Journal of Zhejiang University SCIENCE A
ISSN 1673-565X (Print); ISSN 1862-1775 (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

A semi-custom design methodology for design performance optimization*

Dong-ming LV[†], Pei-yong ZHANG[†‡], Dan-dan ZHENG, Xiao-lang YAN, Bo ZHANG, Li QUAN

(*Institute of VLSI Design, Zhejiang University, Hangzhou 310027, China*)

[†]E-mail: lvdm@vlsi.zju.edu.cn; zhangpy@vlsi.zju.edu.cn

Received Aug. 24, 2007; revision accepted Nov. 8, 2007

Abstract: We present a semi-custom design methodology based on transistor tuning to optimize the design performance. Compared with other transistor tuning approaches, our tuning process takes the cross-talk effect into account and prominently reduces the complexity for circuit simulation and analysis by decomposing the circuit network utilizing graph theory. Furthermore, the incremental placement and routing for the corresponding transistor tuning in conventional approaches is not required in our methodology, which might induce timing graph variation and additional iterations for design convergence. This methodology combines the flexible automated circuit tuning and physical design tools to provide more opportunities for design optimization throughout the design cycle.

Key words: Transistor tuning, Cross-talk, Circuit decomposing
doi:10.1631/jzus.A071449 **Document code:** A **CLC number:** TN402

INTRODUCTION

Semi-custom design, which combines the advantages of custom optimization and automated design with a diversity of commercial EDA tools in existence, is becoming more and more significant in modern IC design. Among the custom design methodologies, transistor tuning is an effective and convenient solution adopted by some mature semi-custom design flows, most of which focus on the area-delay tradeoffs problem (Fishburn and Dunlop, 1985; Kao *et al*., 1985; Yu *et al*., 2003; Santos *et al*., 2005b; Kabbani *et al*., 2005; Lu *et al*., 2005). There has been a large amount of work done on the transistor tuning including all kinds of transistor delay models (Fishburn and Dunlop, 1985; Kao *et al*., 1985; Ketkar *et al*., 2000; Santos *et al*., 2005a; Yelamarthi and Chen, 2007) and various algorithms from TILOS (Fishburn and Dunlop, 1985) based on greedy strategy proposed in the 1980s to a series of gain-based circuit tuning methodologies discussed recently (Kabbani *et al*., 2005).

A majority of the transistor tuning approaches ignore the cross-talk effect in the formulation of the transistor timing model or the process of sizing the transistor width, so as to increase the cross-talk delta delay on wires while reducing the propagation delay on cells. Thus in our methodology, we take the cross-talk effect into account according to a local circuit SPICE extraction for simulation and an expression for the cross-talk delay calculation to balance the cell delay and the wire delay for timing path optimization.

Some transistor tuning approaches proposed recently have taken the cross-talk effect into consideration (Vittal *et al*., 1999) by formulating the path delay as a function of the extracted resistance and capacitance connected along the path. The kernel of such method is to get the minimal function output under a series of constraints of timing, area, power, etc. Obviously, these methods either take much runtime for simulating the RC network or require many

‡ Corresponding author
* Project (No. 2005AA1Z1271) supported by the Hi-Tech Research and Development Program (863) of China

complex calculations to obtain an appropriate width for a certain transistor. For simplifying these methods, our methodology decomposes a certain circuit network into small components based on graph theory and abandons the transistor tuning on those which have “strong connections”. Here, a strong connection means all the cells in a certain component, which show a strong correlation with each other, are on propagating the delay. For each of the non-strongly connected components, we evaluate the delay decrease potential for every path in it, and then choose that one with the maximum weight evaluated, and finally tune the transistors belonging to that chosen path to realize the delay optimization.

For a pair of complementary transistors with their gate region connected to the same pin of a certain cell (for convenience, we define such a pair of transistors as a transistor cluster) fabricated in complementary CMOS style, the ratio of P-type transistor gate width divided by N-type transistor gate width directly determines the transistor rise and fall delay which might be distinctly different owing to the differences in the load capacitances. Any difference in the rise and fall delay contributes to the total path delay difference, thus reducing the maximum clock speed (Talukdar and Sridhar, 1996). Our methodology concentrates on the P/N ratio adjustment for a balanced rise and fall delay. The P/N ratio is restricted to a range of discrete values with approximate 10% increase between two adjacent ratios. During the ratio tuning process, we keep the sum of the width of a transistor cluster fixed, and calculate the P and N transistor width respectively according to the discrete ratio value selected in the range. Since the total width is maintained, the area increase is under control. The area-delay tradeoff is thus not involved in our approach so as to decrease the complexity further.

Note that some semi-custom design flows based on P/N ratio tuning aim to provide a huge library for the synthesis tools, which contains all kinds of cells with the transistor clusters tuned in different discrete P/N ratios between the maximum and minimum values (Northrop and Lu, 2001). It is a discrete optimization problem targeted to select appropriate P/N ratio cells from the underlying cell library. Its complexity sharply increases with the number of cells in the library. Hence there will be an exhausted waiting for the synthesis (both the logical and physical) result. Take an embedded system processor based on RISC architecture with about 100 instructions for instance, the logical synthesis using Synopsys Physical Compiler tool and utilizing a library approximately containing 3000 cells lasts about 3 days with 8 CPUs and 2 G memory available. For modern IC design, it usually takes several or even tens of design step iterations to accomplish the design requirements and get an optimum circuit layout. From this perspective, the total design cycle will be intolerable. So in our methodology, the library is composed of no more than 300 cells and kept fixed. Our circuit tuning aims at optimization on the layout circuit but not fabricating different P/N ratio cells to expand the library for providing more opportunities for cell selection during synthesis, thus the total design cycle almost has no increase.

SEMI-CUSTOM DESIGN FLOW

The complete design flow of our semi-custom design methodology is summarized in Fig.1. The physical design is initialized through the logical and physical synthesis step that is a primitive representation for the common IC design process with all the necessary steps included, such as floorplan, placement, route and so on. The kernel of this procedure, which starts from the critical networks extracting and ends with the static timing analysis on the entire circuit, is iterated after re-identifying the critical network subset based on the STA (static timing analysis) result.

Details of each step in the flow from Fig.1 are cataloged in the following.

Vertically located channel CMOS standard cell library

In our approaches, the standard cells are also semi-customized. We use the Kazam tool, which provides an automated solution for creating custom standard cell layouts, to export cells following our design rule and architecture defined. For a compact standard cell construction, the transistor gate channels could be located in both the vertical and horizontal directions. But our standard cell is fabricated with all of its channels located vertically in a complementary CMOS style. And each transistor cluster is grouped

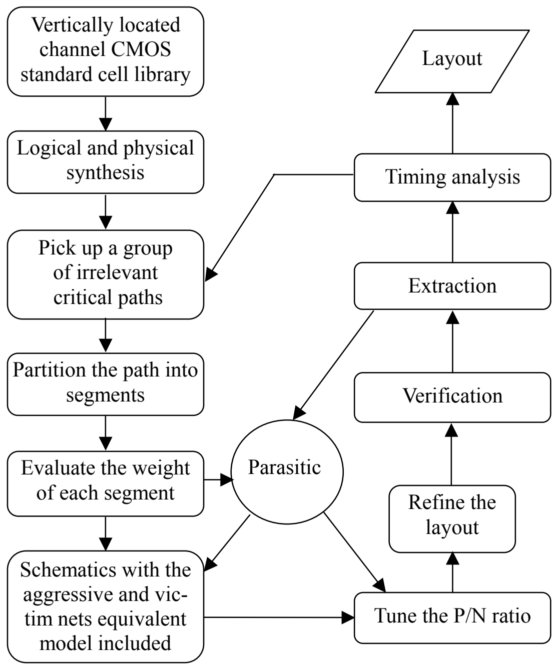
Vertically located channel CMOS standard cell library
Logical and physical synthesis
Pick up a group of irrelevant critical paths
Partition the path into segments
Evaluate the weight of each segment
Schematics with the aggressive and victim nets equivalent model included
Parasitic
Layout
Timing analysis
Extraction
Verification
Refine the layout
Tune the P/N ratio

Fig.1 Complete semi-custom design flow

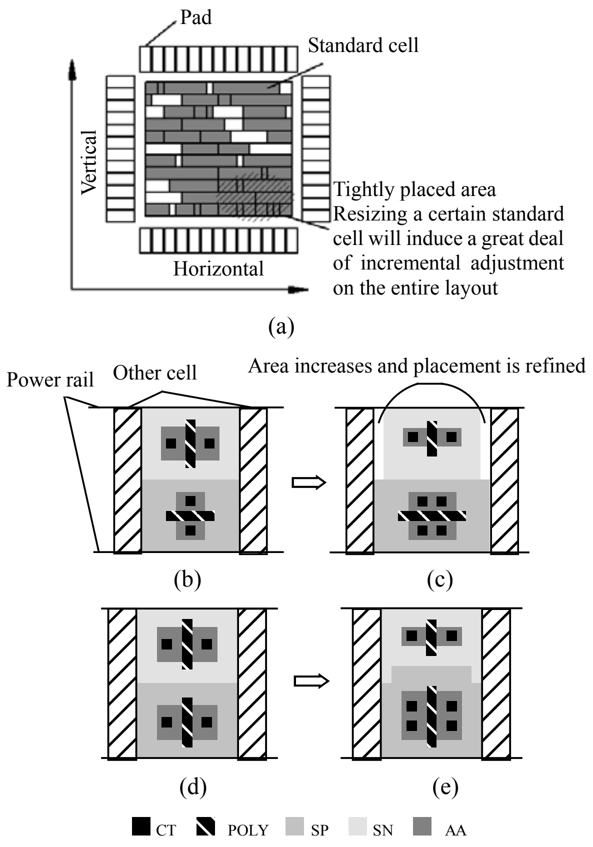
Pad
Standard cell
Vertical
Horizontal
Tightly placed area
Resizing a certain standard cell will induce a great deal of incremental adjustment on the entire layout
(a)
Power rail
Other cell
Area increases and placement is refined
(b)
(c)
(d)
(e)
CT
POLY
SP
SN
AA

Fig.2 Different design layout architectures. (a) A common design layout; (b) Architecture with the N channel located horizontally; (c) Layout evolved from (b) after a certain P/N ratio tuning; (d) Architecture with all the channels located vertically; (e) Layout evolved from (d) with the same P/N ratio and channel size as (c)

together with its N and P channels aligned in the vertical direction. Apparently the cell of this architecture needs a bit more area than the common ones. However, owing to such architecture, the P/N ratio could be tuned by extending or shrinking the active area and jogging the well with no area increase in the horizontal direction which may arouse a series of consequent variations on the circuit as illustrated in Fig.2. Thereby, no incremental placement or routing is needed in our methodology. Note that the vertical dimension of the cell is fixed and the horizontal power rail is abstracted to a straight line in Fig.2 for easy observation.

Our standard cell library contains no more than 300 cells, which could be classified into approximate 30 categories with the adjacent power levels set around 1.5. This type of library provides a considerably fast synthesis, speeds up the design cycle and offers more opportunities to raise circuit performance by custom design.

Critical networks graph

Cell connection that starts from a certain flip-flop and ends with another is defined as a timing path in the synchronous digital circuit. So in most cases, a timing path is truncated by a pair of flip-flops and could be expanded into a corresponding isolated network by tracing through the inputs and outputs of each cell along the path till reaching the terminal, i.e., flip-flops. In our methodology, we find out the critical timing path by STA, and then expand it into a corresponding network defined as critical network. Note that the coupling nets with the coupling capacitance above C_c are also taken into account in the process of expansion. The primary goal of this expansion is to tune the P/N ratio on an overview of the entire network, hence avoiding offsetting the optimization on a local part of the network by the delay deterioration on the related parts. Afterwards the critical network is mapped to a directed graph as follows: each cell is mapped to a node in the graph; each connection between two cells is mapped to an edge connecting the corresponding nodes directed from one cell's output to the other's input, particularly for coupling nets, there exists an edge directed from the aggressive net to the victim net correspondingly; each node is marked with two weights separated by the slash representing the rise delay and the fall delay through the cell, respectively. This transform helps to comprehend the circuit structure more intuitively and prepares for decomposing the network into compo- nents to accelerate the circuit simulation and analysis. A critical network graph example is illustrated in Fig.3.

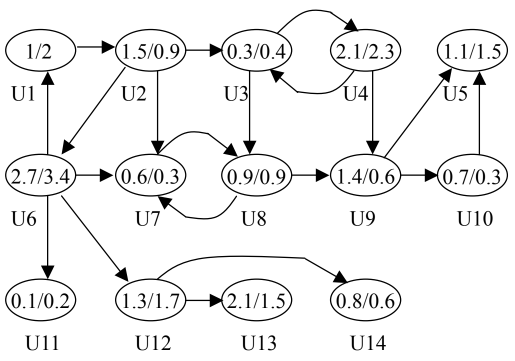

Fig.3 A critical network graph example

Partitioning the network into components

This subsection shows how to decompose the critical network graph into components. This decomposing step brings two primary benefits for the P/N ratio tuning. First, it downsizes the circuit scale, consequently speeds up the spice netlist simulation. Second, the strongly connected components, which are complicated and unnecessary for the tuning, are excluded from the subsequent design steps so as to reduce the complexity remarkably.

A directed graph consists of strongly connected and non-strongly connected components.

Definition 1 A strongly connected component (SCC) of a directed graph $\boldsymbol{G}=(V, \boldsymbol{E})$ is a maximal set of vertices $C\subseteq V$ such that for every pair of vertices u and v in C, there is a directed path from u to v and a directed path from v to u; that is, vertices u and v are reachable from each other.

Definition 2 A non-strongly connected component (nSCC) is a maximal subgraph of a directed graph $\boldsymbol{G}=(V, \boldsymbol{E})$ such that for every pair of vertices u, v in the subgraph, there is an undirected path from u to v and a directed path from v to u.

Corresponding to the circuit, the SCC indicates that for every two cells A and B in it, A's timing closely depends on that of B and vice versa. Concretely, a delay variation on either cell will induce a consequent delay variation on the other, but in an opposite way. Thereby the total delay on such component is almost unalterable or hard to be improved. Thus the SCC is eliminated from the graph $\boldsymbol{G}$ for P/N ratio tuning.

The algorithm to find SCCs of a graph $\boldsymbol{G}$ uses the transpose of $\boldsymbol{G}$ defined as $\boldsymbol{G}^{\mathrm{T}}=(V, \boldsymbol{E}^{\mathrm{T}})$, where $\boldsymbol{E}^{\mathrm{T}}$ consists of the edges of $\boldsymbol{G}$ with their directions reversed. The completed algorithm is described below (Cormen *et al*., 1990):

Component Decomposing Algorithm (*G*)

```
(1) Depth-first search on $G$ to compute the finishing time $f[u]$ for each vertex $u$.
(2) Transform $G$ to $G^{T}$.
(3) Depth-first search on $G^{T}$, but in the main loop of the search, consider the vertices in order of decreasing $f[u]$ as computed in (1).
(4) Output the vertices of each tree in the depth-first forest formed in (3) as a separate SCC.
```

The running time of the above algorithm is about $O(V+\boldsymbol{E})$. Take Fig.3 for instance, the graph decomposing output is illustrated in Fig.4. The SCCs are displayed in shades while the rest are nSCCs.

We delete the SCCs and the related edges from the graph, hence deriving a collection of acyclic directed subgraphs, nSCCs, which is considerably less complicated than before for circuit analysis. The simplified graph is illustrated in Fig.5.

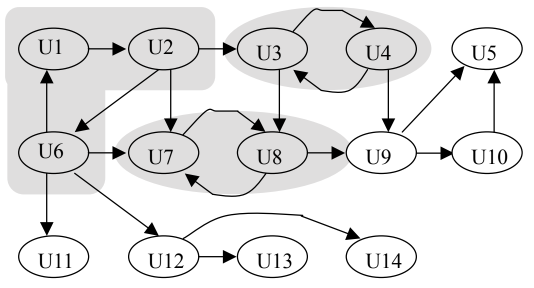

Fig.4 Graph decomposing based on Fig.3

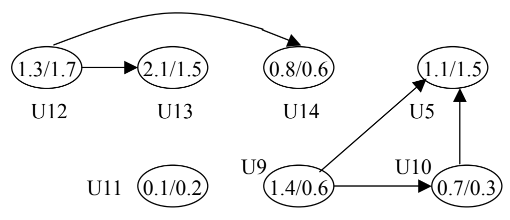

Fig.5 Simplified graph evolved from Fig.4

Path selection

As mentioned before, the nSCC is an acyclic directed subgraph. Thereby there exists at least one root node, i.e., the node with zero input degree, and one leaf node, i.e., the node with zero output degree. In this phase, we evaluate the weight of every path starting from a root and ended with a leaf in each nSCC, then respectively choose the maximum weighted one for P/N ratio tuning. The weight function for such a path $p_{i\to j}$ is formulated as

$$W_{i\to j}=\text{abs}\left(\ln\frac{r_i+r_{i+1}+\ldots+r_{i+n}+r_{i+n+1}+\ldots+r_j}{f_i+f_{i+1}+\ldots+f_{i+n}+f_{i+n+1}+\ldots+f_j}\right),\quad(1)$$

where i represents a root node, j represents a leaf node, r_k and f_k represent the pair of values marked in a node $v_k\in p_{i\to j}$, which are respectively equal to the rise and fall delay of the corresponding cell.

The path selection algorithm for a certain nSCC C is described below:

```
Path Selection (C)
    Find out all the roots and leaves in $C$;
    $p_{\text{select}}=\varnothing$;
    For each root $i$
        For each leaf $j$
            search $C$, find out all the paths $p_{i\to j}$;
            evaluate the weight of $p_{i\to j}$, pick out the maximum
                weighted one $p_{i\to j,\max}$. If its weight is above $W_{\min}$,
                then $p_{\text{select}}=p_{i\to j,\max}$;
            export $p_{\text{select}}$;
        End For
    End For
```

The above algorithm is applied on each nSCC, thus a set of p_{select}'s is derived from the critical network finally.

Spice netlist and P/N ratio tuning

After the path for P/N ratio tuning is chosen from each nSCC, it is remapped to the initial critical network, that is to say, the cells and nets corresponding to each element (node and edge) in the selected path are picked out from the critical network, thereby composing a circuit connection. Subsequently, for the cell, we translate it into its transistor level spice description, and for the nets, we extract the equivalent RC network with the coupling RC network excluded, which will be investigated individually. Thus, we derive a spice netlist from the path in the simplified graph.

The kernel of the P/N ratio tuning is to simulate the circuit for every tiny delta change on the P/N ratio of a transistor cluster. Therefore, on one hand, from the implementation perspective, the time consumed by simulation arises dramatically with the coupling RC network involved. On the other hand, from the algorithm perspective, the conventional Miller factor based methodology for noise analysis calculates the cross-talk delay by growing or shrinking the timing window of every net through an iteration of static timing analysis. It is extremely complicated to calculate the timing window for each relevant net every time a certain P/N ratio varies. Due to the above disadvantages, in our methodology we take the coupling RC network apart from the total network to form a "clean" spice netlist for simulation and add a revisory cross-talk delay value using the cross-talk delay model (Vittal *et al*., 1999; Chen *et al*., 2002) to the "clean" spice model simulation result for offsetting the cross-talk effect. For a victim RC tree coupled to an arbitrary number of aggressors, the revisory value at any node o is calculated based on the topology of the coupled RC network as follows:

$$ct\_delay=K\cdot\frac{\sum_{R_i\in P(o)}X_iR_i}{vdd\left(\sum_{C_i\in C}C_iR_{ii}\right)}\cdot slew(vic).\quad(2)$$

Here, K is a parameter related to design technology, X_i is the sum of downstream coupling capacitances seen from node o, R_i is the upstream resistance along the path, $P(o)$ is the union of the victim driver resistance and the set of resistances in the unique path from the root to the node o, C is the set of all capacitors and R_{ii} is the resistance seen across C_i with all capacitors open. $slew(vic)$ represents the slew rate of victim nets.

Fig.6 shows the variation of cross-talk delay with coupling length as predicted by Eq.(2) and HSPICE. We present experimental results which verify the accuracy and fidelity of Eq.(2) for cross-talk delay. For a pair of coupling nets in a certain test circuit, we obtain the cross-talk delay utilizing a commercial extraction tool followed by HSPICE simulation using an industrial extraction flow for a 0.18-μm CMOS technology. We compare these results with the cross-talk delay Eq.(2). Experimental results of a wide range of test circuits show that the average error is about 10%.

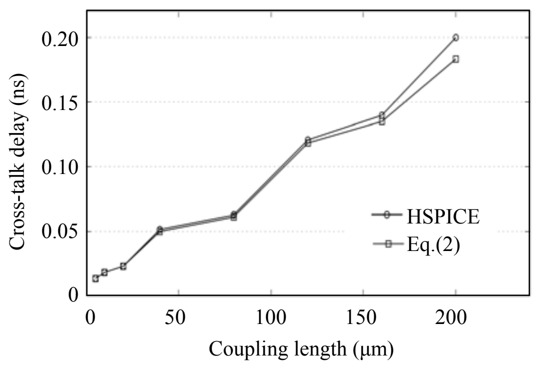

Fig.6 Variation of cross-talk delay with coupling length

The complete P/N ratio tuning process for a certain spice netlist (*S*) is described below:

P/N Ratio Tuning (*S*)

```
(1) For each transistor cluster $tc_i \in S$, get its corresponding $rise\_delay_i$ and $fall\_delay_i$ by simulation, and then choose $tc_{\text{select}}$ with $rf_{\text{select}}=\max(|rise\_delay_i-fall\_delay_i|)$;
(2) If $rf_{\text{select}}<c$, quit; else continue;
(3) Calculate the P/N ratio $r_{\text{select}}$ of $tc_{\text{select}}$;
      if $rise\_delay_{\text{select}}<fall\_delay_{\text{select}}$;
         $tag=-1$;
      else $tag=+1$;
(4) While the total delay of $S$ decreases
      $r_{\text{select}}=tag\times delta+r_{\text{select}}$;
      if $r_{\text{select}}$ is in the range of P/N ratio
         resize the P, N transistors according to $r_{\text{select}}$ while keeping the sum of P and N transistor width fixed;
         simulate $S$, get $delay_{\text{simulate}}$;
         calculate the cross-talk delay $delay_{\text{ct}}$ according to Eq.(2);
         total delay=$delay_{\text{simulate}}+delay_{\text{ct}}$;
      else break;
(5) Back to (1).
```

When the above process converges, we obtain a set of new transistor sizes. We modify the layout for the corresponding transistor resizing by utilizing a SKILL program. The SKILL language is developed by Cadence to be used with their tool suites. It allows the user to write a “script” to perform any layout modification in Cadence.

For verifying the above procedure, experimental results on several circuits in the ISCAS89 benchmark suite are presented in Table 1. For comparison, we implement another widely used EDA tool—Synopsys Amps—for transistor tuning procedure where the cross-talk effect is not involved during the tuning procedure. We observe that the clock cycle improvement by Amps (11.64% on average) is generally smaller than that produced by our methodology (19.71% on average), and that Amps induces more area increase (17.41% on average) than our method does, since there is no area increase in our method as described above. In addition, the tuning procedure utilizing Amps results in a completely different design timing graph, so as to require another 1~6 design iteration(s) for design convergence with all the necessary design steps involved. From this perspective, our method gets rid of a series of complex ECO adjustments through the design flow for the corresponding transistor tuning. The area-delay product produced by either of the two approaches is illustrated in Fig.7. For easy observation, we zoom in the graph on s5378~s298 as illustrated in the subgraph. Obviously, for most of the circuit, the area-delay product of our approach is smaller than that of Amps, about 20.6% on average.

Table 1 Our methodology vs. Synopsys Amps tool

Circuits in ISCAS89	Cells	Initial cycle (ns)	Initial area (μm^2)	Our method optimization (ns)	Amps (ns)	Cycle decrease (%)		Area after Amps (μm^2)	Area increase (Amps) (%)
						Ours	Amps		
s1196	547	4.04	6908.9	2.68	3.50	−33.66	−13.30	7228.4	4.62
s1238	526	4.02	7038.7	2.78	3.48	−30.87	−13.34	4784.6	3.74
s13207	8620	10.55	107 070.2	8.37	8.96	−20.63	−15.12	101 568.0	−5.14
s1494	653	7.39	9051.1	6.23	6.91	−15.64	−6.50	8857.6	−2.14
s15850	10 369	13.91	119 747.0	12.44	13.02	−10.51	−6.39	208 885.5	74.44
s298	133	2.72	2178.8	2.06	2.41	−24.38	−11.37	2828.2	29.80
s344	175	3.44	2461.5	3.05	2.99	−11.37	−13.14	3024.9	22.89
s349	176	3.49	2481.5	3.05	3.09	−12.66	−11.46	2943.5	18.62
s38417	23 815	9.00	287 766.8	8.70	7.81	−3.40	−13.26	299 073.5	3.93
s38584	20 705	11.99	282 444.0	10.46	10.53	−12.76	−12.24	298 206.7	5.58
s386	165	3.59	2418.3	2.68	3.23	−25.10	−9.97	3280.8	35.66
s510	217	3.14	2664.4	2.71	2.70	−13.54	−13.82	4173.9	56.66
s526	214	2.73	3642.4	2.05	2.46	−24.91	−9.80	4014.0	10.20
s5378	2958	5.33	34 421.6	2.90	4.51	−45.61	−15.29	33 023.2	−4.06
s641	398	8.59	4500.6	7.67	7.77	−10.69	−9.54	4787.6	6.31

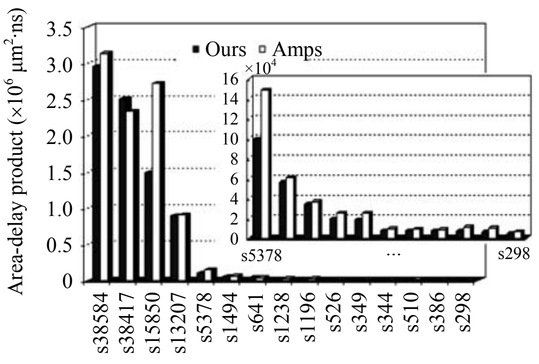

Fig.7 Area-delay product produced by our approach and Amps

Verification

A series of verifications, which is the primary objective of this step, is required for each cell tuned through the previous step, such as checking the design rule, extracting spice model for the LVS check on the top design, setting up a timing model, etc.

Extraction and timing analysis

We use Synopsys StarRC tool for RC network extraction and a commercial static timing analyzer (Synopsys Primetime) to analyze the entire design and report the critical path which is needed for critical network expansion. And the methodology continues to the next iteration as illustrated in Fig.1.

CONCLUSION

The P/N ratio tuning approach described in this paper is effective, convenient and fast convergent in semi-custom design to accelerate clock speed with no additional design flow iteration for ECO placement and routing. It distinctly reduces the complexity of transistor tuning by decomposing the network based on graph theory. It improves the accuracy in the tuning process compared with other conventional transistor tuning methodologies by taking the cross-talk effect into account using HSPICE simulation while requiring lower cost and less runtime on CPUs, owing to taking the cross-talk analysis apart from the timing analysis and calculating the cross-talk delay individually using the cross-talk expression instead of the traditional iterative analysis based on timing window. Furthermore, the area-delay tradeoff problem is not involved in our approaches through a customized standard cell architecture.

References

Chen, R.Y., Yip, P., Konstadinidis, G., Demas, A., Klass, F., Mains, R., Schmitt, M., Bistry, D., 2002. Timing Window Applications in UltraSPARC-IIIi^{TM} Microprocessor Design. Proc. IEEE Int. Conf. on Computer Design: VLSI in Computers and Processors, p.158-163. [doi:10.1109/ICCD.2002.1106764]

Cormen, T.H., Leiserson, C.E., Rivest, R.L., 1990. Introduction to Algorithms. MIT Press, USA, p.641-732.

Fishburn, J.P., Dunlop, A.E., 1985. TILOS: A Posynomial Programming Approach to Transistor Sizing. Proc. Int. Conf. on Computer-Aided Design, p.326-328.

Kabbani, A., Al-Khalili, D., Al-Khalili, A.J., 2005. Logical Path Delay Distribution and Transistor Sizing. Proc. Int. IEEE North-East Workshop on Circuits and Systems Conf., p.391-394. [doi:10.1109/NEWCAS.2005.1496701]

Kao, W.H., Fathi, N., Lee, C.H., 1985. Algorithms for Automatic Transistor Sizing in CMOS Digital Circuits. Proc. ACM/IEEE Conf. on Design Automation, p.781-784.

Ketkar, M., Kasamsetty, K., Sapatnekar, S., 2000. Convex Delay Models for Transistor Sizing. Proc. Design Automation Conf., p.655-660.

Lu, P.F., Northrop, G.A., Chiarot, K., 2005. A Semi-Custom Design of Branch Address Calculator in the IBM Power4 Microprocessor. IEEE VLSI-TSA Int. Symp., p.329-332. [doi:10.1109/VDAT.2005.1500088]

Northrop, G.A., Lu, P.F., 2001. A Semi-Custom Design Flow in High-Performance Microprocessor Design. Proc. Design Automation Conf., p.426-431.

Santos, C., Ferrao, D., Lazzari, C., Wilke, G., Guntzel, J.L., Reis, R., 2005a. Effects of Using a Pin-to-Pin Delay Model on a Library-Free Transistor/Gate Sizing Scheme. Proc. 48th Midwest Symp. on Circuits and Systems, p.315-318. [doi:10.1109/MWSCAS.2005.1594102]

Santos, C., Ferrao, D., Reis, R., Guntzel, J.L., 2005b. Incremental Timing Optimization for Automatic Layout Generation. IEEE Int. Symp. on Circuit and Systems, p.3567-3570. [doi:10.1109/ISCAS.2005.1465400]

Talukdar, D., Sridhar, R., 1996. An Analytical Approach to Fine Tuning in CMOS Wave-Pipelining. Proc. Int. Application Specific Integrated Circuits Conf., p.205-208. [doi:10.1109/ASIC.1996.551995]

Vittal, A., Chen, L.H., Marek-Sadowska, M., Wang, K.P., Yang, S., 1999. Modeling Crosstalk in Resistive VLSI Inter-Connections. Int. Conf. on VLSI Design, p.470-475. [doi:10.1109/ICVD.1999.745200]

Yelamarthi, K., Chen, C.I.H., 2007. Transistor Sizing for Load Balance of Multiple Paths in Dynamic CMOS for Timing Optimization. 8th Int. Symp. on Quality Electronic Design, p.426-431. [doi:10.1109/ISQED.2007.162]

Yu, X.Y., Oklobdzija, V.G., Walker, W.W., 2003. An Efficient Transistor Optimizer for Custom Circuits. Proc. Int. Symp. on Circuits and Systems, p.197-200. [doi:10.1109/ISCAS.2003.1206230]