GATE-LEVEL DUAL-THRESHOLD STATIC POWER OPTIMIZATION METHODOLOGY (GDSPOM) FOR DESIGNING HIGH-SPEED LOW-POWER SOC APPLICATIONS USING 90NM MTCMOS TECHNOLOGY

by

Benjamin Chung
B.A.Sc., University of British Columbia, 2001

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

In the
School
of
Engineering Science

SIMON FRASER UNIVERSITY

Fall 2005

APPROVAL

Name: **Benjamin Chung**

Degree: **Master of Applied Science**

Title of Thesis: **GATE-LEVEL DUAL-THRESHOLD STATIC POWER OPTIMIZATION METHODOLOGY (GDSPOM) FOR DESIGNING HIGH-SPEED LOW-POWER SOC APPLICATIONS USING 90NM MTCMOS TECHNOLOGY**

Supervisory Committee:

Chair: **Dr. John S. Bird**
Professor of the School of Engineering Science

Dr. James B. Kuo
Senior Supervisor
Professor of the School of Engineering Science

Dr. Marek Syrzycki
Supervisor
Professor of the School of Engineering Science

Dr. Karim S. Karim
Internal Examiner
Assistant Professor of the School of Engineering Science

Date Defended/Approved: Thursday, December 1, 2005

ABSTRACT

As integrated-circuits (IC) technology advances into the deep-submicron (DSM) regime, more functionality can be combined onto a single chip. One major challenge in designing such a complex device is to keep the power consumption in check while capitalizing on the highest performance that DSM technology can offer.

In this thesis we describe a novel gate-level dual-threshold static power optimization methodology (GDSPOM), which is based on the static timing analysis technique for designing high-speed low-power SOC applications using 90nm MTCMOS technology. The cell libraries come in fixed threshold – high V_t for good standby power and low V_t 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.

DEDICATION

To my wife, Mila, for her love, support, and believing in me throughout our years together.

ACKNOWLEDGEMENTS

I would like to thank my academic supervisor, Professor James Kuo, for his guidance, motivation, and technical advice during the entire Masters program. Following his guidance, I have learned how to conduct research and look at issues from a broader point of view.

I would also like to thank PMC-Sierra for providing me with real industrial experience. Special thanks to Karim Arabi, Ken Brough, Alan Nakamoto, and Ken Wagner for supporting my graduate studies.

Finally, I would like to thank my family and all SFU and PMC friends for making my life exciting and colourful.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF ACRONYMS

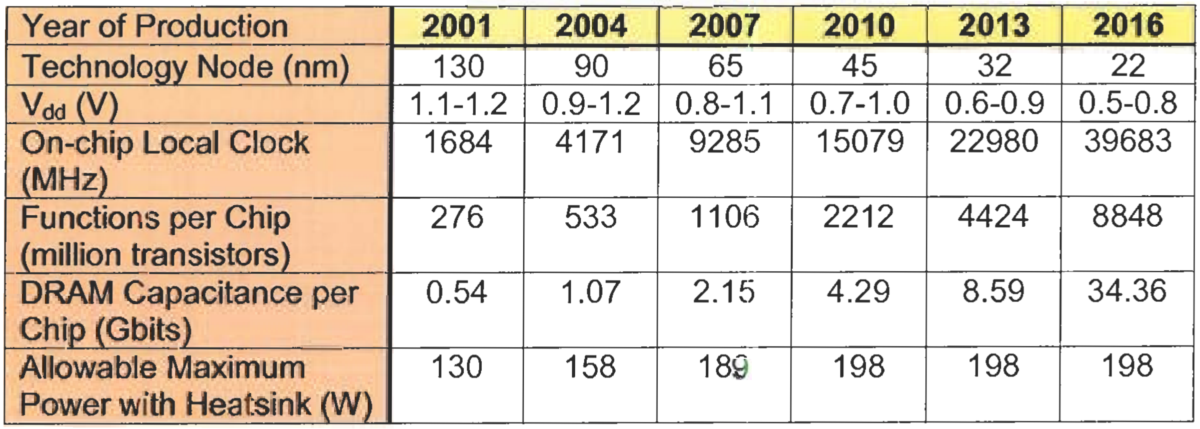
CAD	Computer Aided Design
CMOS	Complementary Metal Oxide Semiconductor
DSM	Deep Sub-micron
GDSPOM	Gate-level Dual-threshold Static Power Optimization Methodology
HVT	High Threshold Voltage
IC	Integrated Circuits
ITRS	International Technology Roadmap for Semiconductors
MTCMOS	Multiple Threshold CMOS
RTL	Register Transfer Language
SOC	System on Chip
STA	Static Timing Analysis
SVT	Standard Threshold Voltage
TCL	Tool Command Language
VLSI	Very Large Scale Integration

CHAPTER 1: INTRODUCTION

1.1 CMOS VLSI Trends

CMOS technology has evolved into sub-micron regime [1]. The mainstream process is 90nm, and 65nm is coming in the near future. Table 1.1 summarizes the VLSI trends and the numbers are from International Technology Roadmap for Semiconductors (ITRS) 2004 update [2].

Table 1.1: Technology Roadmap for Semiconductors

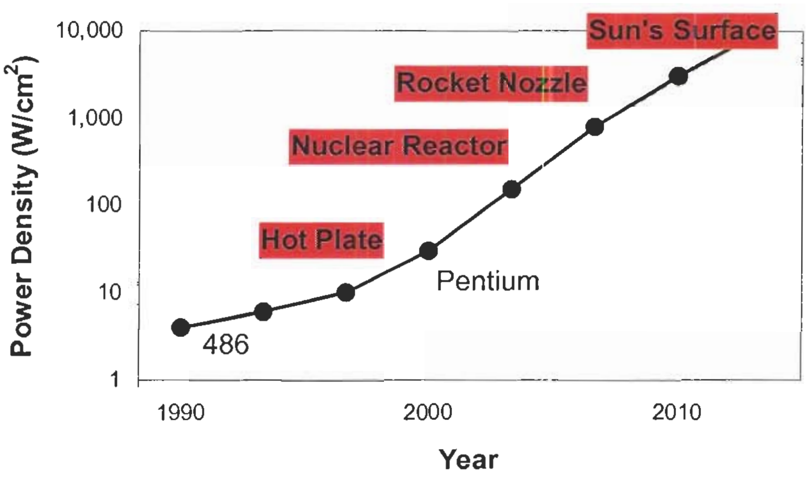


Year of Production	**2001**	**2004**	**2007**	**2010**	**2013**	**2016**
Technology Node (nm)	130	90	65	45	32	22
V_{dd} (V)	1.1-1.2	0.9-1.2	0.8-1.1	0.7-1.0	0.6-0.9	0.5-0.8
On-chip Local Clock (MHz)	1684	4171	9285	15079	22980	39683
Functions per Chip (million transistors)	276	533	1106	2212	4424	8848
DRAM Capacitance per Chip (Gbits)	0.54	1.07	2.15	4.29	8.59	34.36
Allowable Maximum Power with Heatsink (W)	130	158	189	198	198	198

Source: ITRS, http://www.itrs.net/Common/2004Update/2004_000_ORTC.pdf

Following the trends on transistor size scaling down, Gordon Moore's prediction that transistor counts will double every two years (Moore's Law) [3] is still valid for the next decade. However, chip power density is approaching the physical barrier and will limit chip growth if there are no breakthroughs in power reduction [4]. Figure 1.1 illustrates that it is not possible to continue the chip growth trend if power problem is not solved.

Figure 1.1: Power Extrapolation



Source: Pat Gelsinger's Slide from ISSCC 2001, ftp://download.intel.com/technology/silicon/TeraHertzshort.pdf

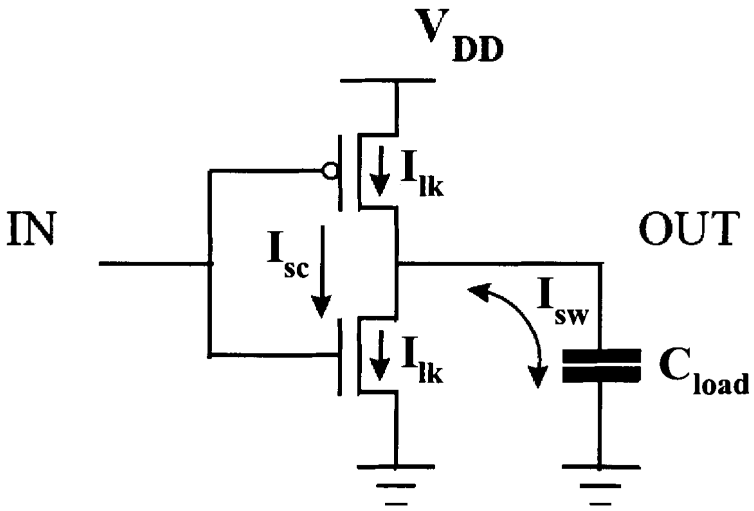
Other advantages of low power designs are as follows: more reliable device operation, cheaper and lighter power supplies, and less expensive cooling system. Low power designs also enable portable products like laptops or mobile telephones to have longer battery life, lighter weight, and smaller size.

Therefore, implementing low power structures has become a required feature in developing a chip, and discovering low power strategies is one of the most active research fields.

1.2 Components of Power Dissipation

A circuit dissipates two kinds of powers: dynamic power and static power [5]. Figure 1.2 shows typical current flow in an active inverter circuit.

Figure 1.2: Current Flow in a CMOS Inverter



Source: Power Compiler User Guide, v2004.12 [5]

Dynamic power is the power dissipated when a cell's input value changes. It contains cell internal power and net switching power. Cell internal power is the power dissipated by the momentary short circuit between power source and ground when a circuit switches, as well as by charging and discharging the internal cell capacitances. Short-circuit current, I_{sc} in Figure 1.2, is the source of cell internal power. Cell internal power is below 10-15% of the total power [6] and can be calculated with the formula:

$$P_{\text{int}ernal} = I_{SC} \cdot V_{dd}$$

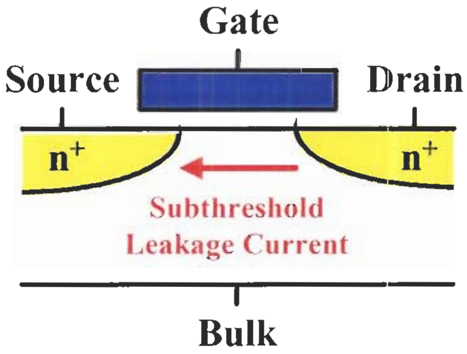
Net switching power is the power dissipated due to charging and discharging of the capacitive load at cell's outputs. This is the dominant component of dynamic power. As shown in Figure 1.2, switching current, $I_{sw,}$ generates net switching power. The value of net switching power is proportional to the logic transition rate of the circuit. Clock frequency, f_{clk}, and output switching factor, α, in the circuit define logic transition rate. Net switching power can be represented as:

$$P_{switching} = C_{load} \cdot V_{dd} \cdot \alpha \cdot f_{clk}$$

Static power is the power dissipated all the time, even when a circuit is held in a steady state [7]. Leakage current, I_{lk} in Figure 1.2, exists because, in reality, a transistor is not an ideal switch. Figure 1.3 shows sources of leakage of currents. Leakage power formula is:

$$P_{leakage} = I_{lk} \cdot V_{dd}$$

Figure 1.3: Leakage Power Sources



Source: Leakage Mechanisms and Leakage Control for Nano-Scale CMOS Circuits [7]

The main component of static power comes from a sub-threshold leakage current. This current runs from source to drain on a transistor even when the transistor is turned off. The amount of sub-threshold leakage current is a function of the threshold voltage: high threshold voltage device has smaller sub-threshold leakage current than low threshold voltage device does [8]. The sub-threshold current of a MOS transistor is approximated as [9]:

$$I_{subthreshold} \approx Ke^{\frac{V_{gs}-V_t}{nV_T}} \cdot \left(1 - e^{\frac{-V_{ds}}{V_T}}\right)$$

where

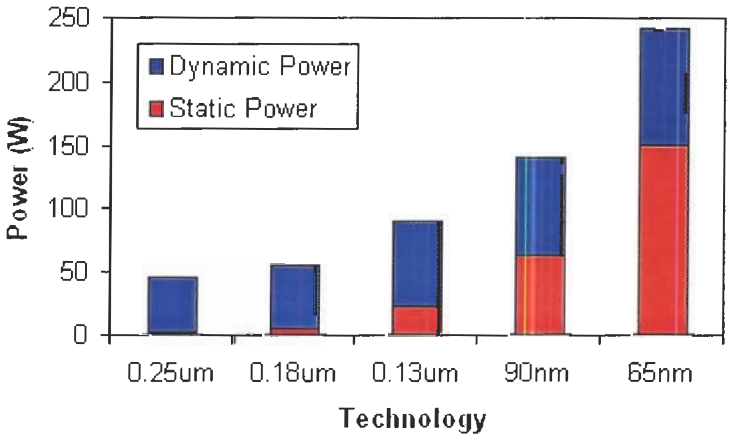
g	= gate terminal
s	= source terminal
d	= drain terminal
V_t	= threshold voltage
V_T	= thermal voltage
K, n	= technology dependent constants

Total circuit power can be represent as:

$$P_{total} = P_{internal} + P_{switching} + P_{leakage}$$

As technology entered the deep sub-micron (DSM) regime, static power became a significant component of the total circuit power. Static power can be larger than dynamic power as shown in Figure 1.4; therefore, static power optimization technique is important in the DSM devices. This is the motivation behind this research work.

Figure 1.4: Dynamic Power Vs. Leakage Power in Various Technologies



Source: Power Modelling and Leakage Reduction,
http://eda.ee.ucla.edu/EE201A-04Spring/leakage_pres.ppt

1.3 Research Goal

The current IC industry is very competitive. A product's cost and time-to-market schedule are usually the most important factors to be successful in this market [10]. Under limited project timeline and budget, it is a challenge to efficiently develop a low power chip without sacrificing its performance. The research goal is to define a practical VLSI design flow for leakage power reduction.

This design flow will need to meet the following objectives:

1. It is easy to use and its runtime is reasonable.
2. It can handle VLSI and system-on-chip (SOC) designs.
3. It is back compatible. In other words, it can be used to optimize the existing designs.

Because of these usage features, the research is focused on applying available process technologies and computer aided design (CAD) tools with minor program implementations.

The research process contains the following stages:

1. Build a 16-bit multiplier following the existing design flow. Measure the multiplier's speed and leakage power with a simulation based power estimation tool.
2. Identify potential low power techniques and commercially available CAD tools to use.
3. Implement low leakage power design flow.

4. Build another 16-bit multiplier following the new flow. Estimate its speed and leakage power with simulation-based approach.
5. Compare both multipliers' speed and power performances. Analyze the comparison data and further improve the leakage power reduction design flow.

The final research result is a gate-level dual-threshold static power optimization methodology (GDSPOM). GDSPOM borrows the multiple threshold CMOS (MTCMOS) concept and applies the static timing analysis (STA) approach to minimize leakage power in VLSI designs.

1.4 Organization of the Thesis

Chapter 2 gives an overview of prior work on MTCMOS leakage power reduction techniques. It first explains the basic MTCMOS principle as a background. Then, it describes the proposed algorithms and experiment results. It also points out the limitations of these MTCMOS strategies.

Chapter 3 presents GDSPOM. First, it describes the characteristics of applied cell libraries and models. Second, it presents GDSPOM with detailed descriptions of STA approach in the flow. Then, it compares two 16-bit multipliers created from non-GDSPOM and GDSPOM flows. Finally, it summarizes GDSPOM power reduction results from designing 16-bit multipliers in different required operating frequencies.

Chapter 4 is the conclusion of this thesis. It also provides directions for future research.

CHAPTER 2: PRIOR WORK ON MTCMOS OPTIMIZATION ALGORITHMS

This chapter presents three published papers of leakage power reduction algorithms using the concept of MTCMOS technology. The idea of using MTCMOS to save power is described first. Then, three papers with different approaches are presented. Limitations of these methods are analysed at the end of this chapter.

2.1 MTCMOS Principle

Threshold voltage (V_t) controls current and signal propagation delay of a MOS device. A transistor with high threshold voltage has low leakage current and long signal propagation delay; a transistor with low threshold voltage has high leakage current and short propagation delay [8].

The basic idea of designing a low leakage power circuit with MTCOMS is as the follows:

1. place low V_t transistors in timing critical paths to satisfy the circuit operating performance requirements,
2. place high V_t transistors off timing critical paths to minimize the circuit leakage current.

It is a challenge to efficiently distribute different V_t cells. Different MTCMOS Vt assignment algorithms have been proposed in prior work and will be presented in the remaining sections.

2.2 Low V_t to High V_t Algorithms

Wei *et al.* proposed two algorithms to replace low V_t cells with high V_t ones: breadth-first search [11] and levelized search [12]. The starting circuit contains all low V_t transistors. A higher V_t transistor with predetermined V_t value is used to replace as many low V_t transistors as possible.

Both algorithms use the same STA tool. In the timing initialization step, each cell's signal arrival time and required time are calculated. The time difference between the arrival time and the required time is defined as "slack". Positive slack indicates the amount of time during which a gate may be slowed down without affecting the circuit speed performance. The path with the longest propagating delay is called a critical path. Cells on critical path have zero 0 slack value.

$$AT_{cell} = \max\{AT_{cell}(fanin)\} + D_{cell}$$
$$RT_{cell} = \min\{RT_{cell}(fanout)\}$$
$$S_{cell} = RT_{cell} - AT_{cell}$$

where

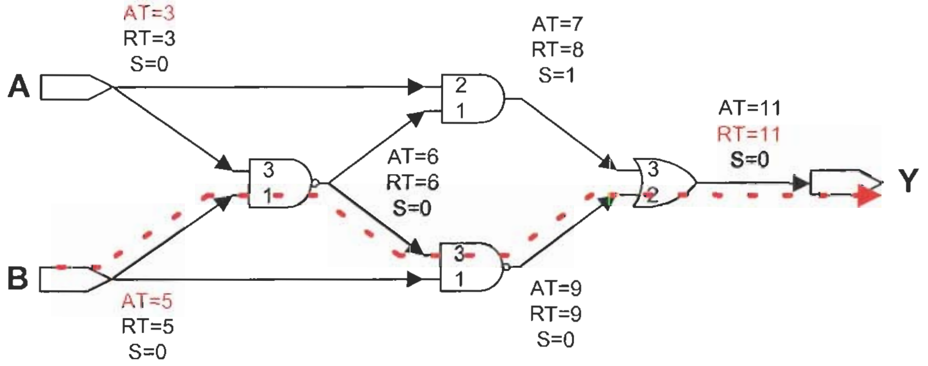
AT_{cell} = cell arrival time

RT_{cell} = cell required time

S_{cell} = cell slack

Figure 2.1 shows a critical path from B to Y and calculated timing values of each cell after the initial STA step. As timing analysis is done in cell bases, it is known as a cell based STA approach [13].

Figure 2.1: Cell Timing Values after STA



Sorce: VISIO Drawing

2.2.1 Breadth-first Search

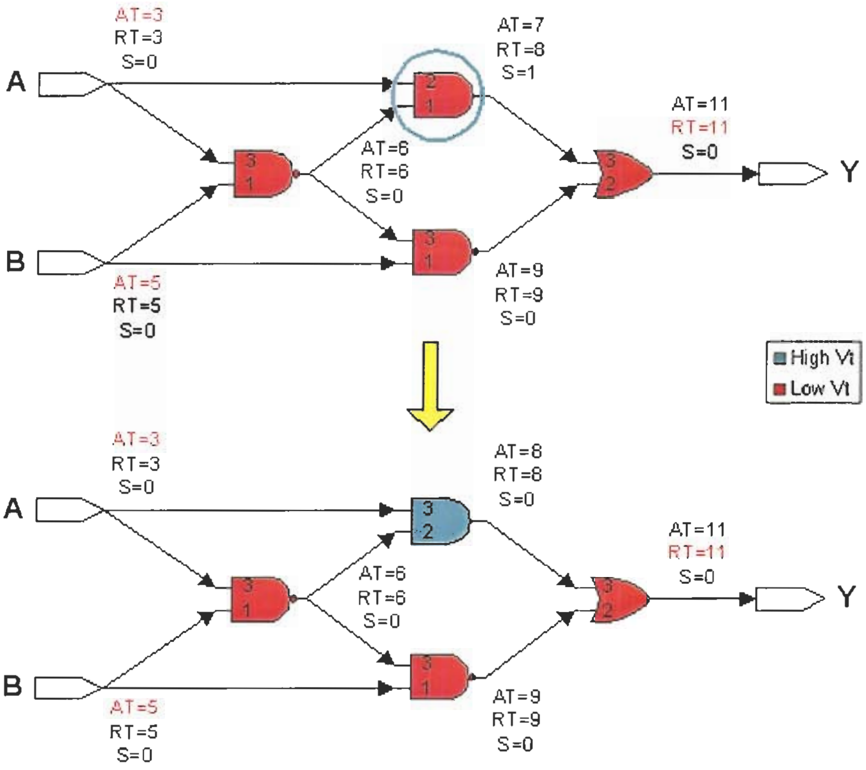
In breadth-first search algorithm, cells are traced backwards starting from one primary output. If a cell's slack value is positive, check if its slack value is still positive after changing its type from low V_t to high V_t. If the new slack value is positive, allow the cell type change, otherwise, keep the original low V_t cell type. When search nodes reach primary inputs, a new search starts again from another primary output. The search stops after all cells are checked. Algorithm 2.1 presents the breadth-first search procedure. Figure 2.2 illustrates how the breadth-first search algorithm works.

Algorithm 2.1: Breadth-first Search

```
Procedure breadthFirstSearch ($inputNetlist, $highVt) {
    @poArray = all primary outputs in $inputNetlist

    foreach $po (@poArray) {
        foreach $cell on paths to $po {
            $type = $cell type
            $slack = $cell slack
            $visited = $cell checked

            if ($type == lowVt && $slack > 0 && $visited == false) {
                $slackHvt = $highVt slack

                    if ($slackHvt > 0) {
                        replace $cell to high vt one
                    } else {
                        keep $cell
                        $visited = true
                    }
                }
            }
        }
    }
}
```

Source: Design and Optimization of Low Voltage High Performance Dual Threshold CMOS Circuits [11]

Figure 2.2: Breadth-first Search Example



Source: Design and Optimization of Low Voltage High Performance Dual Threshold CMOS Circuits [11]

2.2.2 Levelized Search

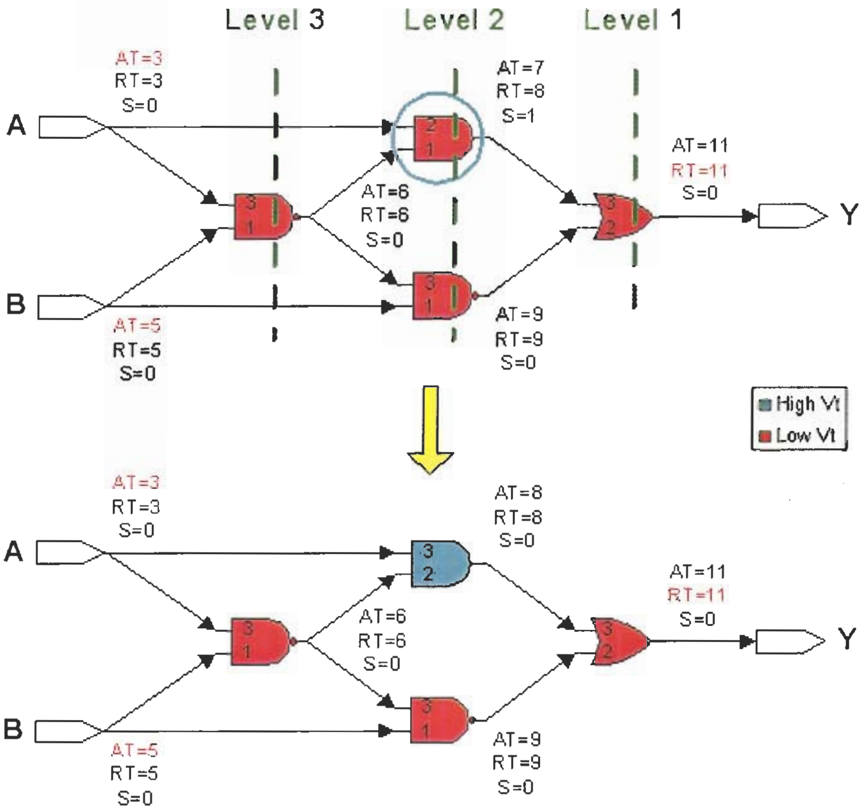
In levelized search, all cells are assigned a level number. Primary outputs have level number 0; cells connect directly to primary outputs have level number 1; cells close to primary inputs have higher-level numbers. The search loop checks cells from maximum level and stops when it reaches level 0. Levelized search is more efficient than breadth-first search and the search procedure is outlined in Algorithm 2.2. Figure 2.3 shows a levelized search example.

Algorithm 2.2: Levelized Search

```
Procedure levelizedSearch ($inputNetlist, $highVt) {
    $currentLevel = maximum level

    while ($currentLevel > 0) {
        foreach $cell on $currentLevel {
            $type = $cell type
            $slack = $cell slack

            if ($type == lowVt && $slack > 0) {
                $slackHvt = $highVt slack

                    if ($slackHvt > 0) {
                        replace $cell to high vt one
                    } else {
                        keep $cell
                    }
                }
            }
        }

        $currentLevel--
    }
}
```

Source: Design and Optimization of Dual-threshold Circuits for Low-voltage Low-power Applications [12]

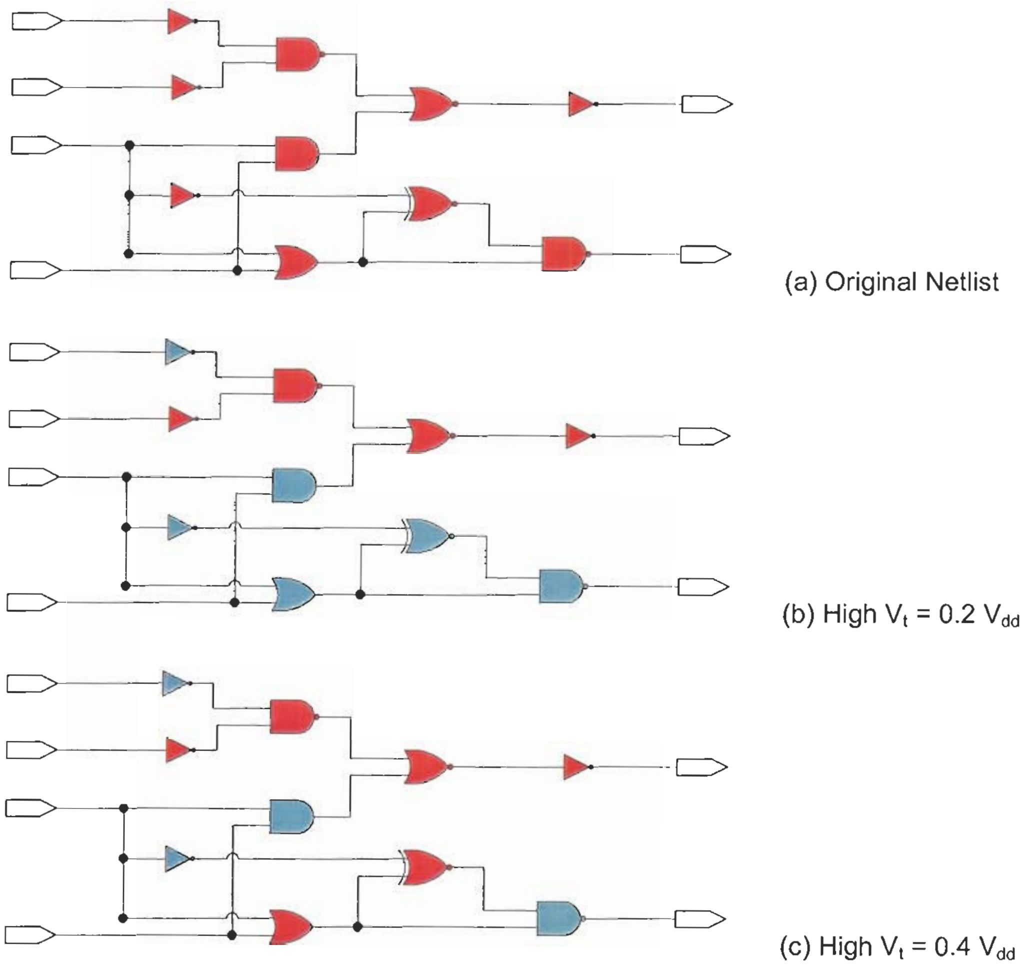
Figure 2.3: Levelized Search Example



Source: Design and Optimization of Dual-threshold Circuits for Low-voltage Low-power Applications [12]

A slightly higher than initial V_t value cell reduces small amount of leakage current and introduces short signal delay. Therefore, more cells in a pure low V_t design can be assigned to this type. A much higher than initial V_t value cell reduces leakage current more efficiently. However, because it also increases a significant amount of delay time, only few cells can be assigned to this type. As shown in Figure 2.4, the higher the replacing V_t value is, the fewer cells can be replaced.

Figure 2.4: High V_t Assignment Results Using Different High V_t Values



(a) Original Netlist

(b) High V_t = 0.2 V_{dd}

(c) High V_t = 0.4 V_{dd}

Source: Design and Optimization of Dual-threshold Circuits for Low-voltage Low-power Applications [12]

Because different replacing high V_t values result in different amount of leakage current savings, Wei *et al.* provides an additional algorithm to find the optimal high V_t value for cell replacing. Algorithm 2.3 is the optimal high V_t value search procedure. This procedure loops the levelized search process by trying different replacing high V_t values. Then, resulting leakage power values are stored and compared. Finally, the optimal high V_t value, which produces the circuit with the least leakage power is reported.

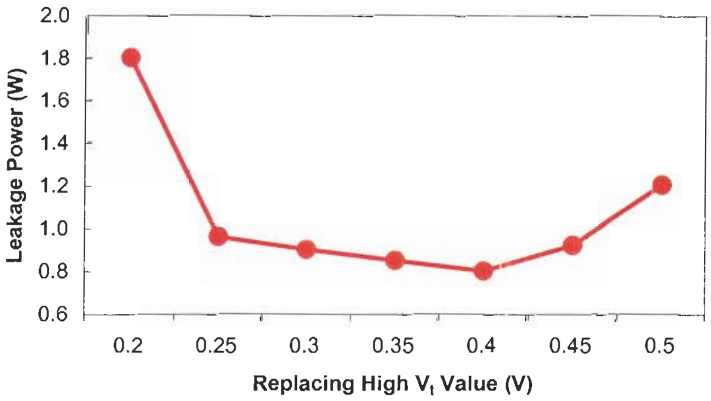
Algorithm 2.3: Optimal High V_t Search

```
Procedure optimalHighVtSearch ($inputNetlist, @highVtArray) {
    $minLeakagePower = measuer $inputNetlist leakage power
    $optimalVt = “”

    foreach $highVt (@highVtArray) {
        &levelizedSearch($inputNetlist, $highVt)
        $leakagePower = measure result netlist’s leakage power

        if ($leakagePower < minLeakgePower) {
            $minLeakagePower = $leakagePower
            $optimalVt = $highVt
        }
    }
}
```

Source: Design and Optimization of Dual-threshold Circuits for Low-voltage Low-power Applications [12]

During optimal high V_t search process, leakage powers of resulting circuits using different replacing V_t values are measured. The power measurement results shown in Figure 2.5 indicate that 0.4 V threshold voltage is the best pick for this particular experiment circuit.

Figure 2.5: Leakage Powers Vs. Replacing V_t values



Source: Design and Optimization of Dual-threshold Circuits for Low-voltage Low-power Applications [12]

2.3 High V_t to Low V_t Algorithm

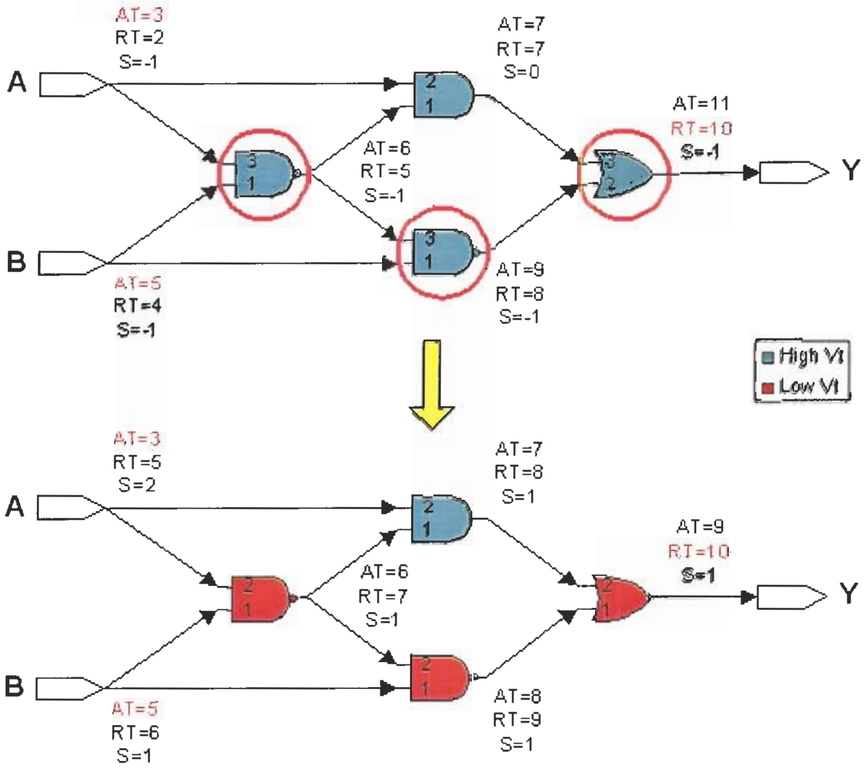
Samanta and Pal modified Wei *et al.*'s breadth-first search algorithm in Chapter 2.2.1 to replace high V_t cells to low V_t ones to satisfy the signal timing constraints [14]. In the search loop, recalculation of cell timing values is performed after tracing through each critical path. Existing critical paths change dynamically due to cell type swapping function in the search process. Performing STA periodically addresses dynamic critical path changing issues. The detailed procedure is shown in Algorithm 2.4 and an example is shown in Figure 2.6.

Algorithm 2.4: High V_t to Low V_t Breadth-first Search

```
Procedure lowVtSearch ($inputNetlist) {
    repeat {
        @toChangeArray = all nodes on critical path
        change @toChangeArray to low Vt
        perform STA
    } until circuits meets timing requirements
}
```

Source: Optimal Dual-VT Assignment for Low-voltage Energy-Constrained CMOS Circuits [14]

Figure 2.6: High V_t to Low V_t Breadth-first Search Example



Source: Optimal Dual-VT Assignment for Low-voltage Energy-Constrained CMOS Circuits [14]

2.4 Fine-Grained V_t Assignment Algorithms

Wang and Vrudhula proposed three fine-grained V_t assignment algorithms: Min-Cut, Max-Cut I, and Max-Cut II [15]. Min-Cut and Max-Cut II replace high V_t transistors to low V_t ones while Max-Cut I exchanges low V_t transistors to high V_t ones. In Chapter 2.2 and 2.3, Wei and Samanta's algorithms assign same V_t transistors to each individual logic block. In fine-grained V_t assignment approach, transistors inside the same logic cell may have different V_t values. For example, gate Z may have high V_t transistors whose input values are driven by the output of gate A and low V_t transistors whose input values are driven by gate B.

2.4.1 Minimum Cut

Min-Cut algorithm starts with a circuit, which contains all high V_t transistors and has timing violations. Use a circuit graph to represent an input circuit, a node is a logic cell partition, an edge is a connection between two nodes. When an edge does not meet timing constraints and one end connects to a high V_t cell, this edge's weight is assigned with the following formula:

$$W_{edge} = \Delta P + \frac{\alpha}{\Delta AT}$$

where

W_{edge} = edge weight

ΔP = power increase after replacing one node to low V_t

ΔAT = arrival time reduction after replacing one node to low V_t

α = scalar factor to balance ΔP and ΔAT

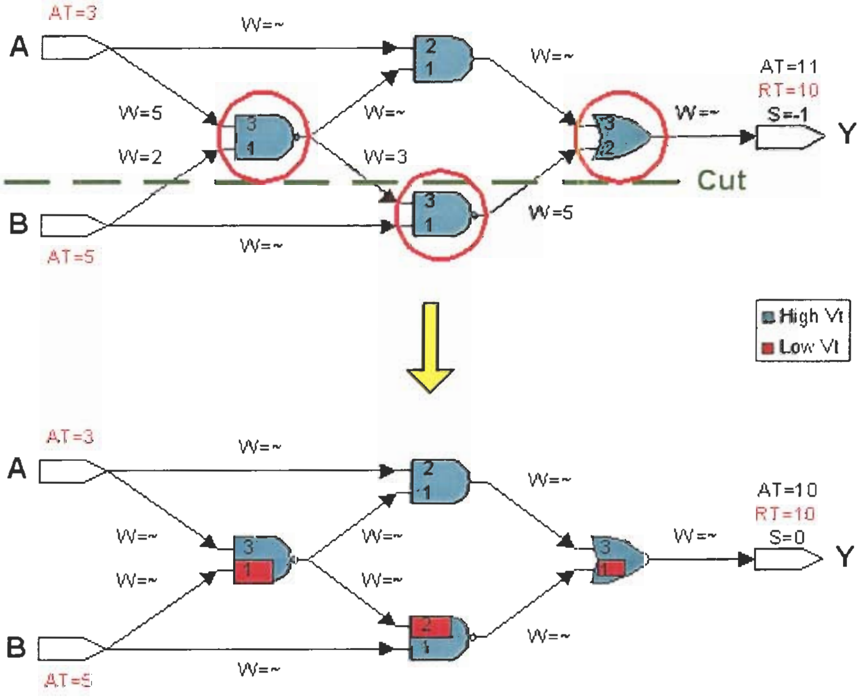
In all other cases, edges have infinity ∞ value. A minimum cut through this circuit graph reveals a set of edges, which, after lowering threshold voltage in one end, causes the minimum leakage power increase and provides the maximum signal speed improvements. Algorithm 2.5 states the minimum cut search procedure. Figure 2.7 demonstrates a simple minimum cut search example.

Algorithm 2.5: Minimum Cut Search

```
Procedure minCut ($inputNetlist) {
    perform static timing anlysis
    compute weights
    $stop = false

    while ($stop == false) {
        $cutSet = minimum weight cut of $inputNetlist
        @candidates = all edges in $cutSet

        if (@candidates = NULL) {
            $stop = true
        } else {
            change all edges in @candidates to low Vt
            perform static timing analysis
            compute weights
        }
    }
}
```

Source: Algorithms for Minimizing Standby Power in Deep Submicrometer, Dual-V_t CMOS Circuits [15]

Figure 2.7: Minimum Cut Search Example



Source: Algorithms for Minimizing Standby Power in Deep Submicrometer, Dual-V_t CMOS Circuits [15]

2.4.2 Maximum Cut I

Max-Cut I algorithm starts with a circuit, which contains all low V_t transistors and does not have timing violations. When one end of the edge can be switched to high V_t node without causing a timing violation, its edge weight is assigned using this formula:

$$W_{edge} = \Delta P$$

where

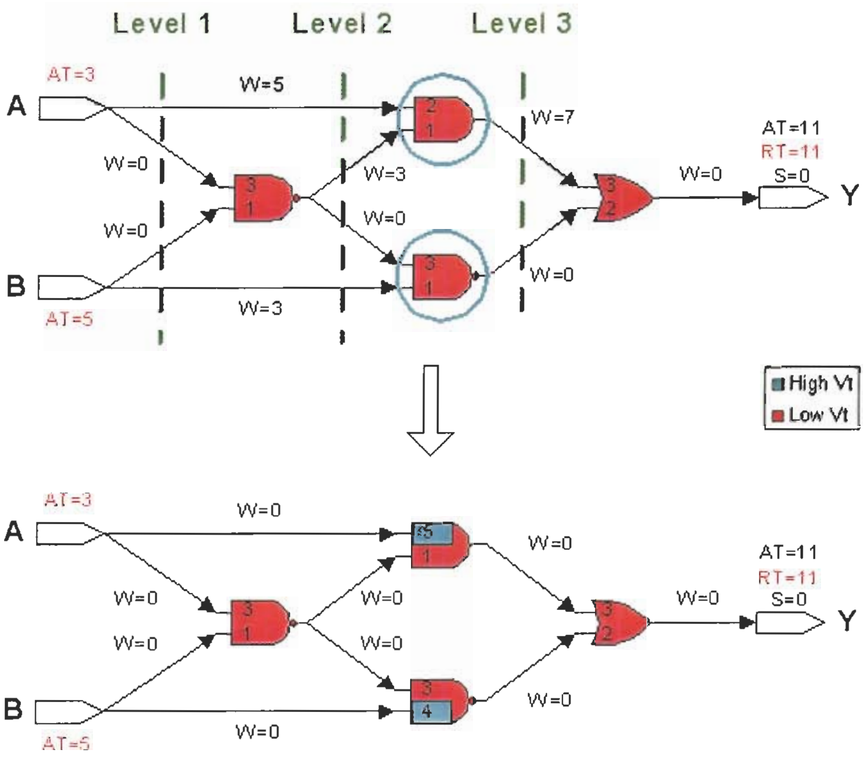
W_{edge} = edge weight

ΔP = leakage power reduction after replacing one node to high V_t

In all other cases, edges have zero 0 value. A maximum cut through this circuit graph reveals a set of edges, which, after raising threshold voltage in one end, causes maximum leakage power reduction without affecting the circuit performance. To address the situation when a node's V_t value changing affects its leaf nodes' timing, cut is only allowed in the level bases. This level restriction is driven from the fact that there is no timing dependency between edges in the same level.

Figure 2.8 illustrates an example of level cut procedure. First, initial circuit's edge weights are calculated. The level 1 has the maximum total edge weight of 8 and, therefore, level 1 cut is chosen for high V_t replacement. After the cell type change, new edge weights are calculated again with new values shown in the second part of the figure.

Figure 2.8: Maximum Cut I Search Example



Source: Algorithms for Minimizing Standby Power in Deep Submicrometer, Dual-V_t CMOS Circuits [15]

The detailed max cut I procedure is stated in Algorithm 2.6.

Algorithm 2.6: Maximum Cut I Search

```
Procedure maxCutI ($inputNetlist) {
    perform static timing anlysis
    compute weights
    $stop = false

    while ($stop == false) {
        compute weights
        $maxCut = 0
        $maxWeight = 0

        foreach $levelCut {
            $levelWeight = total weight of $levelCut

            if ($levelWeight > $maxWeight) {
                $maxWeight = $levelWeight
                $maxCut = $levelCut
            }
        }

        if ($maxCut != 0) {
            change all edges in $maxCut to high Vt
            perform static timing analysis
            compute weights
        } else {
            $stop = true
        }
    }
}
```

Source: Algorithms for Minimizing Standby Power in Deep Submicrometer, Dual-V_t CMOS Circuits [15]

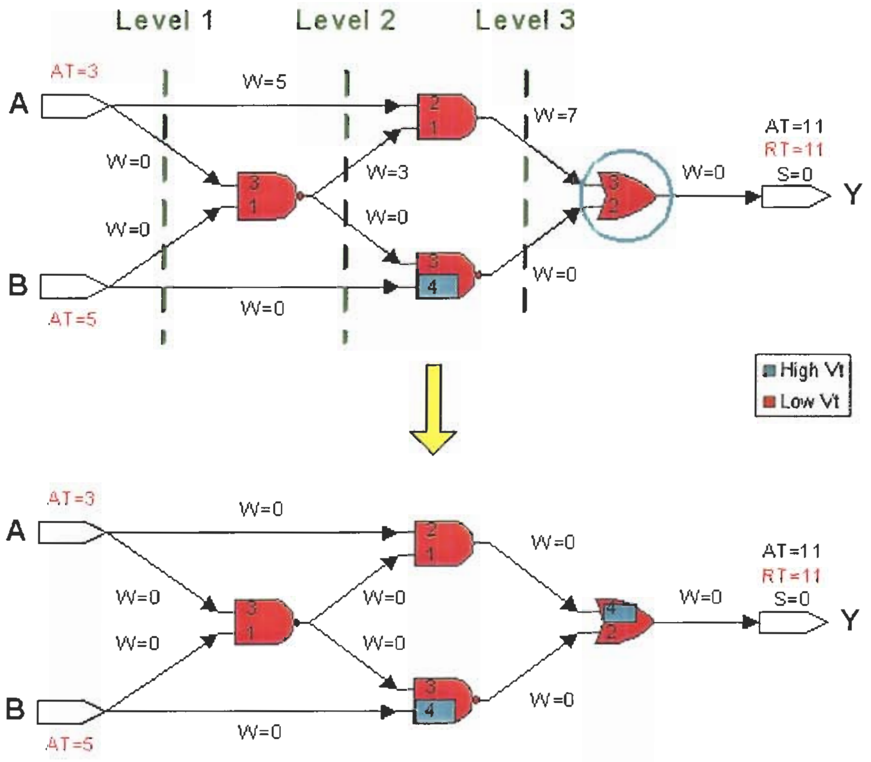
Furthermore, to avoid a locally optimal solution due to level based approach, an additional swap testing procedure is performed after max cut I search. The swap testing procedure takes the product circuit from max cut I as an input. After returning one high V_t node back to low V_t, if more beneficial V_t swapping can be found in other nodes, the high V_t node relocation process is performed. Otherwise, original configurations are restored. Algorithm 2.7 states the swap testing procedure. Following Figure 2.8 example, the swap testing process is illustrated in Figure 2.9.

Algorithm 2.7: Swap Test

```
Procedure swapTest ($inputNetlist) {
    @highVtEdges = high Vt edges in $inputNetlist

    foreach $edge (@highVtEdges) {
        change $edge to low Vt
        perform static timing analysis
        compute weights

        $cost = $edge weight
        $gain = maximum weight sum of level cut

        if ($gain > $cost) {
            $inputNetlist = changed $inputNetlist
        } else {
            restore $inputNetlist
        }
    }
}
```

Source: Algorithms for Minimizing Standby Power in Deep Submicrometer, Dual-V_t CMOS Circuits [15]

Figure 2.9: Swap Test Example

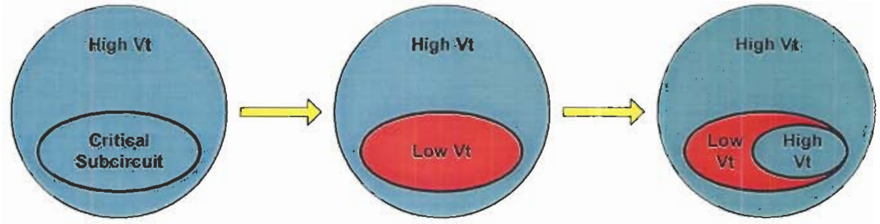


Source: Algorithms for Minimizing Standby Power in Deep Submicrometer, Dual-V_t CMOS Circuits [15]

2.4.3 Maximum Cut II

Max-Cut II algorithm starts with a circuit containing only high V_t transistors. It identifies critical areas and converts all transistors inside the critical areas to low V_t type. Then, Max-Cut I is performed to these critical areas to recover some transistors back to high V_t. Figure 2.10 is the flow chart and Algorithm 2.8 is the procedure of maximum cut II.

Figure 2.10: Maximum Cut II Flow



Source: Algorithms for Minimizing Standby Power in Deep Submicrometer, Dual-V_t CMOS Circuits [15]

Algorithm 2.8: Maximum Cut II

```
Procedure maxCutII ($inputNetlist) {
    @subcircuitArray = critical subcircuits in $inputNetlist

    foreach $subcircuit (@subcircuitArray) {
        replace each edge of $subcircuit to low Vt
        perform maxCutI($subcircuit)
    }
}
```

Source: Algorithms for Minimizing Standby Power in Deep Submicrometer, Dual-V_t CMOS Circuits [15]

The experiment shows that Max-Cut II is the best algorithm among the three. Its runtime is faster than the runtime of others because only subcircuits will go through dual-V_t optimization process. Min-Cut has the worst leakage power reduction performance due to the fact that its edge weight formula contains scalar factor and infinite value. The inaccurate weight values may leads to sub-optimal minimum cut solution.

2.5 Limitations

All of these published work achieved leakage power reduction results. However, all of them focus on how to assign dual V_t cells on pure combinational logic circuits and assume there exists only one clock source. There is no strategy of assigning different V_t values to sequential elements as well as calculating cell timing values on multiple clock domains. Although optimal high V_t value can be identified with Algorithm 2.3, availability of specific V_t cell libraries is limited [16]. Moreover, there is no commercial STA tool to support custom defined timing analysis approaches. Therefore, these algorithms are limited to be applied in simple combinational circuits with custom cell libraries and custom STA engines.

CHAPTER 3: GATE-LEVEL DUAL-THRESHOLD STATIC POWER OPTIMIZATION METHODOLOGY (GDSPOM)

In this chapter, a gate-level dual-threshold static power optimization methodology (GDSPOM) using 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 500MHz. In the following sections, characters of cell libraries 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.

3.1 Cell Libraries and Models

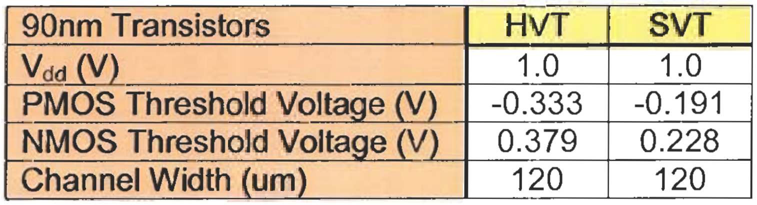
Two cell libraries are used in GDSPOM: one library contains all logic gates built with high threshold voltage transistors; the other library has all logics constructed with low threshold voltage transistors. Gate timing models are used for static timing analysis. Gate power models are used for dynamic and static power estimation. This section introduces cell libraries and models used in GDSPOM.

3.1.1 Cell Libraries

A high threshold voltage cell library and a low threshold voltage cell library are required in this dual MTCMOS design flow. 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 transistors has longer signal propagation delay and blocks more unwanted leakage current; a cell built with all low threshold voltage transistors has faster signal transition time but suffers from generating large amount of sub-threshold leakage current [17] [18].

In the GDSPOM experiment, Artisan 90nm high threshold voltage (HVT) cell library and standard threshold voltage (SVT) cell library are applied. The transistor characteristics are summarized in Table 3.1 and cell characteristics are summarized in Table 3.2.

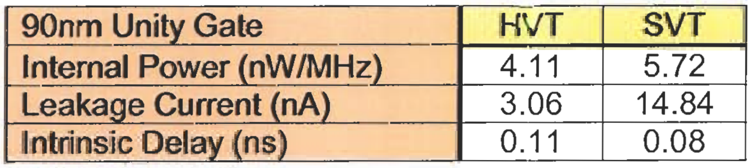
Table 3.1: Typical 90nm Transistors



90nm Transistors	HVT	SVT
V_{dd} (V)	1.0	1.0
PMOS Threshold Voltage (V)	-0.333	-0.191
NMOS Threshold Voltage (V)	0.379	0.228
Channel Width (um)	120	120

Source: artsc90g and artsc90g_hvt Spice Models

Table 3.2: 90nm Unity Gate



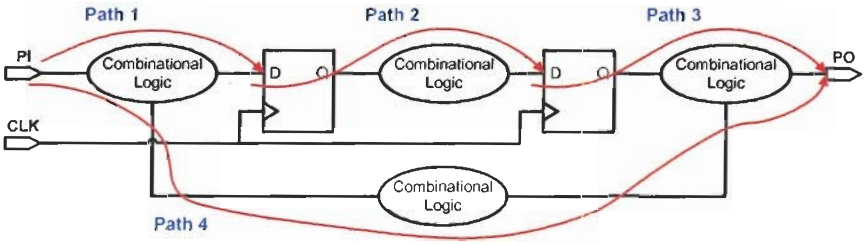
90nm Unity Gate	HVT	SVT
Internal Power (nW/MHz)	4.11	5.72
Leakage Current (nA)	3.06	14.84
Intrinsic Delay (ns)	0.11	0.08

Source: Spice Simulation Results and Average Cell Data on TSMC 90nm Standard Cell Library Databook [17] [18]

3.1.2 Timing Models

Static timing analysis tool uses timing models to estimate signal propagation duration through a timing path [19]. The start point of a timing path is either a primary input (PI) port or a sequential element where the signal is launched from; the end point is either a primary output (PO) port or a sequential element where the signal is captured. Figure 3.1 illustrates four basic types of timing paths:

Figure 3.1: Type of Timing Paths

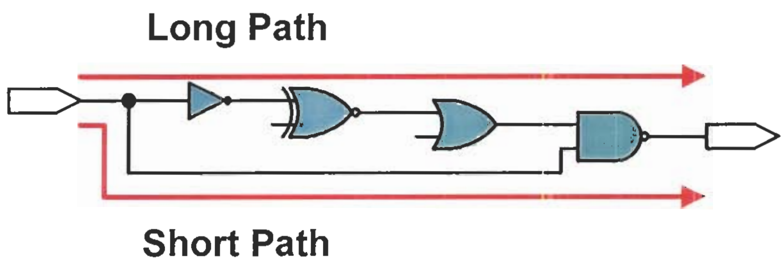


Source: PrimeTime User Guide [19]

As shown in Figure 3.1, Path 1 starts at a PI and finishes at a flop; Path 2 starts from a flop and arrived at another flop; Path 3 begins with a flop and ends with a PO; Path 4 starts from a PI, passes through combinational logics, and arrives at a PO. In typical VLSI design, majority STA checks are performed on the flop-to-flop paths.

It is possible to have different path routes between the same start and end points. Figure 3.2 illustrates one short and one long path routes, which start and finish at the same points.

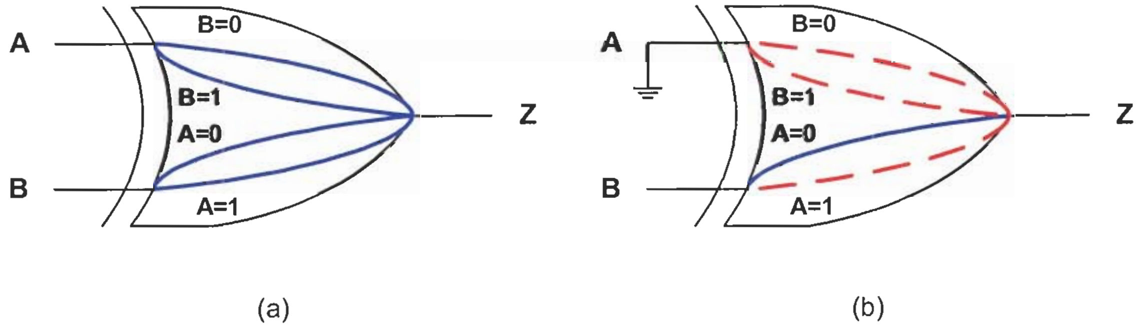
Figure 3.2: Routes of Timing Paths



Source: PrimeTime User Guide [19]

Moreover, 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 delay. As shown in Figure 3.3 (a), a two-input XOR gate has four arcs: A to Z when B is 0, A to Z when B is 1, B to Z when A is 0, and B to Z when A is 0. By default, STA tool will analyze all possible arcs. In the case when a pin is assigned a constant value, STA tool will automatically detect available arcs. Using the same example in Figure 3.3 (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.

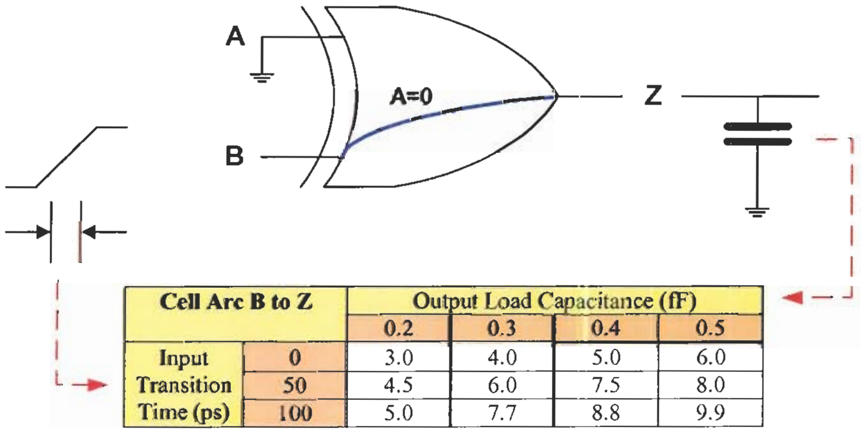
Figure 3.3: Cell Timing Arcs



Source: PrimeTime User Guide [19]

Arc delay calculation formula is a function of input transition time and output load capacitance. Timing lookup table like the one shown in Figure 3.4 is recorded in the cell technology library. Using the example in Figure 3.4, a cell delay time from input B to output Z is 8.8 ns when input transition time is 100 ps and output load capacitance is 0.4 fF. When values of input transition time and output load capacitance are between the table points or outside the table range, STA tool will use interpolation or extrapolation approach to estimate the delay.

Figure 3.4: Arc Timing Lookup Table

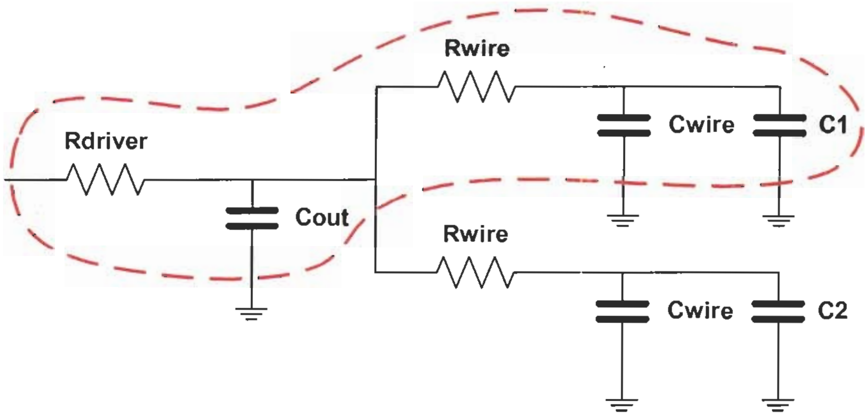


Cell Arc B to Z		Output Load Capacitance (fF)			
		0.2	0.3	0.4	0.5
Input Transition Time (ps)	0	3.0	4.0	5.0	6.0
	50	4.5	6.0	7.5	8.0
	100	5.0	7.7	8.8	9.9

Source: PrimeTime User Guide [19]

Arc delay is cell internal signal propagation time. Net delay is the total time for a signal to travel between two cells. Before the layout phase, absolute cell location and wire length are unknown. To bypass this issue, STA tool uses wire load model to predict the net delay. The wire load model estimates net capacitance and resistance based on the number of fanout pins on this net. As shown in Figure 3.5, the delay time of the circled net will be calculated with net capacitance Cout, Cwire, and C1 along with net resistance Rdriver and Rwire.

Figure 3.5: RC Tree Network



Source: PrimeTime User Guide [19]

Knowing cell and net delays as well as input latency, path arrival time can be calculated as following:

$$AT_{path} = D_{clk} + D_{clk\_net} + \sum D_{cell} + \sum D_{net}$$

where

AT_{path} = path arrival time

D_{clk} = clock source latency

D_{clk_net} = clock network latency

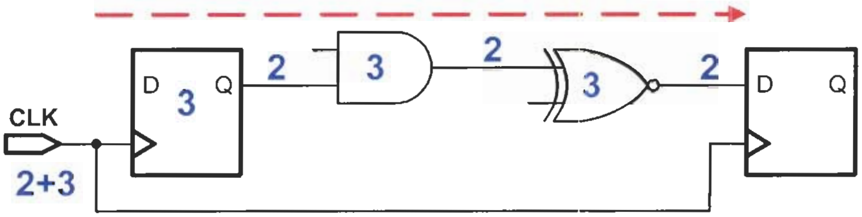
D_{cell} = cell delay

D_{net} = net delay

Assuming that source clock latency is 2 ns, clock network latency is 3 ns, sequential and combinational cell delay are 3 ns, and net delay is 2 ns, the path shown in Figure 3.6 has arrival time of 20 ns.

$$\begin{aligned} &AT_{path} \\ &= D_{clk} + D_{clk\_net} + \sum D_{cell} + \sum D_{net} \\ &= (2) + (3) + (3+3+3) + (2+2+2) \\ &= 20 \end{aligned}$$

Figure 3.6: Path Arrival Time Calculation



Source: http://www.chip123.com

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

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

where

RT_{path_setup} = path setup required time

T_{clk} = clock period

D_{clk} = clock source latency

D_{clk_net} = clock network latency

$T_{clk\_uncertainty}$ = clock uncertainty

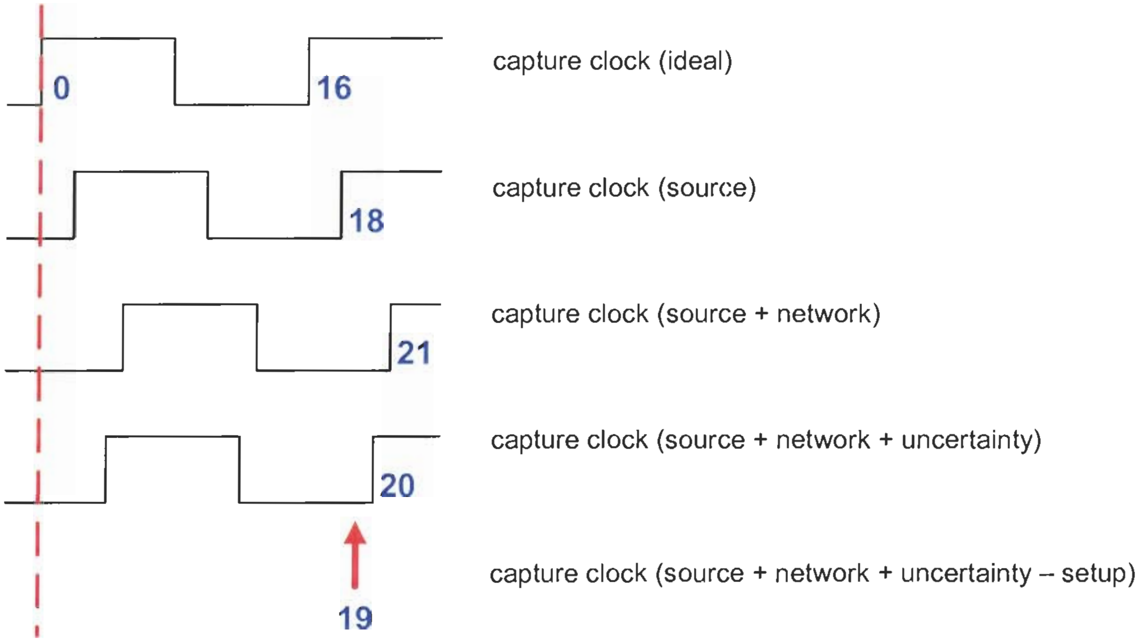
T_{setup} = capture flop setup time

Assuming clock period is 16 ns, source clock latency is 2 ns, clock network latency is 3 ns, clock uncertainty is 1 ns, and flop's setup time is 1 ns, the path required setup time is 19 ns.

$$\begin{aligned} &RT_{path\_setup} \\ &= T_{clk} + D_{clk} + D_{clk\_net} - T_{clk\_uncertainty} - T_{setup} \\ &= 16 + 2 + 3 - 1 - 1 \\ &= 19 \end{aligned}$$

Figure 3.7 illustrates how path required setup time is calculated.

Figure 3.7: Path Required Setup Time Calculation


0
16
capture clock (ideal)
18
capture clock (source)
21
capture clock (source + network)
20
capture clock (source + network + uncertainty)
19
capture clock (source + network + uncertainty – setup)

Source: http://www.chip123.com

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 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:

$$S_{path\_setup} = RT_{path\_setup} - AT_{path}$$

where

S_{path_setup} = path setup slack

RT_{path_setup} = path setup required time

AT_{path} = path arrival time

Positive slack indicates that the path meets timing and negative slack tells that the path has a timing violation. Considering path in Figure 3.6 has arrival time of 20 ns and its clock path in Figure 3.7 has required time of 19 ns, this path's setup slack is -1 ns, which means that a timing violation exists on this path.

Because all timing checks are done per path bases, this type of STA is characterized as path based STA approach. The STA approach presented in Chapter 2.2.1 is block based because timing checks are done per cell bases. GDSPOM uses path based STA.

3.1.3 Power Models

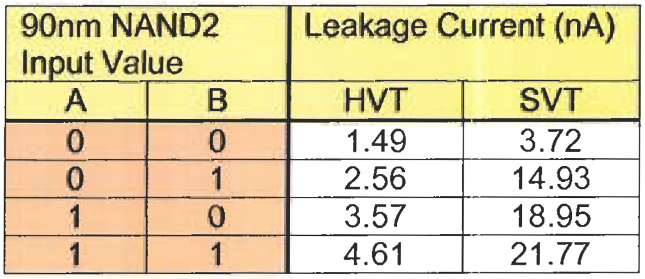
As mentioned in Chapter 1.2, a cell dissipates internal power, switching power, and leakage power. Therefore, a cell power model provides three different power attributes for the power analysis tool to do power estimation [5].

Cell internal power calculation formula is a function of input transition time and output load capacitance. Internal power lookup table similar to the one shown in Figure 3.4 is included in the cell technology library. Like arc delay calculation described in Chapter 3.1.2, a power analysis tool will 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, which has been introduced in Chapter 3.1.2. 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 3.3, cell leakage current can vary in more than 5 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 vary. 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.

Table 3.3: NAND2 Leakage Current



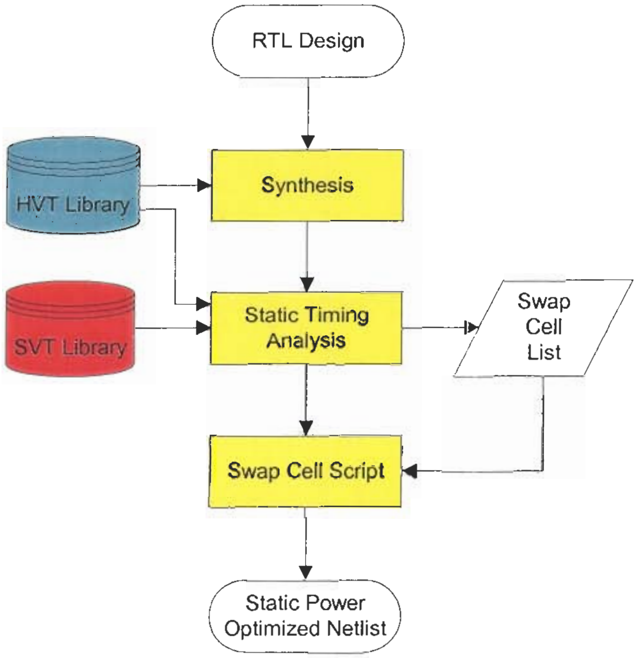
90nm NAND2 Input Value		Leakage Current (nA)	
A	B	HVT	SVT
0	0	1.49	3.72
0	1	2.56	14.93
1	0	3.57	18.95
1	1	4.61	21.77

Source: Spice Simulation Results

3.2 GDSPOM Flow

Figure 3.8 shows the flow chart of GDSPOM used for designing high-speed low-power SOC applications using MTCMOS technology.

Figure 3.8: Flow Chart of GDSPOM



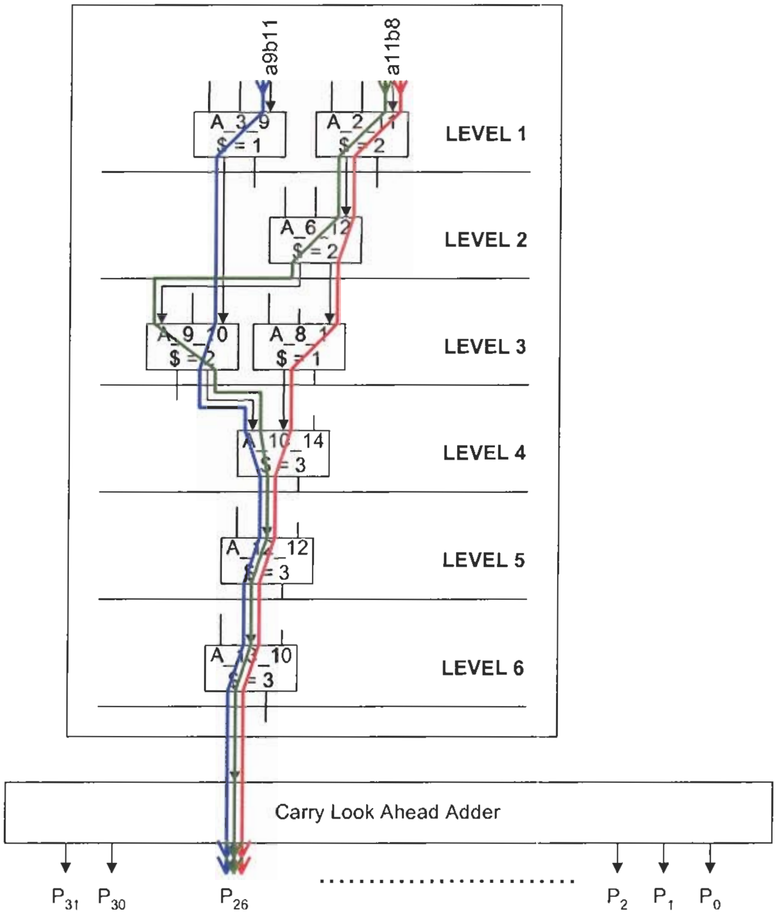
Source: VISIO Drawing

As shown in the figure, a Register Transfer Language (RTL) design is synthesized into gate-level netlist of cells using CMOS devices with a high-threshold voltage (HVT). Then, static timing analysis (STA) 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 500MHz 16-bit multiplier, 400MHz frequency is targeted when converting the multiplier's RTL design to HVT gate-level netlist. The additional 100MHz 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 90nm 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 cell types without altering design architecture and increasing area overhead.

STA is the key component in GDSPOM flow. STA breaks a design into a group of timing paths and calculates the signal propagation delay of each path individually. The concept of path based STA has been introduced in Chapter 3.1.2. When a path's delay is greater than the specified timing constraint, this path has a timing violation. Figure 3.9 illustrates three timing violated paths found inside a 16-bit HVT multiplier's Wallace tree reduction architecture [20] and carry look ahead circuit [21].

Figure 3.9: Cell Timing Violating Cost



Source: VISIO Drawing

As shown in Figure 3.9, three timing violated paths labelled blue, green, and red have been identified. The number of timing violating paths through one cell determined this cell's cost value. For instance, adders A_3_9 and A_8_1 have the cost value of 1; A_2_11, A_2_16 and A_9_10 have the cost value of 2; A_10_14, A_12_12, and A_13_10 have the cost value of 3. The cells with the highest cost value such as A_10_14, A_12_12, and A_13_10 in this example 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.

Algorithm 3.1 explains how cost values are assigned to cells and Algorithm 3.2 shows the STA iterating process. A simple example of GDSPOM procedure is illustrated in Figure 3.10.

Algorithm 3.1: Get Bottleneck Cells

```
procedure getBottleneckCells ($inputNetlist, $requiredTime) {
    @pathArray = all paths in $inputNetlist
    %cellCostHash = all cells in $inputNetlist with initial cost value 0

    foreach $path (@pathArray) {
        $arrivalTime = calculated $path arrival time

        if ($arrivalTime > $requiredTime) {
            foreach $cell in $path {
                incr $cellCostHash{$cell}
            }
        }
    }
}
```

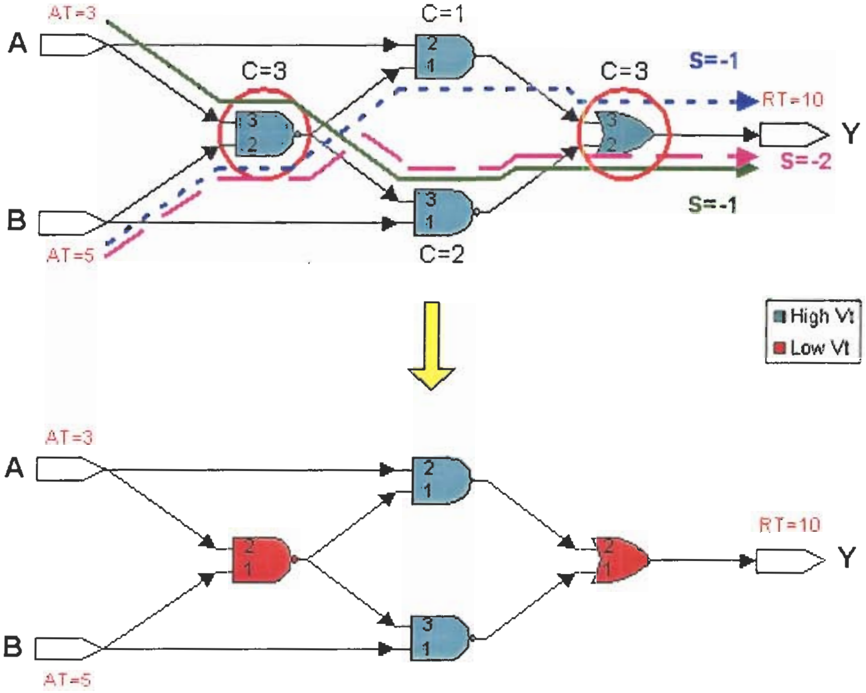
Source: PrimeTime Input Tcl Script

Algorithm 3.2: Get Swap Cell List

```
procedure getSwapCellList ($originalNetlist, $requiredTime) {
    (@bottleneckCellArray, $inputNetlist) =
        &getBottleneckCells ($origianlNetlist, $requiredTime)

    while (@bottleneckCellArray != NULL) {
        @swapCellList = @swapCellList + @bottleneckCellArray

        (@bottleneckCellArray, $inputNetlist) =
            &getBottlenetCells($inputNetlist, $requiredTime)
    }

    return @swapCellList
}
```

Source: PrimeTime Input Tcl Script

Figure 3.10: GDSPOM Example



Source: VISIO Drawing

The bottleneck cell swapping approach is the main difference between GDSPOM and other methodologies mentioned in Chapter 2. Fixing a high cost cell means fixing multiple timing violated paths at once. Always targeting the highest cost 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.

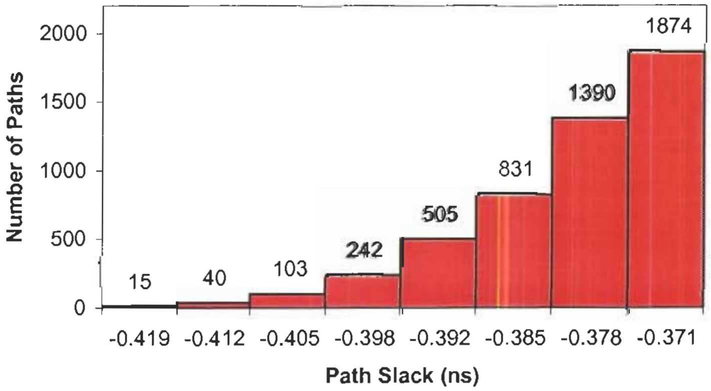
3.3 Performance

In order to assess the effectiveness of GDSPOM for designing low-power high-speed SOC applications using 90nm MTCMOS technology, three 16-bit multipliers with Wallace tree reduction architecture [20] 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 500MHz and 90nm Artisan cell libraries are used in this experiment. A 16-bit multiplier has 7320 unity gates and it contains approximately 30000 transistors.

As shown in Figure 3.11, with 500MHz clock frequency constraint, 5000 paths in the HVT multiplier fail the speed test. Mentioned in Chapter 3.1.2, the negative slack means the overtime for a signal to travel from one input to one output of a path. For instance, a path with -0.37 ns slack means a signal on this path arrives 0.37 ns later than when it is supposed to arrive.

Figure 3.11: Path Slack Chart

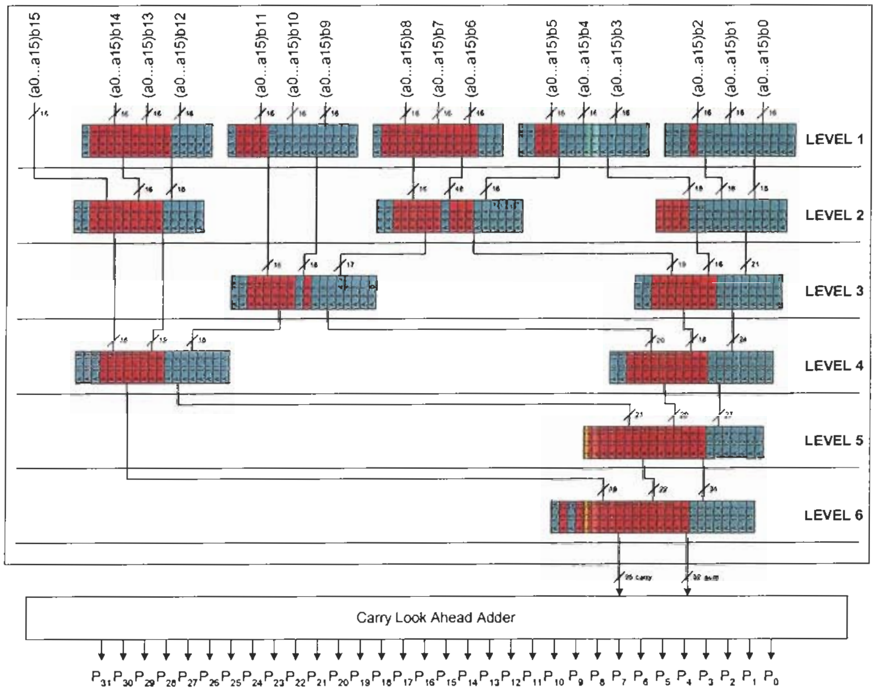


Source: PrimeTime Report Timing Results

In this experiment, GDSPOM reassigned 352 out of total 1715 cells from HVT to SVT to satisfy the 500MHz speed constraint. Figure 3.12 shows the block diagram and Figure 3.13 shows the schematic view of the 16-bit dual-V_t

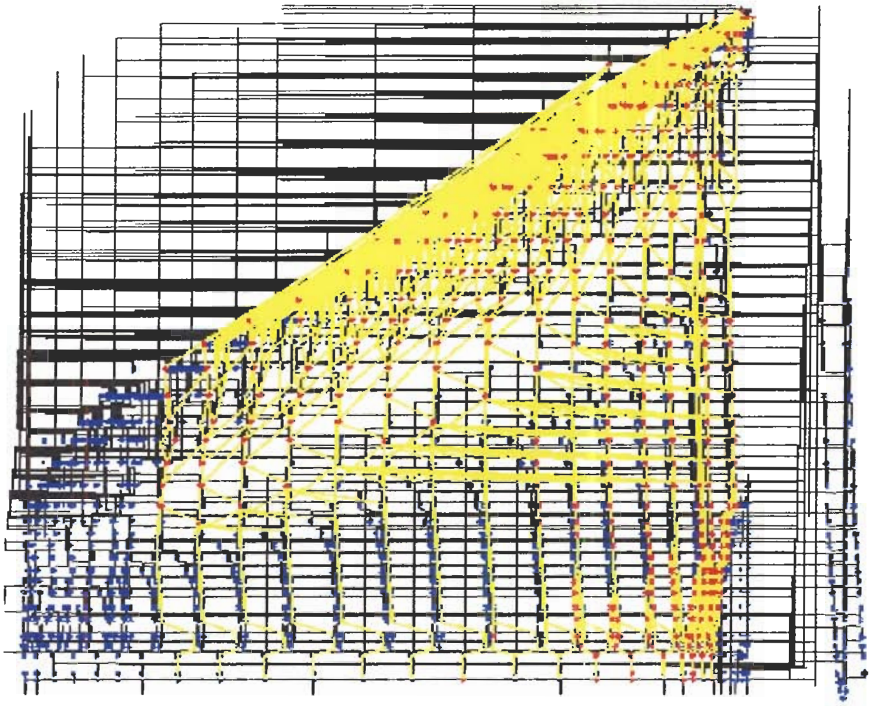
multiplier design optimized by GDSPOM to have HVT (blue) and SVT (red) cells. Note that yellow paths in Figure 3.13 are originally timing violated paths.

Figure 3.12: Block Diagram of a Dual-V_t 16-bit Multiplier



Source: VISIO Drawing

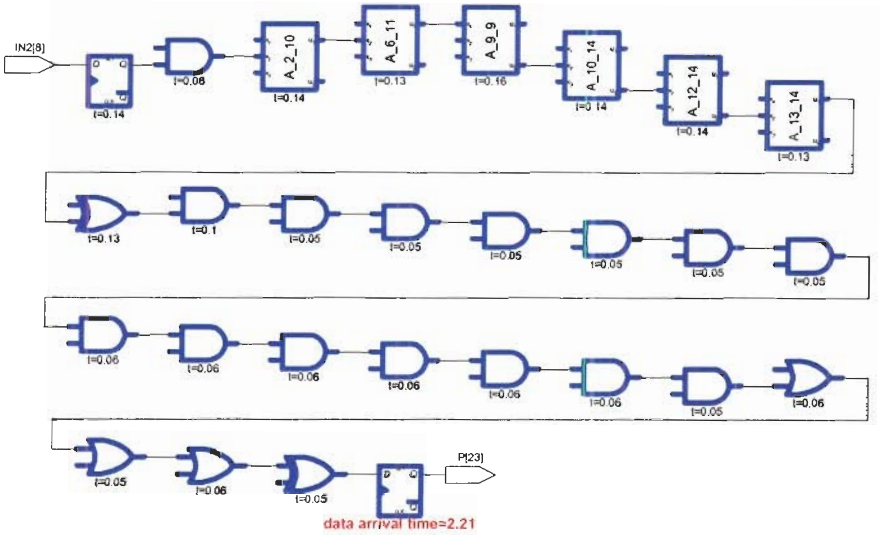
Figure 3.13: Schematic View of a Dual-V_t 16-bit Multiplier



Source: Design Vision Schematic View

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. Figure 3.14 shows this path in the HVT multiplier, whose data arrival time is 2.21 ns, which does not meet the 500MHz operating frequency specification. The arrival time of each cell shown in the figure includes net delay time.

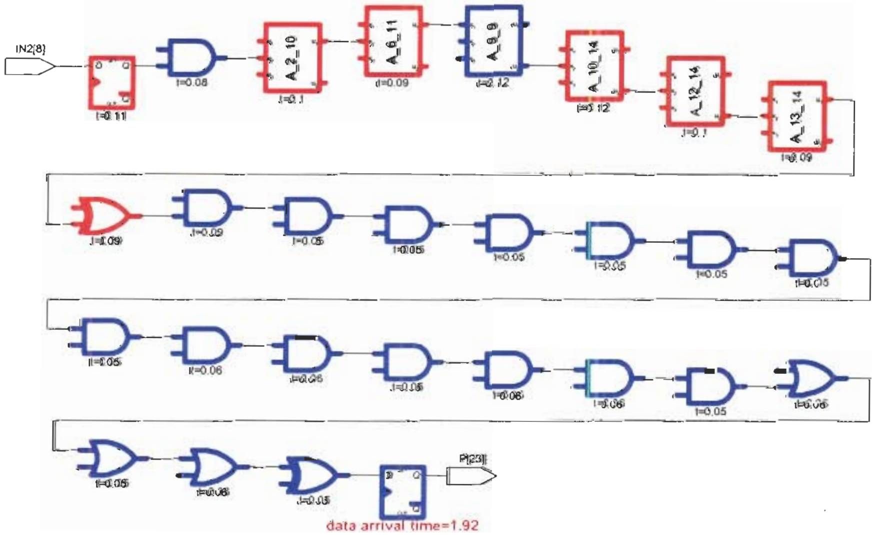
Figure 3.14: A Timing Path in HVT 16-bit Multiplier



Source: PrimeTime Report Timing Results

Figure 3.15 shows the same path in the dual-V_t multiplier. After performing GDSPOM flow, 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.

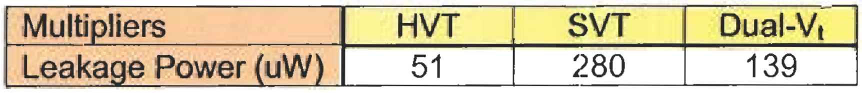
Figure 3.15: A timing Path in Dual-V_t 16-bit Multiplier



Source: PrimeTime Report Timing Results

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 does not meet the speed requirement of 500MHz. 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. Table 3.4 summarizes these multipliers' leakage powers.

Table 3.4: Multiplier Leakage Power Comparison

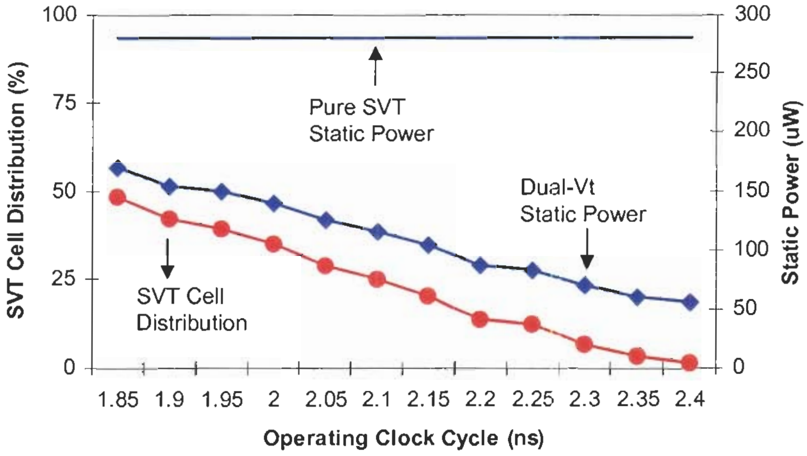


Multipliers	HVT	SVT	Dual-V_t
Leakage Power (uW)	51	280	139

Source: Power Compiler Results

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. Figure 3.16 illustrates that the dual-V_t multiplier dissipates less static power in comparison with the all-SVT multiplier one. It also shows that slower dual-V_t multiplier requires fewer SVT cells, and dissipates less static power as a result.

Figure 3.16: Static Power Chart of Different Speed of Multipliers



Source: Power Compiler Results

CHAPTER 4: CONCLUSION AND FUTURE WORK

In this thesis, a novel gate-level dual-threshold static power optimization methodology (GDSPOM), which is based on the static timing analysis technique for designing high-speed low-power SOC applications using 90nm 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 consumption compared to the all low-threshold voltage one.

Because GDSPOM flow uses commercially available design tools and cell libraries, adopting the flow to the current design environment is straightforward. GDSPOM can be applied in designing a new device as well as be used to improve the power saving in the existing single V_t products. For example, an existing SVT design can be replaced with all HVT cells and go through GDSPOM flow to produce a new dual-V_t device: it has the same performance but dissipates less power.

GDSPOM is not limited to MTCMOS technology for power saving. It can be expanded to combine other types of cells to achieve optimal power reduction. For instance, an adiabatic cell library [22] [23] [24] with characterized timing models can be used in GDSPOM to output a design containing both standard and adiabatic cells. With the same idea, dual V_{dd} cells or other future cell type libraries can be combined through GDSPOM process for better power saving products.

While optimizing the device leakage power, GDSPOM also reduces dynamic power of the design by reducing the number of low V_t cells. This positive side effect is from the fact that low V_t cells have higher short-circuit current during signal switching. More efficient dynamic power optimization strategies like clock gating [25] can be combined with GDSPOM to produce a complete power saving design flow: the flow optimizes both dynamic and static powers.

Another potential research work is to assess the value of moving GDSPOM to post-layout design level. GDSPOM is performed in pre-layout design and values of net delay are estimated numbers based on wire load model. Post-layout design has accurate physical data. Therefore, performing GDSPOM at post-layout level achieves the most accurate results. The drawback of this modification is that it is more difficult and complicated to change circuits at the end of the design cycle.

To sum up, GDSPOM is an efficient method of reducing leakage power. GDSPOM is ready to use and easy to adopt in the traditional design flow. GDSPOM can be applied in the future technology and is also back compatible.

REFERENCE LIST

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

[2] ITRS, "ITRS 2004 Update Documents for Review," http://www.itrs.net/Common/2004Update/2004Update.htm. Accessed on: October 10, 2005.

[3] G.E. Moore, "Progress in Digital Integrated Electronics," International Electron Devices Meeting, Vol. 21, pp. 11-13, 1975.

[4] P.P. Gelsinger, "Microprocessors for the new millennium: Challenges, opportunities, and new frontiers," International Solid-State Circuits Conference, pp. 22-25, Feb. 2001.

[5] Synopsys, "Power Compiler User Guide," v2004.12.

[6] H.J.M. Veendrick, "Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits," IEEE Journal of Solid-State Circuits, Vol. 19, Issue 4, pp. 468-473, Aug 1984.

[7] S. Mukhopadhyay, K. Roy, "Leakage Estimation and Leakage Control for Nano-Scale CMOS Circuits," Design Automation Conference, 2004.

[8] J.B. Kuo, "CMOS Digital IC," McGraw-Hill, Taiwan, 1996.

[9] R.X. Gu, M.I. Elmasry, "Power dissipation analysis and optimization of deep submicron CMOS digital circuits," IEEE Journal of Solid-State Circuits, Vol. 31, Issue 5, pp. 707-713, May 1996.

[10] Synopsys, "Diversifying Design Trends in North America," http://www.synopsys.com/news/pubs/compiler/artlead_designtren-may05.html. Accessed on October 10, 2005.

[11] L. Wei, Z. Chen, M. Johnson, K. Roy, V. De, "Design and optimization of low voltage high performance dual threshold CMOS circuits," Design Automation Conference, pp. 489-494, Jun 1998.

[12] L. Wei, Z. Chen, K. Roy, M.C. Johnson, Y. Ye, V.K. De, "Design and optimization of dual-threshold circuits for low-voltage low-power applications," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 7, Issue 1, pp. 16-24, March 1999.

[13] N.P. Jouppi, "Timing Analysis and Performance Improvement of MOS VLSI Designs," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 6, Issue 4, pp. 650-665, July 1987.

[14] D. Samanta, A. Pal, "Optimal Dual-VT Assignment for Low-voltage Energy-Constrained CMOS Circuits," International Conference on VLSI Design, pp. 193-198, Jan 2002.

[15] Q. Wang, S. Vrudhula, "Algorithms for Minimizing Standby Power in Deep Submicrometer, Dual-V_t CMOS Circuits," IEEE Trans. Computer-Aided Design of IC and Systems, Vol. 21, No. 3, pp. 306-318, March 2002.

[16] CMC Microsystems, http://www.cmc.ca/. Accessed on May 9, 2005.

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

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

[19] Synopsys, "PrimeTime User Guide," v2004.12.

[20] C. S. Wallace, "A suggestion for a fast multiplier", IEEE Trans. Computers, Vol. EC-13, pp. 14-17, February 1964.

[21] "Hardware algorithms for parallel multiplication," http://www.aoki.ecei.tohoku.ac.jp/arith/mg/algorithm.html. Accessed on May 15, 2005.

[22] C. C. Yeh, J. H. Lou, and J. B. Kuo, "1.5V CMOS Full-Swing Energy Efficient Logic (EEL) Circuit Suitable for Low-Voltage and Low-Power VLSI Application," Elec. Lett., Vol. 33, No. 16, pp. 1375-1376, 1997.

[23] Y. Zhang, H. H. Chen, and J.B. Kuo, "0.8V CMOS Adiabatic Differential Switch Logic Circuit Using Bootstrap Techniques for Low-Voltage Low-Power VLSI," Electron. Lett., Vol. 38, No. 24, pp. 1497-1499, 2002.

[24] H.P. Chen and J. B. Kuo, "A 0.8V CMOS TSPC Adiabatic DCVS Logic Circuit with the Bootstrap Technique for Low-Power VLSI," ICECS Proceedings, pp. 175-178, 2004.

[25] Q. Wu; M. Pedram, X. Wu, "Clock-gating and its application to low power design of sequential circuits", IEEE Transactions on Circuits and Systems, Vol. 47, Issue 3, pp. 415-420, March 2000.