors from right
    $T_{\mathrm{reord,R}}$ = procReorderFromRight($T_{\mathrm{act,org}}$)
# (3) select smallest reorder result
    $T_{\mathrm{final}}$ = selMinUnitCell($T_{\mathrm{act,org}}$, $T_{\mathrm{reord,L}}$, $T_{\mathrm{reord,R}}$)
    $U_{\mathrm{final}}$ = numUnitCell($T_{\mathrm{final}}$)
    return $T_{\mathrm{final}}$, $U_{\mathrm{final}}$
end function
```

Algorithm 2 Transistor reorder procedure

```
# data structure of Transistors
# Transistors[i].p/ndiff.l/r : node name of left/right p/n-diffusion
function procReorderFromLeft($T_{\mathrm{reord,L}}$)
    $U_{\mathrm{reord,L}}$ = numUnitCell( $T_{\mathrm{reord,L}}$ ) # reorder transistors from left
    for i=1,..., $U_{\mathrm{reord,L}}$-1 do
# (1) check if $i$th transistor has same node diff. as $(i-1)$th
        if match($T_{\mathrm{reord,L}}$[i-1].pdiff.r, $T_{\mathrm{reord,L}}$[i].pdiff) and match($T_{\mathrm{reord,L}}$[i-1].ndiff.r, $T_{\mathrm{reord,L}}$[i].ndiff) then
# if ith and (i-1)th transistors has same p/ndiff at right, swap
            if match($T_{\mathrm{reord,L}}$[i-1].pdiff.r eq $T_{\mathrm{reord,L}}$[i].pdiff.r then
                swap_diff($T_{\mathrm{reord,L}}$[i].pdiff)
            end if
            if match($T_{\mathrm{reord,L}}$[i-1].ndiff.r, $T_{\mathrm{reord,L}}$[i].ndiff.r then
                swap_diff($T_{\mathrm{reord,L}}$[i].ndiff)
            end if
# add dummy transistor for Double Diffusion Break
            procAddDummyDDB($T_{\mathrm{reord,L}}$)
# remove duplicated dummy transistor, same diff. adjucents
            procRemovDuplication($T_{\mathrm{reord,L}}$)
# (2) check if $i$th transistor has same node diff. as $(i+1)$th
        else if match($T_{\mathrm{reord,L}}$[i].pdiff, $T_{\mathrm{reord,L}}$[i+1].pdiff.l) and match($T_{\mathrm{reord,L}}$[i].ndiff, $T_{\mathrm{reord,L}}$[i+1].ndiff.l then
            if match($T_{\mathrm{reord,L}}$[i].pdiff, $T_{\mathrm{reord,L}}$[i+1].pdiff.l then
                swap_diff_rl($T_{\mathrm{reord,L}}$[i].pdiff)
            end if
            if match($T_{\mathrm{reord,L}}$[i].ndiff, $T_{\mathrm{reord,L}}$[i+1].ndiff.l then
                swap_diff_rl($T_{\mathrm{reord,L}}$[i].ndiff)
            end if
            procAddDummyDDB($T_{\mathrm{reord,L}}$)
            procRemovDuplication($T_{\mathrm{reord,L}}$)
        end if
    end for
    return $T_{\mathrm{reord,L}}$
end function
function procReorderFromRight($T_{\mathrm{reord,R}}$)
# similar to procReorderFromLeft()
    return $T_{\mathrm{reord,R}}$
end function
```

to adjust location. If the VDD/VSS-diffusions of $(2n-i)$-th transistor has already swapped to the left, the VDD/VSS-diffusion of $(2n)$-th transistor might be adjusted to the right (case B in Fig. 6).

Continue this operation until it reaches to the right end of the cell, and store the placement result. The same operation can also be performed from right to left. Finally, the three placement candidates (original, reorder from left, reorder from right) are compared and the placement result with the smallest area is selected.

In this experiment, we manually design layout of the cell library with Cadence Virtuoso. For a fair comparison, we try to use the same or similar shapes of signal wires, contacts, and metal pins. We can design our new cell layout from original cell layout with minor modification. However, some complex logic functions may need major layout modification, even or additional area for routing. These design restrictions reduce the effect of layout compaction of the proposed work.

5. Experimental results

5.1 Experimental setup

Three standard cell libraries were designed to validate the proposed SDB-aware cell layout and SDB-aware transistor reordering. In this experiment, we use Arizona State University 7-nm Predictive PDK (ASAP7) as the design environment [10], and perform Place-and-Route in Synopsys environment [17]. Baseline library $\mathrm{LIB}_{\mathrm{ASAP}}$ comes from original ASAP7 library, but we select very limited primitive logics from PDK. The SDB-aware library $\mathrm{LIB}_{\mathrm{SDB}}$ is the library based on $\mathrm{LIB}_{\mathrm{ASAP}}$. $\mathrm{LIB}_{\mathrm{SDB}}$ uses same transistor placement as $\mathrm{LIB}_{\mathrm{ASAP}}$, however the layout of standard cells are modified to SDB when both PMOS and NMOS diffusions at the cell edge are the source. $\mathrm{LIB}_{\mathrm{REORDER}}$ is the library based on $\mathrm{LIB}_{\mathrm{ASAP}}$ and its diffusions are reordered to improve the SDB ratio. These three libraries have same logic functions, same drive strength families, and the same or similar pin shape to reduce these effects on Place-and-Route.

Table 1 summarizes the logic functions and drive strengths of standard cells. ”xpN” denote the drive strength of ”0.Nx”, defined in the original ASAP7 library. DFFHQN and DFFHQ are the positive edge D-Flip-Flop with negative and positive data polarity, respectively. Async_DFFH is the positive edge D-Flip-Flop with asynchronous set and reset. Each library contaisn 58 cells thus $\mathrm{LIB}_{\mathrm{ASAP}}$ has 116 DDBs and zero SDB. $\mathrm{LIB}_{\mathrm{SDB}}$ has 80 DDBs and 36 SDBs. Transistor reordering achieves $\mathrm{LIB}_{\mathrm{REORDER}}$ to achieve 66 DDBs and 50 SDBs. Figure 7 shows a diffusion reorder example of a normal D-Flip-Flop (DFFHQNx2_ASAP7_75t_SL). Transistor reorder moves source nodes at the cell boundary and it reduces 2-unitCell area in this case.

These libraries are manually designed with Cadence Virtuoso. For a fair comparison, we try to use the same or similar shape of the pin and its surrounding metals from

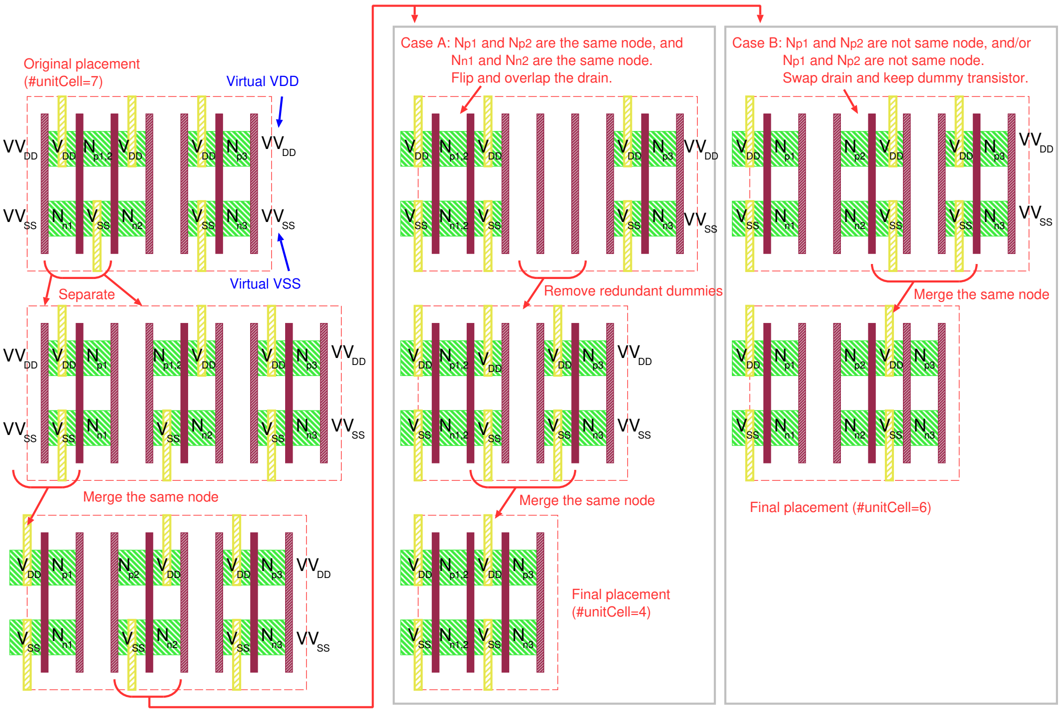

Original placement
(#unitCell=7)
Virtual VDD
Virtual VSS
Separate
Merge the same node
Case A: Np1 and Np2 are the same node, and
Nn1 and Nn2 are the same node.
Flip and overlap the drain.
Remove redundant dummies
Merge the same node
Final placement
(#unitCell=4)
Case B: Np1 and Np2 are not same node, and/or
Np1 and Np2 are not same node.
Swap drain and keep dummy transistor.
Merge the same node
Final placement (#unitCell=6)
VV_DD
VV_SS
V_DD
V_SS
N_p1
N_p1,2
N_p2
N_p3
N_n1
N_n1,2
N_n2
N_n3

Fig. 6 Diffusion reorder flow (Fin layer is not shown). Algorithm try to flip diffusions and check these potentials. Diffusions on same node can be overlapped to save area. If not, keep dummy transistors and use DDB.

Table 1 Logic functions and strength family of three libraries. ”xpN” denote drive strength of ”0.Nx”.

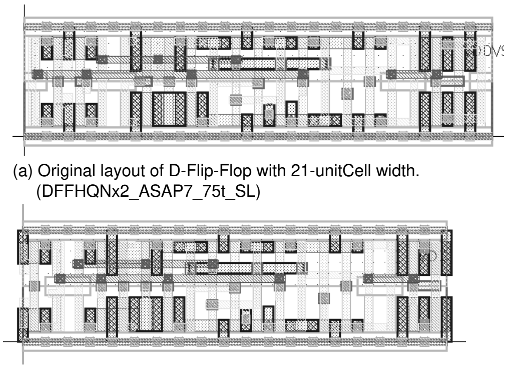
Logic	drive strength	LIB$_{\text{ASAP}}$		LIB$_{\text{SDB}}$		LIB$_{\text{REORDER}}$	
		#DDB	#SDB	#DDB	#SDB	#DDB	#SDB
Inverter	xp33,xp67,x1,x2,x3,x4,x5,x6,x8,x11,x13	22	0	7	15	7	15
NAND2	xp33,xp5,xp67,x1,1p5,x2	12	0	10	2	9	3
NAND3	xp33,x1,x2	6	0	5	1	4	2
NAND4	xp25,xp75	4	0	3	1	2	2
NOR2	xp33,xp67,x1,x1p5,x2	10	0	9	1	8	2
NOR3	xp33,x1,x2	6	0	5	1	4	2
NOR4	xp25,xp75	4	0	3	1	2	2
XOR2	xp5,x1,x2	6	0	3	3	3	3
XNOR2	xp5,x1,x2	6	0	3	3	3	3
AOI21	xp33,xp5,x1	6	0	6	0	6	0
AOI22	xp33,xp5,x1	6	0	6	0	6	0
OAI21	xp33,xp5,x1	6	0	4	2	4	2
OAI22	xp33,xp5,x1	6	0	6	0	6	0
DFFHQN	x1,x2,x3	6	0	5	1	0	6
DFFHQ	x4	2	0	1	1	0	2
Async_DFFH	x1,x2	4	0	2	2	0	4
HalfAdder	x1	2	0	1	1	1	1
FullAdder	x1	2	0	1	1	1	1
Total		116	0	80	36	66	50

the original layout. After all of the cells are designed, they are converted to the physical library for Place-and-Route by Synopsys IC Compiler. The physical layout is scaled 4× by the original library in the same manner as the official Place-and-Route flow of ASAP7 PDK. Timing library is generated by Synopsys PrimeLib, utilizing RC extracted netlist from SimensEDA Calibre xRC.

For the benchmark, we select several circuits from Synopsys DesignWare IPs, such as 64-bit multiplier, 64-bit divider, 512-bit ECC, 64-bit 32-word D-Flip-Flop based two-port RAM, Discrete Cosine Transform (DCT), and 64-bit floating-point arithmetic circuit. ”aes core” is a set of AES cipher de-chipper that comes from OpenCores[18]. D-Flip-Flops are inserted into the inputs and outputs of these circuits. We use 2-ns clock period for synthesis and Place-and-Route. Input pins are driven by 1× size inverter, and output pins have load capacitance as same as the gate capacitance of 4× size inverter.

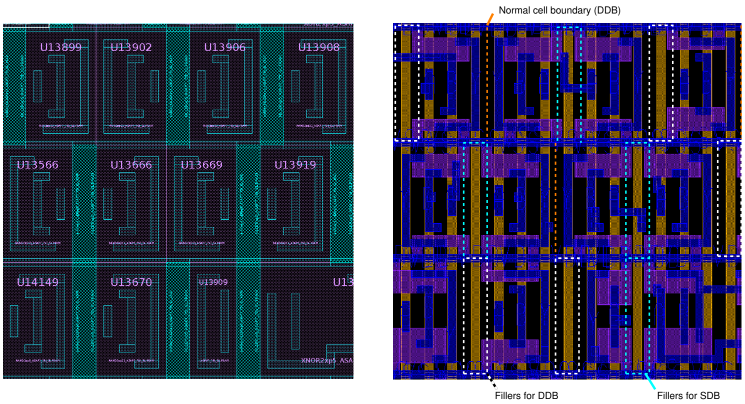
Table 2 Block area result.

Circuit name	DW IP name	Set/reset	Height	LIB$_{\text{ASAP}}$ width	LIB$_{\text{SDB}}$ width		LIB$_{\text{REORDER}}$ width	
			μm	μm	μm	%	μm	%
64-bit Multiplier	DW02_mult	Async.	27	15.6	14.3	91.7	14.0	90.3
64-bit Divider	DW_div	Sync.	27	61.8	59.0	95.4	57.7	93.4
512-bit ECC	DW_ecc	Async.	13.5	42.1	37.8	89.7	37.4	88.7
aes core	(from OpenCores)	Sync.	27	135	124	91.5	123	90.6
64-bit 32-word Register file	DW_ram_2r_w_s_dff	Async.	27	60.5	56.4	93.2	55.3	91.4
Discrete Cosine Transform	DW_dct_2d	Sync.	27	56.6	52.7	93.2	52.3	92.4
64-bit Floating-point add/sub	DW_fp_addsub	Async.	27	19.9	18.8	94.6	18.8	94.6
Average %						92.8		91.6


(a) Original layout of D-Flip-Flop with 21-unitCell width. (DFFHQNx2_ASAP7_75t_SL)

(b) After reorder the transistors of D-Flip-Flop with 19-unitCell width.

Fig. 7 Layout example of diffusion reordering of positive edge negative data polarity D-Flip-Flop (Fin layer is not shown).
(a) Layout of 2x strength D-Flip-Flop from LIB$_{\text{ASAP}}$ (DFFHQNx2_ASAP7_75t_SL) with 21-unitCell width.
(b) D-Flip-Flop after the diffusion reordering from LIB$_{\text{REORDER}}$. 19-unitCell width.



(a) Placement result in IC Compiler (FRAME view)

(b) GDS layout. Only diffusions, gate, M1 layers are shown. Rectangles in white dashed-line show Filler cells with DDB. Rectangles in light-blue dashed-line show Filler cells with SDB. Orange dashed-lines show cell boundary with DDB.

Fig. 8 Block layout example of 64-bit multiplier. (a) Layout in IC Compiler (wires are not shown). Cells with green fill are FILLER cell. Others are logic cells. (b) GDS layout (Diffusions, gate, M1 layers are shown). Some FILLERs (marked as light-blue dashed line) are used to form SDB.

The three libraries are compared in terms of minimum achievable block area. In the Place-and-Route, the block height is fixed, and find minimum block width without violation (short, open, illegal routing and placement, timing violation).

5.2 Results

Figure 8(a) shows a part of the final block layout result of 64-bit multiplier. Here cells with purple instance names are logic cells or sequential cells, and cells with green color are filler cells. Signal and power metal wires are not shown. Figure 8(a) shows a GDS view of the same area. Filler cells with light-blew dashed lines has VDD/VSS diffusions for both cell boundaries; this forms SDB. Filler cells with white dashed lines has one or no VDD/VSS diffusions; this forms normal DDB. On DDB, it can be seen that at least two or more dummy transistors are placed to form DDB. It demonstrates that IC Compiler successfully controls the placement constraints for both SDB and DDB cell boundaries.

Table 2 summarizes the minimum block size achieved by three libraries. Note that Place-and-route in ASAP7 PDK assumes 4× scaling to avoid license issues. This area is re-converted from 4× size to the original 1× size. LIB$_{\text{SDB}}$ which converts DDB to SDB, achieves 7.24% area reduction on average compared to the baseline library LIB$_{\text{ASAP}}$. LIB$_{\text{REORDER}}$ achieves 8.39% area reduction from LIB$_{\text{ASAP}}$ by replacing the transistors to push these source nodes to the boundary of the cell. The block design result demonstrates the proposed layout structure achieves a smaller block area. This work achieves improved area reduction than the previous report [9]. The main reason for this improvement is an extension of the logic families, as shown in Table 2.

6. Conclusion

This paper proposes a method to utilize SDB for common potential nodes with adjusted placement or overlap placement to achieve a smaller block area in DDB-aware FinFET process. The proposed algorithm reorders the diffusions connected VDD/VSS to the cell boundaries and improve the ratio of SDB. Experimental results show proposed library achieves 8.39% block area reduction on average.

Acknowledgment

This work has been supported by KAKENHI JP21K17723. This work is also partly supported by VDEC, the University of Tokyo in collaboration with Nihon Synopsys G.K., Cadence Design Systems, Siemens EDA Japan K.K..

References

[1] A.L.S. Loke, et al., “Analog/mixed-signal design in finFET technologies,” Workshop on Advances in Analog Circuit Design, 2017.

[2] G. Yeap, et al, “5nm CMOS Production Technology Platform featuring full-fledged EUV, and High Mobility Channel FinFETs with densest 0.021μm2 SRAM cells for Mobile SoC and High Performance Computing Applications,” International Electron Devices Meeting, pp.879–882, 2019.

[3] W.C. Jeong, et al., “True 7nm platform technology featuring smallest FinFET and smallest SRAM cell by EUV, special constructs and 3rd generation single diffusion break,” Symposium on VLSI Technology, pp.59–60, 2018.

[4] R. Hpdlo, et al., “A 10nm high performance and low-power CMOS technology featuring 3rd generation FinFET transistors, Self-Aligned Quad Patterning, contact over active gate and cobalt local interconnects,” International Electron Devices Meeting, pp.673–676, 2017.

[5] X. He, et al., “Fin-shape Optimization for Single Diffusion Break Device Performance in FinFET Technology,” International Symposium on VLSI Technology, Systems and Applications, 2-pages, 2022.

[6] Y. Du and M.D.F. Wong, “Optimization of standard cell based detailed placement for 16 nm FinFET process,” Design, Automation and Test in Europe , 6-pages, 2014.

[7] Y.l. Li, et al., “NCTUcell : A DDA-Aware Cell Library Generator for FinFET Structure with Implicitly Adjustable Grid Map,” Design Automation Conference, 6-pages, 2019.

[8] Y.L. Li, et al., “NCTUcell: A DDA-and Delay-Aware Cell Library Generator for FinFET Structure With Implicitly Adjustable Grid Map,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol.41, no.12, pp.5568–5581, 2022.

[9] S. Nishizawa and S. Kimura, “Standard Cell Structure and Transistor Reordering for Mitigating Area Penalty in Double Diffusion Break FinFET Process,” International VLSI Symposium on Technology, Systems and Applications, 4-pages, 2024.

[10] L.T. Clark, et al., “ASAP7: A 7-nm finFET predictive process design kit,” Microelectronics Journal, vol.53, pp.105–115, 2016.

[11] N.H.E. Weste and D.M. Harris, CMOS VLSI Design, 4 ed., Addison Wesley, 2010.

[12] C. Lazzari, et al., “A new transistor-level layout generation strategy for static CMOS circuits,” International Conference on Electronics, Circuits, and Systems, pp.660–663, 2006.

[13] S. Yan, et al., “A novel methodology of layout design by applying Euler path,” International Conference on Solid-State and Integrated Circuit Technology, pp.818–820, 2010.

[14] A. Ziesemer and C. Lazzar, “Transistor Level Automatic Layout Generator for non-Complementary CMOS Cells,” International Conference on Very Large Scale Integration, pp.116–121, 2007.

[15] T. Iizuka, M. Ikeda, and K. Asada, “Exact Minimum-Width Transistor Placement for Dual and Non-dual CMOS Cells,” IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol.E88-A, no.12, pp.3485–3491, 2005.

[16] H. Mair, et al., “A 10nm FinFET 2.8GHz Tri-Gear Deca-Core CPU Complex with Optimized Power-Delivery Network for Mobile SoC Performance,” International Solid-State Circuits Conference, pp.56–58, 2017.

[17] S. Nishizawa, et al., “Supplemental PDK for ASAP7 Using Synopsys Flow,” IPSJ Transactions on System LSI Design Methodology, vol.14, pp.24–26, 2021.

[18] “OpenCores.” *https://opencores.org*

Shinichi Nishizawa received the B.E. degree from Ritsumeikan University, Japan, in 2009, and Ph.D degree from Kyoto University, Japan, in 2015. He is an Assistant Professor in Waseda University.

Shinji Kimura received the B.E., M.E., and Dr.Eng. degrees in information science from Kyoto University, Kyoto, Japan, in 1982, 1984, and 1989, respectively. He has been an Assistant Professor with Kobe University since 1985, has been an Associate Professor with the Nara Institute of Science and Technology since 1993, and has been a Professor with Waseda University since 2002. He was a Visiting Scientist with Carnegie Mellon University from 1989 to 1990 and was a Visiting Scholar with Stanford University from 2000 to 2001. He is interested in the formal and timing verification of logic circuits, the hardware/software co-design methodologies, reconfigurable hardware, and the low-power design. He is a member of the Information Processing Society of Japan and the IEEE Computer Society. He has served an Executive Committee Member of ICCAD 2011 and 2012 and a General Chair of ASP-DAC 2013. He has been a fellow of IEICE since 2015.