Available online at www.sciencedirect.com
ScienceDirect
INTEGRATION, the VLSI journal 41 (2008) 9–16
INTEGRATION theVLSI journal
www.elsevier.com/locate/vlsi

Gate-level dual-threshold static power optimization methodology (GDSPOM) using path-based static timing analysis (STA) technique for SOC application

B. Chung[1], J.B. Kuo*,[2]

School of Engineering Science, SFU, Burnaby, Canada V5A 1S5

Abstract

This paper describes a novel gate-level dual-threshold static power optimization methodology (GDSPOM), which is based on the static timing analysis (STA) technique for designing high-speed low-power SOC applications using 90 nm multi-threshold complementory metal oxide semiconductor (MTCMOS) technology. The cell libraries come in fixed threshold—high V_{th} for good standby power and low V_{th} for high speed. Based on this optimization technique using two cell libraries with different threshold voltages, a 16-bit multiplier using the dual-threshold cells meeting the speed requirement has been designed to have a 50% less leakage power consumption when compared to the one using only the low-threshold cell library.

Keywords: Dual-threshold CMOS; Static timing analysis; SOC; Power optimization

1. Introduction

SOC systems implemented by complementory metal oxide semiconductor (CMOS) very large scale integration (VLSI) circuits using nanometer transistors have been evolving quickly [1]. Low-power consumption has become a significant requirement [1]. For achieving low static power consumption, multi-threshold CMOS (MTCMOS) circuits have been proven to be an effective approach [2,3]. Various algorithms of minimizing static power consumption using dual-threshold techniques have been proposed [4–6]. However, static power optimization of a multi-million gate design using MTCMOS technology at the transistor level is difficult, while at the gate level it is much easier to implement.

In this paper, a gate-level dual-threshold static power optimization methodology (GDSPOM) using path-based static timing analysis (STA) is described. It will be shown that via two cell libraries with different threshold voltages, the design of a 16-bit multiplier circuit has been optimized based on GDSPOM, which has a 50% less leakage power consumption in comparison with the all low-threshold voltage one at the operating frequency of 500 MHz. In the following sections, characters and usages of timing and power models are introduced first, the principle of GDSPOM is presented next, followed by the performance of the test multiplier circuits, discussion and conclusion.

2. Timing models

A cell is a fundamental logic block and it is the basic element in a gate-level netlist. A cell built with all high-threshold voltage (HVT) transistors has a longer signal propagation delay and blocks more unwanted leakage current; a cell built with all low-threshold voltage transistors has a faster signal transition time but suffers from generating a large amount of sub-threshold leakage current [7,8].

A cell may have multiple timing arcs. A timing arc defines a timing relationship between one input pin and one output pin of a cell. Different timing arcs have different cell propagation delays [9]. As shown in Fig. 1(a), a two-input

*Corresponding author.
E-mail address: jbkuo@sfu.ca (J.B. Kuo).
[1]Also with PMC Sierra, Canada.
[2]On leave from NTUEE.

0167-9260/$ - see front matter
doi:10.1016/j.vlsi.2007.03.001

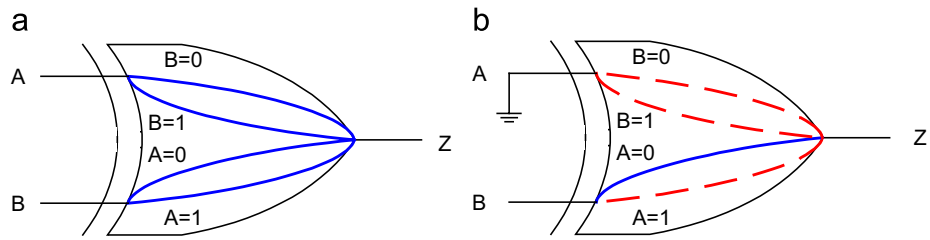


Fig. 1. XOR cell timing arcs.

XOR gate has four arcs: A–Z when B is 0, A–Z when B is 1, B–Z when A is 0, and B–Z when A is 0. In the case when a pin is assigned a constant value as shown in Fig. 1(b), when input A is assigned a constant value of 0, there is only one arc between input B to output Z available for analysis.

Arc delay calculation formula can be simplified as a function of the input transition time and the output load capacitance. Timing lookup table like the one shown in Fig. 2 is recorded in the cell technology library [9]. Using the example in Fig. 2, a cell delay time from input B to output Z is 8.8 ns when the input transition time is 100 ps and the output load capacitance is 0.4 fF. When values of the input transition time and the output load capacitance are between the table points or outside the table range, we use the interpolation or the extrapolation approach to estimate the delay.

Arc delay is the cell internal signal propagation time. Net delay is the total time for a signal to travel between two cells. Knowing cell and net delays as well as input latency, path arrival time can be calculated as following:

$$AT_{\text{path}} = D_{\text{clk}} + D_{\text{clk\_net}} + \sum D_{\text{cell}} + \sum D_{\text{net}}. \tag{1}$$

AT_{path} is path arrival time, D_{clk} is clock source latency, $D_{\text{clk\_net}}$ is clock network latency, D_{cell} is cell delay, and D_{net} is net delay.

Assuming that source clock latency is 2 ns, clock network latency is 3 ns, sequential and combinational cell delays are 3 ns, and net delay is 2 ns, the path shown in Fig. 3 has arrival time of 20 ns.

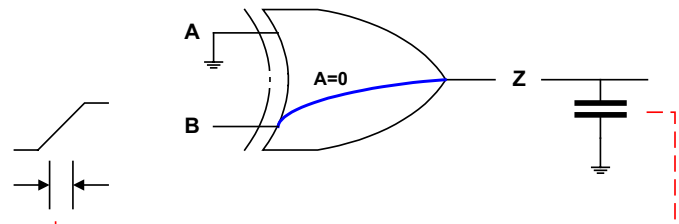
Furthermore, required setup time of a path can be calculated using the following formula:

$$RT_{\text{path\_setup}} = T_{\text{clk}} + D_{\text{clk}} + D_{\text{clk\_net}} - T_{\text{clk\_uncertainty}} - T_{\text{setup}}. \tag{2}$$

$RT_{\text{path\_setup}}$ is setup required time, T_{clk} is clock period, D_{clk} is clock source latency, $D_{\text{clk\_net}}$ is clock network latency, $T_{\text{clk\_uncertainty}}$ is clock uncertainty, and T_{setup} is capture flop setup time.

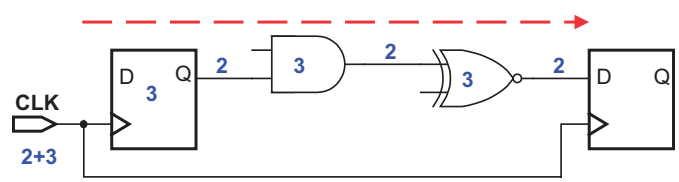
Assuming that clock period is 16 ns, source clock latency is 2 ns, clock network latency is 3 ns, clock uncertainty is 1 ns, and flop setup time is 1 ns, Fig. 4 shows the path has required setup time of 19 ns.

When a path's arrival time is shorter or equal to its required time, this path meets the setup timing constraint. On the other hand, when a path's arrival time is longer



Cell Arc B to Z		Output Load Capacitance (fF)			
		0.2	0.3	0.4	0.5
Input Transition Time (ps)	0	3.0	4.0	5.0	6.0
	50	4.5	6.0	7.5	8.0
	100	5.0	7.7	8.8	9.9

Fig. 2. Arc timing lookup table.



AT_{path}

$= D_{clk} + D_{clk\_net} + \Sigma^{D_{cell}} + \Sigma^{D_{net}}$

$= (2) + (3) + (3 + 3 + 3) + (2 + 2 + 2)$

$= 20$

Fig. 3. Path arrival time calculation example.

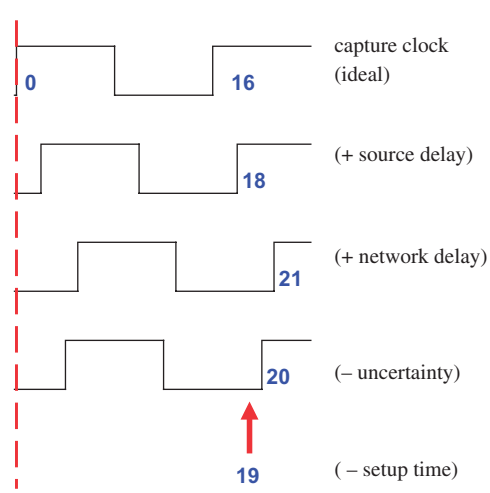


Fig. 4. Path required setup time calculation example.

than its required time, this path does not satisfy its timing constraint and has setup timing violation. The difference between path arrival and required time is called path slack and can be calculated with the formula

$$S_{\text{path\_setup}} = RT_{\text{path\_setup}} - AT_{\text{path}}. \quad (3)$$

Positive slack indicates that the path meets timing and negative slack tells that the path has a timing violation. Considering path in Fig. 3 has arrival time of 20 ns and its clock path in Fig. 4 has required time of 19 ns, this path's setup slack is −1 ns, which means that a setup timing violation exists on this path.

Moreover, it is possible to have different path routes between the same start and end points. Fig. 5 illustrates one short and one long path routes, which start and finish at the same point.

Because all timing checks are done per path bases, this type of STA is characterized as path-based STA approach.

3. Power models

Internal power lookup table similar to the one shown in Fig. 2 is included in the cell technology library. Like arc delay calculation, we use the interpolation or extrapolation method to predict the power when values of input transition time and output load capacitance are between the table points or outside the table range.

Switching power calculation formula is a function of net capacitive load and net switching rate. The net capacitive load can be obtained from the cell technology library and the wire load model. The value of the net switching rate can be calculated by monitoring net toggle activities while running functional simulation. For example, if a net value toggles 25 times in average per 100 clock cycles, its net switching rate is 0.25.

Leakage power is cell state dependent. As shown in Table 1, cell leakage current can vary in more than five orders of magnitude. Leakage current varies because transistors inside a cell have different on and off combinations in different state. As a result, the drain–source voltages V_{DS} of each transistor varies. In VLSI design, the impact of different cell states on the total leakage current is ignorable. Therefore, average leakage value is recorded in the technology library.

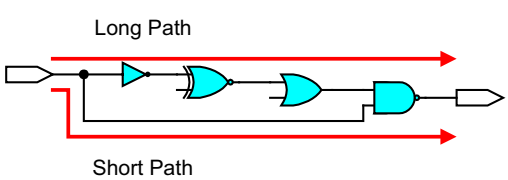


Fig. 5. Routes of timing paths between the same start and end points.

Table 1
90 nm NAND2 leakage current

NAND2 input value		Leakage current (nA)	
A	B	High V_{th}	Low V_{th}
0	0	1.49	3.72
0	1	2.56	14.93
1	0	3.57	18.95
1	1	4.61	21.77

Source: Based on Spice simulation.

4. GDSPOM

Fig. 6 shows the flow chart of GDSPOM used for designing high-speed low-power SOC applications using MTCMOS technology. As shown in the figure, a register transfer language (RTL) design is synthesized into gate-level netlist of cells using CMOS devices with a HVT. Then, static timing analysis is performed to report a list of cells that are required to swap from HVT type to the low-threshold voltage (SVT) type to meet timing constraints. Finally, cell-swapping script is executed to create the netlist built with dual-threshold HVT/SVT cells.

In the synthesis step, 25% slower operation speed is applied. In the example of 500 MHz 16-bit multiplier, 400 MHz frequency is targeted when converting the multiplier's RTL design to HVT gate-level netlist. The additional 100 MHz speed will be caught up in the cell swapping step, which replaces slow HVT cells with fast SVT ones. Comparing speeds of HVT and SVT cells in the 90 nm technology library, SVT ones are about 30% faster than HVT ones. This is the reason why 25% slower speed is chosen to create the initial HVT gate-level netlist and why it is possible to achieve the final speed target by changing

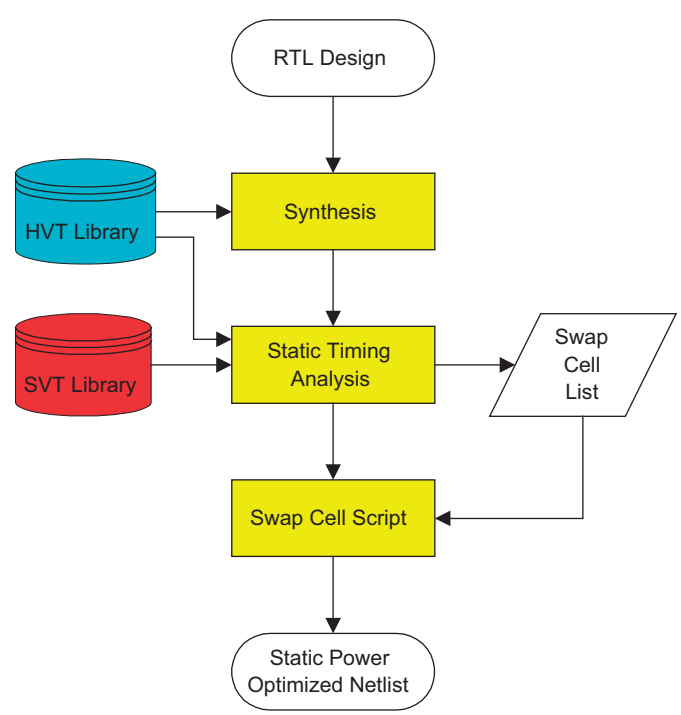


Fig. 6. Flow chart of the gate-level dual-threshold static power optimization methodology (GDSPOM).

cell types without altering design architecture and increasing area overhead.

STA breaks a design into a group of timing paths and calculates the signal propagation delay of each path individually. Fig. 7 illustrates three timing violated paths labeled green, blue, and purple. The number of timing violating paths through one cell defines this cell's cost value. For instance, AND gate has the cost value of 1; bottom NAND gate has the cost value of 2, middle NAND and OR gates have the cost value of 3. The cells with the highest cost value such as middle NAND gate and OR gates will be targeted for cell type change. After changing these bottleneck cells to SVT type, STA is performed again to recalculate cell cost values. This STA process continues until all the timing paths meet the required timing constraints.

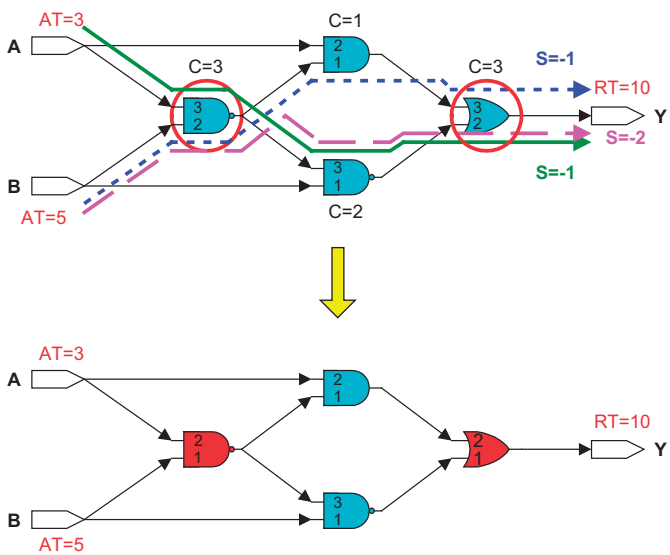


Fig. 7. GDSPOM cell swapping example.

Fig. 8 explains how cost values are assigned to cells and Fig. 9 shows the STA iterating process.

Fixing a high-cost cell means fixing multiple timing violated paths at once. Always targeting bottleneck cells in each STA loop procedure guarantees a highly efficient solution of solving design timing violation problem. In other words, GDSPOM replaces minimum amount of cells from HVT to SVT and results in the least leakage power increase while fixing all timing violations in a design.

5. Performance

In order to assess the effectiveness of GDSPOM for designing low-power high-speed SOC applications using a 90 nm MTCMOS technology, three 16-bit multipliers with Wallace tree reduction architecture [10] have been implemented. All of them are generated based on the same RTL source except that one multiplier uses all HVT cells, another has all SVT cells, and the third one contains both types of cells optimized via GDSPOM. Targeting operating frequency is set to be 500 MHz and 90 nm cell libraries are used in this experiment. A 16-bit multiplier has 7320 unity gates and it contains approximately 30,000 transistors.

```
procedure get Swap Cell List ($ original Net list, $ required Time) {
    (@ bottleneck Cell Array, $ input Net list) =
        & get Bottleneck Cells ($ origianl Netlist, $ required Time)

    while (@ bottleneck Cell Array ! = NULL) {
        @ swap Cell List = @ swap Cell List + @ bottle neck Cell Array

        (@ bottleneck Cell Array, $ input Netlist) =
            & get Bottle net Cells ($ input Net list, $ required Time)
    }

    return @ swap Cell List
}
```

Fig. 9. Get swap cell list algorithm.

```
procedure get Bottle neck Cells ($input Netlist, $ required Time) {
    @ path Array = all paths in $ input Netlist
    % cell Cost Hash = all cells in $ input Netlist with initial cost value 0

    foreach $path (@path Array) {
        $arrival Time = calculated $path arrival time

        if ($arrival Time > $required Time) {
            foreach $cell in $path {
                incr $ cell Cost Hash {$cell}
            }
        }
    }

    sort % cell Cost Hash by  cost value
    @ bottleneck Cell Array = first n cells with positive cost value in %cell Cost Hash
    $output Netlist = $input Net list after @bottleneck Cell Array cell type change

    return (@ bottle neck Cell Array, $output Netlist)
}
```

Fig. 8. Get bottleneck cells algorithm.

As shown in Fig. 10, with the 500 MHz clock frequency constraint, thousands of paths in the HVT multiplier fail the speed test. This result is expected because the HVT multiplier was synthesized with 400 MHz timing constraint, which is 25% slower than the targeting speed performance.

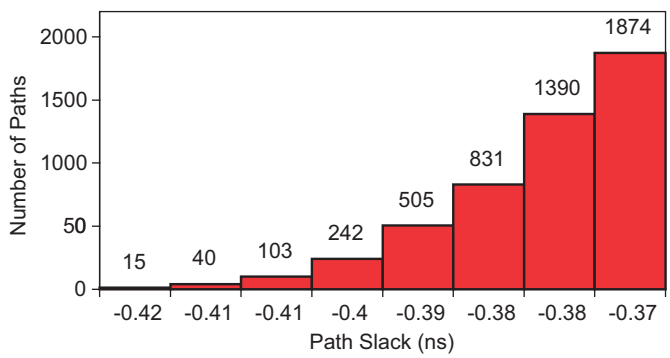


Fig. 10. Number of timing violated paths in the all-HVT multiplier.

Fig. 10 also indicates that 15 paths have the longest delay time and these 15 paths define the operating speed of 16-bit multipliers.

In this experiment, GDSPOM reassigned 352 out of total 1715 cells from HVT to SVT to satisfy the 500 MHz speed constraint. Fig. 11 shows the block diagram and Fig. 12 shows the schematic view of the 16-bit dual-V_t multiplier design optimized by GDSPOM to have HVT (blue) and SVT (red) cells. Note that yellow paths in Fig. 12 are originally timing violated paths.

A path between input $IN2_8$ (the 8th multiplicand bit) and output P_{23} (the 23rd product bit) is randomly selected to demonstrate how the swapping of the cell types has been used to resolve the timing violation. Fig. 13 shows this path in the HVT multiplier, whose data arrival time is 2.21 ns, which does not meet the 500 MHz operating frequency specification. The arrival time of each cell shown in the figure includes net delay time. Fig. 14 shows the same path in the dual-V_t multiplier. After performing GDSPOM flow,

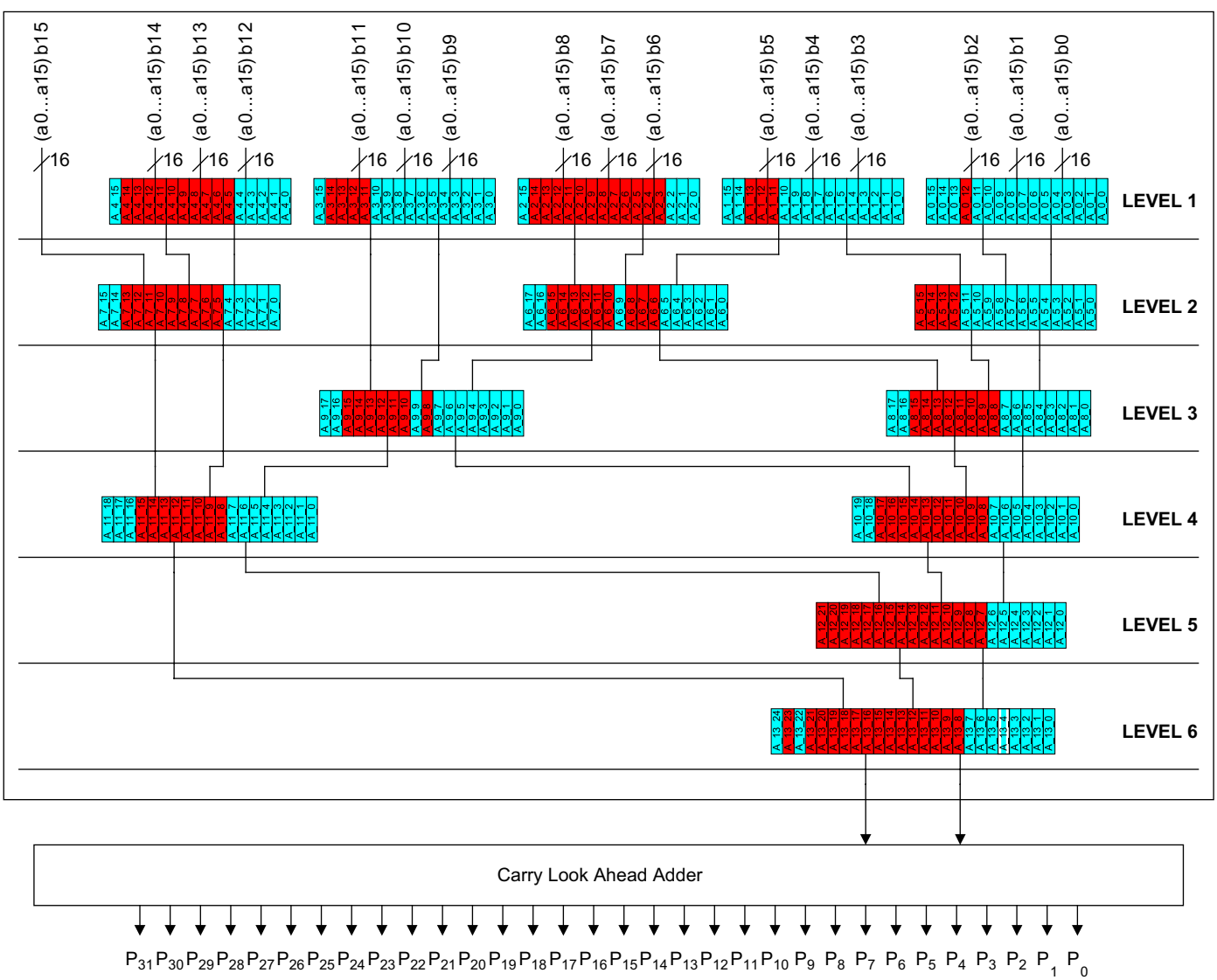


Fig. 11. Block diagram of the 16-bit multiplier design.

seven cells have been swapped from HVT to SVT. The data arrival time of this path becomes 1.92 ns, which meets the operating frequency constraint.

Among three multipliers using all-HVT, all-SVT, and dual-threshold HVT/SVT cells, the all-HVT one has the least leakage power consumption of 51 uW, but can only operate in 400 MHz frequency. All-SVT multiplier has the highest leakage power of 280 uW. Using the dual-threshold HVT/SVT cells adopting the GDSPOM flow, the power consumption of dual-V_t multiplier is 139 uW, which is 50% less than the all-SVT one, and meets the operating frequency constraint.

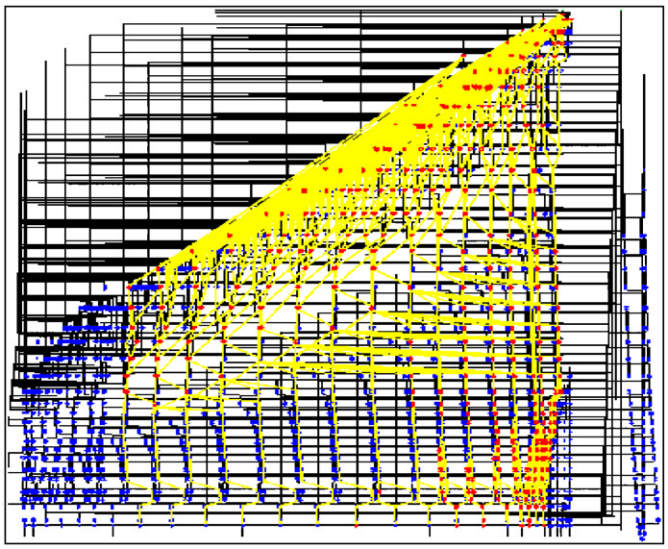


Fig. 12. Schematic view of the 16-bit multiplier design.

6. Discussion

To further assess the performance of this dual-threshold voltage design flow, multipliers meeting different operating frequencies are generated, and their static power dissipation is measured by the power estimation tool. Fig. 15 illustrates that the dual-V_t multiplier dissipates less static power in comparison with the all-SVT multiplier one. It also shows that GDSPOM efficiently assigns fewer SVT cells in the lower speed target and more SVT cells in faster operation requirement.

7. Conclusion

In this paper, a novel GDSPOM, which is based on the static timing analysis technique for designing high-speed low-power SOC applications using 90 nm MTCMOS technology has been reported. Based on this optimization technique, and with the use of two cell libraries of different threshold voltages, a 16-bit multiplier meeting the speed requirement has been designed to have a 50% less power

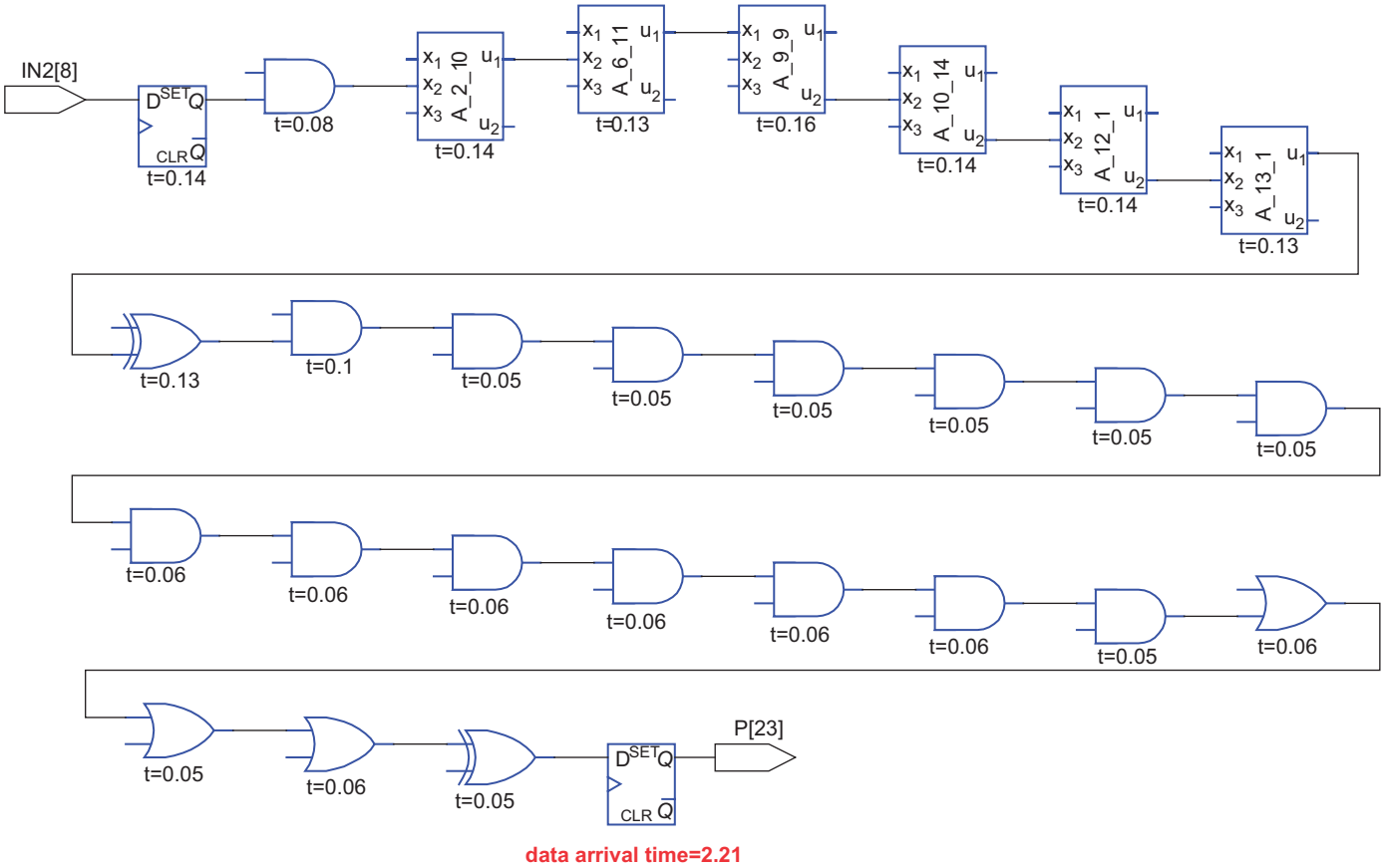


Fig. 13. Timing path from IN28 to P23 in an all-HVT multiplier.

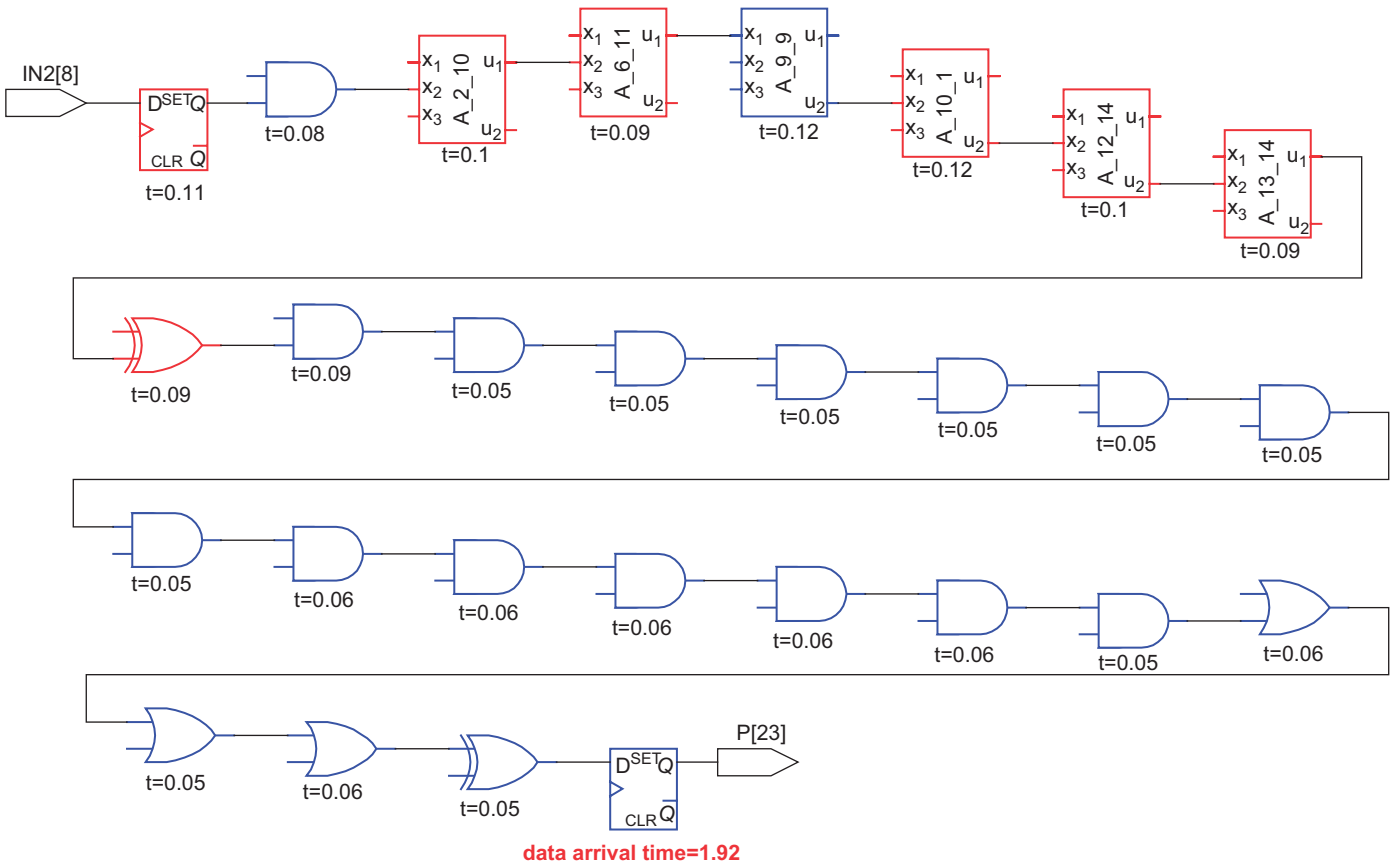


Fig. 14. Timing path from $IN2_8$ to P_{23} in a dual-threshold multiplier.

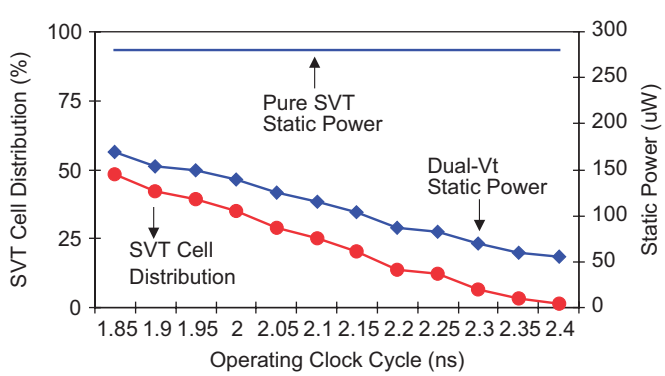


Fig. 15. Power consumption of the 16-bit multiplier using all-HVT cells, all-SVT cells, and dual-threshold HVT/SVT cells operating at different frequencies.

consumption compared to the all low-threshold voltage one.

Not only GDSPOM can be used to design new chips, but also to improve the existing products. The existing high V_{th} designs can use GDSPOM to boost their speed performance with the minimum leakage power increase. Because GDSPOM does not alter the design architecture to meet the final speed target, there is no area overhead.

The existing low V_{th} designs can be converted to high V_{th} ones and go through GDSPOM to produce a dual-V_{th} version of the same product, which keeps the original speed performance but dissipates less leakage power.

References

[1] J. Kuo, J. Lou, Low-Voltage CMOS VLSI Circuits, Wiley, New York, 1999.

[2] J. Kao, S. Narendra, A. Chandrakasan, MTCMOS hierarchical sizing based on mutual exclusive discharge patterns, in: Design Automation Conference Proceedings, June 1998, pp. 495–500.

[3] K. Usami, N. Kawabe, M. Koizumi, K. Seta, T. Furusawa, Automated selective multi-threshold design for ultra-low standby applications, in: Low Power Electronics and Design Conference Proceedings, 2002, pp. 202–206.

[4] L. Wei, Z. Chen, M. Johnson, K. Roy, and V. De, Design and optimization of low voltage high performance dual-threshold CMOS circuits, in: Design Automation Conference Proceedings, June 1998, pp. 489–494.

[5] D. Samanta, A. Pal, Optimal Dual-VT Assignment for low-voltage energy-constrained CMOS circuits, in: International Conference on VLSI Design, January 2002, pp. 193–198.

[6] Q. Wang, S. Vrudhula, Algorithms for minimizing standby power in deep submicrometer, dual-V_t CMOS circuits, IEEE Trans. Comput.-Aided Des. IC Syst. 21 (3) (2002) 306–318.

[7] Artisan, TSMC 90 nm CLN90G Process SAGE-X v3.0 Standard Cell Library Databook.

[8] Artisan, TSMC 90 nm CLN90G HVt Process 1.0-Volt SAGE-X v3.0 Standard Cell Library Databook.

[9] Synopsys, PrimeTime User Guide, v2004.12.
[10] C.S. Wallace, A suggestion for a fast multiplier, IEEE Trans. Comput. EC-13 (1964) 14–17.

Benjamin Chung received his B.A.Sc. degree in electrical engineering from the University of British Columbia, Vancouver, Canada, in 2001, and his M.A.Sc. degree from Simon Fraser University, Burnaby, Canada, in 2005.

He is currently with PMC-Sierra, Burnaby, Canada. His research interests include high-performance and low-power circuit techniques and algorithms in optimization.

Professor James B. Kuo Professor James B. Kuo received a Ph.D.EE. degree from Stanford University in 1985. From 1985 to1987, he was a research associate in IC Lab of Stanford. Since 1987, he has been with National Taiwan University, where he currently is a professor. His research expertise is in the field of low-voltage CMOS VLSI circuits and compact modeling of CMOS VLSI devices. He became an IEEE fellow in 1999 for contributions to modeling CMOS VLSI devices. He served as the VP of membership for the IEEE Electron Devices Society. He has published 300 technical papers and authored eight textbooks on CMOS VLSI. He has graduated 80 graduate students specialized in CMOS VLSI working in leading microelectronics companies.