Power and Area Optimization by Reorganizing CMOS Complex Gate Circuits

M. Tachibana, S. Kurosawa, R. Nojima, N. Kojima, M. Yamada, T. Mitsuhashi, N. Goto

Research Laboratory 1, ULSI Research Laboratories, R & D Center, TOSHIBA Corp.

Abstract

This paper proposes a method for achieving low-power control-logic modules using a combination of CMOS complex gate reorganization, transistor size optimization, and transistor layout. Complex gate reorganization minimizes transistor count and net count without changing the functionality of the circuit. Transistor sizing and layout are interdependent, the optimization of one results in the optimization of the other. The authors applied the reorganization method to a 2400-transistor circuit, and succeeded in reducing the transistor count by 12%, and the net count by 13%. Transistor sizing and layout compaction reduced the average transistor size by one eighth, while the same delay was maintained. Power dissipation was cut to less than half, even when wiring capacitances were dominant.

1 Introduction

A number of transistor size optimizing methods under path delay constraints have been proposed to optimize the power and the area of digital circuits [4]-[6]. These proposed methods minimize power consumption and layout area by minimizing the sum of transistor gate widths under path delay constraints. However, even though many transistor sizing methods are available, estimating the transistor load capacitance is an essential factor for optimization, because in the actual layout, the load capacitance of the transistors may vary by a factor of 15 and more.

We previously reported [9] the development of a new transistor sizing method which combined transistor sizing with the extraction of actual load capacitance from laid-out circuits; after layout compaction with rip-up and rerouting, satisfactory result was achieved. We optimized digital circuits with about 10,000 transistors designed for SOG Gate-Array, and achieved a reduction of approx. one eighth of the total transistor gate width and approx. 54% of the wire capacitance. Also, it was found that most of the transistor gate widths were reduced to the minimum size set by the design rule.

The results of that study suggest that only few transistors are on the critical signal path, and most transistors are laid on paths which have ample room for timing constraints. Further improvements of layout and compaction were required, however, to achieve higher layout density; we felt that another method was needed which could be used to reorganize circuits with fewer transistors and nets by modifying those circuits, because the lower limits of the layout area needed for the circuit are often determined by the number of nets which connect the transistors.

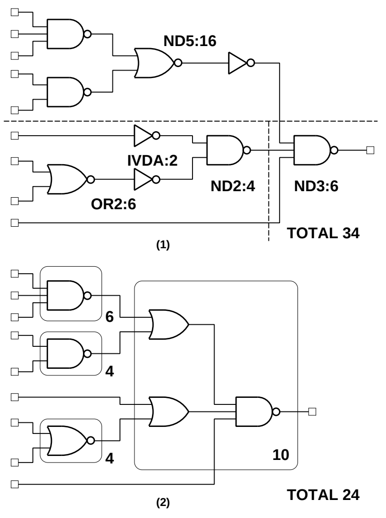

Figure 1: Example of reorganization.

Therefore, we developed a new method for reorganizing the CMOS complex gate circuits by modifying the circuits without causing functional changes, and evaluated its effectiveness. This method, which reorganizes circuits by complex gates thus satisfying the physical constraints such as the maximum number of series-connected transistors, minimizes the number of transistors and nets because only few types of complex gates are registered in the conventional gate library. In fig.1.(1), an example of gate level circuit is shown. The dotted lines are the module boundaries. According to the cell library, the number of transistors needed for the circuit is 34, however, the reorganization of the circuit and minimization of transistors reduced the count to only 24 transistors as shown in fig.1.(2).

If the above assumptions regarding transistor sizing are valid, the area and load capacitance should be reduced as the number of transistors and nets is decreased. Consequently, power consumption should also be reduced. The assumptions regarding transistor sizing are therefore valid; the reorganization of the circuit does not change its functionality, because this method does not affect most of the transistors that are on non-critical signal paths, and the transistors laid on critical signal paths are expected to be as large as those in a circuit that is not reorganized.

The number of realizable logics using CMOS complex gates exceeds 3500 [1], even if we restrict the number of series-connected transistors to 4, and only a few of then are registered in the conventional library. Most complex gates are not registered in the gate library because:

1. It is virtually impossible to design every type of complex gate. It is necessary to design various cells from a gate with different drivability, and maintenance costs are also a consideration.

2. Only few types of complex gates are used frequently. This means that most complex gates are rarely used.

There have been discussions about the size of the cell library [2],[3]. However, large complex gates are difficult to register as part of the library because of the many types and frequency of use. For this reason, the number of complex gates registered in the library does not normally exceed order of 10.

This is due to the fact that there is no transistor sizing after layout. However, if transistor sizing after layout is assumed, the situation changes because only the connection of the transistors is needed to realize a large complex gate. In other words, the size of the transistors in the complex gate which is generated in the reorganization process is optimized by transistor sizing, therefore, the initial size of the transistor should be such as to satisfy the delay constraints. It then becomes unnecessary to prepare all possible complex gates with different drivabilities, while it becomes possible to generate transistor level connections on the line because the algorithm for generating a transistor level circuit is less complex than the algorithm for searching for a cell in a library that contains all possible complex gates.

When the above method was applied to actual circuits, the reorganization process reduced the transistor count by approximately 12%, and the net count by approximately 13%. In addition, transistor sizing and layout compaction following rip-up and rerouting resulted in a reduction of approximately 6% of the layout area and the circuit capacitance, compared with before the reorganization.

In section 2, a brief overview of the system is given, followed by an outline of transistor sizing optimization and layout compaction after rip-up and rerouting. The reorganization by CMOS complex gates is discussed in section 3. The experimental results are in section 4. Section 5 contains the concluding remarks.

2 Layout and transistor sizing

This section provides a brief description of the system, and an outline of transistor sizing optimization and layout compaction after rip-up and rerouting.

The input of module optimization is the nested net lists of gates, and the output is the transistor layout pattern which is optimized under delay constraints. The flow is as follows.

First, the input net lists are reorganized, and the transistor connections are generated. Then, using large transistors, the initial layout is compacted. Next, the actual node capacitance is extracted from the layout; the node capacitance is the sum of the wire capacitance, transistor gate capacitance, and transistor junction capacitance. The size of the transistors is then optimized based on the extracted capacitance; as the initial transistors were large, their size can be reduced. These new transistor sizes are fed back to the layout. Next, wires which were detoured in the initial layout are ripped-up and rerouted, utilizing the space made available by the reduction of transistor size. After rerouting, the compaction yields a denser layout.

Because the transistors load capacitances extracted from the new layout are smaller than those of the initial layout, the above procedures are iterated until no significant improvement in transistor sizing is observed. The procedure for transistor sizing optimization is as follows: In the circuit shown in fig.2, the delay time when transistor $x1$ switches is derived from $(R1 + R2) * C0 + R2 * C1$, using the Elmore formula.

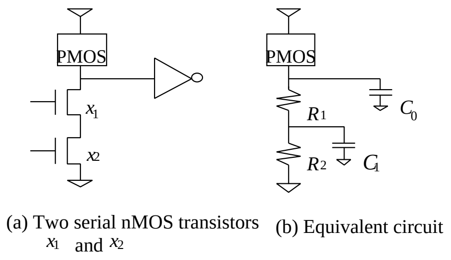

Figure 2: Example of CMOS gate.

In this formula, $R1$ and $R2$ are the transistor gate widths of the charge/discharge path, and $C0$ and $C1$ are the actual wire capacitance and load transistor's gate capacitance. Therefore, the total signal transition delay along a signal path is the sum of the delay time of each transistor on the path. Further, differentiation of the total delay in the above by the transistor gate width gives the sensitivity of the transistor width to total delay. Transistor size is optimized using this sensitivity. Transistor sizing is a well-defined problem [4]-[6] and is essentially a power reduction problem.

Next, the layout is compacted after rip-up and rerouting. The transistor size optimization described above yields more room for wiring, therefore, it is possible to reduce wire length by compacting the layout.

To ensure that wire length will not increase, however, nets are ripped-up one at a time, and are rerouted using a minimum wire length algorithm such as a maze router or a class of line-expansion router [7]. The rip-up and rerouting process always reduces wire length. If the wiring route before ripping up is the shortest path, the new wiring path will pass through the same route, resulting in the same wire length. If there is another route shorter than the original one, the new wiring path will pass through the shorter route, resulting in a shorter wire length. In both cases, a zero-increase in wire length is guaranteed. A simple example of rerouting and compaction is shown in fig.3. In this example, some detours are eliminated and the wire path passes through a new shorter route. After compaction, a denser layout with reduced wire length is obtained.

At the compaction stage, a zero-increase wire length is also necessary. If, for example, the standard constraint graph method [8] is applied, some wire segment may be extended during the packing phase. Maximum distance constraints between objects connected with a wiring segment should be introduced, to ensure zero-increase wire length.

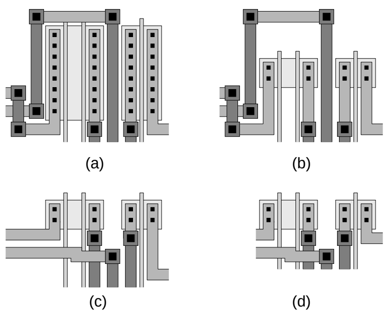

Figure 3: Example for re-routing and compaction.

3 Reorganization process

In this section, the reorganizing process for generating the transistor-level circuit is explained. First, an overview is given of the process according to the flowchart shown in fig.4; then, details are given of the complex gate generation.

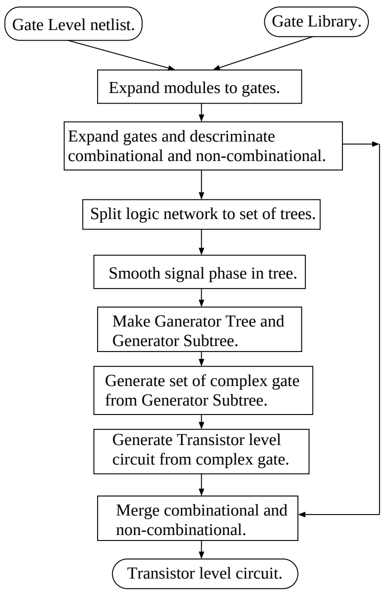
Gate Level netlist.
Gate Library.
Expand modules to gates.
Expand gates and descriminate combinational and non-combinational.
Split logic network to set of trees.
Smooth signal phase in tree.
Make Ganerator Tree and Generator Subtree.
Generate set of complex gate from Generator Subtree.
Generate Transistor level circuit from complex gate.
Merge combinational and non-combinational.
Transistor level circuit.

Figure 4: Flow of complex gate reorganization.

The input is the nested gate level net list and the gate library. The gate library describes the gate by logic level such as And, Or, and Not, if possible, or by transistor level connection if the gate is not a combinational logic such as F/F, latch, or 3-state logic.

First, the input nested gate level net list is expanded to the gate level, then it is discriminated into combinational logic and non-combinational logic according to the library. The non-combinational logic part of the net list is immediately expanded to the transistor connection level and merged with circuits generated from the logic part of the net lists at the end of the process. The combinational logic part is expanded to the And, Or, Not logic network according to the library.

Below, the logic network is treated as a graph in which the node is defined by logic and the (hyper) edge is defined by net. In this system, no signal loop is assumed in the logic network. The graph is split at the edges which connect more than 3 nodes into sets of trees. This process breaks some logic such as exclusive-Or into trees. Because the split tree, which is composed of And, Or, and Not logic, has inconsistent signal phases, Signal phases of the tree are smoothed to change the whole tree to Negative Gate so that Not logics are attached only at root or leaves of the tree. The main step of the reorganizing process is to split the tree into minimum cost sets of complex gates. The complex gate is a Negative Gate and is a subtree of the tree, if needed, with different signal phases. To split the tree into sets of subtrees, first, a Generator Tree is built from the tree, then, a Generator Subtree is built from the Generator Tree to represent sets of subtrees. The reorganized set of complex gates is the minimum cost set of subtrees generated from this Generator Subtree. The cost of the complex gate, in this system, is the number of transistors required to make the complex gate.

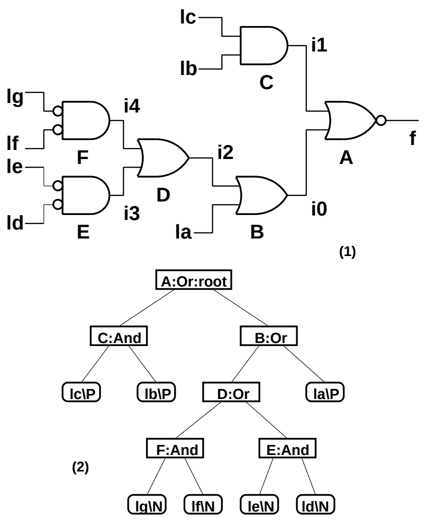

Figure 5: Example of generator tree.

Next, for each complex gate, transistor level circuits are generated and merged with transistor level circuits of the non-logic part.

Fig.3 (1) shows an example of a smoothed signal phase tree. Because this tree is a Negative Gate, if physical constraints are satisfied, the tree can be reorganized as one complex gate and some inverters (4 in this example).

However, such combination of complex gate and inverters often breaks the physical constraints. Even if the physical constraints were satisfied, there is no guarantee that the combination would achieve minimum cost. Therefore, it is necessary to split the tree into complex gates to satisfy physical constraints and to achieve minimum cost. To guarantee that a set of split complex gates and inverters will yield minimum cost, it is necessary to generate all subtrees from the tree, because the complex gate corresponds to the split tree, that is, the subtree.

The process for generating all subtrees from the tree and the data structure for the process are as follows: First, a Generator Tree is built as shown in fig.3 (2) from the tree in fig.3 (1). In fig.3 (2), "\N" and "\P" attached to the leaves of the Generator Tree indicate whether the signal phase is negative or positive. The leaf with the negative logic signal corresponds to the leaf with the Not logic.

Assuming the subtrees of the tree shown in fig.3 (2), there is only one subtree rooted in F. However, there are 4 subtrees rooted in D, that is, {D}, {D,F}, {D,E}, and {D,E,F}. And, the subtrees rooted in B include the subtrees with the B node only, and all the subtrees rooted in B which include the node D; in other words, all subtrees rooted in D are connected to node B (i.e. {B}, {B,D}, {B,D,F}, {B,D,E}, {B,D,E,F}). In general, if node A has two children B and C, the sets of subtrees rooted in A consist of the following 4 types of subtree:

1. Subtree with the A node only.
2. All subtrees rooted in B and connected with A.
3. All subtrees rooted in C and connected with A.
4. All subtrees rooted in B and in C and connected with A.

Therefore, to effectively generate all sets of subtrees, it is necessary to traverse each node in the tree and to generate the subtrees rooted in the nodes in a bottom-up fashion. Because subtrees rooted in a node include subtrees rooted in the child of that node, it is only necessary to connect the subtree root node and to connect the subtrees rooted in the children of the node to generate all the subtrees of that node.

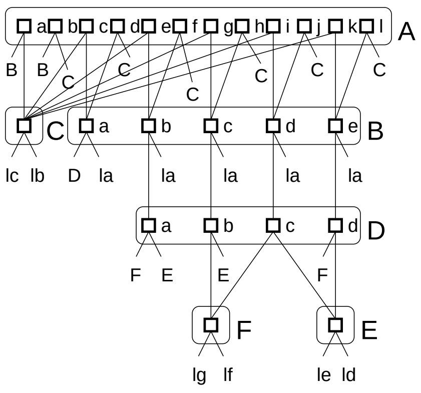

Figure 6: Example of generator subtree.

The Generator Subtree is a data structure in which a list connected to each node of the Generator Tree holds a set of root nodes of subtrees rooted in the node. Fig.3 shows an example of a Generator Subtree for the Generator Tree in fig.3 (2). In this example, there are 5 subtrees, from B.a to B.e, rooted in node B, and each subtree includes the following nodes: B.a:{B}, B.b:{B,D}, B.c:{B,D,F}, B.d:{B.D.E}, and B.e:{B,D,E,F}. All subtrees of the Generator Tree are generated from the above Generator Subtree, then, complex gates are generated from each subtree, and the minimum cost set of complex gates which make up the Generator Tree is achieved.

The cost of the complex gate is represented by the number of transistors required to make the complex gate, which is equal to the number of inputs multiplied by two. It is not necessary to actually generate the complex gate. The physical constraints are posed by the upper limit of the number of transistors series-connected from the power supply (Vss and Vdd) to the output of the complex gate. Simple Depth First Search from the root of the tree to the leaves by the sum of And nodes and the sum of Or nodes gives the maximum number of transistors series-connected without actually generating the complex gate. Therefore, it is possible to determine the cost of the complex gate and whether the complex gate satisfies physical constraints, without generating the transistor level circuit; this minimizes the actual generation of transistor level circuits.

The generation of a transistor level circuit is carried out as follows. First of all, no complex gates can be generated at those nodes in the list attached to the Generator Subtree which do not satisfy the physical constraints. A complex gate which includes a node that violates the physical constraints would also violate those constraints, therefore, the complex gate generation process would be terminated if such nodes were used.

When the root of the subtree is an intermediate node of the Generator Tree, two types of complex gate, a positive and a negative logic output, should be established corresponding to one subtree, because at the time the complex gate is generated, it is not known whether the parent complex gate requires a positive or negative logic signal. Also, these two types of complex gates are the minimum cost complex gates of the subtree rooted in the node. Therefore, two minimum cost complex gates, a positive and a negative logic output, are generated first. Then, by attaching an inverter at the root of these two complex gates, two new implementations are generated and compared, and the lower cost one is selected.

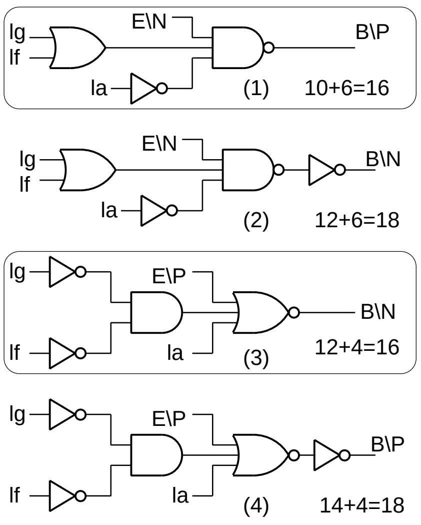

Figure 7: Example of generated complex gates.

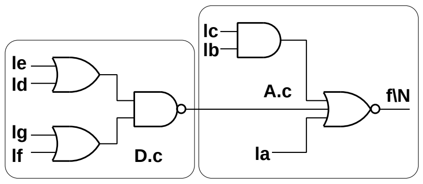

Figure 8: Minimum cost set of complex gates.

In fig.3 (1) to 3 (4), 4 types of complex gate are shown, which were generated from the B.c node of the Generator Subtree in fig.3.

From these complex gates, it is clear that the minimum cost of the positive output is 6 and the minimum cost of the negative output is 4. Therefore, from these 4 complex gate, (1) is selected for the positive output, and (3) is selected for negative output. When the input terminal of a complex gate is not a leaf of the Generator Subtree, that is, it is connected to the root of another subtree, the cost of the complex gate is calculated as if cost of the subtree were attached at the input terminal. Using this cost calculation method and the bottom-up generation of complex gates during the Generator Subtree traverse, selection of the minimum cost subtree generated at the root of the Generator Subtree is guaranteed to generate the minimum cost set of complex gates.

	Positive		Negative			Negative	
F	4	*4	6	*6	A.a	6	18
E	4	*4	6	*6	A.b	4	22
D.a	6	14	4	12	A.c	8	*16
D.b	6	12	8	14	A.d	6	20
D.c	8	*8	10	*10	A.e	10	18
D.d	6	12	8	14	A.f	8	22
C	6	*6	4	*4	A.g	16	20
B.a	6	14	4	*12	A.h	14	24
B.b	8	16	6	14	A.i	22	22
B.c	10	16	12	16	A.j	16	20
B.d	12	*12	14	14	A.k	16	20
B.e	10	16	12	16	A.l	14	24

Table 1: Cost table of complex gate.

Table 1 shows the cost of the positive/negative output for each node in the Generator Subtree in fig. 3. Because nodes from A.a to A.l are roots of the Generator Subtree, there is no need to make a positive output logic. In Table 1, columns designated as Positive and Negative show the positive/negative output cost, respectively. The left half of each column shows the cost of the complex gate itself and the right half shows the cumulative cost. The figures marked with ”*” are the minimum cost nodes. From this table it can be seen that the minimum cost set of complex gates which make up the Generator Tree in 3 is $\{$A.c, C$\}$, as illustrated in fig.3.

Next, we will explain the generation of transistor level circuits from the complex gates generated above. We explained only the N-channel part of the CMOS complex gate, but for the P-channel, the process is the same as the exchanging of the And and Or nodes.

Looking at the interior And nodes and Or nodes of the complex gate, it is clear that at the And node, circuits corresponding to the node children should be connected in series, while at the Or node, circuits corresponding to the node children should be connected in parallel. Therefore, the transistor level circuits of the complex gate are generated by traversing each node in the complex gate in the Depth First Search, saving the stack transistor at the terminal, and connecting a number of partial circuits on the stack corresponding to the fan-in of the node, in series for the And node, and in parallel for the Or node; then, saving on stack the result, until the root is visited.

The vertical position of the partial circuits connected in series is determined as follows: First, the maximum number of parallel-connected transistors, the maximum number of series-connected transistors, and the transistor count of each partial circuit are determined; then, the partial circuits are sorted by the maximum number of parallel-connected transistors. If there are two or more partial circuits having the same maximum number of parallel-connected transistors, these are sorted by transistor count. If there are two or more partial circuits with the same transistor count, they are sorted by the maximum number of series-connected transistors. The sorting places the partial circuit with the largest value near the power supply line. This process does not precisely connect the partial circuits according to their complexity, but it achieves good results in spite of its simplicity.

4 Experimental results

	Transistor Count			Net Count
	Non-Comb.	Comb.	Total	
w.o.	1200	1244	2444	1146
w.	——	956	2156	1001
w./w.o.	1.000	0.786	0.882	0.873

Table 2: Result of reorganization (1).

	Tree	INV	TG	CPX
Count	94	78	26	95
ave.	10.17	——	——	7.87
min.	2	——	——	4
max.	62	——	——	26

Table 3: Result of reorganization (2).

We evaluated a new method using a logic circuit containing approx. 2400 transistors. The results of the reorganization process are shown in Table 2 and Table 3. Table 2 shows the transistor count and net count following the reorganization; transistor size optimization and compaction after rip-up and rerouting are shown in Table 4. Because our reorganization method is applicable only to the logic part of the circuit, the transistor count is shown divided into the combinational part and non-combinational part. This table shows that following the reorganization process, transistor count was reduced by approximately 12% in the entire circuit, and by approximately 23% in the logic part of the circuit; the net count was reduced by approximately 12%. The fact that the reduction rate of the net count and transistor count are close, indicates that the nets reduced by the reorganization are those connected between the reduced transistors.

Table 3 shows the details of the reorganized logic gates. In this table, the Tree column contains the number of trees into which the logic network was split; INV and TG are the number of inverters and transfer gates generated, respectively; the CPX column shows the number of complex gates generated. In the CPX column it is shown that, on the average, one complex gate consists of approximately 7.9 transistors. If we consider that the average transistor count of a complex gate registered in a conventional cell library is 6, there is a mismatch between the cell library and the actual circuit. The table also shows that the maximum transistor count of a generated complex gate is 26. It is well known that it is difficult to register complex gates of this size in the cell library, therefore, the advantage offered by our reorganization method which is able to generate large complex gates is amply demonstrated.

	Total Tr. Size		Total Capacitance	
	w.o.	w.	w.o.	w.
Initial	195520	172480	251.40	224.70
1st Sizing	19552	17736	57.08	54.26
Compaction & 2nd Sizing	19552	17248	57.08	53.90

Table 4: Result for a 2444-transistor circuit.

In Table 4, the results of two layouts, without and with reorganization, are compared. The top row of Table 4 shows the results of the initial layout. In the initial layout, Gate-Array size transistors are used as large transistors. The middle row of the table shows the results of transistor sizing under delay constraints. In this stage, compaction after rip-up and rerouting is not yet done. The bottom row of Table 4 shows the results of the second transistor sizing after rip-up and rerouting, and compaction.

Using our reorganization method, total transistor size after the second sizing was reduced by approximately 12%. This value closely matches the transistors count reduction, indicating that reorganization did not change the size of most transistors. This suggests that the assumptions regarding transistor sizing, that is, there are only a few transistors on the critical signal path, and most transistors are laid on paths which have ample room for timing constraints, are valid. Because the total layout area is not reduced as much as the transistor count or net count, the reduction rate of the total circuit capacitance remains approximately 6%.

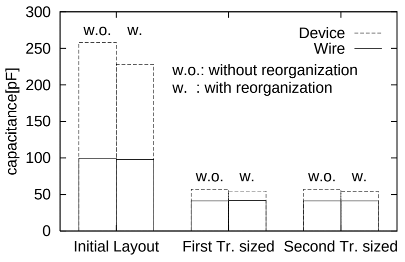

Figure 9: Circuit, wire and device capacitance.

Fig. 9 shows the total circuit capacitance, which is strongly related to power dissipation. When transistor size is reduced, only the device capacitance (transistor's gate capacitance and source/drain junction capacitance – both are proportional to transistor size) is reduced. After rip-up and rerouting and compaction, wire capacitance is also reduced, and as a result, device capacitance is further reduced.

5 Conclusion

In this paper, a method for optimizing power and area based on the reorganization of a CMOS complex gate is presented. This method is especially effective when combined with transistor sizing and layout compaction after rip-up and rerouting to optimize a module originally designed as a semi-custom LSI.

For a 2444-transistor circuit, our method reduced transistor count by approximately 12%, and the net count by approximately 13%. Power consumption was reduced to less than half that of the original circuit. When compared with the results of transistor sizing without reorganization, power consumption was reduced by nearly the same percentage as the reduction achieved in transistor count. These results could not be achieved by the optimization of the cell library or by stand-alone transistor sizing itself without layout compaction.

References

[0] Detjens, E., Gannot, G., Rundell, R., Sangiovanni-Vincentelli, A., "Technology Mapping in MIS" in *Proc. 1987 ICCAD*, pp.116-119, Nov. 1987.

[0] Scott, K., Keutzer, K., "Impact of Library Size and the Quality of Automated Synthesis", in *Proc. 1987 ICCAD*, pp.120-123, Nov. 1987.

[0] Lega, M. C., "Mapping Properties of Multi-level Logic Synthesis", in *Proc. 1988 ICCD*, pp.257-261, Oct. 1988.

[0] Fishburn, J. P., Dunlop, A. E., "TILOS: A posynomial programming approach to transistor sizing", in *Proc. 1985 ICCAD*, pp.326-328, Nov. 1985.

[0] Shyu, J., Fishburn, J. P., Dunlop, A. E., Sangiovanni-Vincentelli, A. L., "Optimization-based transistor sizing", *IEEE J. of SSC*, vol. 23, no. 2, pp.400-409, Apr. 1988.

[0] Sapatnekar, S. S., Rao, V. B., Vaidya, P. M., Kang, S. M. "An exact solution to the transistor sizing problem for CMOS circuit using convex optimization", *IEEE Trans. on CAD*, vol. 12, no. 11, pp.1621- 1634, Nov. 1993.

[0] Ohtsuki, T., "Maze-Running and Line-Search Algorithms", *Layout Design and Verification*, pp.99-131, North-Holland,Amsterdam,1986.

[0] Wolf, W. H., Dunlop, A. E., "Symbolic Layout and Compaction," B. Preas and M. Lorenzatti (ed.), *Physical Design Automation of VLSI Systems*,
pp.211-281, The Benjamin/Cummings Publishing Company, Inc. 1988.

[0] Yamada, M., Kurosawa, S., Nojima, R., Kojima, N., Mitsuhashi, T., Goto, N., "Synergistic Power / Area Optimization with Transistor Sizing and Wire Length Minimization", in *Proc. Symposium on Low Power Electronics*, Oct. 1994.