Practical Methodology of Post-Layout Gate Sizing for 15% More Power Saving

Noriyuki Miura*, Naoki Kato**, and Tadahiro Kuroda*
*Department of Electronics and Electrical Engineering, Keio University
**Central Research Laboratory, Hitachi, Ltd.
miura@kuro.elec.keio.ac.jp, kato@crl.hitachi.co.jp, kuroda@elec.keio.ac.jp

Abstract - This paper presents a practical methodology of post-layout gate sizing for power reduction. Wire capacitance presumed in logic synthesis typically contains excessive margin for better timing closure in layout design. Power waste due to this can be reduced by post-layout gate sizing based on information obtained by backannotation. In this paper, we discuss a theory of optimal gate sizing in a signal path with surplus timing. We also, propose a practical design methodology where standard cells are reselected from a cell library by the theory, replaced by engineering change order, and timing constraints are verified by a static timing analyzer. We have applied the methodology to a 700k-gate commercial application processor for 3G cellular phones. Even though the original design was optimized for 133MHz, 170mW operation in a 0.18μm CMOS technology, power dissipation was further squeezed by 15% in combinational logic without compromising the performance.

I. Introduction

Low-power, high-speed design is an increasingly important engineering challenge in VLSI design, not only for long battery life in mobile applications, but also for cost reduction in high-end systems. The essence of reducing power dissipation is to avoid waste. There may be waste incurred in the automated design process of VLSI chips in compensation for design efficiency. One example can be found in estimation of wire capacitance. Wire capacitance presumed in logic synthesis typically contains excessive margin for better timing closure in the succeeding layout design stage. Recently, placement driven synthesis tools like Physical Compiler, Blast Fusion, PKS or PDS are introduced in logic synthesis to select fine-tuned gate size by estimating wire capacitance from the placement information. However, their estimation is not accurate enough due to their lack of the routing information. Fig.1 depicts wire length in a 120k-net module of a commercial design, one estimated by the synthesis tool in logic synthesis and the other extracted by backannotation in post-layout. It is shown that estimated wire length contains error, which becomes the larger when the length becomes the longer. On top of that, it is not known in the logic synthesis stage, which metal layer will be used in layout, how long adjacent lines will run in parallel, or how densely wires will go orthogonally in the lower and the upper layers. In scaled devices, wire capacitance depends heavily on these surrounding circumstances. Excessive margin of typically 20~30% should be included in estimating wire capacitance from the presumed wire length. As a result, large gate size more than necessary is used even with the current advanced synthesis tools. Post-layout gate sizing can further reduce circuit capacitance, and hence, power dissipation. Clearly, it is as important to try to reduce the waste by investigating wire capacitance in post-layout, as to try to improve accuracy of the estimation in logic synthesis.

Post-layout transistor sizing techniques for power reduction have been reported [1-3]. One of the issues in transistor sizing is accuracy in delay estimation. In gate sizing, propagation delay of sized gates can be characterized precisely by measuring a fabricated testchip. In transistor sizing, on the other hand, number of variants that can be generated by transistor sizing is so large that it is very difficult to characterize them all by a testchip. Precise delay calculation is inevitable. An accurate 4-variable delay model was proposed in [1], but device modeling will be more and more difficult and inaccurate due to non-ideal effects and parasitic effects in scaled devices.

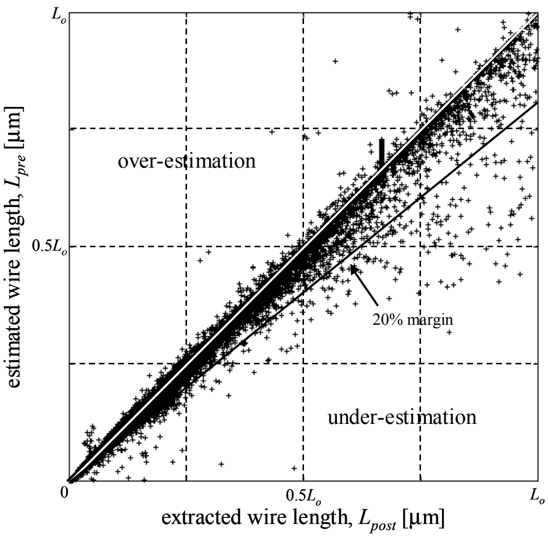

Fig.1. Wire length estimated by the advanced synthesis tool and that extracted by backannotation in post-layout in a 120k-net module.

Gate sizing techniques for power reduction have been investigated [4-7]. Lagrangian Relaxation technique is employed in [4,5] to solve non-linear optimization problem, while in [6,7] gate size is reselected based on delay sensitivity calculation. All gates are optimized exhaustively, so that a gate can be sized larger for global optimization. In this case, the gate cannot be replaced by a larger one without impacting the layout. It is not assumed that there are multiple cells with the same physical size and different drive, since this approach degrades layout density. Therefore, they are not suitable for post-layout design modification, while they can be employed for pre-layout power/area optimization. Furthermore, iterative calculation is required for optimization, which may cause a convergence problem.

We propose a practical methodology for post-layout gate sizing. A timing surplus in a path is utilized to replace a gate with a smaller one in a standard cell library by applying a theory of optimal gate sizing. Layout is then modified by engineering change order (ECO), and timing is verified by a static timing analyzer (STA). Since timing is verified by STA, an industry standard scheme, no new timing issue rises. It is good enough if delay model is accurate only for good guidance of gate sizing. Our proposed methodology runs from start to stop without iteration, and hence very simple. It can be implemented in a small program with ECO and STA, and hence very general. Maximum transition time constraint is also verified for practical design. Gate sizing in a 700k-gate microprocessor was completed within 3 days, mostly for timing verification by STA and its setup time, and fully automated without designer's assistance.

II. Theory of Optimal Gate Sizing

In this section, delay model and power model are defined, as well as path-by-path gate sizing is explained. This sizing method solves a delay constraint power optimization problem of a generic single logic path. We applied this method to combinational logic. w denotes transistor width in the input stage of a gate that corresponds to gate input capacitance C_{in}, is proportion to w. At first, wire capacitance C_w is assumed zero for simplicity, but will be later

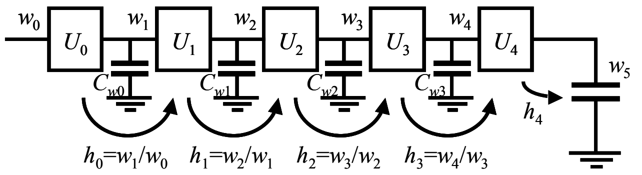

Fig.2. Gate sizing model.

taken into account.

A. Delay Model

Precision of a delay model is not very important in the proposed post-layout gate sizing, since path delay is always verified by STA after the sizing is performed. It is good enough if the delay model is accurate for providing good guidance for gate sizing.

Linear delay model in [8,9] is adopted for simplicity. In this model, delay d_i in a gate U_i is given by

$$d_i = \tau_{inv}(g_i h_i + p_i). \quad (1)$$

d_i is proportional to electrical fanout h_i (a ratio of the output load capacitance w_{i+1} to the driving gate size w_i). Its slope is defined as logical effort g_i and its intercept is defined as parasitic delay p_i. τ_{inv} is the process-dependent constant. g_i, p_i are gate-dependent constants.

B. Power Model

Power consumption in CMOS gate is the sum of switching power, crowbar current power and leakage power. The purpose here is to reduce switching power due to excessive capacitance. Accordingly, only switching power is considered. Power dissipation P_i at the gate U_i is given by

$$P_i = P_{inv} f \alpha_i \left(w_{i+1} + \frac{p_i}{g_i} w_i \right) = P_{inv} f \alpha_i w_0 \left(\prod_{j=0}^{i} h_j + \frac{p_i}{g_i} \prod_{j=0}^{i-1} h_j \right) \quad (2)$$

where P_{inv} is the process dependence constant, f is the operating frequency and α_i is the switching probability. g_i and p_i are equivalent to those in the delay model. The first term of the equations stands for the switching power in the output load. The second term stands for the switching power in the gate inside capacitance.

C. Gate Sizing

Power optimization under delay constraint of a generic single path in Fig.2 is discussed. This optimization problem becomes a nonlinear minimization problem when two constraint conditions are given. The first constraint condition is given by the start/end points. In Fig.2, the gate size of the first stage w_0 and the output load w_5 should be given. The first stage is typically a flip-flop (FF) or a latch, and output load is typically capacitance of a data input of FF or latch. In a N stage logic path, gross product of electrical fanout is given by

$$\prod_{i=0}^{N-1} h_i = \frac{w_N}{w_0} = const. \quad (3)$$

The second constraint condition is given by the delay constraint. The total delay of a N stage logic path, D, is given as follows by sum of each gate,

$$D = \tau_{inv} \left(\sum_{i=0}^{N-1} g_i h_i + D_p \right), \quad (4)$$

where D_p is sum of parasitic delays. Since logic structure is not changed by gate sizing, D_p is constant. D is minimized when $g_i h_i = g_{i+1} h_{i+1} \equiv \hat{f}$ for all i [8,9]. Minimum delay D_{min} is written as

$$D_{\min} = \tau_{inv} \left(N\hat{f} + D_p \right), \quad \hat{f} = \left(\frac{w_N}{w_0} \prod_{i=0}^{N-1} g_i \right)^{1/N}. \quad (5),(6)$$

The electrical fanout $\hat{h}_i$ that gives minimum delay is obtained by $\hat{f}/g_i$. If defined target delay $D_{targ} = \delta D_{min}$, the second constraint condition is rewritten as

$$\sum_{i=0}^{N-1} g_i h_i = \left(\delta + (\delta - 1) \frac{D_p}{N\hat{f}} \right) N\hat{f} \equiv \delta_{con} N\hat{f}, \quad (7)$$

where δ_{con} is delay allowing coefficient. The constraint condition is given to be based on the minimum delay condition. Similarly, h_i is re-defined to be based on $\hat{h}_i$

$$h_i = \lambda_i \hat{h}_i, \quad (8)$$

where λ_i is positive sequence. By (8), (3) and (5) are simplified as follows

$$\prod_{i=0}^{N-1} \lambda_i = 1, \qquad \sum_{i=0}^{N-1} \lambda_i = \delta_{con} N. \quad (9),(10)$$

The minimization function $f(\lambda_i)$ is the sum of switching power P_i (obtained by (2)) in a logic path. Therefore, that is written as

$$f(\lambda_i) = \sum_{j=0}^{N} \left(\kappa_j \prod_{i=0}^{j-1} \lambda_i \right), \quad \kappa_j = \left(\alpha_{j-1} + \alpha_j \frac{p_j}{g_j} \right) \prod_{i=0}^{j-1} \hat{h}_i. \quad (11),(12)$$

Consequently, the delay constraint power optimization of a single path results in the minimization problem, (11), under the two constraint conditions (9), (10). This minimization problem can be solved under the Kuhn-Tacker conditions: $\partial f(\lambda_i) / \partial \lambda_i = 0$ for all i. The solutions of λ_i can be obtained by numerical analysis. Subsequently, the electrical fanouts h_i, optimizing power are obtained by (8), and the gate size w_i are obtained.

D. Wire Capacitance

The minimization problem is extended to the case where the wire capacitance is not zero. At first, the minimum delay D_{min} may be without wire capacitance. What must be extended is only the delay constraint condition in (10). That is written as

$$\sum_{i=0}^{N-1} \lambda_i + \frac{1}{w_0 \hat{f}} \sum_{j=0}^{N-3} \frac{C_{wj+1}}{\prod_{i=0}^{j} \lambda_j \hat{h}_j} = \delta_{con} N - \frac{C_{w0}}{w_0 \hat{f}}. \quad (13)$$

Considering the output load w_N containing the wire capacitance, (9) doesn't change. Furthermore, the minimization function of (11) doesn't change because the consumed power at the wire capacitance is constant. In this case, the numeric analysis solution can be obtained similarly under the Kuhn-Tacker conditions.

The gate size w obtained from the above-mentioned calculation must be used with attention of the following two points. 1)w is calculated by assumed as continuum. Practically the gate size prepared in the library is discrete and finite value w'. Consequently, the nearest value of w' to w is selected. 2)Proposed method is performed post-layout so there is an original design. Increase of the gate size isn't allowed due to the difficulty of layout. If the solution of calculation is larger than that of the original design, the gate maintains its original design. These two points bring up error of path delay from other reasons of delay modeling. However, the sized path is checked with STA if it meet the delay constraint. If it violates, the path is back to that of the original design, as a result timing violation never remain.

III. Design Methodology for VLSI

In this section, a design methodology of post-layout gate sizing in VLSI based on the theory of optimal gate sizing is proposed.

A. Branching and Convergence

Logic networks in VLSI circuits are composed of not only a single path, but also a path with branching and/or convergence. In each case, delay constraint is given to each path between start-point and end-point independently. Assuming the followings, all the networks can be treated as a single path. 1)Gate capacitance at the branching is regarded as a part of wire capacitance. 2)Gate sizing of a gate is carried out in the most critical path through the gate. Once a gate is sized, it will never be resized(A gate that should

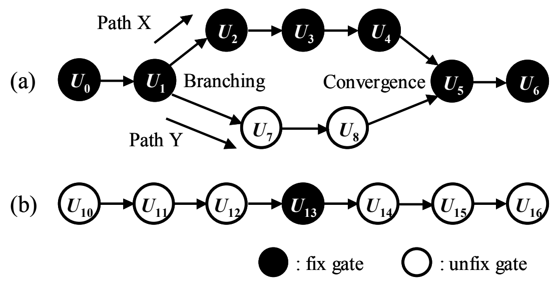

Fig.3. Logic path in VLSI circuits, (a)path including branching and convergence, (b)path containing fix gate inside

never be resized is called as a fix gate). 3)When a fix gate is in a path, the path is divided into two at the fix gate.

To meet a delay constraint, the gate capacitance on the end of branching must be considered. Consequently, 1) is needed. Although gate capacitance can be changed by gate sizing, it doesn't matter because gate capacitance never be enlarged.

By sizing of a path, other paths must not violate the delay constraint. Therefore, 2) is needed. When the logic gate U_1 is sized in the logical network showed in Fig.3 (a), if slack of path X is smaller and more critical than that of path Y, gate sizing should be applied to path X to size U_1. The delay of path Y won't be lager than that of path X, consequently, path Y never violates delay constraint. In addition, as the delay of path X must not increase after that, sized gates U_0-U_6 become fix gates and not to be resized.

Path Y in Fig.3 (a) and Fig.3 (b) contains fix gates partially. Considering as 3), these can be converted into combination of single paths. In case of path Y, it can be sized by single path from U_1 through U_7, U_8 to the input of U_5 after subtraction of the delay of U_0, U_5, U_6 from the entire delay of path Y. In this process, U_1 isn't sized according to the first stage and input capacitance of U_7 won't be larger. Therefore, the delay of path X doesn't increase. In addition, the path that contains fix gate partially as Fig.3 (b) can be objected. In case of that, it can be treated as a combination of single paths by division into path from U_{10} to U_{13} and path from U_{13} to U16. Then, the target delay is divided by a ratio of the minimum delay of each path.

B. Sizing of FF(Latch)

Sizing of FF (latch) is different from that of combinational logic due to its setup/hold time constraints. A timing constraint is typically given from FF to FF, setup/hold time tighten the constraint. If FF is downsized, setup time grow larger, in some cases, it can make timing violation. Changing timing constraint of the previous path due to sizing of first stage FF becomes a problem, though the previous path is not objective of sizing. To solve this problem, we use a heuristic method for the sizing of FF. First, all FFs are extracted from the original design. Next, FFs are decremented, until the most critical path through the input or output pins of the FFs violate the constraint, or the FFs downsized to their lower bounds. The time when this process runs must be considered. If the first stage FF is downsized too small, the gate sizes in combinational logic become larger. To minimize the sum of gate sizes, the gate size of the first stage FF has an optimal solution. However, the switching activity of the pin of FF is generally much higher than that of combinational logic because FF is driven by clock. Therefore, to minimize power, FF should be sized by priority. Consequently, sizing of FF is carried out in pre-process of path-by-path gate sizing.

C. Sizing Algorithm

The sizing algorithm is explained in this section. Firstly, some rules and variables are defined. In our sizing algorithm, all gates have only three states. They are "fix", "subfix", and "unfix". Here, fix gate = (the gate that has been sized once and never be resized.), subfix gate = (the gate that has been sized but is not determined yet.), unfix gate = (the gate except for the both of fix gate and subfix gate). U_i represents a gate, w_i represents the gate size of U_i, and P_i represents the most critical path through U_i. Additionally, S_i is defined as the state of the gate U_i and its initial value is "unfix".

Sizing Algorithm (path-by-path gate sizing)

```
1. foreach gate $U_i$
     if($w_i$ is already the lower limit of the given library)
          $S_i$ = "fix";
2. foreach gate $U_i$
     if($S_i$ = "unfix")
          Extract $P_i$ by STA;
     Sort $P_i$ by slack;
3. foreach path $P_i$
4-1. if($P_i$ contains no subfix gate)
          Size $P_i$;.
4-2.      else
          Don't size $P_i$;
4-3. foreach unfix gates in $P_i$
          $S_i$ = "subfix";
5. foreach sized path $P_i$
     Check $P_i$ by STA if it meets the delay constraint;
6-1 if(meet, slack>=0)
          foreach subfix gates in $P_i$
               $S_i$ = "fix";
6-2 else
          foreach subfix gates in $P_i$ {
               Back the gate to that of original design;
               $S_i$ = "fix";
7. Mark all subfix gates ($S_i$="subfix") unfix gates ($S_i$="unfix");
8. until(all $S_i$="fix")
     Return to process 2;
```

The following explains about each process. Many gates are constructed of the smallest one in the original design. These can't be changed into smaller or larger size. Hence, they are equivalent to fix gates, accordingly the timings through them don't have to be analyzed. Therefore, before STA is performed on these gates, they are marked fix gates in process 1 to shorten the time for STA. The process 2 and process 3 are the process that obey 2) in Section III.A. After that, the process apply gate sizing to each path and mark fix gates with checking timing violation by STA, which is repeated until all gates become fix gates.

However, it is necessary to explain about subfix gates which appear after process 4. The proposed gate sizing uses gate delay obtained by STA, for example, when it calculates the timing constraint in sizing of U_7, U_8 of path Y [Fig.3 (a)]. When the gate size becomes smaller, the gate delay will change. Consequently, the accurate timing constraint of the path that passes this gate can't be obtained except it is obtained by STA again. Therefore, the paths containing subfix gates, whose gate size may be changed, are suspended till such subfix gates are fixed and delays are obtained by STA. Furthermore, unfix gates in suspended paths must not be sized in other path with larger slack as it violates 2) in Section III.A. Thus, the unfix gates in these suspended paths must be marked as subfix gate.

D. Max Transition Time Constraints

On an actual design, not only the timing constraint is important, but also max transition time constraint. Max transition time violation occurs when transition time of gate output pin is over a certain value. As proposed sizing process ignores this so far, the gate size gets too small. As a result, violation occurs. Such violation is resolved in post-process of sizing as follows. On the violation net, its driver gate is incremented under the gate size of original design. However, there are some cases that the violation cannot be improved when the size of the driver gate is back to the gate size of the original design, because input transition time of the driver gate is degraded. In case of that, the gate driving the input pin with worst transition time of the driver gate is incremented.

The proposed methodology of post-layout gate sizing is consisted of three processes. They are 1)pre-process that sizes FF

TABLE I
Post-layout gate sizing in a 700k-gate commercial application processor for 3G cellular phones. Even though the original design was optimized for low-power (170mW), high-performance (133MHz) operation in a 0.18 µm CMOS technology, power dissipation is further reduced by 15.4% in combinational logics, and by 12.9% including register elements.

module	gates	sized gates	normalized power					
			combinational logic + register			combinational logic only		
			before	after	reduction	before	after	reduction
a	169	56	0.044	0.041	−5.8%	0.044	0.041	−5.8%
b	248	6	1.73	1.65	−4.4%	1.12	1.08	−3.8%
c	315	3	0.021	0.020	−5.5%	0.021	0.020	−5.5%
d	471	117	6.30	5.99	−5.1%	6.31	5.99	−5.1%
e	504	240	1.30	1.12	−14.2%	0.21	0.16	−24.2%
f	520	233	7.07	6.51	−7.9%	7.07	6.51	−7.9%
g	928	469	3.04	2.56	−15.7%	1.93	1.50	−22.1%
h	1830	654	1.57	1.44	−8.3%	0.83	0.72	−13.0%
i	2006	798	4.80	3.42	−28.8%	4.80	3.41	−28.8%
j	2289	861	0.028	0.024	−15.2%	0.016	0.012	−27.4%
k	2336	752	1.25	1.11	−11.5%	1.18	1.06	−10.4%
l	2460	1138	4.72	2.85	−39.7%	4.72	2.85	−39.7%
m	2914	1126	0.49	0.47	−3.8%	0.49	0.47	−3.8%
n	3621	1732	2.38	1.74	−26.9%	2.27	1.66	−26.8%
o	3771	1100	0.097	0.095	−1.4%	0.0061	0.0053	−12.9%
p	3986	980	0.79	0.77	−2.6%	0.40	0.38	−5.03%
q	4860	1772	0.93	0.77	−17.5%	0.66	0.54	−18.1%
r	5796	1726	0.085	0.084	−0.9%	0.0013	0.00098	−28.6%
s	6434	1811	0.96	0.96	−0.7%	0.32	0.32	−0.25%
t	9415	3014	17.26	15.82	−8.3%	7.49	6.63	−11.4%
u	10163	3448	40.64	36.25	−10.8%	22.12	19.07	−13.8%
v	10901	4028	0.70	0.57	−19.1%	0.44	0.37	−17.6%
w	11608	3805	3.77	2.87	−23.9%	0.0080	0.0071	−12.2%
total	87545	29869	100	87.14	**−12.9%**	62.46	52.81	**−15.4%**

(latch), 2)main-process that sizes combinational logic by path-by-path gate sizing, 3)post-process that improves max transition time violations.

IV. Experimental Results

We implemented the proposed algorithm using Perl and developed the proposed design methodology by employing the Synopsys Design Compiler for STA. It was applied to a 700k-gate commercial application processor for 3G cellular phones [10], which integrates 260,187 cells in 126 modules in a 0.18µm CMOS technology. An existing standard cell library is employed and nothing special is prepared. Power dissipation is evaluated by the dhrystone benchmark.

The experimental results are summarized in Table I where only those modules activated by the dhrystone benchmark are listed. 34% cells are sized to reduce the gate area by 10.4%. Even though the original design was optimized for low-power (170mW), high-performance (133MHz) operation, power dissipation is further reduced by 15% in combinational logics, and by 13% including register elements. No timing error was reported by STA. The post-layout gate sizing was performed fully automatically by our program within 3 days, mostly for timing verification by STA and its setup time. Such time can be eliminated by embedding customized STA in our sizing program without calling external STA. Moreover, since the process runs individually module-by-module, parallel processing is possible to shorten the design time considerably.

Delay distributions of the entire chip before and after gate sizing are depicted in Fig.4. The discrepancy between the two lines corresponds to the amount of power dissipation reduced by the gate sizing.

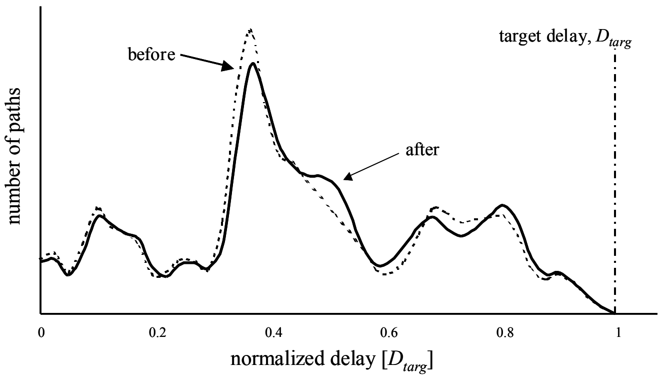

Fig.4. Delay distribution, before (broken line) and after (solid line) gate sizing

V. Conclusions

Post-layout gate sizing is effective to save waste in estimation of wire capacitance in logic synthesis. Our proposed design methodology further reduces power dissipation of CMOS LSI by 15%, compared to the conventional optimal designed for low power, without any compromise in performance and reliability. Since the proposed methodology is based upon an industry standard design methodology using the same validation suites, it can be introduced with minimum effort and risk, and hence, very practical.

Acknowledgements

The authors are grateful to Kazuo Yano and Naohiko Irie from Hitachi, Ltd. Ikuya Kawasaki and Kenji Hirose from Renesas Technology Corp. for the provision of the design data and the design environment.

References

[1] M. Hashimoto, and H. Onodera, “Post-layout transistor sizing for power reduction in cell-based design,” *Proc. DAC*, pp.359-365, 2001.
[2] M. Yamada, *et al*., “Synergistic power/area optimization with transistor sizing and wire length minimization,” *IEICE Trans. Electrons*, Vol. E78-C, No. 4, pp. 441-446, 1995.
[3] J. P. Fishburn and A. E. Dunlop, “TILOS: Aposynomial programming approach to transitor sizing,” *IEEE Trans. on CAD*, pp. 326-328, Nov 1985.
[4] C. P. Chen, C. C. N. Chu, and D. F. Wong, “Fast and exact simultaneous gate and wire sizing by Lagrangian Relaxation,” *Proc. ICCAD*, pp. 617-624, November 1998.
[5] H. Tennakoon, and C. Sechen, “Gate sizing using Lagrangian Relaxation combined with a fast gradient-based pre-processing step,” *Proc. ICCAD*, pp. 395-402, November 2002.
[6] H. R. Lin and T. T. Hwang, “Power reduction by gate sizing with path-oriented slack calculation,” *Proc. ASP-DAC*, pp.7-12, 1995.
[7] M. Hashimoto, H. Onodera, and K. Tamaru, “A practical gate resizing considering glitch reduction for low power design,” *Proc. DAC*, pp.446-451, 1999.
[8] T. Sakurai, “A Unified theory for mixed CMOS/BiCMOS buffer optimization” *IEEE J. Solid-State Circuits*, vol. 27, no. 7, pp1014-1019, July 1992.
[9] I. Sutherland, B. Sproull and D. Harris, *Logical Effort: Designing Fast CMOS Circuits*, Morgan Kaufmann Publishers, Co.
[10] T. Yamada, *et al*., “A 133MHz 170mW 10µA standby application processor for 3G cellular phones,” *Proc. ISSCC*, pp. 370-371, 2002.