Chapter 3

Hierarchical Extension for Large Cell Layout Synthesis

3.1 Introduction

This chapter describes a hierarchical extension of the layout synthesis method explained in Chapter 2 for layout synthesis of large dual CMOS cells. The proposed method partitions a given transistor-level netlist into blocks considering the transistor connections by diffusions and places all transistors hierarchically. Intra-block placement uses an exact transistor placement method which is proposed in Chapter 2. During intra-block transistor placement, a new optimization cost is introduced to maximize the number of the connections by diffusion sharing between logic blocks in the subsequent inter-block placement step. This new cost function is a key to realize a hierarchical transistor placement and the drastic runtime reduction with little increase in cell width. Intra-cell routability of the generated transistor placement is also checked in the same manner as in Chapter 2 in order to generate a routable cell layout.

The layout styles of the hierarchical cell layout synthesis method are defined in Section 3.2. Section 3.3 and Section 3.4 explain the formulation of the hierarchical transistor placement and intra-cell routings. In Section 3.6, experimental results of the proposed cell layout synthesis are presented. Finally, Section 3.7 summarizes this chapter.

3.2 Layout Styles

Our cell layout styles for the hierarchical cell layout synthesis are described in Table 3.1 and illustrated in Figure 3.1. These styles are basically same as the styles used in Chapter 2. Particularly the layout styles No. 5 and 10 in Table 3.1, the alignment style of the multiple-

Table 3.1 Our layout styles of the hierarchical SAT-based cell layout synthesis for dual CMOS cells.

1. Static dual CMOS logic circuits.
2. Transistors are drawn up in two horizontal rows. The upper row is for P type transistors and the bottom row is for N type transistors.
3. Two transistors which have the same gate input signals are vertically aligned.
4. Two transistors which have the same diffusion terminals are placed in the neighboring columns to share their diffusions.
5. The top of the P diffusions are aligned to Top-Region and the bottom of the N diffusions are aligned to Bottom-Region.
6. The intra-cell routing uses polysilicon and first metal layers.
7. All nets which connect diffusion terminals of P type transistors are in P-region.
8. All nets which connect diffusion terminals of N type transistors are in N-region.
9. All nets which connect GATE terminals are in G-, Top- or Bottom-regions.
10. Gate terminals are connected by the polysilicon layer in Top- or Bottom-region, and by the first metal or polysilicon layer in G-region.
11. The same signals in P-region and N-region are connected by the vertical first metal through G-region at the top of N-region and the bottom of P-region.
12. Vertical first metals and the GATE terminals are connected by the horizontal first metals in G-region.
13. VDD are connected from the top of P-diffusion through Top-region by the vertical first metal.
14. GND are connected from the bottom of N-diffusion through Bottom-region by the vertical first metal.
15. Single contact is assumed to be enough to connect between metal and diffusion or polysilicon.
16. No dogleg is used.

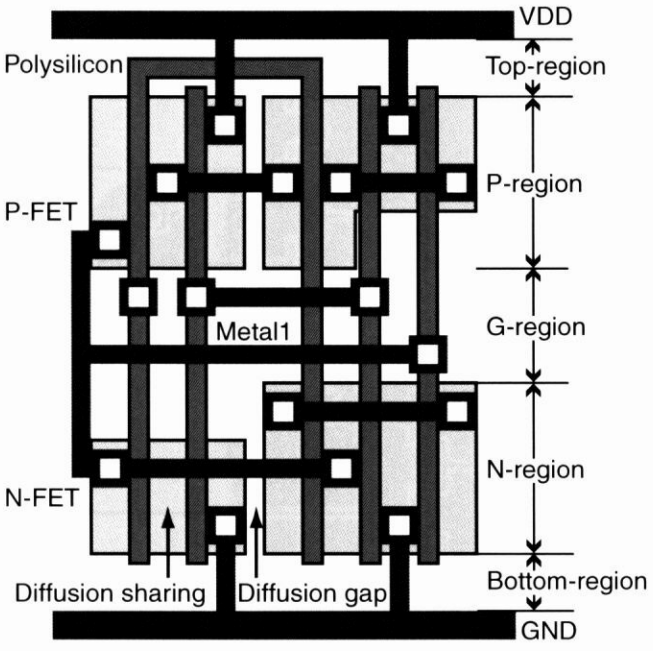

Figure 3.1 An illustration of the layout styles of the hierarchical SAT-based cell layout synthesis for dual CMOS cells.

sized P and N type transistors and the usage of polysilicon layer inside the G-region are different from the styles in Chapter 2. Although these styles are changed to improve the intra-cell routability, they do not have a significant effect on the results of the cell layout synthesis. Under these layout styles, the proposed method synthesizes the layout of the dual CMOS logic cells hierarchically.

3.3 Hierarchical Transistor Placement

In this section, we explain the formulation of the hierarchical transistor placement. The proposed hierarchical transistor placement method has three steps, partitioning, intra-block placement, and inter-block placement. The proposed method partitions a given transistor-level netlist into blocks considering the transistor connections by diffusions, then place all transistors inside each block in minimum width via Boolean satisfiability. After all the intra-block transistor placements are finished, all the blocks are placed in minimum-width in the same manner as the intra-block transistor placement. Detailed explanation of each step is described as follows.

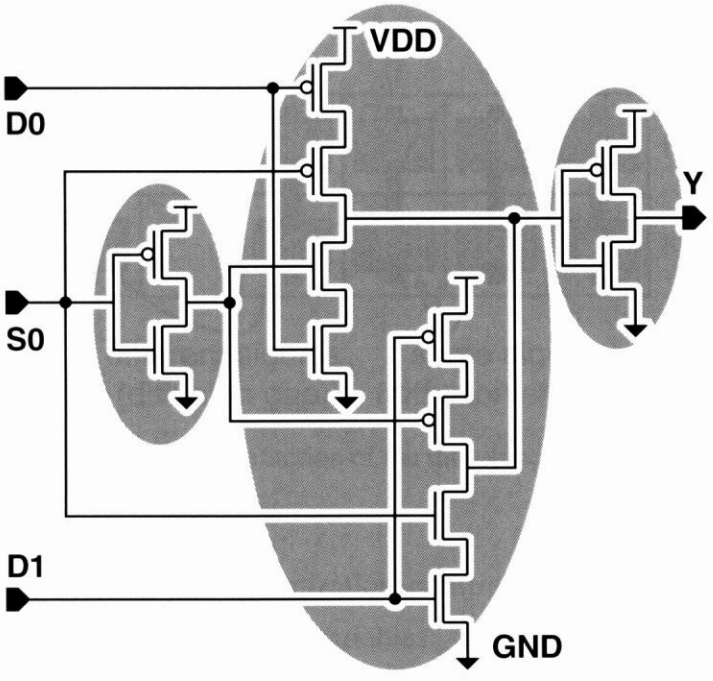

Figure 3.2 Schematic of 2-input multiplexer and its logic block partitioning.

3.3.1 Partitioning into Logic Blocks

Our approach first partitions the given circuit into logic blocks. Conventionally, several hierarchical layout synthesis methods for runtime reduction have been proposed[28, 29]. The proposed method partitions a given transistor network into logic blocks in the same manner as these methods[28, 29]. A logic block consists of transistors that are connected together by their diffusions except power and ground. An example of the partitioning result of 2-input multiplexer is shown in Figure 3.2. Two clocked inverters whose outputs are connected are identified as one logic block using our partitioning procedure. Since it is important to maximize the diffusion sharing between transistors in order to minimize the placement width, the partitioning procedure based on the diffusion connections is not expected to worse the results severely.

3.3.2 Intra-Block Transistor Placement Formulation

Next, we explain the intra-block transistor placement procedure. The proposed method uses Boolean satisfiability to generate the minimum-width intra-block transistor placement. In this chapter, we use Pseudo-Boolean formulation[30] to formulate the hierarchical cell layout synthesis. Pseudo-Boolean formulation can handle Conjunctive Normal Form (CNF),

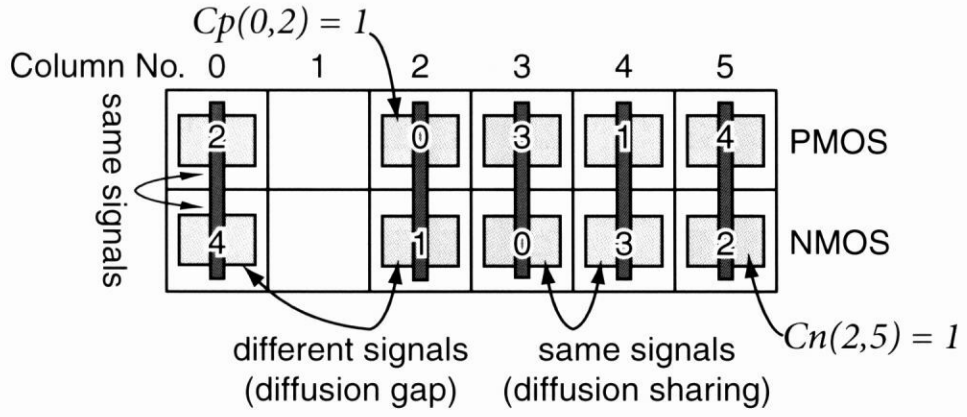

Figure 3.3 The problem definition of the intra-block transistor placement.

Pseudo-Boolean Form (PBF), and optimization constraints[30]. PBF constraints are expressed by linear inequalities of Boolean variables. Since this intra-block transistor placement formulation requires an optimization constraint, we select this formulation.

When N N type transistors and N P type transistors are given, this problem is defined as the problem to place these $2N$ transistors in the minimum area. This problem can be transformed into the problem that places all transistors using the minimum number of columns as illustrated in Figure 3.3. As explained in Section 3.2, the P type transistors are aligned in the upper row and the N type transistors are in the bottom. Two neighboring transistors must face the diffusions which belong to the same signals each other to connect their source or drain by diffusion sharing. The empty columns result in the diffusion gaps in the final layout. These transistor placement constraints are same as those explained in Chapter 2. Under these placement constraints, we search for a possible placement with minimum number of columns of the placement area.

When the intra-block transistor placement is formulated, a new optimization cost is introduced to maximize the number of the connections by diffusion sharing between logic blocks in the subsequent inter-block placement step. The diffusions which can be shared between logic blocks are only power and ground in our partitioning procedure. For maximizing the number of the connections by diffusion sharing in the inter-block placement step, it is better to place the diffusions which belong to VDD and GND on the edge of the placement of each block as illustrated in Figure 3.4. Therefore, we introduce new variables E_l and E_r, and a new optimization function into the formulation of transistor placement to maximize the number of the connections by diffusion sharing. E_l (E_r) takes the value of 1 unless the diffusions of

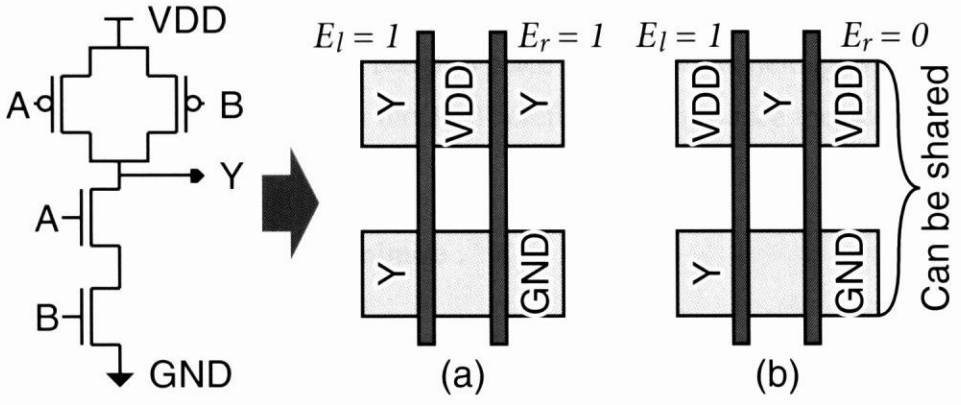

Figure 3.4 Additional variables introduced to maximize the number of the connections by diffusion sharing between logic blocks.

Table 3.2 The variables used for the intra-block transistor placement formulation.

name	*number*	*condition which makes the value 1*
$C_n(i,k)$	$N \times W$	N-FET i is placed in the column k.
$C_p(i,k)$	$N \times W$	P-FET i is placed in the column k.
$F_n(i)$	N	N-FET i is flipped.
$F_p(i)$	N	P-FET i is flipped.
E_r, E_l	2	The diffusions on the right/left edge of the intra-block placement do not belong to VDD and GND.

the left (right) edge are assigned to the pair of VDD and GND as illustrated in Figure 3.4. We explain the details of the placement formulation in the following paragraph.

In our formulation of intra-block transistor placement, $W + 1$ Boolean variables are introduced for each transistor to identify its location and whether it is flipped or not, where W is the number of the columns of the placement area. In addition, 2 Boolean variables are required to identify whether the diffusions of left and right edges of the intra-block placement is a pair of VDD and GND signals or not. All variables required for this formulation are listed in Table 3.2. Table 3.2 shows the names of the variable type, the numbers of each type of variables, and the conditions when each type of variables takes the value of 1. Some example cases of such conditions are also illustrated in Figure 3.3. These variables are used to formulate the transistor placement constraints as Conjunctive Normal Form (CNF) and Pseudo-Boolean Form (PBF) constraints. We formulate the following Boolean constraints which express all the possible transistor placements in W columns under our layout style.

Objective function: For maximizing the number of the connections by diffusion sharing in the inter-block placement step, it is better to place diffusions that belong to VDD and GND on the edge of the placement of each block. Therefore, we introduce the objective function as follows.

$$\text{Minimize}: \quad E_r + E_l \tag{3.1}$$

Transistor overlap constraint: N (P) type transistors must not overlap in the same column. This constraint is expressed by the following PBF constraints.

$$\sum_{i=0}^{N-1} C_n(i,k) \leq 1, \quad 0 \leq k < W \tag{3.2}$$

$$\sum_{i=0}^{N-1} C_p(i,k) \leq 1, \quad 0 \leq k < W \tag{3.3}$$

Transistor instantiation constraint: Each transistor must be instantiated once. The following PBF constraints express this constraint.

$$\sum_{k=0}^{W-1} C_n(i,k) = 1, \quad 0 \leq i < N \tag{3.4}$$

$$\sum_{k=0}^{W-1} C_p(i,k) = 1, \quad 0 \leq i < N \tag{3.5}$$

Different gate constraint: P and N type transistors which have different gate input signals must not be placed in the same column. This constraint is expressed as following PBF constraints.

$$C_n(i,k) + C_p(j,k) \leq 1, \quad 0 \leq k < W \tag{3.6}$$

where i and j meet following condition.

$$0 \leq i < N, \quad 0 \leq j < N, \quad GATE_n(i) \neq GATE_p(j) \tag{3.7}$$

where $GATE_n(i)$ and $GATE_p(j)$ means that the gate input signal of N type transistor i and P type transistor j, respectively.

Neighboring transistors constraint: Two transistors facing the diffusions which belong to the different signals each other can not be placed in the neighboring columns. For example, N type transistors 4 and 1 in Figure 3.3 can not be placed in the neighboring columns. This

constraint is expressed by the following logic equation for N type transistors.

$$GAP_n(i,j) \wedge \bigvee_{k=0}^{W-2} (C_n(i,k) \wedge C_n(j,k+1)) = 0 \qquad (3.8)$$
$$i \neq j, \quad 0 \leq i < N, \quad 0 \leq j < N$$

Here, $GAP_n(i, j)$ is the logic function of $F_n(i)$ and $F_n(j)$, and takes the value of 1 if N type transistor i can not share its diffusion with N type transistor j placed to its immediate right, otherwise 0. This logic equation is expressed as CNF. The same constraint is also expressed as follows for P type transistors.

$$GAP_p(i,j) \wedge \bigvee_{k=0}^{W-2} (C_p(i,k) \wedge C_p(j,k+1)) = 0 \qquad (3.9)$$
$$i \neq j, \quad 0 \leq i < Y, \quad 0 \leq j < Y$$

Finally, we can determine whether given transistors can be placed in W columns or not using these constraints. If no satisfiable assignment is found by the Boolean solver, it is guaranteed that there is no possible placement of W columns. Therefore, we can find an exact minimum-width transistor placement using the procedure described below.

1. For a given transistor netlist, count the number of N and P type transistors. The initial number of column W is set to the number of N type transistors(=#P-FET).

2. Search for a satisfiable assignment for the Boolean constraints constructed for W columns. If a satisfiable assignment is found, these transistors can be placed in W columns and this procedure terminates. Otherwise, go to step 3.

3. $W = W + 1$. Go to step 2 again.

3.3.3 Inter-Block Placement Formulation

After intra-block placement of all the blocks are finished, these blocks are placed in the minimum area. In the proposed method, inter-block placement is also based on Boolean Satisfiability. The inter-block placement problem is formulated as Pseudo-Boolean formulation in the same manner as the intra-block placement. The problem definition of the inter-block placement is shown in Figure 3.5. In the formulation of the inter-block placement, each logic block is placed in one column and two logic blocks must not overlap in the same column. Two logic blocks facing the diffusions that can not be shared each other must not be placed on the neighboring column, as illustrated in Figure 3.5.

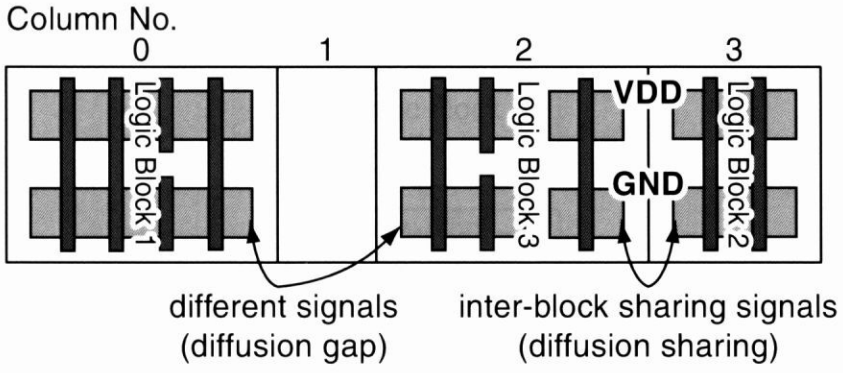

Figure 3.5 The problem definition of the inter-block placement.

These constraints are expressed as CNF and PBF constraints in the same manner as explained in Section 3.3.2. Therefore, we obtain an exact minimum-width block placement using the following procedure similar to that in Section 3.3.2.

1. For a given partitioned block list, count the number of logic blocks. The initial number of column W is set to the number of the blocks.

2. Search for a satisfiable assignment for the Boolean constraints constructed for W columns. If a satisfiable assignment is found, these blocks can be placed in W columns and this procedure terminates. Otherwise, go to step 3.

3. $W = W + 1$. Go to step 2 again.

3.4 Intra-Cell Routing

The routability check of the intra-cell wiring is formulated in the same manner as explained in Chapter 2. Since the routability check consumes extremely less time than the transistor placement procedure, the routability check is executed flatly to the generated transistor placement in the proposed method.

3.5 Overall Flow

Figure 3.6 shows the overall cell layout synthesis flow of the proposed method. In this layout synthesis flow, the routability check is executed once for each block after intra-block transistor placement. Therefore, all blocks are guaranteed to be routed individually before inter-block placement step. After inter-block placement, the routability is checked again

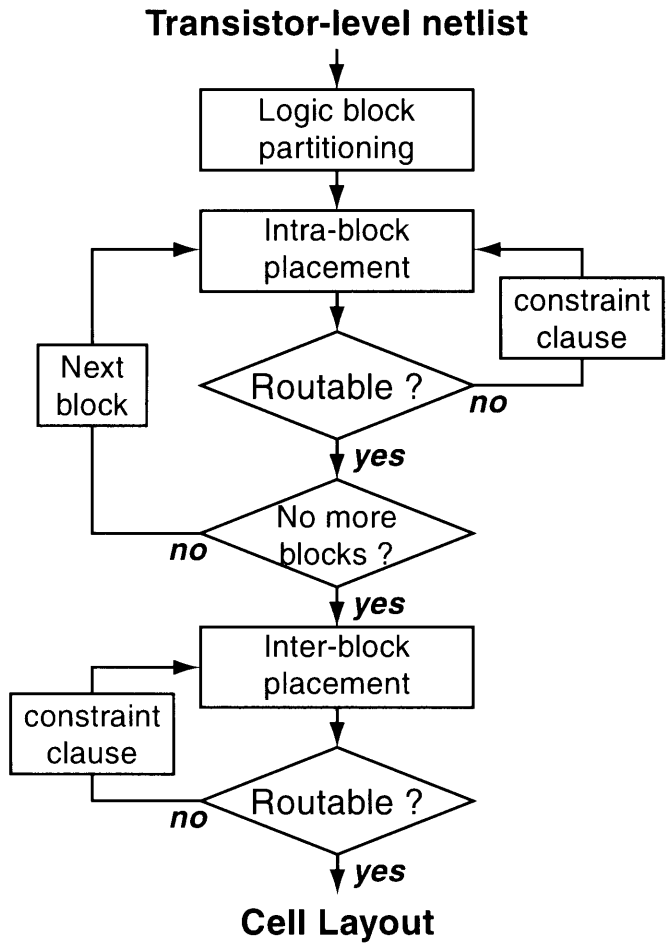

Figure 3.6 Our hierarchical SAT-based cell layout synthesis flow.

for whole transistor placement iteratively until a routable block placement is found. If a placement is found to be unroutable in the routability check point, a constraint clause is added to the set of placement constraints in order not to generate this unroutable placement again.

3.6 Experimental Results

The proposed hierarchical cell layout synthesis method is implemented to show its effectiveness. In this experiment, we used Pseudo-Boolean Solver, *PBS*[30] to solve formulated satisfiability problems. *PBS* is a complete Boolean solver which can handle CNF, PBF, and optimization constraints.

3.6.1 Transistor Placement

First, the comparison results between the proposed hierarchical transistor placement method and the original flat placement method explained in Chapter 2 are shown. In the case of only transistor placement generation, all the routability check in the flow shown in Figure 3.6 is not executed. Table 3.3 summarizes the comparison results. This table shows the circuit name, the number of transistors inside each circuit, the number of columns of the generated placement, and runtime. *Flat* in this table indicates the original flat method and *Hierarchical* indicates the proposed hierarchical method. Although this table shows the results of only 5 circuits, the experiment is conducted for 30 circuits. The row Total(30 circ.) shows the total of 30 circuits. The results show that all the placements generated by the proposed method have the same width as those generated by the flat method. The runtimes are drastically reduced by the hierarchical method particularly in the case of large cells such as fad1 (full adder).

3.6.2 Cell Layout Synthesis

Next, we compared the proposed cell layout synthesis method with a commercial cell layout synthesis tool *ProGenesis*[13] and the original flat synthesis method explained in Chapter 2. In this comparison, the commercial tool only generates the symbolic layout without layout compaction. The comparison results are shown in Table 3.4. This table shows the circuit name, the number of columns of the generated placement, and runtime comparison. *Commercial*, *Flat*, and *Hier.* indicate the commercial tool, the original flat synthesis method, and the proposed hierarchical method, respectively. Although this table also shows the results of only 5 circuits, the comparison is conducted for 30 circuits.

Compared with the original flat method, the proposed method generates 1 column larger cell layout only for mux2 circuit. On the other hand, the runtime of the proposed method is drastically reduced compared with the flat method. These results show that the proposed hierarchical cell layout synthesis method realizes a drastic runtime reduction with little increase in cell width. Figure 3.7 illustrates a layout of a buffered full adder cell generated by the proposed method. This cell has 40 transistors and can not be solved by the original flat method in practical runtime, whereas the proposed method can solve it less than 1 minute.

Cell width comparison with the commercial tool shows that the total cell width of the proposed method is increased about 4%. The reason of this cell width increase is the layout style restriction that only the P and N type transistors which have the same gate input signal

Table 3.3 Comparison results between the proposed hierarchical transistor placement and the original flat method proposed in Chapter 2.

Circuits		*Width (#column)*		*Runtime (sec.)*	
name	*#transistors*	*Flat*	*Hierarchical*	*Flat*	*Hierarchical*
aoi21	6	3	3	0.02	0.03
mux2	12	8	8	0.23	0.13
ao33	16	9	9	0.25	0.06
oa44	20	11	11	0.69	0.04
fad1	28	15	15	201.05	0.28
Total(30 circuits)	—	174	174	205.86	2.49
Ratio	—	1.000	1.000	1.000	0.012

Table 3.4 Comparison results with the commercial cell generation tool and the original flat method proposed in Chapter 2.

Circuits	*Width (#column)*			*Runtime (sec.)*		
name	*Commercial*	*Flat*	*Hier.*	*Commercial*	*Flat*	*Hier.*
aoi21	3	3	3	18.78	0.04	0.06
mux2	*6*	8	**9**	24.23	6.42	1.45
ao33	*9*	10	10	28.12	392.26	2.57
oa44	*11*	12	12	32.00	116.41	4.60
fad1	15	15	15	73.06	305.76	1.24
Total(30 circ.)	172	178	179	789.45	1907.78	19.93
Ratio	1.000	1.035	1.041	1.000	2.416	0.025

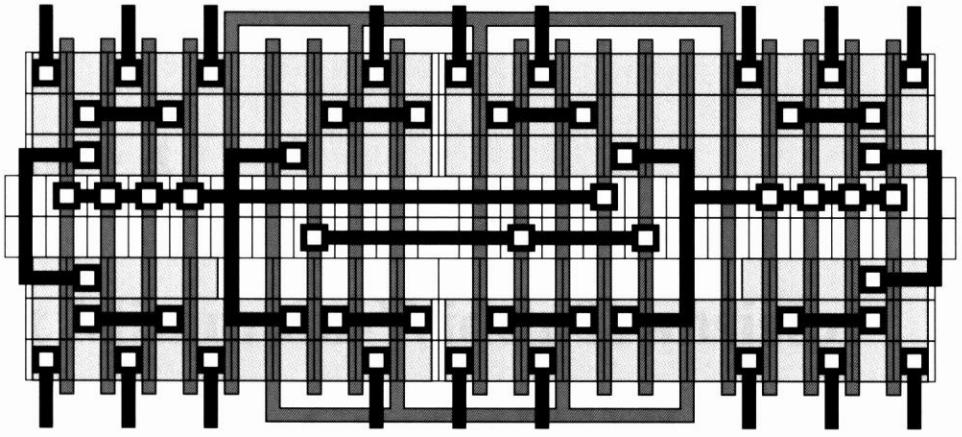

Figure 3.7 A layout of buffered full adder generated by the proposed hierarchical SAT-based cell layout synthesis method.

can be placed in the same column in the proposed method, whereas the commercial tool does not have such restriction. The runtime of the proposed method, however, is only about 3% of that of the commercial tool. These results show that the proposed method can generate the cell layout much more quickly with a little width increase compared with the commercial tool.

3.7 Summary

This chapter proposed a hierarchical layout synthesis method for large dual CMOS cells via Boolean Satisfiability. Experimental results showed that the proposed hierarchical method generates the same width placement as the exact flat method explained in Chapter 2 and drastically reduces the runtime for transistor placement. The comparison results of the cell layout synthesis for 30 benchmark circuits showed that the proposed method generates the same width layout as the flat method except one circuit. The comparison results with the commercial cell generation tool without cell layout compaction showed that the total cell width of the proposed method is increased about 4% due to the layout style restriction, whereas the runtime is only about 3% of that of the commercial tool. From these results, we can conclude that the proposed method can be used as a quick layout generator in the area of transistor-level circuit optimization such as on-demand cell layout synthesis.