LOW-POWER MULTI-THRESHOLD CMOS CIRCUITS OPTIMIZATION AND CAD TOOL DESIGN

A Thesis

Presented to

The Faculty of Graduate Studies

of

The University of Guelph

by

WENXIN WANG

In partial fulfilment of requirements

for the degree of

Master of Science

May, 2004

ABSTRACT

LOW-POWER MULTI-THRESHOLD CMOS CIRCUITS OPTIMIZATION AND CAD TOOL DESIGN

Wenxin Wang
University of Guelph, 2004

Advisors:
Professor Shawki Areibi, Mohab Anis

Over the last two decades, low-power design has become a concern in digital VLSI design, especially for portable and high performance systems. As technology scales into the Deep Sub-Micron (DSM) regime, standby subthreshold leakage power increases exponentially with the reduction of the threshold voltage. Therefore, effective leakage minimization techniques are becoming a necessity. Multi-Threshold CMOS (MTCMOS) has emerged as an effective circuit-level technique that attains a high performance, while standby subthreshold leakage is minimized by cutting off the power of the inactive blocks by sleep transistors. As a result, the proper sizing of the sleep transistor is pivotal to the performance and the leakage power saving of the MTCMOS circuit.

The gate-clustering MTCMOS technique has been proposed as an effective method to size the sleep transistor. The sizing problem has been modelled as a Bin-Packing

Problem (BPP) and a Set-Partitioning Problem (SPP). However, the computation time for these solutions is high. In this thesis, two Genetic Algorithms (GAs) are implemented to reduce the computation time of the CPLEX solver which is applied to the BPP and SPP problems. In addition, to improve the solution quality and the computation time, a First-Fit (FF) technique and a Set-Covering (SC) model are proposed. The FF technique achieves a 12% and 92% reduction, on average, in leakage power and CPU time, respectively, compared to the leakage power and CPU time of the BPP technique. The SC model reduces the objective cost of the problem and the computation time by 9% and 99%, respectively, compared to those of the SPP model.

The MTCMOS low-power design methodology involves an iterative design process that involves an area versus power tradeoff and a timing versus power tradeoff. As a result, the technique needs to be integrated into the principal design environment. In this thesis, an automated vector generation engine is introduced to build a vector for each gate in the gate-level netlist. Based on the vector representation, a MTCMOS design environment is devised and integrated into the Canadian Microelectronics Corporation (CMC) digital ASIC design flow.

Acknowledgements

I would like to thank my supervisor professor Shawki Areibi for his guidance and assistance. Without his criticism, seriousness, and kindness, this work would never have been possible. Thanks also to my co-advisor professor Mohab Anis who encouraged me to work harder and meet the deadline.

I am extremely grateful to the Canadian Microelectronics Corporation for providing CAD tools and making this work possible. Special thanks to Sean Smith, who always solves the CAD tool problem immediately. Also, many thanks to Phil Regier for making sure I could use facilities at University of Waterloo.

Finally, I would like to express my extreme gratitude to my wife, my parents, my brother, and my sisters. They have been a constant support, help, and encouragement during this work.

Wenxin Wang

Guelph, Ontario

To

my family

whose love and encouragement helped accomplish this thesis.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Low-Power VLSI Design

Since the invention of the first Integrated Circuit (IC) four decades ago, silicon technology down scaling continues to meet the increasing demands for higher functionality and better performance at a lower cost. Power dissipation, though not entirely ignored, has been of little concern until recently. The advances in VLSI integration technology have made it possible to put a complete System on a Chip (SoC) which facilitates the development of portable systems. Portable battery-powered applications such as notebook computers, cellular phones, Personal Digital Assistants (PDAs), and military equipments profile power dissipation as a critical parameter in digital VLSI design.

With the increasing prominence of portable systems, it is important to prolong the battery life as much as possible, since it is the limited battery lifetime that typically imposes strict demands on the overall power consumption of such systems.

Although the battery industry has been making efforts to develop batteries with a higher energy capacity than that of conventional Nickel-Cadmium (NiCd) batteries, a revolutionary increase of the energy capacity does not seem imminent. Therefore, portable applications have led to rapid and innovative developments in low-power circuit designs.

Power dissipation is also crucial for Deep Sub-Micron (DSM) technologies. To further improve the performance of the circuits and to integrate more functions on a chip, the feature size has to continue to shrink. As a result, the power dissipation per unit area grows, increasing the chip temperature. Since the dissipated heat needs to be removed to maintain an acceptable chip temperature, large cooling devices and expensive packaging are required in portable devices and high-performance digital systems such as microprocessors. A recently announced Pentium IV [1] CPU, operating at a 3.4GHz frequency and 1.3V supply voltage, consumes 130W of power [Inc04b]. This high power dissipation also requires special Printed Circuit Board (PCB) technology to deliver large currents from the power supply to the various devices in the system.

Another important reason for low-power design is reliability. As technologies continue to scale, not only does the power density increase, but also the current density increases. Large current densities cause serious problems such as electro-migration and hot-carrier induced device degradation [Kang03]. In addition, the heat gradient across the chip causes thermal and mechanical stress leading to early breakdown. Therefore, the reliability can only be enhanced if power consumption is reduced.

[1]Pentium and Pentium IV are trademarks of Intel Corporation.

Although power dissipation is important for modern VLSI design, performance (speed) and area are still the main requirements of a design. However, low-power design usually involves making tradeoffs such as timing versus power and area versus power. Increasing performance, while the power dissipation is kept constant, is also considered to be a low-power design problem.

1.2 Motivation

1.2.1 MTCMOS Technique

Low-power design methodologies range from the device/process level to the algorithmic level. Of all these techniques, lowering the supply voltage (V_{DD}) is the one that significantly reduces the power consumption because of the quadratic relationship between the supply voltage and the dynamic power consumption [Raba96]. To compensate for the performance loss due to a lower supply voltage, a transistor's threshold voltage (V_T) should also be reduced. However, this causes an exponential increase in the subthreshold leakage current [Kang03]. Therefore, an important research area today is to develop circuit techniques to reduce the subthreshold leakage currents that are caused by the reduced V_T.

Multi-Threshold Complementary Metal Oxide Semiconductor (MTCMOS) is an effective circuit-level technique that provides a high performance and low leakage power design strategy [Muto95]. However, the technique employs transistors[2] at the standby mode to isolate the power supply. As a result, the circuit speed at the active

[2]They are called sleep transistors.

mode degrades due to the presence of sleep transistors [Anis02]. Consequently, the sleep transistor sizing is critical to the performance, the leakage power saving, and the noise immunity of MTCMOS circuits.

For the past few years, a number of sleep transistor sizing techniques have been reported in the literature. An innovative gate-clustering MTCMOS technique has been introduced in [Anis02]. In this technique, the sleep transistor sizing problem is modelled as a Bin-Packing Problem (BPP) and a Set-Partitioning Problem (SPP). However, the BPP and the SPP consume an increasingly larger computation time by the ILP CPLEX solver to find the optimal solutions as the circuit size increases. Consequently, one motivation for this thesis is to develop heuristic methods to find solutions that are close to the optimum for the BPP and the SPP in less computation time. In addition, more simple and effective methods to model sleep transistor sizing are proposed so that both the solution quality and the computation time are improved.

1.2.2 Automatic Design Environment

Creating optimal low-power MTCMOS techniques involves tradeoffs such as timing versus power and area versus power at the different design stages. For designers to accurately and efficiently balance these tradeoffs, it is necessary for this technique to be integrated with and applied throughout the entire RTL-to-GDSII flow.

Therefore, another goal of this thesis is to introduce a MTCMOS automatic design environment and integrate it within the Canadian Microelectronics Corporation (CMC) digital ASIC design flow.

1.3 Research Approach

The overall research approach for the sleep transistor sizing problem and the developed MTCMOS design environment are illustrated in Figure 1.1.

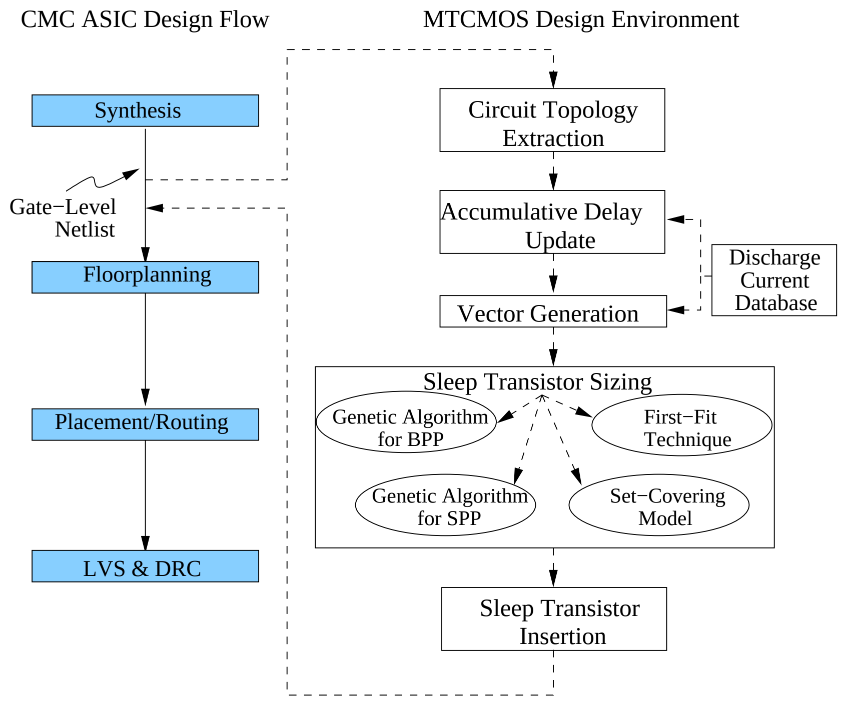

CMC ASIC Design Flow
MTCMOS Design Environment
Synthesis
Gate−Level Netlist
Floorplanning
Placement/Routing
LVS & DRC
Circuit Topology Extraction
Accumulative Delay Update
Discharge Current Database
Vector Generation
Sleep Transistor Sizing
Genetic Algorithm for BPP
First−Fit Technique
Genetic Algorithm for SPP
Set−Covering Model
Sleep Transistor Insertion

Figure 1.1: Overall approaches and developed MTCMOS environment.

Within the MTCMOS design environment, several heuristic methods are developed to handle the circuit extraction and vector generation. A discharge current database, based on the technology library, is also constructed. In addition, a CPLEX solver interfacing engine is built to identify the effectiveness of the Meta-heuristics for solving the BPP and the SPP, compared to the effectiveness of the CPLEX solver. Finally, a First-Fit (FF) technique and a Set-Covering (SC) model are proposed to effectively solve the sleep transistor sizing problem.

1.4 Contributions

The main contributions of the thesis can be summarized as follows:

- An investigation of the applicability of several heuristic methods for a MTCMOS low-power design is conducted.
- The digital design flow for the CMC is automated by incorporating the MTCMOS approach.
- Several publications in the form of conference papers [?] and journal papers have resulted from this thesis.

1.5 Thesis Organization

Chapter 2 presents various power dissipation mechanisms in CMOS digital circuits. The increased subthreshold leakage, caused by technology scaling, is then discussed. For each subthreshold leakage reduction approach, the advantages and disadvantages are analyzed. This chapter provides a background and motivates the need for the work that is presented in later chapters.

The gate-clustering MTCMOS technique is introduced in detail in the first part of Chapter 3. Two GAs, specifically designed for the BPP and the SPP, are implemented and compared with techniques in the literature. In addition, an effective FF technique and a SC model are described in Chapter 3 with experimental results.

In Chapter 4, the Canadian Microelectronics Corporation (CMC) digital ASIC design flow is introduced. An automated vector generation engine and a MTCMOS design environment are developed and integrated into the CMC design flow. Finally, Chapter 5 provides conclusions and suggestions for future work.

Chapter 2

Background

2.1 Introduction

With the smaller geometries in Deep Sub-Micron (DSM) technology, the number of gates that need to be integrated on a single chip, power density, and total power are increasing rapidly. Also, designing for low-power has become increasingly important in a wide variety of applications. However, creating optimal low-power designs involves tradeoffs such as timing versus power and area versus power at the different stages of the design flow. Successful power-sensitive designs require engineers to have the ability to accurately and efficiently perform these tradeoffs.

To address these issues directly, it is essential to understand the different types and sources of power dissipation in digital Complementary Metal Oxide Semiconductor (CMOS) circuits. The reason for choosing the CMOS technology is that it is currently the most dominant digital IC implementation technology.

In this chapter, the most significant power dissipation sources in CMOS circuits are identified. Then some low-power design techniques to handle the leakage power are discussed. Finally, several heuristics for combinatorial optimization problems in VLSI design automation are introduced.

2.2 Power Dissipation in CMOS Digital Circuits

Power dissipation in CMOS digital circuits is categorized into two types: peak power and time-averaged power consumption. Peak power is a reliability issue that determines both the chip lifetime and performance. The voltage drop effects, caused by the excessive instantaneous current flowing through the resistive power network, affects the performance of a design due to the increased gate and interconnect delay. This large power consumption causes the device to overheat which reduces the reliability and lifetime of the circuit. Also noise margins are reduced, increasing the chance of chip failure due to crosstalk.

The time-averaged power consumption in conventional CMOS digital circuits occurs in two forms: dynamic and static. Dynamic power dissipation occurs in the logic gates that are in the process of switching from one state to another. During this process, any internal and external capacitance associated with the gate's transistors has to be charged, thereby consuming power. Static power dissipation is associated with inactive logic gates (i.e., not currently switching from one state to another). Dynamic power is important during normal operation, especially at high operating frequencies, whereas static power is more important during standby, especially for battery-powered devices. An overview of the different power dissipation types is given in Figure 2.1.

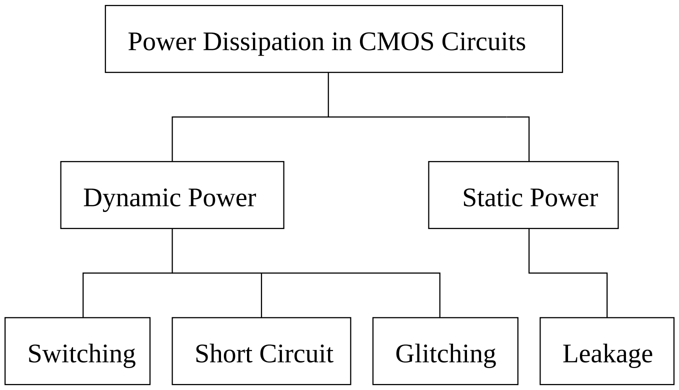


Figure 2.1: Different power dissipation types in CMOS circuits.

2.2.1 Dynamic Power Dissipation

Dynamic power, primarily caused by the current flow from the charging and discharging of parasitic capacitances, consists of three components: switching power, short-circuit power, and glitching power.

2.2.1.1 Switching Power Dissipation

In digital CMOS circuits, the switching power is dissipated when current is drawn from the power supply to charge up the output node capacitance. During this switching event, the output node voltage typically makes a full transition from 0 to V_{DD}, and one-half of the energy drawn from the power supply is dissipated as heat in the conducting pMOS transistors. The energy stored in the output capacitance during charge-up is dissipated as heat in the conducting nMOS transistors, when the output voltage switches from V_{DD} to 0. A CMOS inverter circuit, depicted in Figure 2.2, is presented to illustrate this dynamic power dissipation during switching. The total capacitive load C_{load} at the output of the inverter consists of the diffusion capacitance of the drains of the inverter transistors, the total interconnect capacitance, and the input gate oxide capacitance of the driven gates that are connected to the inverter's output. In most CMOS digital circuits, the switching power

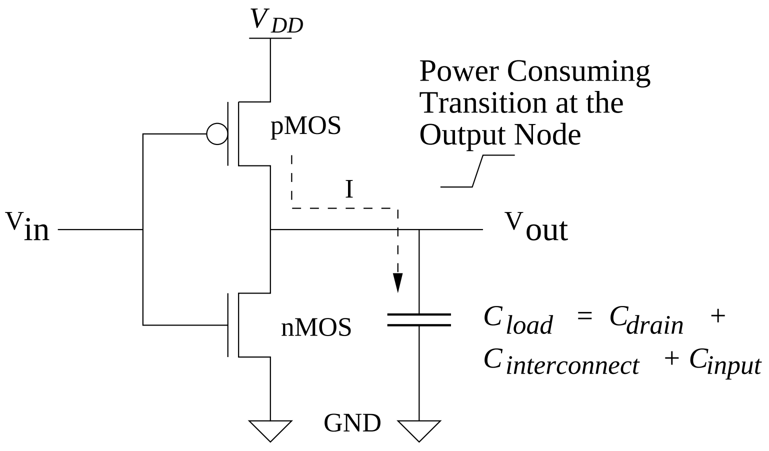

V_{DD}
Power Consuming Transition at the Output Node
pMOS
I
V_{in}
V_{out}
nMOS
$C_{load} = C_{drain} + C_{interconnect} + C_{input}$
GND

Figure 2.2: CMOS inverter for switching power calculation.

is the dominant component in power dissipation. Figure 2.3 exhibits the supply current waveform of the inverter circuit. The average switching power dissipation of the inverter can be calculated from the energy, required to charge up the output node to V_{DD} and discharge the total output load capacitance to ground (GND). The generalized expression for the switching power dissipation of a CMOS logic gate can be written as

$$P_{avg} = \alpha_T \cdot C_{load} \cdot V_{DD}^2 \cdot f_{CLK}, \tag{2.1}$$

where α_T is the switching activity factor of the gate, C_{load} represents the total load capacitance, V_{DD} is the supply voltage, and f_{CLK} represents the operating frequency. The switching activity α_T is computed by multiplying the probability that the output of a gate will be at zero by the probability that the output will be at one [Kang03]. The parameter α_T is a function of several factors, including the Boolean function performed by the gate, the logic style, and the input signal statistics.

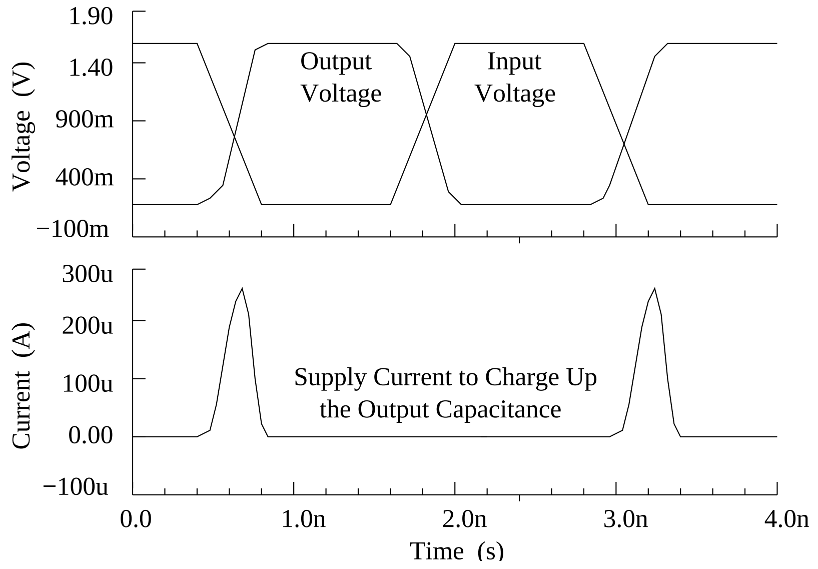


Figure 2.3: Supply current used to charge up the load capacitance.

Equation 2.1 indicates that the supply voltage is the dominant factor in the switching power dissipation. Thus, reducing the supply voltage is the most effective technique to reduce the power dissipation. Other methods such as reducing the switching activity and the load capacitance [Kang03], for reducing the power consumption are also suggested by the equation.

2.2.1.2 Short-Circuit Power Dissipation

In static CMOS circuits, short circuit power dissipation is generated by the short circuit current flowing through both the nMOS and the pMOS transistors during switching. The short circuit current occurs if a logic gate is driven by the input voltage waveforms with the finite rise and fall times, as shown in Figure 2.4. Thus,

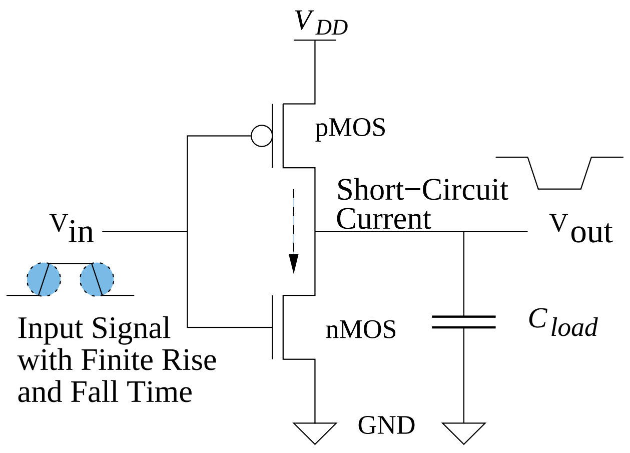


Figure 2.4: CMOS inverter for short-circuit power calculation.

both the nMOS and the pMOS transistors in the circuit conduct simultaneously for a short period of time during the transitions, forming a direct current path between the power supply and GND. This short circuit current does not contribute to the charging of the capacitance in the circuit. Figure 2.5 illustrates the input-output waveforms and the short circuit current of the inverter circuit with zero load capacitance in Figure 2.4. If a symmetric CMOS inverter has the same transconductance (i.e., $k_n = k_p = k$) and threshold voltage parameters (i.e., $V_{T,n} = |V_{T,p}| = V_T$), and if the input voltage waveform has equal rise and fall times ($\tau_{rise} = \tau_{fall} = \tau$), the averaged short-circuit power dissipation with a very small capacitive load is calculated as follows [Kang03]:

$$P_{avg} = \frac{1}{12} \cdot k \cdot \tau \cdot f_{CLK} \cdot (V_{DD} - 2V_T)^3, \tag{2.2}$$

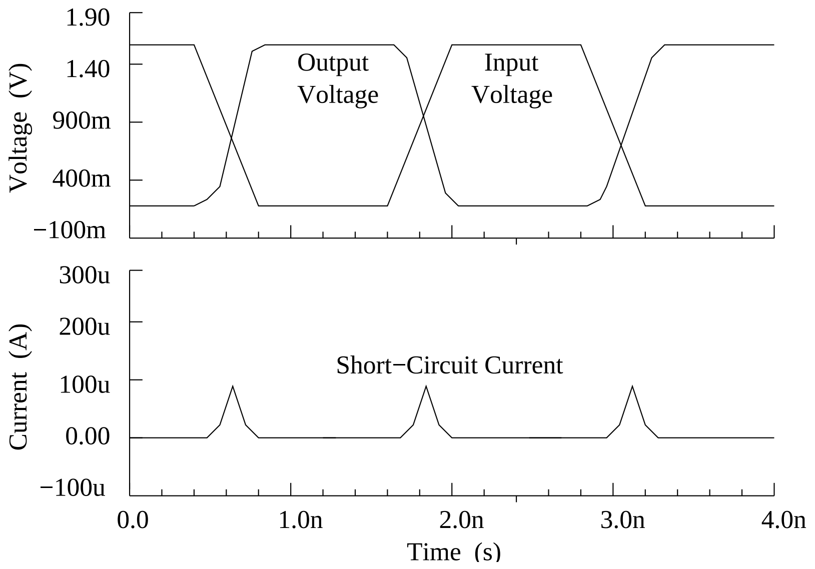


Figure 2.5: Short-circuit current during switching.

where k is transconductance of the transistors, V_T is threshold volatge, and τ represents the equal rise and fall times. Note that the short-circuit power dissipation is linearly proportional to the input signal's rise and fall times. Therefore, reducing the input transition times will decrease the short-circuit current component. However, the increased load capacitance (i.e., the output rise/fall time is larger than the input rise/fall time) can also lead to less short-circuit power dissipation [Veen84]. Yet, this goal should be balanced carefully against other performance goals such as propagation delay.

2.2.1.3 Glitching Power Dissipation

Glitching power is the power dissipated in the intermediate transitions during the evaluation of the logic function of the circuit [Ragh96]. In multi-level logic circuits, the propagation delay from one logic block to the next can cause the input signals to the block to change at different times. Thus, a node can exhibit multiple transitions in a single clock cycle before settling to the correct logic level. These intermediate erroneous outputs lead to a power loss in charging and discharging the output load capacitance.

Primarily, glitches occur due to a mismatch or imbalance in the path lengths in the logic network [Raba96]. Such a mismatch in the path lengths results in a mismatch in the signal timing with respect to the primary inputs. Figure 2.6 denotes a simple multi-level network. If both NAND gates have the same delay and three input signals arrive at the same time, the network will suffer from glitching, as seen in Figure 2.7.

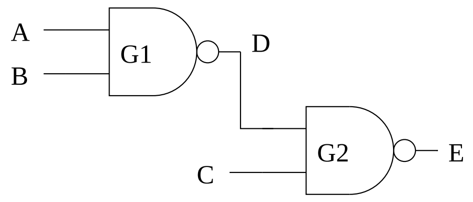


Figure 2.6: Multi-level static CMOS circuit.

To avoid such power loss, designers can use synchronous circuits in which all the outputs are either latched or gated to synchronize the inputs to the next stage. Also, dynamic circuits avoid the problem of glitching power by synchronizing the output with the clock signal. Finally, a careful layout can reduce the skew among the input signals to each logic gate, leading to lower glitching activity.

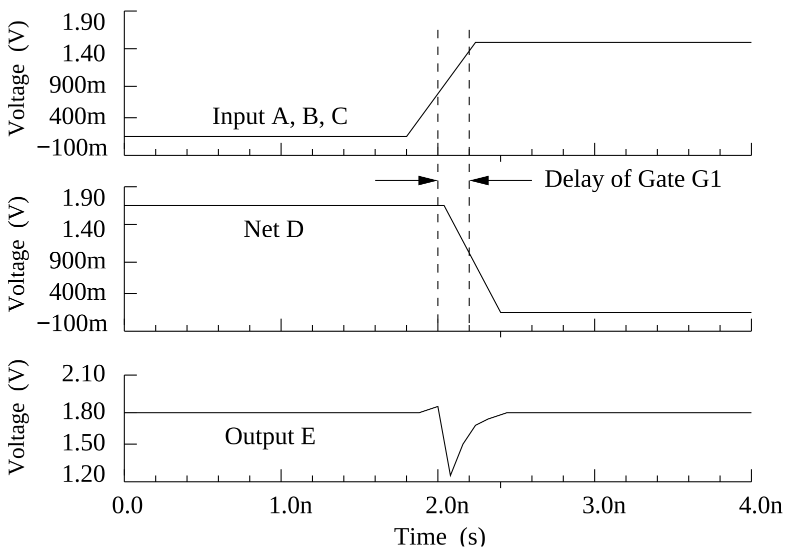


Figure 2.7: Signal glitching in multi-level CMOS circuit.

2.2.2 Static Power Dissipation

Static power is caused by leakage currents while the gates are idle; that is, no output transitions. Theoretically, CMOS gates should not be consuming any power in this mode. This is due to the fact that either pull-down or pull-up networks are turned off, thus preventing static power dissipation. In reality, however, there is always some leakage current passing through the transistors, indicating that the CMOS gates do consume a certain amount of power. Even though the static power consumption, associated with an individual logic gate is extremely small, the total effect becomes significant when tens of millions of gates are utilized in today's integrated circuits (ICs). Furthermore, as transistors shrink in size (as the industry moves from one technology to another), the level of doping has to be increased, thereby causing leakage currents to become larger.

Leakage currents come from a variety of sources within the transistor [Roy00].

For long-channel transistors, the leakage current is dominated by the reverse diode leakage and the subthreshold leakage. Other leakage mechanisms are peculiar to the small-device geometries. In Figure 2.8, a summary of the leakage mechanism in a short-channel transistor is presented.

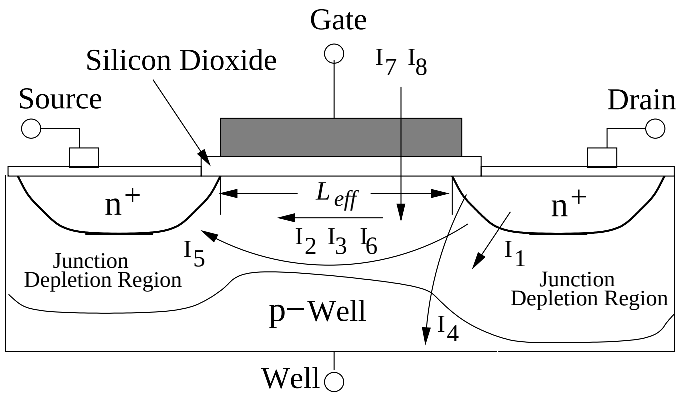


Figure 2.8: Leakage mechanism in short-channel nMOS transistor.

2.2.2.1 Reverse Diode Leakage Current (I_1)

The reverse diode leakage occurs when the pn-junction between the drain and the bulk of the transistor is reverse-biased. The reverse-biased drain junction conducts a reverse saturation current which is drawn from the power supply. The reverse leakage current of a pn-junction is expressed as

$$I_{reverse} = A \cdot J_S(e^{\frac{qV_{bias}}{kT}} - 1), \tag{2.3}$$

where V_{bias} is the reverse bias voltage across the junction, J_S is the reverse saturation current density, and A is the junction area. Since the leakage current is proportional to the junction area, it is advisable to minimize the area as much as possible in the layout. The reverse saturation current density is exponentially proportional to the temperature as well so that the J_S increases dramatically at higher temperatures [Pier96].

2.2.2.2 Subthreshold Leakage Current (I_2)

The subthreshold leakage current (also known as the weak inversion current) occurs when the gate voltage is below the threshold voltage V_T. The subthreshold leakage current can be approximately formulated as [Anis03]

$$I_{subthreshold} = \mu_0 \cdot C_{ox} \cdot \frac{W}{L} \cdot V^2 \cdot e^{1.8} \cdot e^{\frac{(V_{gs}-V_T)}{nV}}, \tag{2.4}$$

where μ_0 is the zero bias mobility, C_{ox} is the gate oxide capacitance, and (W/L) represents the width to the length ratio of the leaking MOS device. The variable V in (2.4) is the thermal voltage constant, and V_{gs} represents the gate to the source voltage. The parameter n in (2.4) is the subthreshold swing coefficient given by $1 + C_d/C_{ox}$ with C_d being the depletion layer capacitance of the source/drain junction. One important point about (2.4) is that the subthreshold leakage current is exponentially proportional to $(V_{gs} - V_T)$. Traditionally, the threshold voltage V_T has been high enough that with $V_{gs} = 0$, the subthreshold current is very small. However, with today's smaller geometry processes such as $90nm$, reduced power supply voltages require the V_T to be reduced also, and thus, the subthreshold leakage at $V_{gs} = 0$ becomes significant.

Equation 2.4 indicates that the subthreshold leakage can be reduced by increasing the V_T or reducing the V_{gs}. However, increasing the V_T affects performance, as will be described later on, and so there is a strong tradeoff between performance and the power dissipation of a design.

2.2.2.3 Drain-Induced Barrier-Lowering Effect (I_3)

Drain-Induced Barrier Lowering (DIBL) occurs when the depletion region of the drain interacts with the source near the channel surface to lower the source potential barrier [Roy00]. The source then injects carriers into the channel surface without the gate playing a role. As a result, the DIBL is enhanced at a higher drain voltage and shorter L_{eff}. DIBL reduces the V_T for short-channel devices [Kao02].

2.2.2.4 Gate-Induced Drain Leakage (I_4)

The Gate Induced Drain Leakage (GIDL) current arises in the high electric field under the gate/drain overlap region which causes a deep depletion [Brew90]. GIDL occurs at a low V_G and high V_D bias and generates carriers into the substrate and drain from the surface traps.

2.2.2.5 Punch-Through (I_5)

Punch-Through occurs when the drain and the source depletion region approach each other and electrically "touch" deep in the channel [Kang03]. Punch-through is a space-charge condition that allows the channel current to exist deep in the subgate region, causing the gate to lose control of the subgate channel region. Punch-Through is regarded as a subsurface version of DIBL, and is obviously an undesirable condition and should be prevented in normal circuit operation.

2.2.2.6 Narrow-Channel Effects (I_6)

MOS transistors, which have channel widths W of the same order of magnitude as the maximum depletion region thickness, are defined as narrow-channel devices [Bohr96]. The most significant narrow-channel effect is that it increases the actual threshold voltage V_T.

2.2.2.7 Gate Oxide Tunnelling (I_7)

The gate oxide tunnelling current arises due to the finite (non-zero) probability of an electron directly tunnelling through the insulating SiO_2 layer. The probability, and thus, the gate tunneling current itself, is a strong exponential function of the gate oxide layer thickness (t_{ox}) and the voltage potential across the gate oxide [Lee03b]. For $t_{ox} \geq 2nm$, the gate tunnelling current is typically very small, compared to that of the other forms of leakage current [Yeo00]. In the most recent generation (i.e., $90nm$ CMOS technology), the gate oxide thickness is scaled down to a range of 1.2-1.6nm to provide a large current at the reduced voltage supply and to suppress the short-channel effects [Ono01]. This results in the presence of a significant gate tunnelling leakage current, which, in some cases, has caught up to the subthreshold leakage in magnitude [Lee03b].

2.2.2.8 Hot-Carrier Injection(I_8)

Reducing the device dimensions to the DSM regime, accompanied by increasing the substrate doping densities, results in a significant increase of the horizontal and vertical electrical fields in the channel region. However, electrons and holes that gain high kinetic energies in the electric field (hot carriers) can be injected into the gate oxide. This causes permanent changes in the oxide-interface charge distribution, degrading the current-voltage characteristics of the MOSFET.

Out of all these leakage currents, the subthreshold leakage is the dominant source of static power [Kao02]. This thesis focuses on circuit-level techniques to handle the exponentially increased subthreshold leakage that is caused by technology scaling.

2.3 Technology Scaling

Since the invention of the first Integrated Circuit (IC) in the early 1960s, CMOS technology has continued to scale down at a dramatic rate. In 1975, Moore predicted that the number of transistors per square inch in an IC will double every 18 months [Moor75]. With each new process generation, all of the lateral and some of the vertical dimensions of the transistors are scaled down to allow a higher level of integration. Figure 2.9 reflects the reduction of the key dimensions of a typical MOSFET with the corresponding increase of the doping densities. Scaled

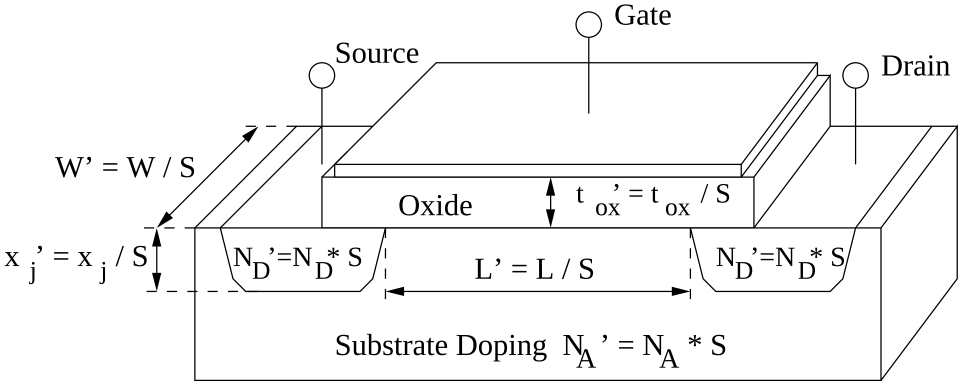


Figure 2.9: Scaling of a typical MOSFET by a factor of S.

dimensions and doping densities have an immediate impact on reducing the power dissipation, as well as increasing the circuit speed. The primary effect of process scaling is the reduction of all the capacitance, which provides a proportional decrease in the power and circuit delays. As today's technology scales below $90nm$, the transistor density will continue to grow. The transistor delay will also continue to improve, at least modestly, to a 30% reduction per generation [Karn02a]. The continued scaling of the technology has meant that designs that were limited by the amount of functionality on a chip are now limited by the amount of constrained power.

In practice, there are two types of scaling strategies for MOSFET devices: full scaling and constant voltage scaling. In full scaling (also called constant field CF scaling), proposed by Dennard et al. in 1974 [Denn74], all the horizontal and vertical dimension of the transistor, as well as the power supply, are scaled down by a factor of S. In order to preserve the magnitude of the internal electric field, the doping densities need to be increased by the same factor S. In constant voltage (CV) scaling, proposed by Chatterjee et al. in 1980 [Chat80], all the dimensions of the MOSFET are reduced by a factor of S, as in full scaling, but the power supply voltage and the terminal voltage remain unchanged. The doping densities are increased by a factor of S^2 in order to preserve the charge-field relation. Table 2.1 summarizes the scaling factors for all the significant dimensions, power supply, doping densities of the MOS transistors, and changes in the key device characteristics for these two scaling strategies [Kang03].

It is evident that CF scaling reduces both the drain and the supply voltage by a factor of S. Therefore, the power dissipation of the transistor decreases by a factor of S^2, and increases by the factor S in CV scaling. This significant reduction of

Parameter	Constant Field (CF)	Constant Voltage (CV)
Channel length (L)	$1/S$	$1/S$
Channel width (W)	$1/S$	$1/S$
Gate oxide thickness (t_{ox})	$1/S$	$1/S$
Junction depth (x_j)	$1/S$	$1/S$
Power supply voltage (V_{DD})	$1/S$	1
Threshold voltage (V_{T0})	$1/S$	1
Doping densities (N_A, N_D)	S	S^2
Oxide capacitance (C_{ox})	S	S
Drain current (I_D)	$1/S$	S
Delay (τ)	$1/S$	$1/S^2$
Power dissipation (P)	$1/S^2$	S
Leakage power ($P_{leakage}$)	exp	1
Power density ($P/Area$)	1	S^3
Power delay produce (PDP)	$1/S^3$	$1/S$

Table 2.1: Influence of scaling on MOS device characteristics.

the power dissipation is one of the most attractive features of CF scaling. However, Intel has used CV scaling in their microprocessors until the appearance of 0.8μm technology, where a 5V supply voltage has been used to maintain the compatibility with the supply voltage of conventional systems, and also to obtain a higher operation speed [Inc04a]. CF scaling has been used since 0.5μm technology has evolved. The main reason for the supply voltage scaling that began in the 0.5μm generation is that CV scaling increases the drain current densities and the power density by a factor of S^3. This large increase in the current and power densities can eventually cause serious reliability problems such as electromigration, hot carrier degradation, oxide breakdown, and electrical over-stress, for the scaled transistor. Another reason for reducing the power supply voltage is to decrease the power consumption of the chip. However, the CF scaling causes the subthreshold leakage currents to grow exponentially and become an increasingly larger component of the total power dissipation [Kao02]. Therefore, effective leakage minimization techniques need to be designed.

2.4 Subthreshold Leakage Reduction Techniques

Equation 2.1 denotes that the average switching power dissipation is proportional to the square of the power supply voltage. Therefore, the reduction of the V_{DD} significantly reduces the power consumption. Although such a reduction is usually very effective, the inevitable design tradeoff is an increase in the circuit delay [Raba96]. This is obvious from the following propagation delay expressions for the CMOS inverter circuit which are

$$\tau_{PHL} = \frac{C_{load}}{k_n(V_{DD} - V_{T,n})}\left[\frac{2V_{T,n}}{V_{DD} - V_{T,n}} + ln\left(\frac{4(V_{DD} - V_{T,n})}{V_{DD}} - 1\right)\right] \tag{2.5}$$

and

$$\tau_{PLH} = \frac{C_{load}}{k_p(V_{DD} - |V_{T,p}|)}\left[\frac{2|V_{T,p}|}{V_{DD} - |V_{T,p}|} + ln\left(\frac{4(V_{DD} - |V_{T,p}|)}{V_{DD}} - 1\right)\right]. \tag{2.6}$$

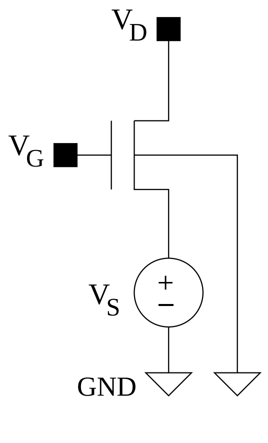
The propagation delays in (2.5) and (2.6) indicate that the negative effect of reducing the power supply voltage on delay can be compensated for, if the threshold voltage of the transistor V_T is reduced accordingly. However, a reduction in the V_T will cause an exponential increase in the device subthreshold leakage, as mentioned in Section 2.2.2.2. In turn, this increases the static power of the device to unacceptable levels [FTsa03]. This clearly justifies the need for leakage reduction techniques, even for current technologies.

Recently, an important area of research focuses on the development of circuit techniques to reduce subthreshold leakage currents that are affected by the supply voltage and the threshold voltage [Kao02]. The most commonly used leakage reduction techniques such as source biasing, dual V_T partitioning, VTCMOS, and

MTCMOS are first reviewed. These techniques reduce leakage currents during the standby states. As technology continues to scale down, leakage currents become excessive, and therefore, need to be balanced during the active mode as well [Kao02].

2.4.1 Source Biasing

The concept of source biasing refers to the application of a positive bias voltage to the source terminal of an "off" nMOS transistor during the standby mode, which raises the threshold voltage of the transistor [Bell95], as illustrated in Figure 2.10. By taking advantage of the body effect phenomenon the subthreshold leakage current can be exponentially reduced. In addition, the gate to source voltage (V_{gs}) becomes negative. The net effects are that the "off" transistor is turned off more strongly and the leakage currents can be reduced during the standby mode.



(a)

Figure 2.10: Source biasing.

2.4.2 Stack Effect

Two series-connected "off" transistors or transistor "stacks" will have lower leakage currents than those of a single "off" device due to the self-reverse biasing effects [Kawa93]. This so-called "stack effect", shown in Figure 2.11, causes the leakage current to vary with the circuit primary input vector [Ye98]. Therefore, special flip-flops can be inserted in the circuit to produce the input vector that provides the least amount of leakage at the primary input flip-flops [Hatl97]. This technique can reduce leakage power at the standby mode. However, determining the input vector that minimizes the leakage current is a difficult problem due to the inherent logic correlations in the circuit.

In [John02], the stack effect is extended to inserting an extra series of "off" devices into the single stack paths. This provides a moderate leakage reduction while a standard single threshold voltage technology is used. One drawback of this approach is that there are no appropriate CAD tools to identify the single stack candidates with enough slack such that inserting extra series devices will not adversely affect the performance.

2.4.3 Dual V_T Partitioning

In modern process technology, multiple threshold voltages are provided for each transistor. A dual V_T process provides the designer with transistors that are either fast (with a high leakage) or slow (with a low leakage). Therefore, a circuit can be partitioned into high and low threshold voltage gates or transistors, which is a tradeoff between performance and reduced leakage currents. For instance, in Figure

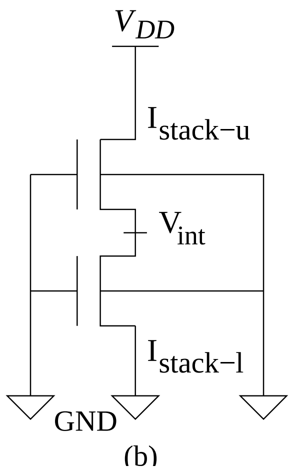


Figure 2.11: Stack effect.

2.12, the critical path within the circuit should be implemented with a low V_T to maximize the performance, whereas non-critical paths should be implemented with high V_T devices to minimize the leakage currents. As a result, the leakage currents are significantly reduced in both the standby and active modes, compared to an all low V_T implementation. At the same time, circuit performance is maintained at low supply voltages.

A limitation of this technique is that CAD tools need to be developed and integrated into the design flow to optimize the partitioning process. A gate-level method for assigning dual threshold voltage is described in [Kato00]. Another method, based on a transistor-lever for a dual V_T assignment, achieves a better leakage reduction because the individual transistors within the gates themselves are optimized to have multiple threshold voltage options [Ketk02]. The dual V_T partitioning technique can also be combined with other techniques to provide an even better performance and leakage reduction. A simultaneous input vector con-

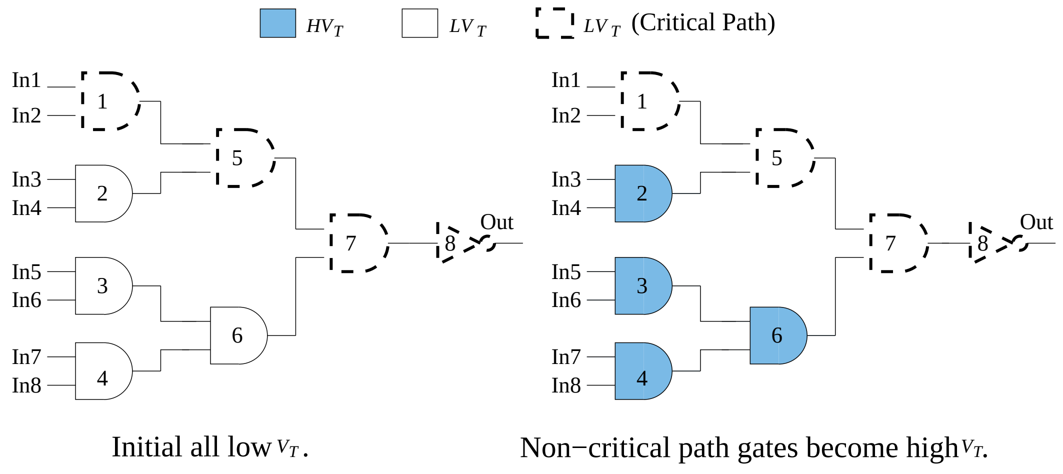


Figure 2.12: Dual V_T partitioning scheme.

trol and dual V_T assignment approach is proposed in [Lee03a], whereas a dual V_T partitioning technique in conjunction with a transistor sizing algorithm is explored in [Siri99, Karn02b].

2.4.4 Variable Threshold CMOS (VTCMOS)

Variable Threshold CMOS (VTCMOS) is a circuit design technique that has been developed to reduce standby leakage currents in low V_{DD} and low V_T applications [Hyun01]. Rather than employ multiple threshold voltage process options, a VTCMOS circuit inherently uses low threshold voltage transistors, and the substrate bias voltages of the nMOS and pMOS transistors are generated by the variable substrate bias control circuit that is depicted in Figure 2.13.

When the inverter circuit in Figure 2.13 is operating in its active mode, the inverter transistors work as conventional CMOS transistors and do not experience any body effect. The circuit operates with a low-power dissipation (due to a low

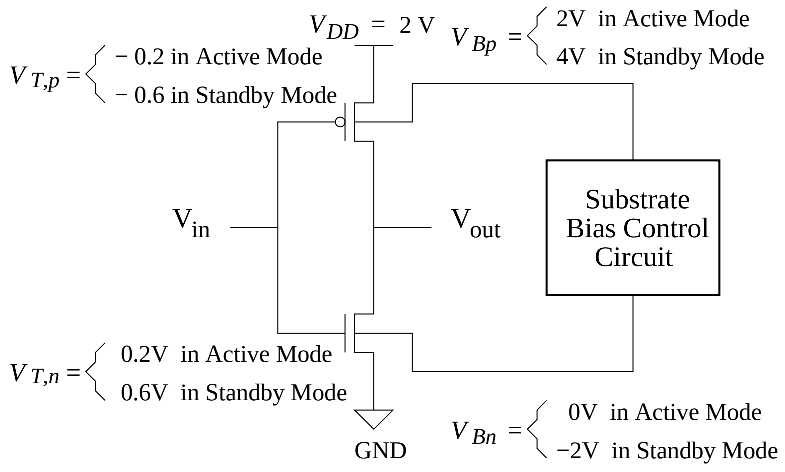

V_{DD} = 2 V
V_{Bp} = 2V in Active Mode / 4V in Standby Mode
$V_{T,p}$ = − 0.2 in Active Mode / − 0.6 in Standby Mode
V_{in}
V_{out}
Substrate Bias Control Circuit
$V_{T,n}$ = 0.2V in Active Mode / 0.6V in Standby Mode
GND
V_{Bn} = 0V in Active Mode / −2V in Standby Mode

Figure 2.13: VTCMOS inverter circuit.

V_{DD}) and a high switching speed (due to a low V_T). When the circuit is in the standby mode, however, the substrate bias control circuit generates a lower substrate bias voltage for the nMOS transistor and a higher substrate bias voltage for the pMOS transistor. As a result, the magnitudes of the threshold voltages $V_{T,n}$ and $V_{T,p}$ both increase in the standby mode due to the body effect. Therefore, the leakage power dissipation in the standby state can be significantly reduced with this circuit design technique.

However, with technology scaling, it has been proved that the effectiveness of VTCMOS reduces as the channel lengths become smaller, or the V_T values are lowered [FTsa03, Karn02a]. Also, VTCMOS is intrinsically more problematic for reliability since the high voltage across the oxide decreases the lifetime of the device [FTsa03].

2.4.5 Multi-Threshold CMOS (MTCMOS)

MTCMOS is a very effective technique to reduce the leakage current of circuits in the standby mode [Muto95]. The principle of the MTCMOS technique is the employment of low V_T transistors to design the logic gates for which the switching speed is essential, and the high V_T transistors (also called sleep transistors) are used to effectively isolate the logic gates in the standby state and reduce the leakage dissipation. The generic circuit structure of the MTCMOS circuit is offered in Figure 2.14. This method will be explained in more detail in Chapter 3.

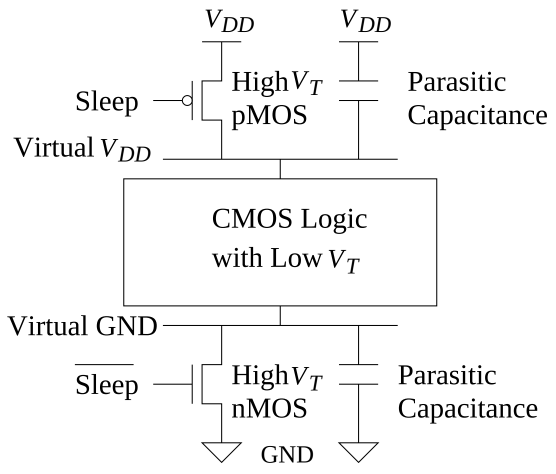


Figure 2.14: Generic structure of a MTCMOS logic gate.

Although the MTCMOS circuit technique is effective for controlling leakage currents in combinational logic, a drawback is that the technique can cause the internal nodes to float and result in the loss of the stored state for the memory units and the flip-flops [Liao02]. As a result, researchers have explored MTCMOS latch designs that can eliminate the leakage currents, yet maintain a state during the standby mode in [Muto95, Shig97, Kao01]. Another problem of the MTCMOS circuit technique comes from the presence of series-connected sleep transistors which increase the overall circuit area and also add extra parasitic capacitance and delay. Consequently, the appropriate sleep transistor sizing is pivotal to the performance, as well as to the leakage power of the entire circuit [Anis02]. Thus, an optimization problem is introduced.

2.5 Heuristics for Combinatorial Optimization

Most of the problems encountered in VLSI design automation are *combinatorial* optimization problems and many of them are *NP-hard* problems [Gere99]. Combinatorial optimization problems are referred to those problems where the variables used to specify the optimal solution are discrete (i.e., they only can assume a finite number of distinct values). If the variables range over real numbers, the problem is called a *continuous* optimization problem. The qualification NP-hard is often encountered when discussing the computational complexity of a problem. The complexity class NP consists of those problems that can not be solved exactly in polynomial time on a common (deterministic) computer.

A combinatorial problem can be solved exactly if the problem size is sufficiently small using an algorithm, such as *exhaustive search* and *branch-and-bound*, that has an exponential (or even a higher order) time complexity in the worst case. The existence of NP-hard problems justifies the use and design of heuristic algorithms such as *local search, simulated annealing,* and *genetic algorithms* that do not guarantee an optimal solution.

2.5.1 Local Search

Local search is the simplest search heuristic. It subsequently visit a number of feasible solutions in the search space until some feasible solution that is better than all its neighbors is found. The transition from one solution to the next is called a *move*. The major disadvantage of local search is that it can get stuck in a local minimum because it only accepts moves that reduce the total cost. In order to overcome the main disadvantage of local search, one should be able to escape the local minimum by accepting the moves with a higher cost (also called *hill-climbing ability*). However, this should be done in a way that still guarantees convergence to some solution.

2.5.2 Simulated Annealing

In 1983, a new algorithm technique—*Simulated Annealing* is presented in [Kirk83]. The idea of simulated annealing is originated from the observation of crystal formation. When a material is heated, the modules move around randomly. When the temperature slowly decreases, the modules move less and finally form a crystal structure. The cooling process is done more slowly, the crystal lattice is more stronger. The simulated annealing technique has been successfully used in many phases of VLSI physical design, including circuit partitioning and placement. The basic procedure in simulated annealing is to start with an initial solution and accept all perturbations or moves which result in a reduction in cost. Moves that result in a cost increase are accepted with a probability that decreases with the increase in cost. A parameter T, called the temperature, is used to control the acceptance probability of the cost-increasing moves [Shah91]. Obviously, simulated annealing is a method with hill-climbing ability. The cooling schedule (i.e., the rate of temperature change) determines the quality of the final solution. It produces good quality results when given a long-enough time and a good cooling schedule but the computation time is also large.

2.5.3 Genetic Algorithms (GA)

The theoretical foundations of the genetic algorithm (GA) were originally developed by Holland [Holl92]. The concept of the GA is based on the evolutionary process of biological organisms in nature. During the course of evolution, populations evolve according to the principles of natural selection and "survival of the fittest". Individuals that are more successful in adapting to their environment will have a better chance of surviving and reproducing, whereas individuals that are less fit will be eliminated. This means the *genes* from the highly fit individuals will spread to an increasing number of individuals in each successive generation. The combination of good characteristics from highly adaptive ancestors can produce even more fit offspring. In this way, the species evolve to become increasing more suitable to their environment.

A GA simulates these processes by taking an initial population of individuals and applying genetic operators in each reproduction phase. In optimization terms, each individual in the population is encoded into a string or *chromosome* which represents a possible *solution* to a given problem. The *fitness* of an individual in evaluated with respect to a given objective function. Highly fit individuals or *solutions* are given opportunities to reproduce by exchanging pieces of their genetic information in a *crossover* procedure with other highly fit individuals. This produces new "offspring" solutions (*children*) who share some of the characteristics from both parents. *Mutation* is often applied after crossover by altering some genes in the strings. The offspring can either replace the whole population or replace the less fit individuals. This evaluation-selection-reproduction cycle is repeated until a satisfactory solution is found.

2.6 Test Circuits

Six combinational circuits are used in [Anis02] and [Anis03] as test benchmarks: a 4-bit Carry Look Ahead Adder (CLAD), a 32-bit parity checker, a 6-bit array multiplier design, a 4-bit ALU/Function generator (74181 ISCAS'85 benchmark), a 32-bit error correcting circuit (C499 ISCAS'85 benchmark), and a 27-bit channel interrupt controller (C432 ISCAS'85 benchmark). These benchmarks are chosen to offer a variety of circuits with different structures employing various gates with different fanouts. In addition, a 16-bit multiplier circuit is developed in this thesis by using the developed MTCMOS design environment. Table 2.2 lists the general information of these benchmarks.

Circuit	4-bit CLA Adder	32-bit Parity Checker	6-bit Multiplier	4-bit 74181 ALU	32-bit Error Correcting	27-channel Interrupt Controller	16-bit Multiplier
Name	CLAD	Parity	Mult1	ALU	Error	AllCh	Mult2
No. of Gates	28	31	30	61	202	160	688
No. of Inputs	9	32	12	14	41	36	32
No. of Outputs	5	1	8	8	32	7	32

Table 2.2: ISCAS'85 benchmarks used for testing.

All the experiments in this thesis are implemented on a SUN Solaris Unix workstation that is equipped with dual 440 MHz CPUs and with a 2-Gbit RAM.

2.7 Summary

The main power dissipation mechanisms in CMOS logic circuits have been described in this chapter. Various phenomena associated with smaller transistor sizes, the result by technology scaling, have also been discussed.

The methodologies of the CF and the CV scaling are introduced and compared with each other. To compensate for the performance loss, caused by the scaling down of the supply voltage, the transistor's threshold voltage V_T should also be reduced which results in an exponential increase in the leakage current. Therefore, circuit techniques have to be developed to handle this rising leakage. Dual V_T partitioning, VTCMOS, and MTCMOS are considered to be the most effective leakage reduction techniques. The optimization problem introduced by the MTCMOS technique can be solved by heuristic algorithms.

This thesis focuses on the research of the MTCMOS technique. In the following chapter, several MTCMOS approaches are introduced to effectively reduce the leakage current.

Chapter 3

MTCMOS Technique

3.1 Introduction

MTCMOS is an enabling technology that provides a high speed performance and low-power operation by utilizing both high and low threshold voltage (V_T) transistors [Muto95, Muto96]. By using low V_T transistors in a signal path, the supply voltage (V_{DD}) can be lowered to reduce the switching power dissipation without affecting the performance. Although the switching power can be reduced quadratically according to the V_{DD} reduction, the V_T that has been decreased for the performance compensation incurs an exponential increase in the subthreshold leakage current. In fact, the increased leakage power can dominate the switching power if the voltage is scaled down aggressively [Chan96]. In many event driven applications, such as a processor running an X-server or a mobile media terminal, circuits are usually in an idle state when no computation is being performed. During this standby state, it is very wasteful to have a large subthreshold leakage current. This static power dissipation in the standby mode can be reduced dramatically by using high V_T transistors (sleep transistors) with very low leakage currents to gate the power supply. Figure 3.1 illustrates the basic circuit scheme of the MTCMOS technique. Note that, in Chapter 2, both the pMOS and nMOS transistors were utilized as sleep transistors as illustrated in Figure 2.14. However, it has been proved that for a purely combinational logic circuit, the circuit performance (speed) can be increased by employing nMOS transistors alone [Muto95] as illustrated in Figure 3.1(a).

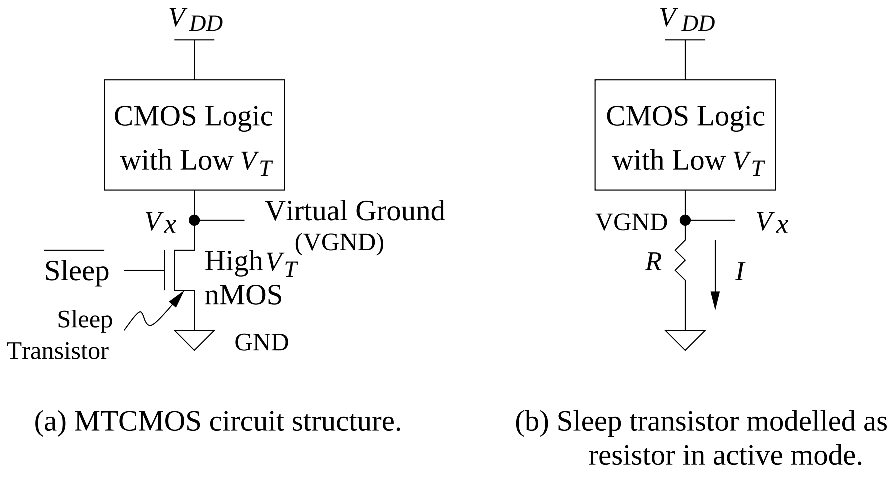

V_{DD}
CMOS Logic with Low V_T
V_X
Virtual Ground (VGND)
$\overline{\text{Sleep}}$
High V_T nMOS
Sleep Transistor
GND
(a) MTCMOS circuit structure.
V_{DD}
CMOS Logic with Low V_T
VGND
V_X
R
I
(b) Sleep transistor modelled as resistor in active mode.

Figure 3.1: Sleep transistor in MTCMOS circuits.

Sleep transistors in MTCMOS circuits are controlled by a "sleep" signal that is used for the active/standby mode control. During the active mode (sleep = 0), the sleep transistor can be approximated by the linear resistor R as seen in Figure 3.1 (b) [Kao97]. This creates a finite voltage drop ($V_X = I \times R$) across the virtual ground nodes as the gates are discharging. This voltage drop causes the internal logic to slow down for two reasons: (*i*) it reduces the gate's driving capability from V_{DD} to $V_{DD}-V_X$, and, (ii) the internal transistor threshold voltages increase due to the body effect [Raba96]. Therefore, the resistor should be small, and consequently, the size of the sleep transistor should be large. This is, however, a tradeoff between the circuit speed in the active mode and the standby leakage current, because the total standby leakage current in a chip is proportional to the width of the sleep transistor [Muto95]. As a result, optimal sizing of the sleep transistor for an arbitrary circuit to meet a performance constraint is difficult. In other words, the current "I" flowing through the sleep transistor in the active mode needs to achieve the required speed.

In the last decade, a number of sleep transistor sizing methodologies have been reported in the literature [Kao02]. The use of a single sleep transistor to support the whole circuit has been proposed in [Muto95]. However, sharing a single sleep transistor for the whole circuit increases the interconnect resistance for the distant blocks. As a result, the sleep transistor has to be sized larger than expected to compensate for the added interconnect resistance. An excessively large size sleep transistor augments the dynamic and leakage power, as well as the area. In order to reduce the area overhead effectively, a hierarchical sizing method, based on the mutual exclusive discharge pattern, was introduced in [Kao98]. The cascaded gates are grouped together by this method, because simultaneous current discharges can not take place. This method can be efficient for balanced circuits with tree configurations, where mutually exclusive discharging gates are easily detected. However, this method is not efficient for circuits with complicated interconnections and unbalanced structures [Anis03]. Therefore, a gate-clustering MTCMOS technique is presented in [Anis01, Anis02]. Although the cluster-based MTCMOS design is better than the hierarchical sizing design [Long03], the former proposed algorithms are inefficient and consume a relatively large CPU time and impractical for large circuit design.

Due to the drawbacks of these techniques, two genetic algorithm (GA) meta heuristics are proposed in this thesis to improve the computation time of the gate-clustering MTCMOS technique. Two new models are also proposed to solve the sleep transistor sizing problem. Both of the models achieve a better solution with less CPU time than those in [Anis02]. Figure 3.2 is a summary of the overall approaches that will be introduced in this chapter. This chapter begins with a brief introduction of the gate-clustering MTCMOS technique in Section 3.2, which is fundamental to the proposed techniques. Section 2.5.3 presents the implementation of the GAs, including the experimental results. The two models for sizing the sleep transistor are presented in Section 3.5 and 3.6.

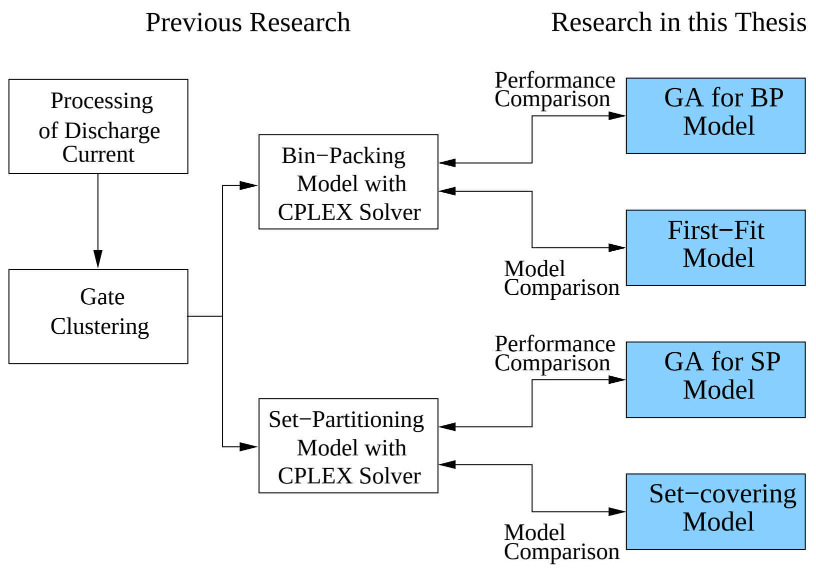


Figure 3.2: Approaches for gate-clustering MTCMOS technique.

3.2 Gate-Clustering MTCMOS Technique

When sleep transistors are absent, the propagation delay (τ_d) of a CMOS gate can be expressed as

$$\tau_d \propto \frac{C_L V_{DD}}{(V_{DD} - V_{TL})^\alpha} \tag{3.1}$$

where C_L is the load capacitance, V_{TL} is the threshold voltage in the low V_T module, and α is the velocity saturation index for modelling the short-channel effects [Kao97]. In the presence of a sleep transistor, the delay of the gate (τ_d^{sleep}) increases to

$$\tau_d^{sleep} \propto \frac{C_L V_{DD}}{(V_{DD} - V_X - V_{TL})^\alpha} \tag{3.2}$$

where V_X is the potential of the virtual ground. If the gate is assumed to tolerate a 5% degradation in performance due to the presence of the sleep transistor, then

$$\frac{\tau_d}{\tau_d^{sleep}} = 95\%. \tag{3.3}$$

According to the analysis in [Anis02], to maintain the gate performance (within a 5% degradation), the size of the sleep transistor can be expressed as

$$\left(\frac{W}{L}\right)_{sleep} = \frac{I_{sleep}}{0.05\mu_n C_{ox}\left(V_{DD} - V_{TL}\right)\left(V_{DD} - V_{TH}\right)} \tag{3.4}$$

where I_{sleep} is the discharge current flowing through the linearly-operating sleep transistor, μ_n is the N-mobility, C_{ox} is the oxide capacitance, and V_{TH} is the threshold voltage of the sleep transistor[1].

[1]V_{TL} and V_{TH} are set at 350mV and 500mV, respectively, for the 0.18μm technology in this thesis.

If the size of the sleep transistor is designed as a constant (i.e., the width W is fixed), (3.4) implies that the discharge current passing through the sleep transistor has to be less than the limit value I_{sleep} in order to keep gate performance degradation within 5%. This is a key parameter for the gate-clustering technique to guarantee the performance constraint; that is, each gate in the cluster-based MTCMOS circuit satisfies the performance criteria. This ensures that any combination of the gates in the path, including the worst case delay path caused by the worst case input vector, and thus, the whole circuit, also meets the performance requirements.

To overcome the increased interconnect resistance drawback, introduced by using a single sleep transistor for the whole circuit in [Muto95, Kao98], the gate-clustering technique employs multiple sleep transistors, of equal size, distributed evenly over the circuit. The optimum size W, and consequently I_{sleep}, are chosen to have a minimal dynamic and leakage power dissipation. This optimization problem is accomplished by a discharge current processing stage, illustrated in the next section, and the cluster assignment modelling, presented in Sections 3.2.2 and 3.2.3, respectively.

3.2.1 Processing of Discharge Currents

The accuracy of sizing the sleep transistor, while the system performance requirements are maintained, is heavily dependent on how well the discharge currents at the output of each gate in the circuit are modelled. Cascaded gates that have exclusive discharge currents are clustered together in [Kao98], because these discharge currents do not occur at the same time. Therefore, these gates can share one sleep transistor to reduce the total number of sleep transistors, and consequently, the total area. In fact, taking a triangular shape waveform, the discharge current itself varies in the time domain. Therefore, more gates that have a partially overlapped discharge current with the cascaded gates can be added to this cluster to further reduce the total sleep transistor number. This idea is adopted by the gate-clustering technique, and a fundamentally better solution is then achieved in [Anis03].

An innovative method is proposed in [Anis02] to model the discharge current as a vector. Each discharge current at the output of a gate is represented by a vector, where the time axis is divided into 10psec time slots (an adequate accuracy for 0.18μm CMOS technology). Each time slot holds a value that represents the magnitude of the discharge current at that specific time. For each gate, all the input transitions, causing the output to switch from logic “1” to logic “0”, are applied (e.g., 00 to 11, 10 to 11, and 10 to 11 for the two-input NAND gate G1 in Figure 3.3(a)), and the highest discharge current (peak-value I_{max}) at the output of every gate is monitored (a worst case discharge current), while the gate’s fanout is taken into consideration. Since the delay of a gate changes with the different transitions, the earliest peak-current time and the latest peak-current time are also documented (e.g., the t_{1min} and t_{1max} for gate G1 in Figure 3.3(b)). Therefore, a trapezoid waveform, which is the bold line in the timing diagram of Figure 3.3(b) and (c), is adopted to represent the discharge current for each gate in the circuit. This guarantees that the sleep transistor is sized properly, and the circuit meets the target performance.

Based on the trapezoid waveform, a vector is constructed for each gate (i.e., Vec1, Vec2 in Figure 3.3). The vector carries all the information about this gate in

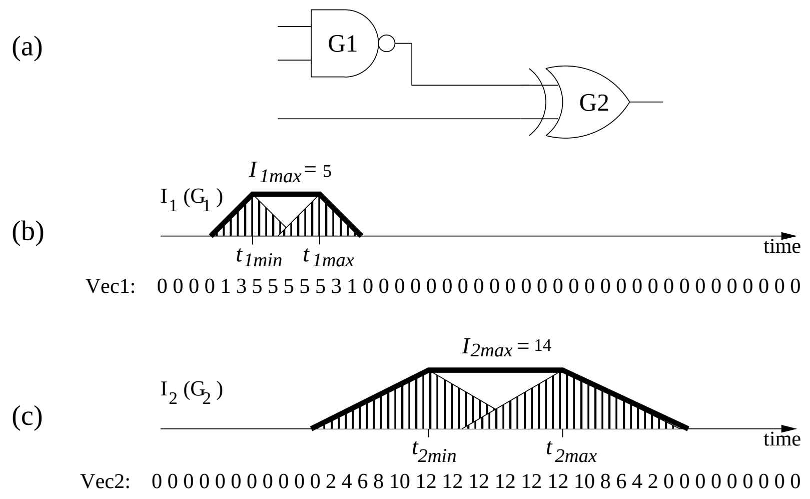

(a)
G1
G2
(b)
I_{1max} = 5
I_1 (G_1)
t_{1min} t_{1max}
time
Vec1: 0 0 0 0 1 3 5 5 5 5 5 5 3 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
(c)
I_{2max} = 14
I_2 (G_2)
t_{2min} t_{2max}
time
Vec2: 0 0 0 0 0 0 0 0 0 0 0 2 4 6 8 10 12 12 12 12 12 12 12 10 8 6 4 2 0 0 0 0 0 0 0 0 0

Figure 3.3: Discharge current timing diagram and vector modelling.

the circuit such as the delay of the gate, the fanout of the gate, the delay level, and the magnitude of the discharge current in each time slot [Anis03]. Figure 3.4 illustrates the heuristic used in the gate-clustering MTCMOS technique to efficiently group the gates into clusters. The main objective of this heuristic is to cluster the circuit such that their combination does not exceed the maximum current of any gate within the cluster [Anis02].

3.2.2 Bin-Packing Problem (BPP)

Following the formation of clusters, the maximum discharge current value (i.e., the peak-value) in the time domain of each cluster is modelled as the cluster capacity (i.e., the worst case). This static value is then adopted to represent the equivalent discharge current of the cluster. The main objective here is to assign as many clusters as possible to a sleep transistor such that the accumulated current of clusters does not exceed the current limit I_{sleep} of the sleep transistor, as show in Figure

```
PROCESSING HEURISTIC
1. Initialize current vectors;
2. Set all gates free to move to any cluster;
3. **For** all gates in circuit
        **If** gate $G_i$ is not clustered yet
            assign gate $G_i$ to **new cluster** $C_k$
            update cluster current vector
            calculate max current, start, and end time
        **End If**
        **For** all other gates in circuit
            **If** gate $G_j$ is not clustered yet
                add current of gate $G_j$ to existing cluster $C_k$
                **If** combination $\leq$ max current
                    append gate to cluster
                    update cluster info
                    set gate $G_j$ locked in cluster $C_K$
                **End If**
        **End For**
   **End For**
4. Return all clusters formed.
```

Figure 3.4: Heuristic for forming clusters [Anis02].

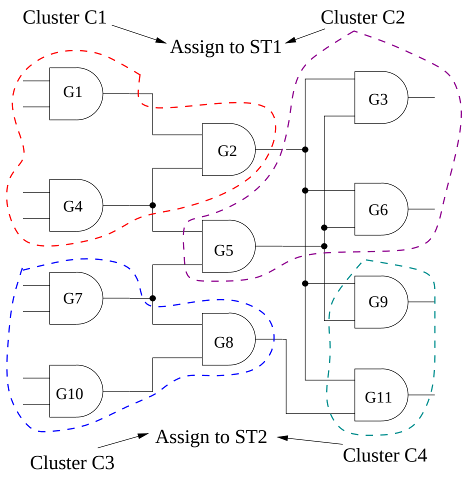


Figure 3.5: Gate-clustering and BPP techniques.

3.5. In addition, the number of sleep transistors allocated should be kept to a minimum. This problem presentation is analogous to the Bin-Packing Problem (BPP) in operation research. The BPP [Rard98] can be described as follows: given n *items* (a set of *clusters*) and m *bins* (*sleep transistors*) with

I_{EQ_j} = *capacity* of cluster j and

I_{max} = *capacity* of each *sleep transistor*,

the objective is to assign each I_{EQ} to one bin so that the total capacity in each bin does not exceed I_{max}, and the number of bins is minimized.

The mathematical formulation of the problem is as follows:

$$\text{Minimize Z } = \sum_{i=1}^{m} y_i, \tag{3.5}$$

subject to

$$\begin{aligned} &\sum_{j=1}^{n} I_{EQ_j} x_{ij} \leq I_{max} y_i \quad i \in \{1, ..., m\}, \\ &\sum_{i=1}^{m} x_{ij} = 1, \end{aligned} \tag{3.6}$$

where

$$y_i = \begin{cases} 1, & \text{if bin } i \text{ is used} \\ 0, & \text{otherwise} \end{cases} \qquad x_{ij} = \begin{cases} 1, & \text{if item } j \in \text{bin } i \\ 0, & \text{otherwise.} \end{cases}$$

This model is a pure Binary Integer Programming problem (BIP). The objective function Z to be minimized is analogous to the minimum number of sleep transistors that are used. The variable y_i is analogous to the available sleep transistors. The variable x_{ij} takes a value of "1", if cluster I_{EQ_j} is assigned to sleep transistor i. (3.6) guarantees that the total discharge current flowing through the sleep transistor is less than the limit value, and each cluster is covered by a sleep transistor once. CPLEX 7.5 [Inc02], a commercial ILP solver, is used in [Anis02] to solve this BPP problem and to determine which clusters should be grouped together and to which sleep transistor they are assigned.

Six different sleep transistor sizes which correspond to six different discharge current limit values (I_{sleep}) are considered in [Anis02]. The optimum I_{sleep} value that dissipates the least leakage power in the standby mode is then recorded for each test circuit. Table 3.1 shows the different values for I_{sleep} and the corresponding $(W/L)_{sleep}$ and W_{sleep} by using (3.4), where L_{sleep} is taken as 180nm for a 0.18μm CMOS technology.

I_{sleep} (μA)	150	200	250	300	350	400
$(W/L)_{sleep}$	3.67	4.89	6	7.5	8.56	9.78
W_{sleep} (μm)	0.66	0.88	1.1	1.32	1.54	1.76

Table 3.1: Values for I_{sleep}.

Even though the Bin-Packing technique is effective for sizing the sleep transistor, the problems such as the increased wirelength and the routing complexity, caused by sleep transistor insertion, are also introduced.

3.2.3 Set-Partitioning Problem (SPP)

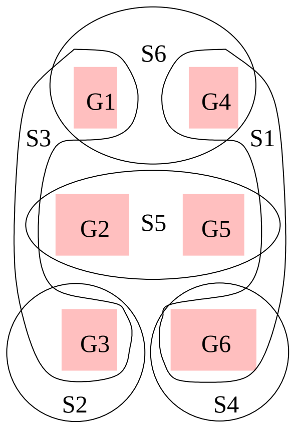
In order to take the physical location of the gates on the chip into account and reduce the routing complexity of large circuits, the Set-Partitioning technique is applied in [Anis02].

The Set-Partitioning Problem (SPP) [Rard98] can be described as follows: given

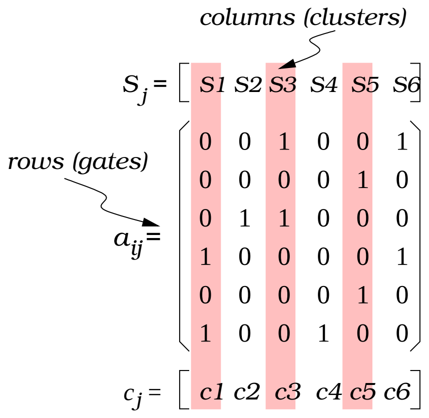
m *rows* (all the *gates* in the circuit in this case) and n *columns* (a set of *clusters* in this case), each *column* covers several *rows*, with

c_j = *cost* associated with column j.

The objective is to find a subset of *columns* that covers all the *rows* once, and the total cost is minimized. Figure 3.6(b) depicts a feasible solution ($S1 = S3 = S5 = 1,\ S2 = S4 = S6 = 0$) for a simple example of the SPP in Figure 3.6(a).



(a) A set of clusters.



$$S_j = \begin{bmatrix} S1 & S2 & S3 & S4 & S5 & S6 \end{bmatrix}$$

$$a_{ij} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

$$c_j = \begin{bmatrix} c1 & c2 & c3 & c4 & c5 & c6 \end{bmatrix}$$

(b) A feasible solution..

Figure 3.6: Simple example of the Set-Partitioning Problem (SPP).

The mathematical formulation of the SPP is as follows:

$$\text{Minimize Z } = \sum_{j=1}^{n} c_j S_j, \tag{3.7}$$

subject to

$$\sum_{j=1}^{n} a_{ij} S_j = 1$$
$$i = 1, ..., m$$

$$S_j \in 0, 1 \quad j = 1, ..., n, \tag{3.8}$$

where

$$S_j = \begin{cases} 1, & \text{if the } j\text{th column is selected} \\ 0, & \text{otherwise} \end{cases} \qquad a_{ij} = \begin{cases} 1, & \text{if row } i \text{ is covered by column } j \\ 0, & \text{otherwise.} \end{cases}$$

In this model, n is equivalent to the number of clusters that are generated. Each row ($i = 1, ...m$) represents a constraint, where gate m should belong. The column ($j = 1, ...n$) represents the feasible clusters (i.e., sleep transistors) that accommodate a set of the gates in the circuit. Therefore, the objective of the low-power SPP is to find the *best* collection of clusters such that each gate is covered by exactly one cluster. The model is also a 0-1 pure integer Linear Programming (LP) problem which is again solved by using the CPLEX solver.

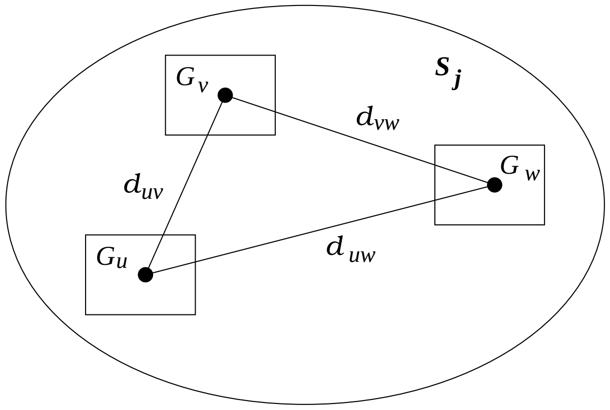


Figure 3.7: Cost function calculation example.

The cost associated with each cluster is formulated as follows [Anis02]:

$$c_j = (w_1 \times c_{j1}) + (w_2 \times c_{j2}), \tag{3.9}$$

where c_{j1} is a distance function (i.e., the rectilinear distance between the gates in the cluster), and c_{j2} represents the difference between the maximum cluster capacity and the sum of all the currents of the gates in the cluster. For example, in Figure 3.7, group S_j is composed of gates G_u, G_v, and G_w. Coefficients c_{j1} and c_{j2} are calculated by

$$c_{j1} = d_{uv} + d_{vw} + d_{uw}$$

and

$$c_{j2} = I_{Sleep} - \sum current_i \ \ \forall i.$$

w_1 and w_2 in (3.9) are the weights, associated with the cost of the two constraints. In [Anis02], the weights w_1, w_2 are assigned equal values of 0.5 to balance the distance and capacity constraints.

Figure 3.8 shows the heuristic presented in [Anis02] to generate groups of clusters that are used by the SPP technique. Step 3 is specifically designed to guarantee a feasible solution.

Both Bin-Packing and Set-Partitioning have been proven to be Non-deterministic Polynomial (NP)-hard problems [Gare79]. These NP-hard problems can be solved by exact method such as branch-and-bound technique available in the CPLEX solver. However, as the problem size (i.e., the number of gates in the circuit) increases, it becomes quite inefficient to solve these problems by an exact method (e.g., the CPLEX) in a reasonable amount of time. The experimental results in [Anis03] indicate that the CPU time, used by the CPLEX solver to solve the BPP and the SPP, increases dramatically as the circuit size increases. Therefore, heuristic methods need to be developed for large circuit designs to produce solutions close to the optimum in a reasonable amount of time.

```
               CLUSTERING HEURISTIC
Create_Cluster()
1. Calculate distances between all gates;
2. Initialize maxgates_per_cluster = n;
3. Create clusters with single gate;
4. For  cl = 2; cl $\leq$ maxgate_per_cluster
          Create_n_Gate_Clusters(cl)
   End For
5. For all clusters created calculate_cost();
6. Return().

Create_n_Gate_Clusters(cl)
1. For  cluster of type cl
          create_new_cluster()
          While not done
             choose gate with minimum distance
             If sum of currents $\leq$ capacity
                append gate to newly created cluster
             End If
             If total gates within cluster $\geq$ limit
                break;
          End While
   End For
```

Figure 3.8: Heuristic for grouping gates into clusters [Anis02].

3.3 Genetic Algorithm (GA) for the BPP

With the increasing interest in evolutionary algorithms and their applications to combinatorial optimization problems, two genetic algorithms (GAs) are designed and applied to the BPP and SPP for low-power MTCMOS circuit design.

Representation and Fitness Function

The first step in designing a GA for a particular problem is to devise a suitable representation scheme. The group based representation scheme [Falk94] is adopted in this BPP oriented GA. Figure 3.9(a) reflects the layout view of a circuit. Eight clusters (namely, cluster 1, 2... 8) are formed according to the discharge current processing heuristic, presented in Section 3.2.1. Figure 3.9(b) illustrates the solution string (*chromosome*) representation of the group-based encoding method.

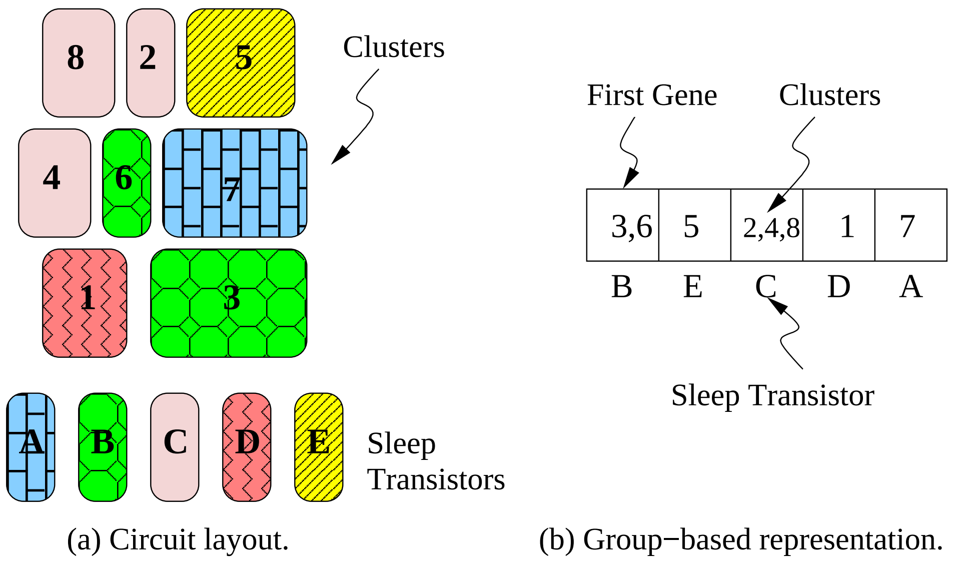


Figure 3.9: Chromosome representation.

With this encoding scheme, the genes represent both items (clusters) and bins (sleep transistors). In this example (as illustrated in Figure 3.9(b)), eight clusters are assigned to five sleep transistors (namely, A, B...E), where the first gene indicates that sleep transistor B is used to cover clusters 3 and 6.

The fitness function of an individual chromosome is a function of the number of sleep transistors (bins) that are involved to construct this solution. The fitness of the chromosome in Figure 3.9(b), constructed with five sleep transistors, is five, consequently. Apparently, group-based representation can cause chromosomes to have a variable length.

Initial Population

The initial population in this GA is generated by the First-Fit (FF) heuristic, based on the previous representation scheme. The first cluster (item), randomly chosen from all the clusters, is assigned to the first sleep transistor (bin). Once a cluster is collapsed into a sleep transistor, it is locked in and is unable to participate in the formation of a new sleep transistor. All the other clusters are then randomly considered to be appended to the sleep transistor. The criterion that is used to append a cluster to a sleep transistor is based on the maximum current capacity of the sleep transistor. Each unlocked cluster is assigned to the lowest-indexed initialized sleep transistor that it fits. When the current cluster cannot fit into any initialized sleep transistor, a new sleep transistor is introduced. This process terminates when it is not possible to append any further clusters to the sleep transistor.

Parent Selection Method

Parent selection is the task of assigning reproductive opportunities to each individual in the population according to their relative fitnesses. The commonly used binary tournament selection [Gold91] method is applied to the GA. In a binary tournament selection, two individuals are chosen randomly from the population. The more fit individuals (fewer sleep transistors are involved) is then allocated a reproductive trial. In order to produce a child, two binary tournaments are held, each of which produces one parent string. These two parent strings are then recombined, using crossover and mutation, to produce two children.

Crossover and Mutation Operator

The crossover operator takes genes from each parent string and *combines* them to create an offspring. By creating new strings from sub-strings of fit parent strings, new and promising areas of the search space are explored. As pointed out earlier, group-based representation consists of two parts: sleep transistors and groups of clusters. Therefore, the crossover operator operates on variable-length chromosomes, with genes representing the sleep transistors. The crossover procedure is illustrated in Figure 3.10, and the concept is presented in Figure 3.11.

Typically, a mutation provides a small random search. Not only does mutation play the crucial role of replacing the gene value lost during the crossover process, but also expands the search space. However, due to the small solution space after grouping the gates into clusters, the mutation operator is not implemented in this GA.

CROSSOVER OPERATOR

1. Select at random two crossing sites, delimiting the crossing section in each of the two parents;
2. Inject the contents of the crossing section of the first parent at the first crossing site of the second parent;
3. Eliminate all the clusters occurring twice from the sleep transistors that were members of the second parent;
4. Adapt the resulting sleep transistors by applying First-Fit heuristic;
5. Apply steps 2 to 4 to the two parents to generate the second child.

Figure 3.10: Crossover procedure.

Population Replacement Scheme

Following the crossover operator, the population for the next generation is then chosen from the combined set of parents and two children. In order to keep the best individuals around all of the time, a method, called *elitism*, is used. Two worst individuals in the population and two children are compared and the two fittest individuals are kept to next generation. The *elitism* strategy guarantees that the best individual in the current generation will appear in the subsequent generation, protecting the search from regression [Mitc96].

3.3.1 BPP Results

After the generation of the clusters for each test circuit by using the heuristic in [Anis02], 322 clusters are formed for the 16-bit multiplier circuit which is the

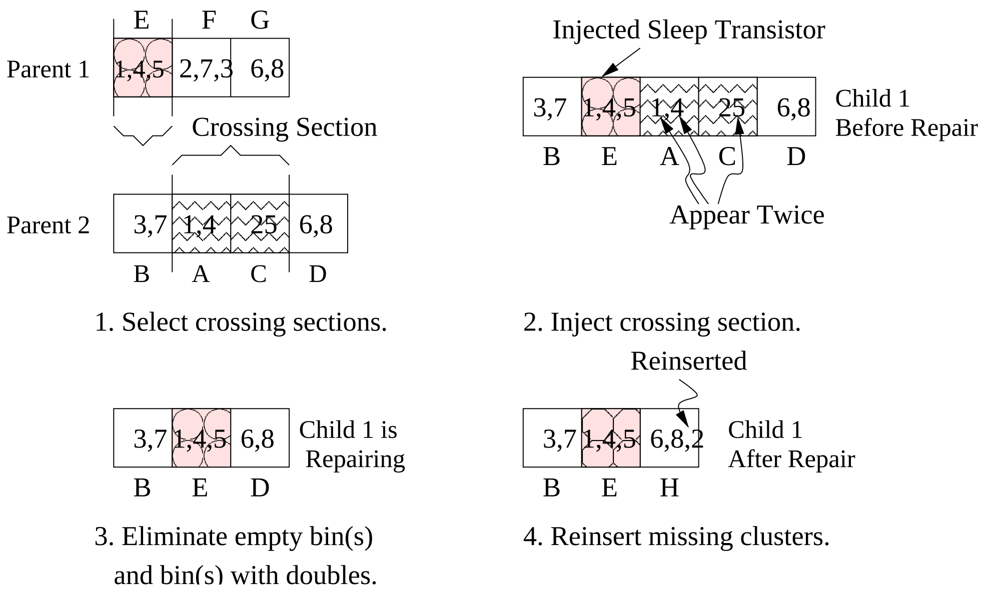


Figure 3.11: Crossover operator.

largest BPP size in these experiments. Figure 3.12 illustrates the convergence of the developed GA for this circuit. Employing 10 populations, the GA find the optimal solution after 40 generations, as seen from the figure. Therefore, only 10 chromosomes are used to evolve within 100 generations for all the experiments. The experimental results indicate that these parameters are *large* enough for the proposed GA to find the optimal solution for all the test circuits. Since the subthreshold leakage power dissipation of a circuit is proportional to the total number of sleep transistors [Muto95], the leakage power saving, achieved by the GA for each test circuit, is equivalent to that obtained by the CPLEX solver. Furthermore, the execution time of the GA for all the experiments, especially for the largest circuit (Mult2), is largely reduced, compared to that of the CPLEX exact method. Table 3.2 reflects the CPU time in seconds, used by the CPLEX solver and the GA, respectively. The column “CPL” lists the CPU time for each test, executed by the CPLEX ILP solver, whereas column “GA” designates the computation time that is consumed by the genetic algorithm. Even though most of the CPU times are small because of the small problem size, the proposed GA still achieves a 89%, on average, reduction in the CPU time, compared to that of the ILP CPLEX solver.

The principal advantage of the GA implementation arises from the controllable computation time, compared to that of the CPLEX solver. Figure 3.13 plots the effect of the size of the sleep transistor (I_{sleep}) with respect to the CPU time, used by the CPLEX solver and the GA for the ALU and AllCh benchmarks. For both circuits, there exists a spur point for the CPLEX solver (150 μA for ALU, 250 μA for AllCh), where the computation time is significantly larger than the rest of the values. However, the GA computational complexity is quite even for all the benchmarks, which indicates that it scales well for larger circuits.

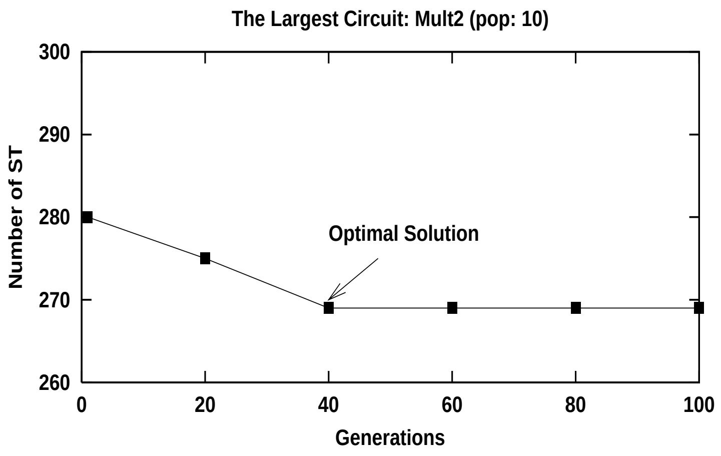


Figure 3.12: Convergence of the GA.

Circuit	I_{sleep}=150		I_{sleep}=200		I_{sleep}=250		I_{sleep}=300		I_{sleep}=350		I_{sleep}=400	
	CPL	GA	CPL	GA	CPL	GA	CPL	GA	CPL	GA	CPL	GA
CLAD	0.22	**0.01**	0.16	**0.01**	0.03	**0.01**	0.10	**0.01**	0.15	**0.01**	0.19	**0.01**
Mult1	0.10	**0.01**	0.13	**0.01**	0.03	**0.01**	0.02	**0.01**	0.05	**0.01**	0.03	**0.01**
Parity	0.09	**0.01**	0.46	**0.01**	0.04	**0.01**	0.08	**0.01**	0.08	**0.01**	0.02	**0.01**
ALU	4233.2	**0.01**	1.36	**0.1**	0.25	**0.01**	0.16	**0.01**	0.18	**0.01**	0.13	**0.01**
Error	0.56	**0.1**	0.89	**0.01**	0.81	**0.01**	0.25	**0.01**	0.06	**0.01**	0.26	**0.01**
AllCh	14.01	**0.1**	37.26	**0.1**	11284	**0.1**	3.11	**0.1**	4.45	**0.1**	3.53	**0.1**
Mult2	-	-	-	-	-	-	-	-	16299.5	**3.0**	15286.1	**3.0**
Avg Imp	92.39%		95.72%		83.68%		85.66%		90.64%		85.08%	

- I_{sleep} violation, no feasible solution is found.

Table 3.2: Comparison of CPU time(s) for CPLEX and GA.

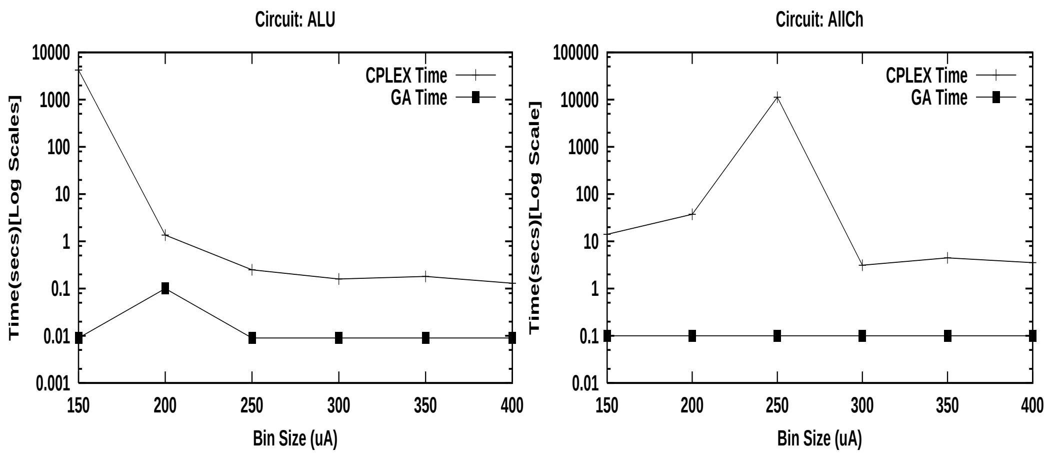


Figure 3.13: Computation time for CPLEX and GA.

3.4 Genetic Algorithm (GA) for the SPP

The evolutionary algorithm implemented in this thesis to solve the Set-Partitioning problem for low-power MTCMOS is a modification of the GA, presented in [Chu95], where problem-specific knowledge of the SPP is considered. A brief overview of this GA is presented in Figure 3.14, and the detailed implementation is discussed in the following sections.

```
GENETIC ALGORITHM
1. Generate an initial population of $N$ random solutions,
   set iteration counter t = 0 ;
2. Select two solutions $P_i$ and $P_j$ from the population using
   the MCS method;
3. Combine $P_i$ and $P_j$ to form a new solution C using the
   fusion crossover operator;
4. Mutate selected bits in C and perform
   dynamic mutation if required;
5. Apply the heuristic feasibility operator to C in an
   attempt to make C more feasible;
6. If C is identical to any one of the solutions in the
   population, go to step 2, otherwise, set t = t+1;
7. Replace a solution in the population with C using
   the SOR scheme;
8. Repeat step 2-7 until t = M non-duplicate solutions have
   been generated. The best feasible solution found is the
   one with the smallest fitness in the population.
```

Figure 3.14: Genetic Algorithm (GA) for Set-Partitioning Problem (SPP).

Chromosome Representation

Usually, a 0-1 binary representation is used for the SPP since it is a natural representation of the underlying 0-1 integer variables. A n-bit binary string is considered as the chromosome structure, where n is the number of columns (clusters) in the SPP. Figure 3.15 illustrates the binary representation of a chromosome. A value of 1 for the i-th bit implies that column "i" is in the solution.

Column(Gene)	1	2	3	4	5	...	n−1	n
Bit String	1	0	1	1	0	...	1	0

Figure 3.15: Binary representation of chromosome.

Fitness Function

Each individual in the population is evaluated and assigned a fitness accordingly. Fitness is used in the selection and replacement phases to decide which individuals should be chosen for reproduction and replacement. For the SPP, the lower the fitness score, the more fit the solution is, because the SPP is a minimization problem. The traditional operational research algorithms restrict their search to feasible solutions, and so, no additional term is included in the SPP objective function to handle constraint violations. Since finding a feasible solution to the SPP is difficult, in the GA approach, the majority of the solutions, generated by the crossover operator and mutation operator, may be infeasible [Chu95]. Therefore, the fitness function used in this GA not only takes into account the costs of the columns included in the solution (i.e., the SPP objective function value), but also the degree of the feasibility of a solution.

There are two methods to define the fitness of an individual. The most common approach is to employ a penalty function to allow the constraints to be violated.

According to the magnitude of the violation, a penalty that is proportional to the size of the infeasibility is generated to degrade the SPP solution fitness. If the penalty is large enough, highly infeasible individuals will rarely be selected for reproduction, and the GA will concentrate on feasible or near-feasible solutions. A typical penalty function $p(x)$ may be defined as follows:

$$p(x) = \sum_{i=1}^{m} \lambda_i \phi_i(x), \tag{3.10}$$

where λ_i is a scalar weight that penalizes the violation of the constraint, and $\phi_i(x)$ is a function of the violation. However, choosing the optimal value for λ_i is difficult [Levi94].

Another alternative approach for defining fitness is proposed in [Chu95]. This method involves separating the single fitness measure into two: one is called *fitness* and the other is called *unfitness*. Each individual can then be represented by a pair of values f_p and u_p. The fitness f_p of an individual p equals its SPP objective function value which is calculated by

$$f_p = \sum_{j=1}^{n} c_j s_{pj}, \tag{3.11}$$

where s_{pj} is the value of the j-th bit (column) in the string corresponding to the p-th individual, and c_j is the cost of bit (column) j. The unfitness u_p of an individual p measures the amount of infeasibility and is defined as

$$u_p = \sum_{i=1}^{m} |w_i - 1|, \tag{3.12}$$

where $w_i = \sum_{j=1}^{n} a_{ij} s_{pj}$ is the number of columns that cover row i. The absolute value in (3.12) implies that an individual is feasible, if $u_p = 0$, and infeasible, if $u_p > 0$. Originally, this approach was adopted in [Chu95] to evaluate the individual chromosomes for the general SPP. However, the MTCMOS SPP problem is specially formulated in [Anis02] so that the feasibility operator, combined with this GA, can guarantee that all the solutions during the evolution are feasible. Therefore, the *unfitness* value of each individual is zero.

Initial Population

The initial population is generated randomly. Each of the initial solutions S_p is formed by using the following method:

1. Set $S_p = \emptyset$, Set $U = I$;
2. Randomly select a row $i \in U$;
    (a) randomly select a column $j \in \alpha_i$ such that $\beta_j \cap (I - U) = \emptyset$;
    (b) if no such j exists, set $U = U - i$;
    else, add j to S_p, and set $U = U - i, \forall_i \in \beta_j$;
3. Repeat step 2 until $U = \emptyset$.

where

I = the set of all rows

α_i = the set of columns that cover row i

β_j = the set of rows covered by column j

U = the set of uncovered rows

Parent Selection Method

The maximum compatibility selection (MCS) method, introduced in [Chu95], is considered to select the parents. In this selection method, one parent P_i is first selected by using a binary tournament [Gold91], based on the *fitness* value. The other parent P_j is then selected to give a maximum compatibility score measured by

$$|R_{Pi} \cup R_{Pj}| - |R_{Pi} \cap R_{Pj}|, \tag{3.13}$$

where R_{Pi} and R_{Pj} are the set of rows that are covered by P_i and P_j, respectively. If j is not unique, then the tie-breaking rule is to select the member of the population with the lowest fitness score. The goal of this method is to find two parents that can cover as many rows as possible and have as few rows in common as possible.

This MCS method is specifically designed for a general SPP, where most populations are infeasible. However, the MTCMOS SPP is specially formulated such that the feasibility operator will fully repair a solution after crossover and mutation. Therefore, the commonly used roulette wheel and binary tournament selection methods are considered instead of the MCS method.

The roulette wheel is a proportionate selection scheme in which the slots of a roulette wheel are sized according to the fitness of each individual in the population. An individual is selected by spinning the roulette wheel and locating the position of the marker. Therefore, the probability of selecting an individual is proportional to its fitness. In a binary tournament selection, two individuals are picked randomly from the population, and the individual with least fitness value is selected, as mentioned in Section 3.3.

Tournament selection provides more pressure in later generations, when the fitness values of individuals are not significantly different. Thus, the roulette wheel selection is more likely to converge to a suboptimal result than the tournament selection, if the individuals have large variations in fitness values. Several experiments are conducted as plotted in Figure 3.16 , and accordingly, the binary tournament selection is incorporated into the algorithm.

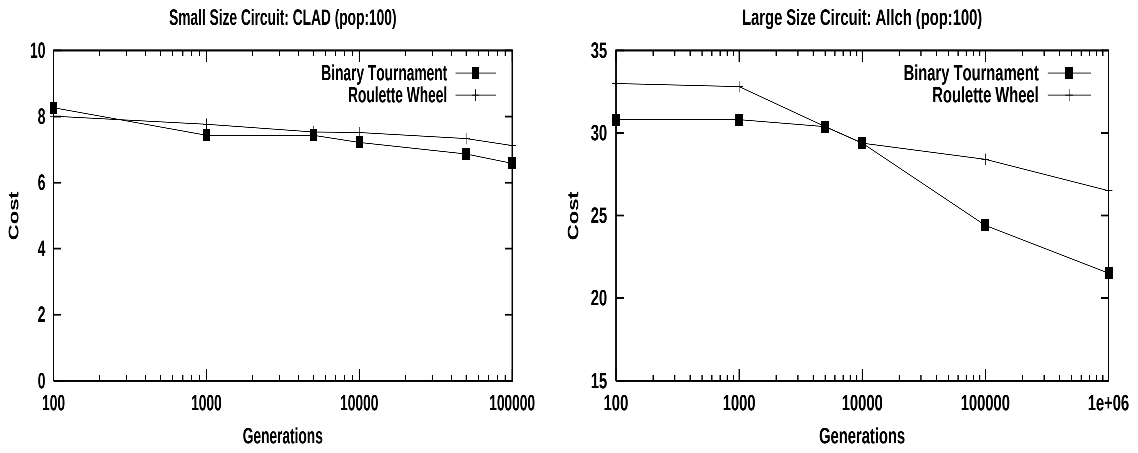


Figure 3.16: Different parent selection methods.

Crossover and Mutation Operator

The fitness based fusion operator [Beas96] and the uniform operator [Sysw89] are considered to perform the crossover.

The fusion operator takes into account both the structure and relative fitness of the parent solutions [Beas96]. Such an operator produces only a single child, whereas a conventional crossover operator usually produces two children. Let P_1 and P_2 be the parent strings. f_{p1} and f_{p2} represent the fitness of parent P_1 and P_2, respectively. C is the child string, and $P_1[i]$ denotes the i-th gene of parent P_1. The fusion operator works by following the procedure in Figure 3.17.

```
FUSION CROSSOVER
1. $i$ =1
2. if $P_1[i] = P_2[i]$, then $C[i]= P_1[i] = P_2[i]$.
3. if $P_1[i] \neq P_2[i]$, then
   (1) C[i] = $P_1 i[i]$ with probability $P = f_{P_2}/(f_{P_1} + f_{P_2})$
   (2) C[i] = $P_2 i[i]$ with probability $1 - P$
4. if $i = n$, stop; otherwise, set $i = i + 1$, and go to step 1.
```

Figure 3.17: Fusion crossover operator.

The idea of the uniform operator is to generate a random *crossover mask* which can be represented by the binary string in Figure 3.18. According to the crossover mask, the offspring is formed by picking the gene from parent P_1, if the corresponding binary bit of the crossover mask is 0; otherwise, from parent P_2.

Figure 3.19 presents the experimental results for the two benchmarks for the different crossover operators. Clearly, the fusion crossover operator produces better results than the uniform operator. Thus, all the results in this thesis are based on the fusion operator.

Mutation, applied to each offspring after the crossover, works by inverting M randomly chosen bits of a string, where M is experimentally determined.

Heuristic Feasibility Operator

Since the SPP is highly constrained, new solutions, generated by crossover and mutation are often infeasible (i.e., some rows may be under-covered and some rows

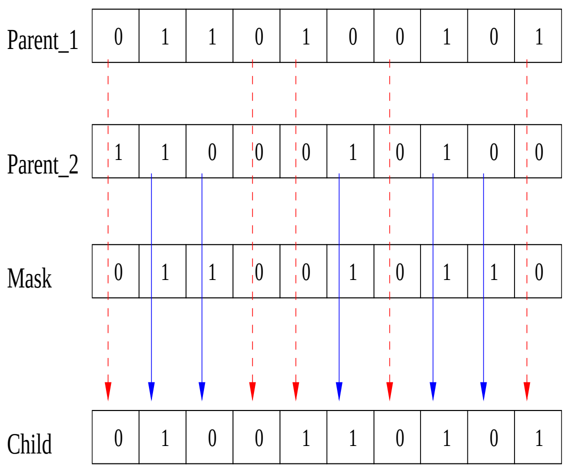


Figure 3.18: Uniform crossover operator.

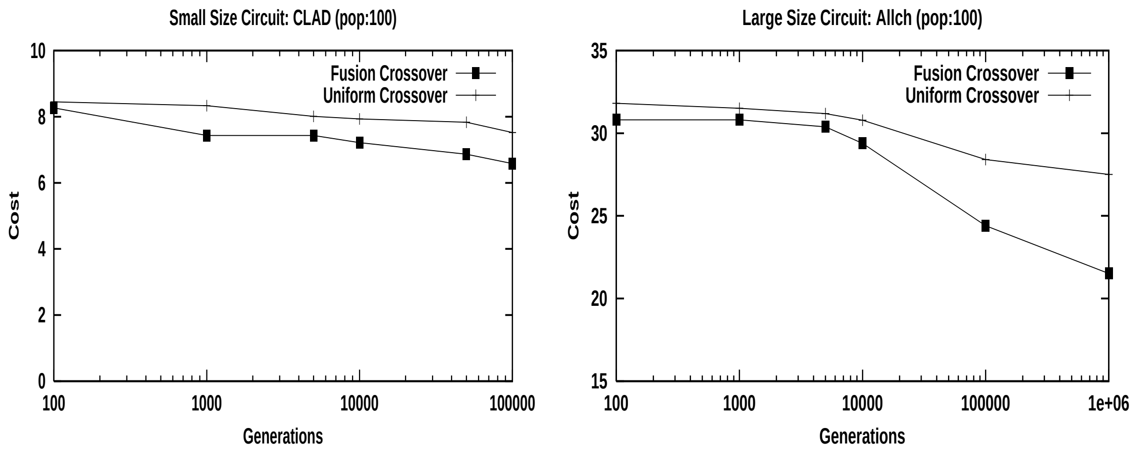


Figure 3.19: Different crossover operators.

may be over-covered). A heuristic feasibility operator, presented in [Chu95], is then used to repair infeasible solutions. The operator begins by identifying all the over-covered rows, and randomly removes columns, until all the rows are covered by, at most, one column. The following step is to identify all the under-covered rows. Some new columns are then selected such that as many under-covered rows as possible can be covered without causing any other rows to be over-covered. The solution that is produced by the heuristic operator does not have over-covered rows, but may have under-covered rows. In general, this operator does not guarantee a feasible solution. However, since special clusters (*columns*) that cover only one gate (*row*) are introduced by the clustering heuristic (Figure 3.8), the feasibility operator attempts to fully repair solutions of the MTCMOS SPP following the crossover and mutation operations. In addition, after the feasibility operation, if the solution is similar to any parent (i.e., duplication), the algorithm will discard the child and the evolution process of this generation will be repeated again.

Population Replacement Scheme

Once a child solution has been created via the GA operators, the child will replace a *less fit* member of the population. The average fitness and unfitness values of the population will be improved if the child solution has a lower fitness and unfitness value than those of parent solutions. During the replacement process, a duplicate solution needs to be prevented from entering the population. A duplicate child is one whose solution structure is identical to any one of the solution structures in the population. Allowing duplicate solutions to exist can reduce the diversity of the whole population and may lead to premature convergence.

A Subgroup Ordering Replacement (SOR) scheme is specifically designed for population replacement in [Chu95], where the population is divided into four mutually exclusive subgroups with respect to the child. Figure 3.20 illustrates the fitness-unfitness landscape of the population, where the "x" and "y" axes represent unfitness and fitness, respectively [Chu95]. Each point in the plot represents an individual in the population and is positioned according to the individual's fitness and unfitness values. These points are separated into four subgroups with respect to the fitness and unfitness of the child. A child will replace a selected individual of the first non-empty subgroup in the order of "a", "b", "c", and "d". Based on the situation in Figure 3.20, a child solution will first attempt to replace a solution in subgroup "a", because it has higher fitness and unfitness values than the child's[2]. If subgroup "a" is empty, then subgroup "b" is considered next. The reason for considering subgroup "b", before subgroup "c", is that the feasible solutions are harder to find, and therefore, the priority is to first search for a feasible solution before trying to improve the fitness of the solution. Accordingly, by considering subgroups "a" and "b" first, the average unfitness of the population will decrease.

3.4.1 SPP Results

GA Parameter Tuning

The generation size and population size are two important parameters in any GA to determine the solution quality and average execution time. Figure 3.21 and 3.22 portray the effect of different generation and population sizes, respectively,

[2]The higher the fitness and unfitness a solution has, the inferior the solution.

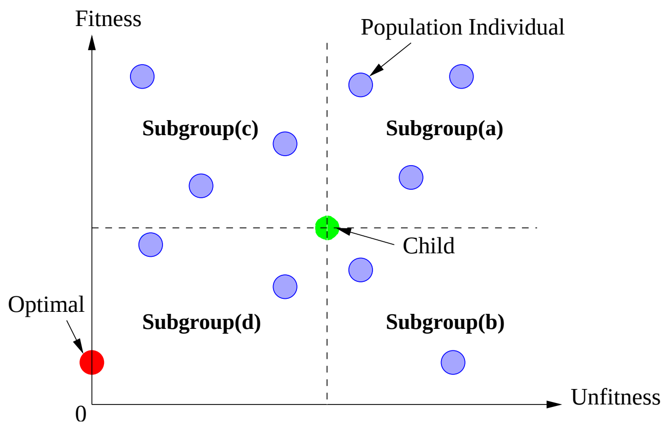


Figure 3.20: Population subgroups and fitness-unfitness landscape.

for a small (CLAD) and a large (AllCh) circuits when I_{sleep} equals 300 μA. Both figures prove that the solution quality (cost) is improved (decreased) with the larger population or generation size. However, the two figures also indicate that the computation time will increase as more generations and populations are evolved. As a result, there is a tradeoff between the solution quality and the CPU time. For a large circuit, this tradeoff becomes particularly severe.

Figure 3.23 reflects the effect of different mutation rates for two different size circuits. The experimental results indicate that the larger the mutation rate is, the worse the solution quality is. This denotes that duo to too much random perturbation, the offspring has lost its resemblance to the parents. Therefore, the algorithm loses the ability to converge to a good solution. However, small mutation rates also result in a larger execution time. The reason for this is that the offspring with small mutation rates is more likely to be a duplicate child. Under these circumstances, the evolution process of this iteration needs to be repeated again. Consequently, the mutation rate is set at ten for the rest of the tests.

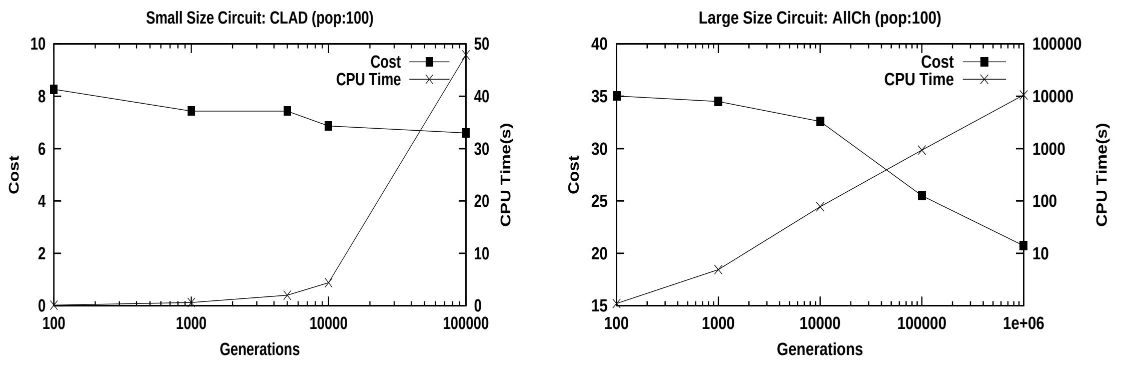


Figure 3.21: Results of different generation sizes ($I_{sleep} = 300\mu$A).

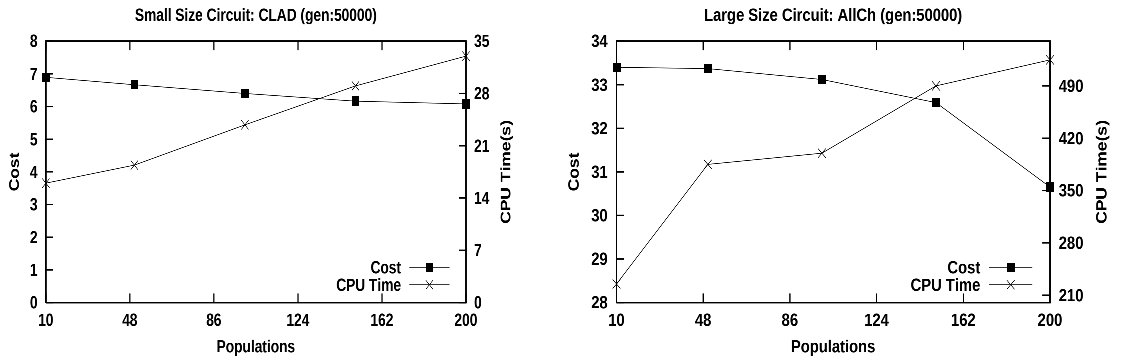


Figure 3.22: Results of different population sizes ($I_{sleep} = 300\mu$A).

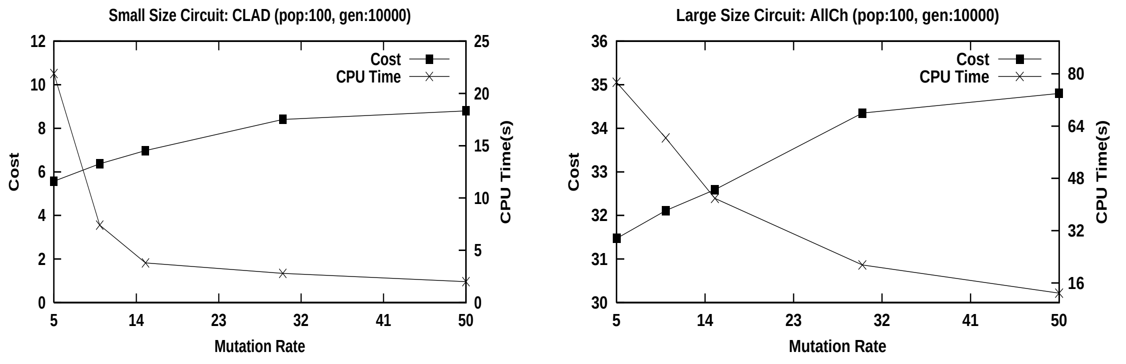


Figure 3.23: Results of different mutation rates ($I_{sleep} = 250\mu$A).

GA Results of Different Circuits

Tables 3.3 and 3.4 compare the performance of the CPLEX ILP solver and GA for the CLAD and the parity circuits, respectively. The generation size and population size parameters in the GA heuristic are set at 10,000 and 100, respectively, for each benchmark. The solution cost ("Cost" column), total sleep transistor number ("ST-Num" column), and solution time (in CPU seconds) are recorded for each sleep transistor size (I_{sleep}). The last row in each table evaluates the average performance improvement, based on the optimal solution obtained by the ILP CPLEX solver. For CPLEX, these two small SPPs are fairly easy to solve with the optimal solution being found after only a few branch-and-bound iterations. However, the proposed GA is not computationally competitive with the CPLEX solver.

CLAD	CPLEX			GA (Gen=10,000 Pop=100)		
I_{sleep}	Cost	ST-Num	Time	Cost	ST-Num	Time
150	3.68	9	0.33	4.52	10	2.7
200	4.33	8	2.16	5.59	9	3.0
250	4.68	6	0.57	6.37	8	3.2
300	5.46	6	3.82	7.05	7	4.1
350	5.78	5	3.93	7.42	7	4.5
400	6.44	5	12.27	8.32	7	4.8
Avg Imp	0%			-29.3%	-23.1%	-0.9%

Table 3.3: Comparison of CPLEX and GA for "CLAD benchmark".

The advantage of the GA arises from the controllable computation time for the Mult and ALU test circuits. For both circuits, the modelled SPPs become extremely more difficult to solve in one particular case than in the others. This can be seen by examining the CPU time, executed by the CPLEX solver, for the different sleep transistor sizes in Table 3.5 and 3.6, respectively. The worst solution time in both

Parity	CPLEX			GA (Gen=10,000 Pop=100)		
I_{sleep}	Cost	ST-Num	Time	Cost	ST-Num	Time
150	3.41	9	0.7	4.16	9	7.0
200	4.33	8	2.65	4.98	8	5.9
250	4.88	7	16.57	6.33	8	7.2
300	4.99	6	1.58	5.99	7	85.3
350	5.49	6	2.29	7.59	7	11.7
400	5.87	6	2.46	7.82	7	11.0
Avg Imp	0%			-27.3%	-9.5%	-388%

Table 3.4: Comparison of CPLEX and GA for "Parity benchmark".

circuits is three orders longer than the shortest one. However, the GA's more even and less computation time, which is directly proportional to the generation and population sizes, can be carefully adjusted to generate an acceptable solution in a reasonable amount of time.

Mult1	CPLEX			GA (Gen=10,000 Pop=100)		
I_{sleep}	Cost	ST-Num	Time	Cost	ST-Num	Time
150	5.72	18	349.7	5.75	18	2.0
200	3.91	12	1.58	4.32	12	2.4
250	3.62	9	0.11	4.13	10	3.0
300	4.96	9	0.32	6.38	10	3.3
350	5.07	8	0.47	6.38	9	4.5
400	5.36	7	1.19	5.93	8	4.6
Avg Imp	0%			-14.8%	-6.3%	+94.4%

Table 3.5: Comparison of CPLEX and GA for "Mult1 benchmark".

The 32-bit error correcting, the 27-bit channel interrupt controller, and the 16-bit multiplier are the three largest circuits for the MTCMOS SPP modelling in this thesis. These three circuits contain more rows (gates) and more columns (clusters) than the other benchmarks. The CPLEX solver, therefore, requires a large amount of computation time to find the optimal solution for each case, as

ALU	CPLEX			GA (Gen=100,000 Pop=100)		
I_{sleep}	Cost	ST-Num	Time	Cost	ST-Num	Time
150	4.76	17	9.41	6.98	19	246.1
200	5.61	14	47.13	6.74	15	299.7
250	6.63	12	271.47	8.60	15	320.8
300	7.51	11	1299.09	11.27	14	349.8
350	8.43	11	4478.45	11.26	14	391.8
400	9.22	11	8589.01	13.09	12	657.0
Avg Imp	0%			-37.4%	-17.1%	+84.6%

Table 3.6: Comparison of CPLEX and GA for “ALU benchmark”.

shown in Table 3.7, 3.8, and 3.9. The GA, implemented in this section to solve the SPP, is principally applied to improving the performance for these large circuits. The experimental results indicate that the GA solution is not very sensitive to the size of the population. Therefore, a population size $P = 100$ is used for all circuits with different sleep transistor sizes. Various generation sizes are then applied to attempt to find an acceptable solution in a reasonable amount of time. Tables 3.7, 3.8, and 3.9 present the solutions and average performance improvements achieved by the GA with two generation sizes; namely, 100,000 and 1,000,000. Clearly, more generations will improve the solution quality (less cost function), but the execution time also increases to the same order as that of the CPLEX solver. This tradeoff results in less effectiveness for this pure GA to solve the SPP.

Even the results, generated by the GA, might be unsatisfactory, for most NP-hard problems in CAD for VLSI design, heuristic methods seem to be the only way to solve the problems [Gere99]. Heuristics are acceptable if they perform well for most typical problem instances. It can be envisaged that this GA implementation can become more effective if a local search heuristic is combined as a hybrid. An alternative method to significantly improve the solution quality for this MTCMOS sleep transistor sizing problem is using a simpler and more effective model, as discussed in Section 3.6.

Error I_{sleep}	CPLEX			GA (G:100,000 P:100)			GA (G:1,000,000 P:100)		
	Cost	ST-N	Time	Cost	ST-N	Time	Cost	ST-N	Time
150	6.83	32	11125.2	10.02	36	365.8	9.89	36	4822.2
200	7.27	24	60451.3	14.99	32	488.1	13.61	31	9520.4
250	8.42	22	26731.2	16.19	27	585.6	15.38	27	13169.7
300	9.13	20	18875.6	18.46	25	553.9	14.87	23	12535.4
350	10.05	18	18034.6	17.24	23	618.2	15.64	21	15448.6
400	10.97	17	17557.8	22.22	26	849.9	18.15	22	9192.3
A-Imp	0%			-88.2%	-27.1%	+97.7%	-66.2%	-20.3%	+57.7%

Table 3.7: Comparison of CPLEX and GA for “Error benchmark”.

AllCh I_{sleep}	CPLEX			GA (G:100,000 P:100)			GA (G:1,000,000 P:100)		
	Cost	ST-N	Time	Cost	ST-N	Time	Cost	ST-N	Time
150	11.10	55	1994.5	20.12	61	463.0	14.40	58	5489.6
200	9.20	39	89390.2	23.76	52	555.6	16.72	46	7309.0
250	9.32	32	116229.4	24.41	44	625.8	21.52	42	8874.7
300	12.65	30	29552.7	25.51	38	936.5	20.73	34	10574.8
350	12.71	26	32080.9	27.92	39	967.7	24.17	36	13713.7
400	14.74	25	40308.9	26.57	37	1075.1	23.95	33	15075.5
A-Imp	0%			-112.7%	-30.9%	+98.5%	-74.3%	-20.2%	+80.3%

Table 3.8: Comparison of CPLEX and GA for “AllCh benchmark”.

3.5 A First-Fit (FF) Technique

In [Anis02, Anis03], a discharge current processing heuristic is first used to cluster the gates in the circuit. The sleep transistor sizing and distribution problem is then solved by an Integer Linear Programming (ILP) Bin-Packing formulation that was

Mult2	CPLEX			GA (G:100,000 P:100)			GA (G:1,000,000 P:100)		
$I^{\dagger}_{sleep}$	Cost	ST-N	Time	Cost	ST-N	Time	Cost	ST-N	Time
350	88.96	415	969.5	120.62	443	1345.2	108.15	443	28994.6
400	25.49	309	59767.8	67.47	349	1796.3	50.35	335	32299.1
A-Imp	0%			-110.9%	-9.1%	+94.8%	-59.5%	-7.5%	-2.4%

† when $I_{sleep} < 350\mu$A, no feasible solution is found.

Table 3.9: Comparison of CPLEX and GA for “Mult2 benchmark”.

explained in Sections 3.2.1 and 3.2.2. The discharge current processing heuristic is utilized to form a set of clusters of gates that, when combined, does not exceed the maximum current of any gate in the cluster. Following the processing heuristic, the problem is modelled as a BPP. The objective of the BPP formulation is to assign each cluster to one sleep transistor such that the number of sleep transistors used is minimized. Total current in each sleep transistor is constrained such that it does not exceed the discharge current limit I_{sleep}.

The original Bin-Packing problem consists of placing n objects into a number of bins. Each object has a weight and each bin has a limited bin capacity. In [Anis02, Anis03], the weight of each object is represented by the maximum discharge current of each cluster, and the bin capacity equals the limit current of the sleep transistor. One problem of using the maximum current of each cluster to represent the object weight is over-estimating the discharge current at the different time slots. As illustrated in Figure 3.24(c), the maximum current of the cluster, which is formed by adding the discharge currents of gate G1 and G2 in the timing diagram, is used to represent the weight of the cluster (object). It is assumed that the discharge current of this cluster is equal to $Area_1$ plus $Area_2$. However, the real discharge current combination of these two gates in the timing diagram consists of only $Area_2$.

As a result, more sleep transistors can be used due to the over-estimation.

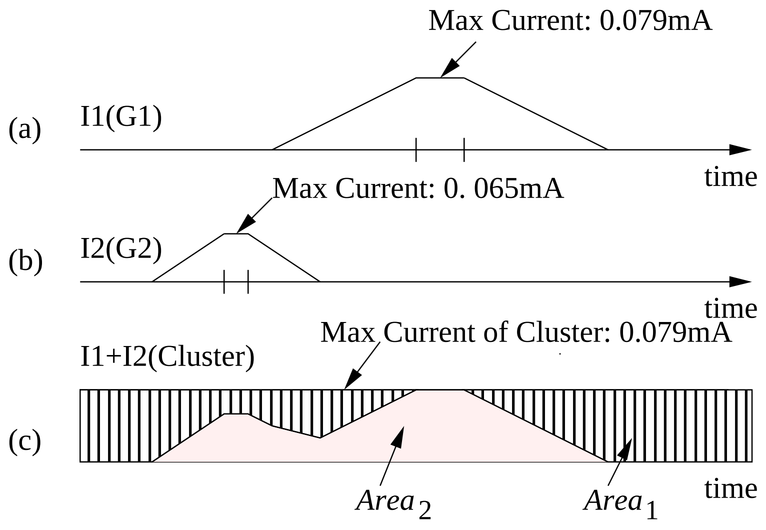


Figure 3.24: Formation of a single cluster.

Accordingly, a more efficient First-Fit (FF) heuristic technique is proposed to solve the sleep transistor sizing and the distribution problem directly. As seen in the pseudo-code of Figure 3.25, the limit on the current of each sleep transistor is directly taken as the criterion to assign a gate to the sleep transistor. The algorithm terminates when all the gates are assigned. Without transforming the dynamic discharge current of a cluster into a static maximum current, the current over-estimation is avoided, and therefore, the number of sleep transistors can be reduced. Furthermore, from the pseudo-code of Figure 3.25, it can be seen that the computational complexity of the FF technique is $O(n^2)$, which is similar to the current processing heuristic proposed in [Anis02]. Thus, the CPU time for solving the sleep transistor sizing and the distribution problem is improved dramatically, since the ILP BPP is avoided.

```
            FIRST-FIT HEURISTIC
1. Initialize current vectors;
2. Set all gates free to be assigned to a sleep transistor;
3. **For** all gates in circuit
        **If** gate $G_i$ is not assigned yet
            assign gate $G_i$ to new sleep transistor $S_k$
            update sleep transistor info
            calculate max current, start, and end time
        **End If**
        **For** all other gates in circuit
            **If** gate $G_j$ is not assigned yet
                add current of gate $G_j$ to sleep transistor $S_k$
                **If** combination $\leq$ current limit of ST
                    append gate to sleep transistor
                    update sleep transistor info
                    set gate $G_j$ locked in sleep transistor $S_K$
                **End If**
        **End For**
   **End For**
4. Return all sleep transistors used.
```

Figure 3.25: First-Fit (FF) heuristic for MTCMOS sleep transistor sizing.

3.5.1 Experimental Results of the First-Fit (FF) Approach

With the performance degradation within 5% as a basis, Table 3.10 displays a comparison of the techniques used in [Anis02] and the proposed FF technique in terms of the total sleep transistor (ST) area and subthreshold leakage power. The column "BPP" are CPLEX solver results that are generated by the current processing heuristic and the BPP technique, whereas the column "FF" gives the results that are generated by the FF technique. Obviously, the FF technique achieves less or equal leakage power than the techniques in [Anis02] due to the correct estimation of the discharge current. The FF technique reduces the leakage power by 12.1%, on average, for seven test circuits. However, the major advantage of the proposed FF heuristic is the simplified problem model. Consequently, the large computation time of the CPLEX solver to solve the BPP is avoided. The speed-up factor, achieved by the FF approach indicated in the column "S-U" of Figure 3.11, is significant. The decrease in the CPU time predicates that the proposed FF heuristic will be more effective than the techniques in [Anis02], as problem increases in size.

Circuit	Optimal I_{sleep}	Total number of STs (BPP)	Total width of STs (BPP)	Total number of STs (FF)	Total width of STs (FF)	Leakage savings by FF
CLAD	300μA	3	3.96μm	3	3.96μm	0%
Mult1	300μA	3	3.96μm	2	2.64μm	33%
Parity	300μA	2	2.64μm	2	2.64μm	0%
Alu	250μA	6	6.6μm	4	4.4μm	33%
Error	350μA	5	7.7μm	5	7.7μm	0%
AllCh	300μA	13	16.94μm	11	14.52μm	14.3%
Mult2	350μA	269	414.3μm	258	397.3μm	4.1%

Table 3.10: Leakage comparison between BPP and FF techniques.

Circuit	I_{sleep}=150			I_{sleep}=250			I_{sleep}=300			I_{sleep}=350			I_{sleep}=400		
	BPP	FF	S-U	BPP	FF	S-U	BPP	FF	S-U	BPP	FF	S-U	BPP	FF	S-U
CLAD	0.32	**0.01**	32	0.13	**0.01**	13	0.2	**0.01**	2	0.25	**0.01**	25	0.29	**0.01**	29
Mult1	0.11	**0.01**	11	0.04	**0.01**	4	0.03	**0.01**	3	0.06	**0.01**	6	0.04	**0.01**	4
Parity	0.19	**0.1**	2	0.14	**0.1**	1	0.18	**0.1**	2	0.18	**0.1**	2	0.13	**0.1**	1
Alu	4.2e+3	**0.1**	4e+4	0.35	**0.1**	3	0.26	**0.1**	2	0.28	**0.1**	3	0.23	**0.1**	2
Error	0.66	**0.1**	6	0.91	**0.1**	9	0.35	**0.1**	3	0.16	**0.1**	1	0.36	**0.1**	3
AllCh	14.1	**0.1**	140	1.1e+4	**0.1**	1e+5	3.21	**0.1**	32	4.55	**0.1**	45	3.63	**0.1**	36
Mult2	-	-	-	-	-	-	-	-	-	1.6e+4	**1.5**	1e+4	1.5e+4	**1.3**	1e+4
Avg S-U	7031			18338			7			1440			1439		

- I_{sleep} violation, no feasible solution is found.

Table 3.11: CPU time(s) comparison between BPP and FF techniques.

3.6 A Set-Covering Problem (SCP) Model

To take the physical locations of the gates on the chip into consideration, and thereby reduce the routing complexity of larger circuits, a Set-Partitioning technique is proposed in [Anis02]. The objective of the low-power SPP is to find an optimal collection of clusters such that each gate is covered by exactly one cluster (i.e., one sleep transistor in the MTCMOS technique), while the lowest cost value is achieved.

The concept and formulation of the Set-Partitioning technique was presented in Section 3.2.3. The equality constraint in the SPP mathematical formulation, (3.8), guarantees that all the gates in the circuit are covered once by a single sleep transistor. Due to these limitations, the SPP is considered to be a highly constrained problem. As the problem size (i.e., the number of gates in the circuit) increases, it becomes quite inefficient to solve the model by an ILP solver (CPLEX) in a reasonable amount of time. The experimental results in [Anis03] signify that the CPU time, used by CPLEX to solve the SPP, increases dramatically as the circuit size increases.

Accordingly, a SCP model is considered in this thesis to further reduce the CPU computation time. The same clustering heuristic, Figure 3.8, is used within the SCP formulation. The major difference between the SPP and the SCP is in the restriction of the constraints, imposed by the model. By relaxing the sensitivity of the constraints of the SPP model, the SPP is transformed to a SCP, as shown in (3.14) and (3.15),

$$\text{Minimize Z } = \sum_{j=1}^{n} c_j S_j, \tag{3.14}$$

subject to

$$\begin{aligned} &\sum_{j=1}^{n} a_{ij} S_j \geq 1 \\ &i = 1, ..., m \\ &S_j \in 0, 1 \quad j = 1, ..., n, \end{aligned} \tag{3.15}$$

where

$$S_j = \begin{cases} 1, & \text{if the } j\text{th column is selected} \\ 0, & \text{otherwise} \end{cases} \qquad a_{ij} = \begin{cases} 1, & \text{if row } i \text{ is covered by column } j \\ 0, & \text{otherwise.} \end{cases}$$

Constraint (3.15) in the SCP model guarantees that all gates in the circuit will be covered by the sleep transistor at least once, which indicates that some of the gates can be covered by two or even more sleep transistors. In [Anis02, Anis03], the current capacity of each sleep transistor is assumed to be of fixed value. As a result, the relaxed constraints of the SCP technique can increase the number of sleep transistors. However, when one gate is connected to more than one sleep transistor, the virtual ground wires of the different sleep transistors become common which balances the discharge currents. As the authors reported in [Long03], the total area of all sleep transistors can be reduced with the presence of such a current discharging balance. Therefore, it is useful to have a gate that is assigned to more than one sleep transistor. Furthermore, the relaxed constraints of the SCP can result in a better optimization solution because of the larger solution space. For the computation time, the CPLEX solver spends much less time to search for the feasible solutions of the SCP, compared to the SPP, due to the relaxation of constraints.

3.6.1 Comparison of SPP and SCP

Tables 3.12, 3.13, and 3.14 compare the results, produced by the SPP and the SCP techniques, in terms of the solution cost, total number of sleep transistors, and the CPU time, respectively. All the results are derived from the same gate-clustering technique that is introduced in [Anis02], and the ILP CPLEX solver is applied to solve the integer linear programming models.

Table 3.12 confirms that the SCP technique generates a better solution (lower cost) than the SPP for each test circuit. A reduction in the cost function indicates a more efficient solution in terms of the distance between the clustered gates and the efficiency of the clustering mechanism. For all the test circuits, an 8.9% cost reduction, on average, is gained by the SCP modelling due to the larger solution space. Column "Imp" in Table 3.13 indicates the reduction of the number of sleep transistors achieved by the SCP technique compared to the SPP technique. Although the SCP technique improves the cost for each test circuit, the number of sleep transistors is not necessarily reduced. The reasoning behind this is that the objective function of the SPP and SCP problems is the cost, and not directly the number of sleep transistors; that is, the leakage power of the circuit. However, Table 3.13 does signify that the SCP can achieve a comparable solution in terms of the number of sleep transistors compared to the SPP.

Circuit	Optimal I_{sleep} (μA)	Cost of the SPP	Cost of the SCP	Cost reduction by the SCP
CLAD	150	3.68	3.50	4.9%
Mult1	250	3.62	3.48	3.9%
Parity	150	3.41	2.76	19.1%
Alu	150	4.76	4.48	5.9%
Error	150	6.83	6.27	8.2%
AllCh	250	9.32	8.39	9.9%
Mult2	400	25.5	22.7	10.9%

Table 3.12: Cost comparison between SPP and SCP techniques.

Circuit	I_{sleep}=150			I_{sleep}=250			I_{sleep}=300			I_{sleep}=350			I_{sleep}=400		
	SPP	SCP	Imp	SPP	SCP	Imp	SPP	SCP	Imp	SPP	SCP	Imp	SPP	SCP	Imp
CLAD	**9**	10	-11%	6	6	0%	6	6	0%	5	5	0%	5	5	0%
Mult1	18	18	0%	**9**	10	-11%	9	9	0%	8	8	0%	7	7	0%
Parity	9	9	%0	7	**6**	+14%	6	6	0%	6	**5**	+17%	6	**5**	+17%
Alu	**17**	18	-6%	12	12	0%	11	11	0%	11	**9**	+18%	11	**9**	+18%
Error	**32**	33	-3%	22	22	0%	20	**18**	+10%	18	**16**	+11%	17	**16**	+6%
AllCh	**55**	57	-4%	**32**	36	-13%	30	**28**	+7%	**26**	27	-4%	25	**24**	+4%
Mult2	-	-	-	-	-	-	-	-	-	**415**	444	-7%	**309**	327	-6%
Avg Imp	-3.4%			-1.4%			+2.4%			+5.0%			+5.6%		

- I_{sleep} violation, no feasible solution is found.

Table 3.13: Comparison of sleep transistor number for SPP and SCP.

The main advantage of the SCP model arises from the large reduction in the CPU time, compared to that of the SPP model due to relaxation of constraints. Table 3.14 lists the solution times of the CPLEX solver to solve the SPP and the

SCP models. Column "S-U" indicates the speed-up factor, achieved by the SCP, compared to that of the SPP. From the "Avg Imp" row in Table 3.13 and the "Avg S-U" row in Table 3.14, it is clear that the SCP technique can produce comparable solutions to those of the SPP with much less computation time. Figure 3.26 plots the effect of the size of the sleep transistor (I_{sleep}), with respect to the computation time, for two different benchmarks. Obviously, the speed-up achieved by the SCP technique is more evident for the larger circuit (i.e., the "AllCh" circuit), which indicates that this method can scale well for the larger benchmarks.

Circuit	I_{sleep}=150			I_{sleep}=250			I_{sleep}=300			I_{sleep}=350			I_{sleep}=400		
	SPP	SCP	S-U	SPP	SCP	S-U	SPP	SCP	S-U	SPP	SCP	S-U	SPP	SCP	S-U
CLAD	0.33	**0.19**	2	0.57	**0.40**	1	3.82	**0.77**	5	3.93	**0.44**	9	12.27	**0.65**	19
Mult1	3.5e+2	**0.09**	3889	0.11	**0.07**	2	0.32	**0.18**	2	0.47	**0.19**	2	1.19	**0.13**	10
Parity	0.7	**0.06**	12	16.57	**0.48**	35	1.89	**1.58**	1	2.29	**2.26**	1	2.46	**2.05**	1
Alu	9.41	**1.91**	5	2.7e+2	**1.78**	152	1.2e+3	**8.99**	14	4.4e+3	**5.10**	863	8.5e+3	**9.96**	853
Error	1.1e+4	**39.2**	282	2.6e+4	**3033.1**	9	1.9e+4	**47.2**	404	1.8e+4	**124.7**	150	1.8e+4	**4606.3**	4
AllCh	2.0e+3	**2.45**	801	1.2e+5	**457.8**	261	3.0e+4	**7722.8**	4	3.2e+4	**876.1**	36	4.0e+4	**4317.4**	9
Mult2	-	-	-	-	-	-	-	-	-	9.6e+2	**28.7**	34	5.9e+4	**1609.7**	37
Avg S-U	832			72			72			156			133		

- I_{sleep} violation, no feasible solution is found.

Table 3.14: Comparison of CPU time(s) for SPP and SCP.

3.7 Summary

The MTCMOS low-power digital circuit design technique was introduced in this chapter. The tradeoff between the system performance and the standby leakage that is associated with this method was then analyzed. Proper sleep transistor sizing is a key issue that affects the speed, as well as the leakage power of the circuit.

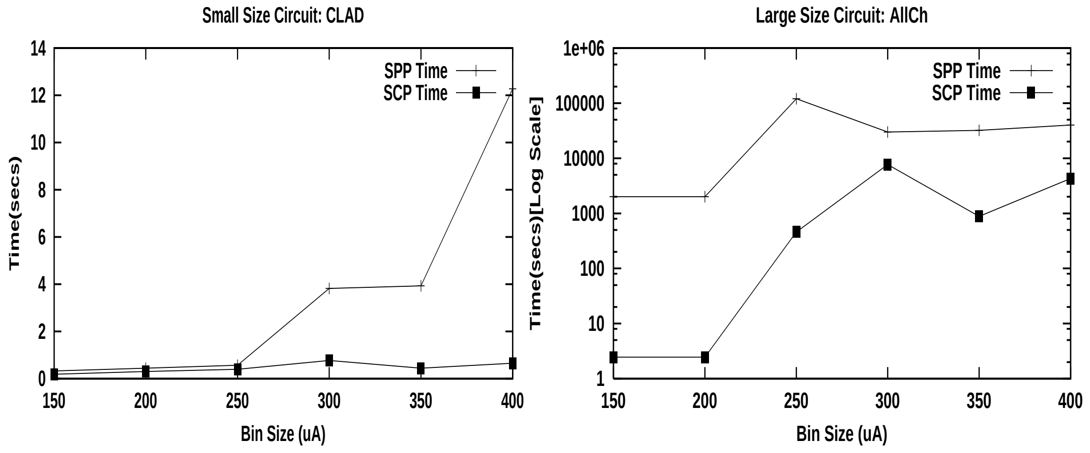


Figure 3.26: Computation time for SCP and SPP.

The accuracy of sizing the sleep transistor depends on how well the discharge currents are modelled. Based on the timing diagram, the discharge current is modelled as a trapezoid vector in the innovative gate-clustering technique, proposed in [Anis02]. The BPP and SPP models were then employed by the gate-clustering technique to effectively solve the sleep transistor sizing and distribution problem. To reduce the large computation time required by the CPLEX solver (to find the optimal solutions for the BPP and the SPP), two GAs were implemented and compared. The experimental results show that the BPP can be solved quite well by the GA. For the hard SPP, however, some work is required to improve the effectiveness of the GA in the future.

In addition, two models were proposed to effectively handle the sleep transistor sizing and distribution problem. By eliminating the over-estimated discharge current caused by the gate-clustering technique, the proposed FF technique achieved a leakage reduction of 12.1%, on average, compared to that of [Anis03]. Furthermore, the simplified problem model results in much less computational complexity. In order to reduce the routing complexity for the large circuits, caused by sleep transistor inserting, the original SPP in [Anis02] is substituted by a SCP model. Experimental results indicate that the SCP modelling Produces better results than the SPP modelling in a fraction of the time.

All test circuits in [Anis03] were designed manually by a schematic view. This design method not only consumes time, but also generates errors easily. For large circuit designs, CAD (Computer Aided Design) tools, integrated in the design flow, are required to implement iterative design process. The following chapter focuses on automating the design flow for MTCMOS.

Chapter 4

MTCMOS Design Environment

4.1 Introduction

Typically, Application Specific Integrated Circuit (ASIC) design is based on a flow that uses Hardware Description Languages (HDLs). In this flow, the design and implementation of the logic circuit are coded in either Verilog [Lee03c] or VHDL [Rush98]. Simulations are performed to check the logic circuit's functionality. This is followed by synthesis, where the HDL code is converted to logic gates. Following the synthesis, the physical design of the ASIC is realized. In this step, the synthesized gates are placed and routed. It is also the step to synthesis the clock tree and route the clock tree correctly with an acceptable clock skew. Figure 4.1 is a diagram of an ASIC design flow, beginning with specification of an ASIC design to a logic synthesis, and finally, the tapeout.

Optimal low-power designs involve decisions about the timing versus power and area versus power tradeoffs at different stages of the design flow. To enable designers to accurately and efficiently perform these tradeoffs, it is necessary for low-power optimization techniques to be integrated within, and applied throughout, the entire RTL-to-GDSII flow.

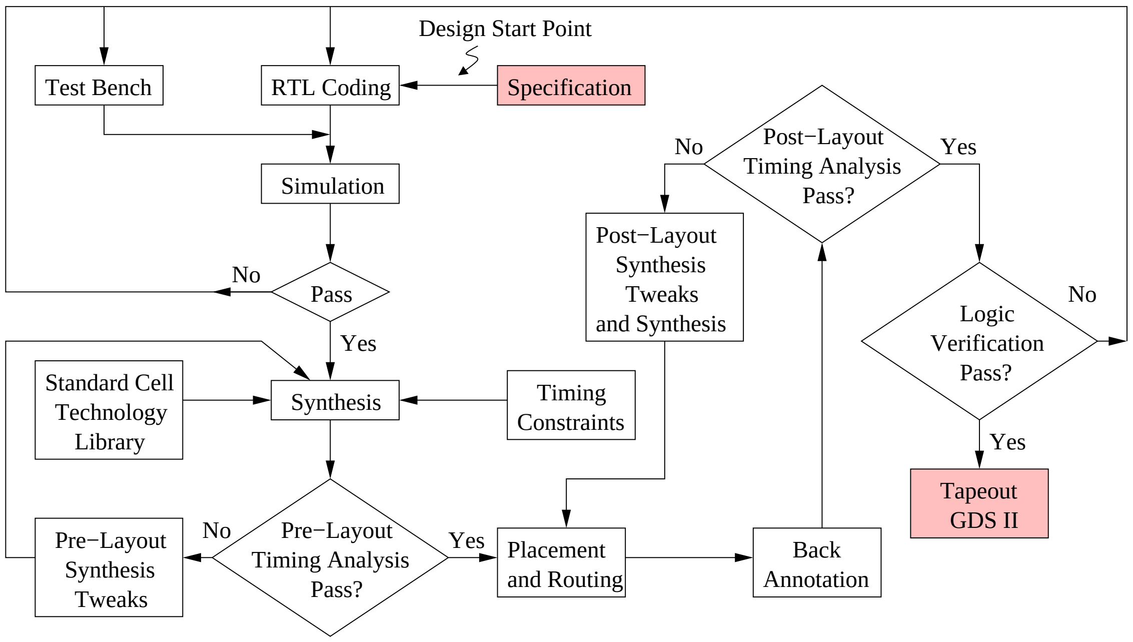


Figure 4.1: Diagram showing ASIC design flow.

This chapter first introduces the Canadian Microelectronics Corporation (CMC) digital ASIC design flow in Section 4.2. A new database is then constructed in Section 4.3 to effectively model the discharge current, based on the MTCMOS gate-clustering technique. Finally, the MTCMOS low-power design environment is developed and integrated with the CMC design flow in Section 4.4.

4.2 CMC Digital ASIC Design Flow

The 0.18 μm CMOS technology from Taiwan Semiconductor Manufacturing Company (TSMC), supported by CMC, is adopted in this thesis to design all the circuits. The standard cell library, designed by Virtual Silicon Technology Inc. using the 0.18 μm TSMC process, is adopted to implement the logic functions. The circuit design begins with a Register Transfer Level (RTL) model and ends with the physical verification which includes the stream file creation. Computer Aided Design (CAD) tools, working seamlessly together for the logic synthesis and physical design, play an important role in this iterative design process.

In order to integrate the MTCMOS automatic design environment with the traditional design flow, the CMC digital IC design process is introduced first. Figure 4.2 exhibits the flow of this design. A 2-bit pure combinational integer multiplier circuit is employed as an example to illustrate the design procedures.

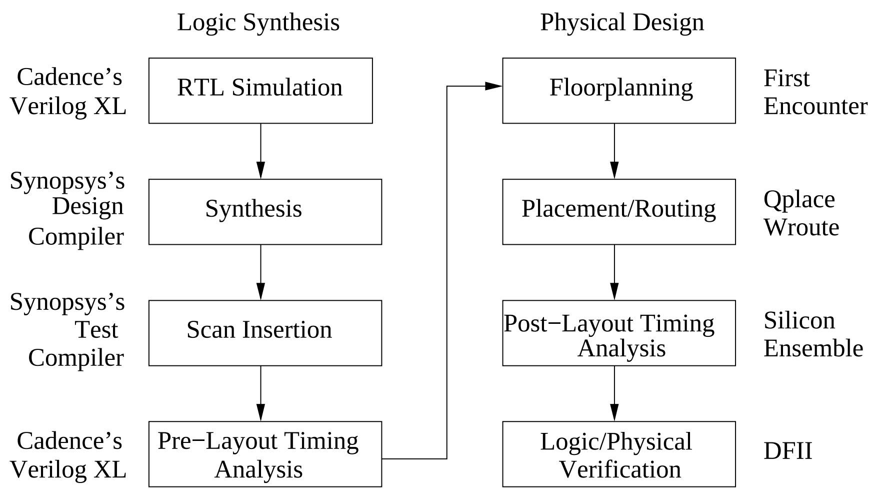


Figure 4.2: CMC digital ASIC design flow [Corp02].

4.2.1 Front-End Portion of IC Design

The purpose of this design portion is to synthesis the design from the RTL code to the logic level and create scan-based test and test vectors. The timing constraints for the placement and routing tools are also generated at this stage. The tool that is integrated with the CMC design flow to perform these tasks is Synopsys's *Design Compiler (DC)*.

RTL Simulation

Utilizing either Verilog or VHDL hardware description languages, RTL coding is the preferable starting point for most designs today. Appendix B provides the RTL netlist of the 2-bit multiplier circuit in Verilog. In order to ensure that the RTL code functions properly, *Cadence's Verilog XL* simulator is used to simulate the RTL circuit description, based on the test bench file. In this test bench file, test vectors with circuit input values and expected output results are utilized to verify the circuit functionality.

Logic Synthesis

Synopsys's *Design Compiler* is a tool combined with the CMC design flow to perform logic synthesis. In this step, the RTL code is converted to a netlist of logic gates. The synthesis process requires two other input files to conduct the conversion from the RTL to logic gates. The first input file is the *technology library* file that contains the standard cells. The library that is selected for this multiplier design is from the technology foundry TSMC. The second input file *constraints file*, helps to optimize the logic being synthesized. Usually, this file consists of information such as timing and loading requirements and optimization algorithms that the tool needs to optimally synthesize the logic gate.

Logic synthesis is a very important step in the ASIC design flow [Lee03c], since it ensures that the synthesis is tweaked to meet the specified timing performance and area constraints.

Scan Chain Insertion

The scan-based Design For Testability (DFT) is a structure approach to designing sequential circuits for testability. This technique allows the chip to be easily tested, following manufacturing, to ensure that no manufacturing errors nor subsequent problems (bonding problems, for example) exist within the chip. Usually, Synopsys's *Test Compiler* is used to implement the scan-based design technique. The goal of this technique is to connect all sequential elements (flip-flops) to form a long serial shift register (or scan chain) with multiplexers. The *Test Compiler* is then used to create a set of test vectors which can detect "stuck at 1" and "stuck at 0" [Kang03] faults in the chip. Since the multiplier circuit design is for illustration purposes the scan-based DFT is ignored in the 2-bit multiplier circuit.

Pre-Layout Timing Analysis

When the logic synthesis and scan chain insertion are completed, a structural netlist (either in Verilog or VHDL format) is generated by the *Design Compiler*. A static timing analysis, derived from this netlist, is then performed to detect any possible timing violation. The circuit netlist, including the scan chain, is again simulated to guarantee the functionality of the synthesized logic gates. This gate-level netlist, based on the technology library, also connects the logic synthesis and the physical design during the iterative design process. Appendix C shows the gate-level netlist of the 2-bit multiplier circuit following logic synthesis and simulation. The MTCMOS automatic design environment that will be introduced in Section 4.4 works on this netlist.

4.2.2 Physical IC Design

Once the pre-layout timing analysis of the gate-level netlist is completed, the netlist, together with the timing information from Synopsys's *Design Compiler*, is imported to the Cadence physical design environment. In this physical portion of the design, the synthesized logic gates are placed and routed. Following the post-layout timing analysis, the physical version of the design is verified by the Layout Versus Schematic (LVS) and Design Rule Check (DRC) tools. Finally, the design is converted into a GDSII (stream) format and is ready for fabrication.

Floorplanning

The objective of this design stage is to create a floorplan for the design, including a default group of cells, an I/O ring that is connected by an abutment, and defined placement sites for all the cells. Cadence's *First Encounter* is the design tool utilized for floorplanning stage.

Any macro-cells (such as RAM or hard-core), combined with the design, are included in this step. The positions of all the I/O cells, combined with the power pads, corner cells, and I/O feeder cells, are also defined. The power planning function, called from the *First Encounter*, can be used to create the power rings and power stripes for the design.

Placement and Routing

The task of the placement step is to construct a layout that indicates the positions of the cells. The timing driven placement tool, *Qplace*, is adopted for this job. During the timing driven placement, the placer balances the timing constraints with routability. Following the placement phase, global and detailed routing are performed by using *Wroute*. The *Wroute* tool can route the high-priority nets (e.g., clock, VDD, and VSS) before attempting optimization of other nets to allow for the most optimal routing. A particularly important task during this design stage is the clock tree synthesis. *CTPKS*, the clock tree synthesis engine in the Cadence *PKS*, is involved to create a balanced clock tree with an acceptable clock skew.

Post-Layout Timing Analysis

The post-layout timing analysis allows real timing violations such as hold and setup to be identified. This step is similar to the pre-layout timing analysis except that in the post-layout timing analysis, the accurate net delay information of the physical layout is used. In the pre-layout timing analysis, the net delay information is estimated. The post-layout synthesis is tweaked to fix timing violations during the resynthesis process, as shown previously in Figure 4.1. Following resynthesis, the floorplanning, placement and routing are performed iteratively, until all the timing violations are fixed.

Logic/Physical Verification and Tape Out

Once the post-layout timing analysis is completed, the design flow is directed towards the final logic verification to ensure correct functionality. *DivaLVS* in the Cadence *Design Framework II (DF II)* is employed to compare the schematic view (based on the gate-level netlist) with the layout view (based on the DEF file after *Wroute*). The DRC is also implemented to verify the geometric constraints. Once the LVS and the DRC are verified, the design is converted into the GDSII (steam) format and is ready for manufacturing.

4.3 Discharge Current Database Construction

One of the key issues concerning the gate-clustering MTCMOS technique is modelling the discharge current of each gate in the circuit as a vector. This offers an automated design environment so that the gate-clustering MTCMOS design environment can be easily integrated within the CMC digital design flow. As previously explained in Section 3.2.1, the construction of the vector for each gate is based on a trapezoid discharge current waveform. This waveform contains information about the delay of the gate (when the earliest and latest peak discharge currents occur), fanout (the duration of the current), and the magnitude of the current in each time slot.

In order to construct the vector for each gate in the circuit automatically, each standard cell in the technology library is simulated by Hspice [Comp00], and the information of the trapezoid waveform is documented. Thus, a database is formed which works as a look-up table to generate the vectors automatically (this will be further described in Section 4.4.1). Figure 4.3 illustrates a two-input NAND gate test circuit for the Hspice simulation.

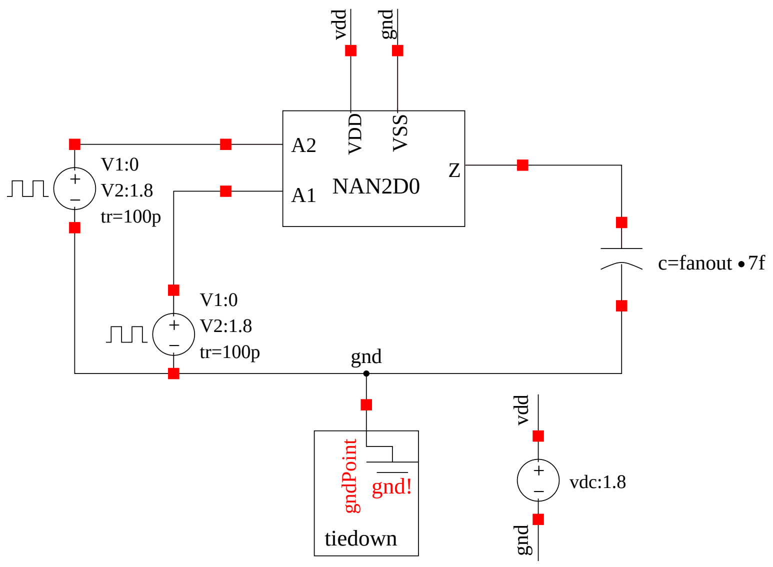


Figure 4.3: Standard cell (NAN2D0) test circuit.

Some environment variables of this test circuit are: $V_{DD} = 1.8$V, and temperature = 25 °C. The operational frequency is 500MHz, and the rising/falling time of the input signals are set to 100ps (5% of the signal period). For each fanout, a load of 6fF is applied to the output of the gate, since this represents the average input capacitance of cells in the standard-cell library [Inc99]. Furthermore, a load of 1fF is applied to the output of each fanout, contributing to the interconnection capacitance. Therefore, the total load capacitance of each fanout is 7fF. The threshold voltages V_T of the transistors in the standard cell are set to ±350mV by adjusting the values of parameters “dvthn” and “dvthp” in the model file [Comp00].

For each standard cell, all the possible input transitions, causing the discharge current to occur at the output node, are applied, as discussed in Section 3.2.1.

Figures 4.4, 4.5 and 4.6 present the discharge currents of the two-input NAND gate of the test circuit (Figure 4.3). These figures indicate that the discharge current varies with different input transitions. Consequently, the highest discharge current value (peak value I_{max}), the earliest (t_{min})/latest (t_{max}) delay time, and the longest duration time, among these discharge currents, are monitored. In order to correctly estimate the average discharge current, the peak value is multiplied by the switching activity of the gate, as explained in [Anis02]. The parameters that are collected by the database are used to represent the characteristics of the standard cell. For each standard cell, all parameters are documented at different fanout values. Figure 4.7 reveals the discharge current waveforms of the NAND gate at four different fanouts. Clearly, the larger the fanout, the longer the switching duration. Figure 4.8 gives an example of the structure of the discharge current database. Following the database construction, which is fundamental to the developed MTCMOS design environment (to be introduced in the next section), the MTCMOS design flow will be described in more detail.

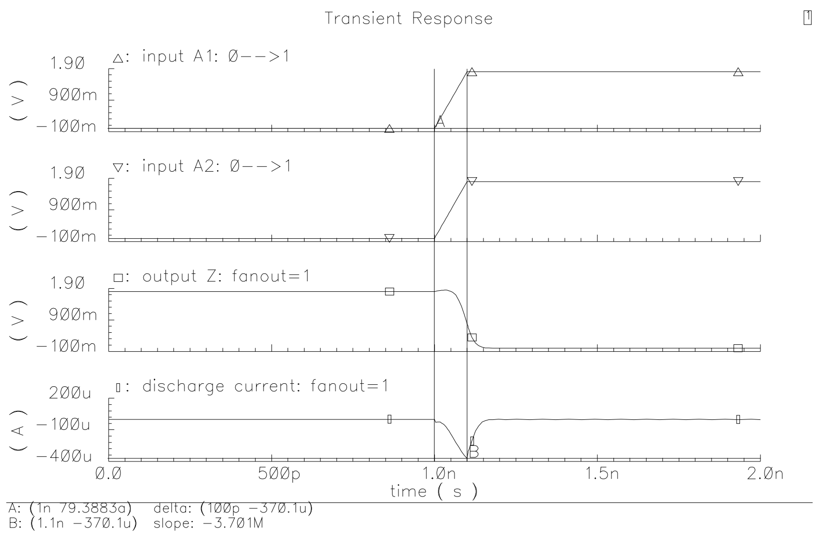


Figure 4.4: Standard cell NAN2D0: 00 to 11 discharge current.

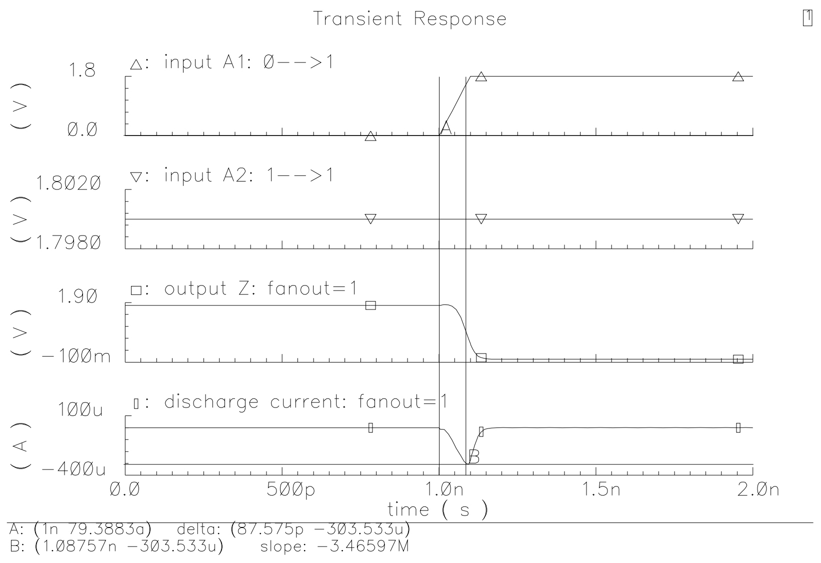


Figure 4.5: Standard cell NAN2D0: 01 to 11 discharge current.

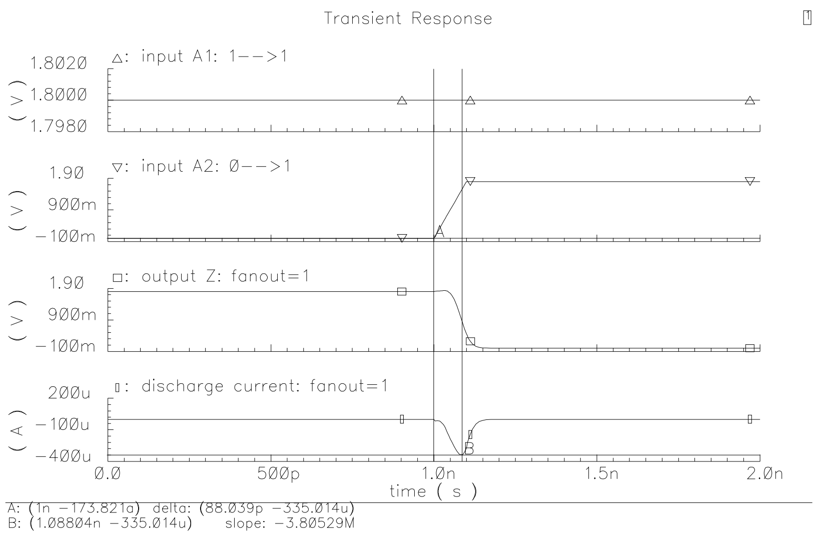


Figure 4.6: Standard cell NAN2D0: 10 to 11 discharge current.

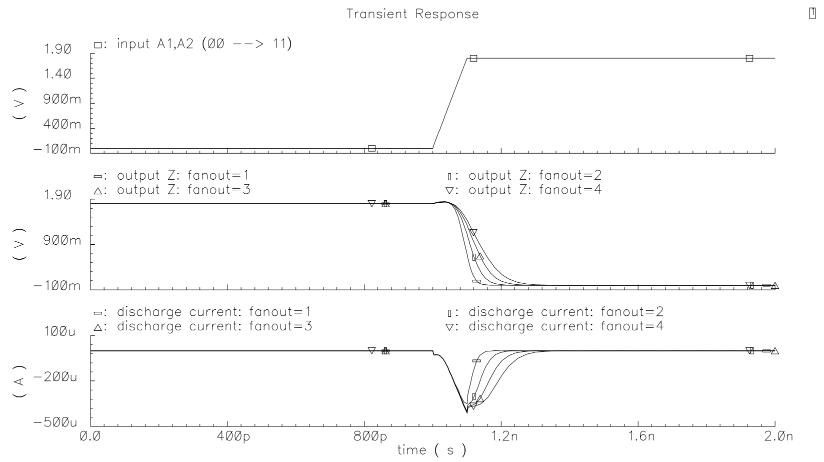


Figure 4.7: Effect of different fanouts on discharge current.

```
* ...
* The data format is as follows ;
* cell name    fanout   $t_{min}$   $t_{max}$   duration    $I_{max}$ ($\mu$A) ;

STANDARD_CELL ;
    .
    .
    .
  NAN2D0       1        9       10        8          69.15 ;
  NAN2D0       2       10       10       10          76.26 ;
  NAN2D0       3       10       11       10          77.71 ;
  NAN2D0       4       11       14       11          78.47 ;
  NAN2D0       5       12       15       12          79.04 ;
  NAN2D0       6       14       16       13          79.23 ;
    .
    .
    .
END STANDARD_CELL
```

Figure 4.8: Standard cell NAN2D0 discharge current database.

4.4 MTCMOS Design Environment

The integration of the CMC digital ASIC design flow with the developed MTCMOS design environment is illustrated in Figure.4.9.

Following the synthesis of the RTL code (using Synopsys's *Design Compiler*), the gate-level netlist is imported to the MTCMOS design environment. Next, the circuit topology is extracted and the delay parameters (t_{min} and t_{max}) of each gate, read from the database, are updated to account for the accumulative delay. The vectors are generated for each gate in the circuit from the updated delay information and discharge current database built apriori. In addition, sleep transistors with different W/L ratios (i.e., different I_{sleep}) are developed in the layout view by using (3.4). The sizing of the sleep transistor is then estimated by using techniques

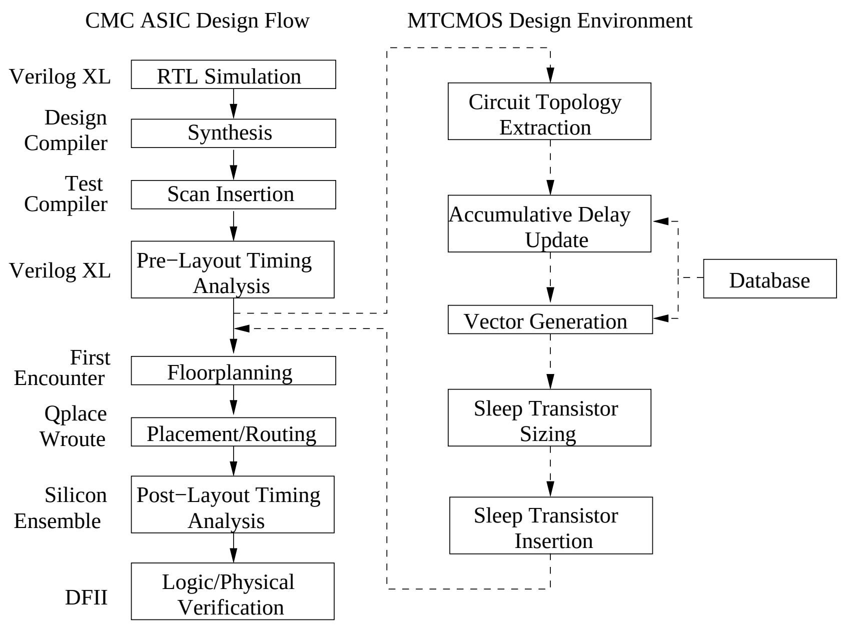


Figure 4.9: MTCMOS design flow.

presented in Chapter 3. Based on the optimization results, sleep transistors with optimal sizes, combined with the sleep control signal, are inserted into the netlist. Finally, the new gate-level netlist is exported to the Cadence physical design environment. The rest of the design follows the conventional design flow.

Several key issues that are related to the MTCMOS design environment are explained in more detail in the following subsections; namely, the vector generation, optimization environment, sleep transistor insertion, and layout implementation.

4.4.1 Automatic Vector Generation from RTL

In order to achieve a proper estimation of the sleep transistor size, the timing analysis of the discharge current needs to incorporate a number of important issues: (1) the different input transitions, (2) the circuit topology, (3) the fanout associated with each gate, and (4) the glitching currents. Several parameters were introduced in Section 3.2.1 to characterize the trapezoid discharge current of the gate. Among the parameters, the earliest delay time (t_{min}) and the latest delay time (t_{max}) of each standard cell (due to the different input transitions) are documented in the database construction phase, as discussed in Section 4.3. However, the values for these two parameters are recorded, based on the test circuit in Figure 4.3. Consequently, the true propagation delay of a gate in the circuit has to be updated according to the real circuit topology and delay level. In general, the propagation delay T_k of gate G_k is expressed as follows:

$$T_{k_{min}} = min\{(T_{i_{min}} + t_{k_{min}}), (T_{j_{min}} + t_{k_{min}}), ...\} \tag{4.1}$$

and

$$T_{k_{max}} = max\{(T_{i_{max}} + t_{k_{max}}), (T_{j_{max}} + t_{k_{max}}), ...\}, \tag{4.2}$$

given that the output of gates G_i, G_j, ... are inputs to gate G_k.

Step 2 in Figure 4.10 presents a simple heuristic that is developed to update the propagation delay for each gate in the circuit. Following the delay parameter updating, an automatic vector generation engine is developed to build a vector for each gate, according to the information of the updated delay and the database. The vectorially modelled circuit can then be optimized by using the heuristic techniques introduced earlier in Chapter 3.

```
MTCMOS DESIGN ENVIRONMENT
1. Read gate-level netlist;
2. While (updating the accumulative delay) not done
     For all gates in circuit
       If $T_k$ of $G_k$ is not updated
         If all the fan-in $(G_i, G_j ...)$ are updated
           update $T_k$
         End If
       End If
     End For
   End While
3. Build a vector for each gate;
4. Solve the ST sizing problem using FF or SCP technique;
5. Insert STs and sleep control signal into netlist;
6. Export the new gate-level netlist.
```

Figure 4.10: MTCMOS automatic design environment.

4.4.2 Environment Optimization

The objective of step 4 in the MTCMOS design environment as seen in Figure 4.10, is to solve the sleep transistor (ST) sizing problem by using the First-Fit or SCP technique, proposed in Sections 3.5 and 3.6, respectively. To evaluate the physical locations of the gates that are utilized by the SCP technique, an *Amoeba* placement tool, combined with the Cadence *First Encounter*, is applied. This trial placement is implemented after the logic synthesis, based on the gate-level netlist. The X, Y coordinates of each gate are then extracted from the Design Exchange Format (DEF) file. The SCP can thus be formulated in terms of the extracted coordinates and the constructed vectors. The next step involves calling the CPLEX library to solve the low-power MTCMOS optimization problem. To facilitate an automatic design and avoid data translation, a CPLEX solver interface engine is developed and integrated into the design environment. The MTCMOS design environment, illustrated in Figure 4.10, is implemented in the 'C' programming language on a Sun Solaris workstation.

4.4.3 Sleep Transistor Insertion and Circuit Layout

The problem solution, generated by the CPLEX solver, gives the optimal combination of gates and corresponding sleep transistors. According to the CPLEX results, the optimally sized sleep transistors are then inserted into the gate-level netlist. Also, extra nets are added to the netlist, contributing to the virtual ground rails (VGND) which distribute to the drain terminal of each sleep transistor. All the source terminals of the sleep transistors are connected to the real ground (GND). The connection of the "VSS" pin [1] of each gate is then modified to the corresponding virtual ground net, according to the CPLEX solution. Finally, a sleep control signal is introduced as an extra circuit input signal, and all the gate terminals of the sleep transistors are connected to this input signal. Appendix C shows the modified netlist after the MTCMOS technique is applied for the 2-bit multiplier design example.

The new gate-level netlist, resulting from the MTCMOS design flow, is imported to the Cadence physical design environment. During the floorplan phase, a cavity is defined where the sleep transistors will be located. The remaining steps of the physical design follow the conventional ASIC design. Figure 4.11 exemplifies the sleep transistor layout implementation. Although this style incurs routing

[1]Originally, this pin was connected to GND.

complexity because of the additional virtual ground rails (VGND), the conventional placement and routing methods can be adopted with minimal modification by using commercially available tools (i.e., *Qplace* and *Wroute*).

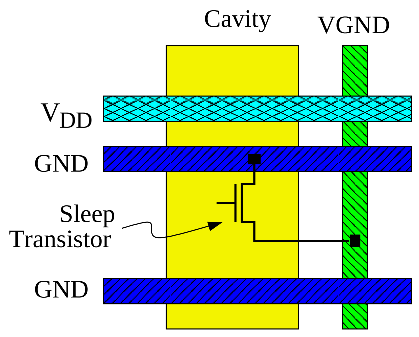


Figure 4.11: Layout example with placed sleep transistor.

4.5 Summary

A true low-power design technique has to be tightly integrated into the main design environment to ensure that the timing versus power and area versus power design tradeoffs are easily handled during the iterative design process. The Canadian Microelectronics Corporation (CMC) digital ASIC design flow was introduced in this chapter. In order to integrate the developed MTCMOS design environment into the CMC design flow, a discharge current database that characterizes the standard cells in the technology library was constructed. An automated vector generation engine and a CPLEX solver interface engine were then designed and fully integrated with the flow. The gate-level netlist works as the design media, switching the iterative low-power design among synthesis, MTCMOS design, and physical design.

Chapter 5

Conclusion

Low-power design is attracting a great deal of attention in VLSI digital design, especially for portable systems and high performance systems. The reduction of the transistor size allows a higher integration density and increases the operating frequency. The rapid switching of millions of transistors dissipates tremendous power and overheats the chip, reducing the reliability of the chip and necessitating expensive and large cooling systems. For many event driven applications, such as mobile devices where circuits spend most of their time in an idle state with no computation, standby leakage power is the dominant power dissipation. The MTCMOS technique has emerged as an effective way to achieve high performance and low-power designs. Since the size of the sleep transistor directly affects the timing versus power tradeoff, there is an obvious need for a MTCMOS design environment in which the sleep transistor sizing problem can be solved.

This thesis introduces a new MTCMOS design environment, where effective methods to solve the sleep transistor optimization problem are integrated.

5.1 Heuristic/Mathematical Approaches

Two genetic algorithms (GAs) are implemented in this thesis to solve the low-power MTCMOS BPP and SPP. The performance of the GAs is compared with that of a state-of-the-art ILP solver. For all the MTCMOS test circuits in this paper, the BPP oriented GA found all the optimum solutions. Even though most of the computation time of the GA is small due to the small problem size, the proposed GA still achieves a 89%, on average, reduction in the CPU time, compared to that of the ILP CPLEX solver. However, the performance of the SPP oriented GA is not computationally competitive with the CPLEX solver in terms of the solution quality. Although the GA can improve the solution quality by employing more populations or generations, the execution time also increases to the same order used by the CPLEX solver, which degrades the effectiveness of the heuristic method.

To improve both the solution quality and the computation time of the sleep transistor sizing problem, a First-Fit(FF) heuristic and a Set-Covering (SCP) model are proposed in this thesis. By eliminating the over-estimation of the discharge current, introduced by the gate-clustering technique, the proposed FF technique achieves a 12% and 92%, on average, reduction in leakage power and CPU time compared to those of the BPP technique, respectively. As for the SCP technique, the model is formulated by relaxing the constraints of the SPP model, resulting in discharge current sharing in the circuit. This phenomena reduces not only the total area of all sleep transistors but also the noise bouncing on the virtual ground. The experimental results reveal that the SCP technique improves the cost function, leakage power, and computation time by 8.9%, 2%, and 99%, respectively. Large reductions in the CPU time by the FF and SCP techniques prove that the proposed methods are effective and scale well with larger circuits.

5.2 MTCMOS Design Environment

VLSI design is an iterative design process to balance area versus power and timing versus power tradeoffs. It requires a true low-power design technique to be tightly integrated into the main design environment so that the designers can accurately and efficiently perform these tradeoffs. In order to develop a MTCMOS automatic design environment, a discharge current database is constructed that contains all the information about the standard cells in the technology library. An automated vector generation engine is then developed to build vectors for all the gates in the netlist. An ILP CPLEX solver interface is also developed and integrated to solve the sleep transistor optimization problem. Based on these two tools, a MTCMOS design environment is developed and fully integrated into the Canadian Microelectronics Corporation (CMC) digital ASIC design flow.

5.3 Future Work

Although this work presents solutions to existing problems, it has also opened the door for other research ventures.

For example, the benchmarks used in this thesis are six pure combinational logic circuits without memory units and flip-flops. The issue is how to design the sequential circuits by using the gate-clustering MTCMOS techniques. Furthermore, how can a MTCMOS latch be effectively designed?

The high and low threshold voltages (V_{TH} and V_{TL}) of the design are adjusted by modifying the parameters "dvthn" and "dvthp" in the Hspice model file. Future work in this area should take the form of applying physical layouts to characterize high and low threshold voltages. Another extension to this work will be to develop a MTCMOS standard cell library and sleep transistors.

Future work should involve the design and fabrication of a larger MTCMOS circuit using the newly developed design flow.

Appendix A

Glossary

ALU	:	Arithmetic Logic Units
ASIC	:	Application Specific Integrated Circuit
BPP	:	Bin-Packing Problem
CAD	:	Computer Aided Design
CF	:	Constant Field
CMC	:	Canadian Microelectronics Corporation
CMOS	:	Complementary Metal Oxide Semiconductor
CMOSP18	:	The 0.18-micron CMOS Technology
CV	:	Constant Voltage
DC	:	Design Compiler
DEF	:	Design Exchange Format
DIBL	:	Drain Induced Barrier Lowering
DFT	:	Design for Testability
DRC	:	Design Rule Check

DSM	:	Deep Sub-Micron
DSP	:	Digital Signal Processing
FF	:	First-Fit
GA	:	Genetic Algorithm
GIDL	:	Gate Induced Drain Leakage
ILP	:	Integer Linear Programming
IC	:	Integrated Circuit
IP	:	Intellectual Property
LVS	:	Layout Versus Schematic
MTCMOS	:	Multi-Threshold CMOS
NP-hard	:	Non Deterministic Polynomial Hard
PCB	:	Printed Circuit Board
PDA	:	Personal Digital Assistance
RTL	:	Register Transfer Logic
SoC	:	System on Chip
SCP	:	Set-Covering Problem
SPP	:	Set-Partitioning Problem
ST	:	Sleep Transistor
TSMC	:	Taiwan Semiconductor Manufacturing Company
UDSM	:	Ultra Deep Sub-Micron
VHDL	:	Very High Speed Integrated Circuit Hardware Description Language
VLSI	:	Very Large Scale Integration
VTCMOS	:	Variable Threshold CMOS

Appendix B

RTL Code

```
module Circuit6288_2bit (prod_out, a_in, b_in);
        output [3:0] prod_out;
        input [1:0] a_in, b_in;

        wire [3:0] prod_top;
        wire [1:0] a_top, b_top;


       // Circuit6288_2bit is an I/O wrapper for Ckt6288

        TopLevel6288 Ckt6288 (prod_top, a_top, b_top);

        PDO08CDG pprod00 ( .PAD(prod_out[0]), .I(prod_top[0]) );
        PDO08CDG pprod01 ( .PAD(prod_out[1]), .I(prod_top[1]) );
        PDO08CDG pprod02 ( .PAD(prod_out[2]), .I(prod_top[2]) );
        PDO08CDG pprod03 ( .PAD(prod_out[3]), .I(prod_top[3]) );
        PDIDGZ pa00 ( .C(a_top[0]), .PAD(a_in[0]) );
        PDIDGZ pa01 ( .C(a_top[1]), .PAD(a_in[1]) );
        PDIDGZ pb00 ( .C(b_top[0]), .PAD(b_in[0]) );
        PDIDGZ pb01 ( .C(b_top[1]), .PAD(b_in[1]) );
endmodule
```

```
module TopLevel6288 (prod_out, a_in, b_in);

        output [3:0] prod_out;
        input [1:0] a_in, b_in;

        assign prod_out = a_in * b_in;

endmodule
```

Appendix C

Gate-Level Netlist

Netlist After Synthesis

```
module Circuit6288_2bit ( prod_out_3_, prod_out_2_, prod_out_1_,
                          prod_out_0_, a_in_1_, a_in_0_, b_in_1_,
                          b_in_0_, VDD, VSS );

    output prod_out_3_ ;
    output prod_out_2_ ;
    output prod_out_1_ ;
    output prod_out_0_ ;
    input a_in_1_ ;
    input a_in_0_ ;
    input b_in_1_ ;
    input b_in_0_ ;
    input VDD ;
    input VSS ;

    wire b_top_1_ ;
    wire \Ckt6288_2bit|n5 ;
    wire \Ckt6288_2bit|n1 ;
    wire prod_top_3_ ;
    wire \Ckt6288_2bit|n6 ;
    wire a_top_0_ ;
    wire \Ckt6288_2bit|n2 ;
    wire prod_top_2_ ;
    wire a_top_1_ ;
    wire \Ckt6288_2bit|n3 ;
    wire prod_top_1_ ;
    wire b_top_0_ ;
    wire \Ckt6288_2bit|n4 ;
    wire prod_top_0_ ;

    supply1 VDD ;
    supply0 VSS ;

    NOR3D1 \Ckt6288_2bit|U1 ( .Z(prod_top_3_) , .A3(\Ckt6288_2bit—n3 ) ,
                        .A2(\Ckt6288_2bit—n2 ) , .A1(\Ckt6288_2bit|n1 ) ,
                        .VSS(VSS) , .VDD(VDD) ) ;
    NOR3D1 \Ckt6288_2bit|U2 ( .Z(prod_top_2_) , .A3(\Ckt6288_2bit|n2 ) ,
                        .A2(prod_top_0_) , .A1(\Ckt6288_2bit|n3 ) ,
                        .VSS(VSS) , .VDD(VDD) ) ;
    INVD0 \Ckt6288_2bit|U3 ( .Z(\Ckt6288_2bit|n2 ) , .A(b_top_1_) ,
                        .VSS(VSS) , .VDD(VDD) ) ;
    INVD0 \Ckt6288_2bit|U4 ( .Z(\Ckt6288_2bit|n3 ) , .A(a_top_1_) ,
                        .VSS(VSS) , .VDD(VDD) ) ;
    INVD0 \Ckt6288_2bit|U7 ( .Z(prod_top_0_) , .A(\Ckt6288_2bit|n1 ) ,
                        .VSS(VSS) , .VDD(VDD) ) ;
    INVD0 \Ckt6288_2bit|U10 ( .Z(\Ckt6288_2bit|n4 ) , .A(\Ckt6288_2bit|n5 ) ,
                        .VSS(VSS) , .VDD(VDD) ) ;
    NAN2D0 \Ckt6288_2bit|U5 ( .Z(\Ckt6288_2bit|n1 ) , .A2(b_top_0_) ,
                        .A1(a_top_0_) , .VSS(VSS) , .VDD(VDD) ) ;
    NAN2D0 \Ckt6288_2bit|U8 ( .Z(\Ckt6288_2bit|n5 ) , .A2(b_top_0_) ,
                        .A1(a_top_1_) , .VSS(VSS) , .VDD(VDD) ) ;
    NAN2D0 \Ckt6288_2bit|U9 ( .Z(\Ckt6288_2bit|n6 ) , .A2(a_top_0_) ,
                        .A1(b_top_1_) , .VSS(VSS) , .VDD(VDD) ) ;
    MUXB2D0 \Ckt6288_2bit|U6 ( .Z(prod_top_1_) , .SL(\Ckt6288_2bit|n6 ) ,
                              .A1(\Ckt6288_2bit|n5 ) , .A0(\Ckt6288_2bit|n4 ) ,
                              .VSS(VSS) , .VDD(VDD) ) ;

      PDO08CDG pprod00 ( .PAD(prod_out_0_) , .I(prod_top_0_) ) ;
      PDO08CDG pprod01 ( .PAD(prod_out_1_) , .I(prod_top_1_) ) ;
      PDO08CDG pprod02 ( .PAD(prod_out_2_) , .I(prod_top_2_) ) ;
      PDO08CDG pprod03 ( .PAD(prod_out_3_) , .I(prod_top_3_) ) ;

      PDIDGZ pa00 ( .PAD(a_in_0_) , .C(a_top_0_) ) ;
      PDIDGZ pa01 ( .PAD(a_in_1_) , .C(a_top_1_) ) ;
      PDIDGZ pb00 ( .PAD(b_in_0_) , .C(b_top_0_) ) ;
      PDIDGZ pb01 ( .PAD(b_in_1_) , .C(b_top_1_) ) ;

      PVDD1DGZ vdd_core ( .VDD(VDD) ) ;
      PVSS1DGZ vss_core ( .VSS(VSS) ) ;

endmodule
```

Netlist With STs

```
module Circuit6288_2bit ( prod_out_3_, prod_out_2_, prod_out_1_,
                         prod_out_0_, a_in_1_, a_in_0_, b_in_1_,
                         b_in_0_, sleep, VDD, VSS );

      output prod_out_3_ ;
      output prod_out_2_ ;
      output prod_out_1_ ;
      output prod_out_0_ ;
      input sleep ;
      input a_in_1_ ;
      input a_in_0_ ;
      input b_in_1_ ;
      input b_in_0_ ;
      input VDD ;
      input VSS ;

      wire VSS_1 ;
      wire VSS_2 ;
      wire VSS_3 ;
      wire VSS_4 ;
      wire VSS_5 ;
      wire sleep_top ;
      wire b_top_1_ ;
      wire \Ckt6288_2bit|n5 ;
      wire \Ckt6288_2bit|n1 ;
      wire prod_top_3_ ;
      wire \Ckt6288_2bit|n6 ;
      wire a_top_0_ ;
      wire \Ckt6288_2bit|n2 ;
      wire prod_top_2_ ;
      wire a_top_1_ ;
      wire \Ckt6288_2bit|n3 ;
      wire prod_top_1_ ;
      wire b_top_0_ ;
      wire \Ckt6288_2bit|n4 ;
      wire prod_top_0_ ;

      supply1 VDD ;
      supply0 VSS ;

      ST250   sleep_1   ( .G(sleep_top) , .S(VSS) , .D(VSS_1) )  ;
      ST250   sleep_2   ( .G(sleep_top) , .S(VSS) , .D(VSS_2) )  ;
      ST250   sleep_3   ( .G(sleep_top) , .S(VSS) , .D(VSS_3) )  ;
      ST250   sleep_4   ( .G(sleep_top) , .S(VSS) , .D(VSS_4) )  ;
      ST250   sleep_5   ( .G(sleep_top) , .S(VSS) , .D(VSS_5) )  ;
      NOR3D1 \Ckt6288_2bit|U1 ( .Z(prod_top_3_) , .A3(\Ckt6288_2bit—n3 ) ,
                          .A2(\Ckt6288_2bit—n2 ) , .A1(\Ckt6288_2bit|n1 ) ,
                          .VSS(VSS_1) , .VDD(VDD) ) ;
      NOR3D1 \Ckt6288_2bit|U2 ( .Z(prod_top_2_) , .A3(\Ckt6288_2bit|n2 ) ,
                          .A2(prod_top_0_) , .A1(\Ckt6288_2bit|n3 ) ,
                          .VSS(VSS_5) , .VDD(VDD) ) ;
      INVD0 \Ckt6288_2bit|U3 ( .Z(\Ckt6288_2bit|n2 ) , .A(b_top_1_) ,
                          .VSS(VSS_4) , .VDD(VDD) ) ;
      INVD0 \Ckt6288_2bit|U4 ( .Z(\Ckt6288_2bit|n3 ) , .A(a_top_1_) ,
                          .VSS(VSS_3) , .VDD(VDD) ) ;
      INVD0 \Ckt6288_2bit|U7 ( .Z(prod_top_0_) , .A(\Ckt6288_2bit|n1 ) ,
                          .VSS(VSS_4) , .VDD(VDD) ) ;
      INVD0 \Ckt6288_2bit|U10 ( .Z(\Ckt6288_2bit|n4 ) , .A(\Ckt6288_2bit|n5 ) ,
                          .VSS(VSS_3) , .VDD(VDD) ) ;
      NAN2D0 \Ckt6288_2bit|U5 ( .Z(\Ckt6288_2bit|n1 ) , .A2(b_top_0_) ,
                          .A1(a_top_0_) , .VSS(VSS_3) , .VDD(VDD) ) ;
      NAN2D0 \Ckt6288_2bit|U8 ( .Z(\Ckt6288_2bit|n5 ) , .A2(b_top_0_) ,
                          .A1(a_top_1_) , .VSS(VSS_4) , .VDD(VDD) ) ;
      NAN2D0 \Ckt6288_2bit|U9 ( .Z(\Ckt6288_2bit|n6 ) , .A2(a_top_0_) ,
                          .A1(b_top_1_) , .VSS(VSS_1) , .VDD(VDD) ) ;
      MUXB2D0 \Ckt6288_2bit|U6 ( .Z(prod_top_1_) , .SL(\Ckt6288_2bit|n6 ) ,
                          .A1(\Ckt6288_2bit|n5 ) , .A0(\Ckt6288_2bit|n4 ) ,
                          .VSS(VSS_2) , .VDD(VDD) ) ;
```

```
    PDO08CDG pprod00 ( .PAD(prod_out_0_) , .I(prod_top_0_) ) ;
    PDO08CDG pprod01 ( .PAD(prod_out_1_) , .I(prod_top_1_) ) ;
    PDO08CDG pprod02 ( .PAD(prod_out_2_) , .I(prod_top_2_) ) ;
    PDO08CDG pprod03 ( .PAD(prod_out_3_) , .I(prod_top_3_) ) ;

    PDIDGZ psleep ( .PAD(sleep) , .C(sleep_top) ) ;
    PDIDGZ pa00 ( .PAD(a_in_0_) , .C(a_top_0_) ) ;
    PDIDGZ pa01 ( .PAD(a_in_1_) , .C(a_top_1_) ) ;
    PDIDGZ pb00 ( .PAD(b_in_0_) , .C(b_top_0_) ) ;
    PDIDGZ pb01 ( .PAD(b_in_1_) , .C(b_top_1_) ) ;

    PVDD1DGZ vdd_core ( .VDD(VDD) ) ;
    PVSS1DGZ vss_core ( .VSS(VSS) ) ;

endmodule
```

Bibliography

[Anis01] M. Anis, M. Mahmoud, and M. Elmasry, "Efficient Gate Clustering for MTCMOS Circuits," In *Proceedings of the 14th Annual International ASIC/SOC Conference*, pp. 34–38, Washington, DC, 2001.

[Anis02] M. Anis, S. Areibi, M. Mahmoud, and M. Elmasry, "Dynamic and Leakage Power Reduction in MTCMOS Circuits Using an Automated Efficient Gate Clustering," In *Proceedings of the 39th Design Automation Conference*, pp. 480–485, New Orleans, 2002.

[Anis03] M. Anis, S. Areibi, and M. Elmasry, "Design and Optimization of Multithreshold CMOS (MTCMOS) Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, No. 10, pp. 1324–1342, 2003.

[Beas96] J. E. Beasley and P. C. Chu, "A Genetic Algorithm for the Set Covering Problem," *Europaen Journal of Operational Research*, vol. 94, pp. 392–404, 1996.

[Bell95] A. Bellaouar and M. I. Elmasry, *Low-Power Digital VLSI Design Circuits and Systems*, Kluwer Academics Publications, MA, 1995.

[Bohr96] M. Bohr and et al., "A high-performance 0.25-μm logic technology optimized for 1.8V operation," In *Proceedings of the International Electron Devices Meeting*, pp. 847–850, 1996.

[Brew90] J. Brews, *High Speed Semiconductor Devices*, Wiley, New York, 1990.

[Chan96] A. Chandrakasan, I. Yang, C. Vieri, and D. Antoniadis, "Design Considerations and Tools for Low-Voltage Digital System Design," In *Proceedings of the 33rd Design Automation Conference*, pp. 113–118, 1996.

[Chat80] P. Chatterjee, W. Hunter, T. Holloway, and Y. Lin, "The Impact of Scaling Laws on the Choice of n-channel or p-channel for MOS VLSI," *IEEE Electron Device Letters*, vol. 1, No. 10, pp. 220–223, 1980.

[Chu95] P. C. Chu and J. E. Beasley, “A Genetic Algorithm for the Set-Partitioning Problem,” *European Journal of Operational Research*, vol. 94, pp. 392–404, 1995.

[Comp00] Taiwai Semiconductor Manufacturing Company, “TSMC SPICE Model,” 2000.

[Corp02] Canadian Microelectronics Corporation, “Tutorial on CMC’s Digital IC Design Flow,” 2002.

[Denn74] R. Dennard, F. Gaensslen, H. Yu, V. Rideout, E. Bassous, and A. LeBlanc, “Design of Ion-Implanted MOSFETs with Very Small Dimensions,” *IEEE Journal of Solid-State Circuits*, vol. SC-9, No. 5, pp. 256–268, 1974.

[Falk94] E. Falkenauer, “A New Representation and Operators for Genetic Algorithms Applied to Grouping Problems,” *Evolutionary Computation*, vol. 2, No. 2, pp. 123–144, 1994.

[FTsa03] Y. F.Tsai, D. Duarte, N. Vijaykrishnan, and M. J. Irwin, “Implications of Technology Scaling on Leakage Reduction Techniques,” In *Proceedings of the 40th Design Automation Conference*, pp. 187–190, Anaheim, 2003.

[Gare79] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, New York, 1979.

[Gere99] S. H. Gerez, *Algorithms for VLSI Design Automation*, Wiley, Chichester, West Sussex, 1999.

[Gold91] D. E. Goldberg and K. Deb, “A comparative analysis of selection schemes used in genetic algorithms,” *Foundations of Genetic Algorithms*, pp. 181–186, 1991.

[Hatl97] J. Hatler and F. Najm, “A Gate-Level Leakage Power Reduction Method for Ultra Low-Power CMOS Circuits,” In *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 475–478, Santa Clara, CA, 1997.

[Holl92] J. H. Holland, *Adaption in Natural and Artificial Systems*, MIT Press, Cambridge, Massachusetts, 1992.

[Hyun01] I. Hyunsik, T. Inukai, H. Gomyo, T. Hiramoto, and T. Sakurai, “VTC-MOS Characteristics and Its Optimum Conditions Predicted by a Compact Analytical Model,” In *Proceedings of the International Symposium on Low-Power Electronics and Design*, pp. 123–128, 2001.

[Inc02] ILOG Inc., “ILOG CPLEX 7.5 User’s Manual,” 2002.

[Inc04a] Intel Inc., “Intel Microprocessor Quick Reference Guide,” 2004.

[Inc04b] Intel Inc., “Intel Pentium 4 Processor with 512-KB L2 Cache on 0.13 Micron Process and Intel Pentium 4 Processor Extreme Edition Supporting Hyper-Threading Technology Datasheet,” 2004.

[Inc99] Virtual Silicon Technology Inc., “Native-18 Standard Cell Library Databook,” 1999.

[John02] M. Johnson, D. Somasekhar, L. Chiou, and K. Roy, “Leakage Control with Efficient Use of Transistor Stacks in Single Threshold CMOS,” *IEEE Transactions on VLSI Systems*, vol. 10, No. 1, pp. 1–5, 2002.

[Kang03] S. Kang and Y. Leblebici, *CMOS Digital Integrated Circuits*, McGraw-Hill, New York, 2003.

[Kao01] J. Kao and A. Chandrakasan, “MTCMOS Sequential Circuits,” In *Proceedings of the 27th European Solid State Circuits Conference*, pp. 861–869, 2001.

[Kao02] J. Kao, S. Narendra, and A. Chandrakasan, “Subthreshold Leakage Modeling and Reduction Techniques,” In *Proceedings of the International Conference on Computer Aided Design*, pp. 141–148, 2002.

[Kao97] J. Kao, A. Chandrakasan, and D. Antoniadis, “Transistor Sizing Issues and Tools for Multi-threshold CMOS Technology,” In *Proceedings of the 34th Design Automation Conference*, pp. 409–414, Las Vegas, Nevada, 1997.

[Kao98] J. Kao, S. Narendra, and A. Chandrakasan, “MTCMOS Hierarchical Sizing Based on Mutual Exclusive Discharge Patterns,” In *Proceedings of the 35th Design Automation Conference*, pp. 495–500, Las Vegas, Nevada, 1998.

[Karn02a] T. Karnik, S. Borkar, and V. De, “Sub-90nm Technologies – Challenges and Opportunities for CAD,” In *Proceedings of the International Conference on Computer Aided Design*, pp. 203–206, 2002.

[Karn02b] T. Karnik, Y. Ye, J. Tschanz, L. Wei, S. Burns, V. Govindarajulu, V. De, and S. Borkar, “Total Power Optimization by Simultaneous Dual-Vt Allocation and Device Sizing in High Performance Microprocessors,” In *Proceedings of the 39th Design Automation Conference*, pp. 486–491, New Orleans, 2002.

[Kato00] N. Kato, Y. Akita, M. Hiraki, T. Yamashita, T. Shimizu, F. Maki, and K. Yano, “Random Modulation: Multi-Threshold-Voltage Design Methodology in Sub-2-V Power Supply CMOS,” *IEICE Transactions on Electronics.*, vol. E83-C, No. 11, pp. 1747–1754, 2000.

[Kawa93] T. Kawahara, M. Horiguchi, Y. Kawajiri, G. Kitsukawa, T. Kure, and M. Aoki, “Subthreshold current reduction for decoded-driver by self-reverse biasing,” *IEEE Journal of Solid-State Circuits*, vol. 28, No. 11, pp. 1136–1143, 1993.

[Ketk02] M. Ketkar and S. Sapatnekar, "Standby Power Optimization via Transistor Sizing and Dual Threshold Voltage Assignment," In *Proceedings of the International Conference on Computer Aided Design (ICCAD)*, pp. 375–378, 2002.

[Kirk83] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, "Optimization BY Simulated Annealing," *Science*, vol. 220, No. 4598, pp. 671–680, May 1983.

[Lee03a] D. Lee and D. Blaauw, "Static Leakage Reduction through Simultaneous Threshold Voltage and State Assignment," In *Proceedings of the 40th Design Automation Conference*, pp. 191–194, Anaheim, 2003.

[Lee03b] D. Lee, W. Kwong, D. Blaauw, and D. Sylvester, "Analysis and Minimization Techniques for Total Leakage Considering Gate Oxide Leakage," In *Proceedings of the 40th Design Automation Conference*, pp. 175–180, Anaheim, 2003.

[Lee03c] W. F. Lee, *Verilog Coding for Logic Synthesis*, Wiley, Hoboken, New Jersey, 2003.

[Levi94] D. Levine, *A Parallel Genetic Algorithm for the Set Partitioning Problem*, Ph.D. thesis, Illinois Institute of Technology, Department of Computer Science, 1994.

[Liao02] W. Liao, J. M. Basile, and L. He, "Leakage Power Modeling and Reduction with Data Retention," In *Proceedings of the International Conference on Computer Aided Design*, pp. 714–719, 2002.

[Long03] C. Long and L. He, "Distributed Sleep Transistor Network for Power Reduction," In *Proceedings of the 40th Design Automation Conference*, pp. 181–186, Anaheim, 2003.

[Mitc96] M. Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, Massachusetts, 1996.

[Moor75] G. E. Moore, "Progress in Digital Integrated Circuits," In *Proceedings of the International Electronic Devices Meeting*, pp. 11–13, 1975.

[Muto95] S. Mutoh, T. Douseki, Y. Matsuya, T. Aoki, S. Shigematsu, and J. Yamada, "1-V Power Supply High-Speed Digital Circuit Technology with Multi-Threshold Voltage CMOS," *IEEE Journal of Solid-State Circuits*, vol. 30, No. 8, pp. 847–854, 1995.

[Muto96] S. Mutoh, S. Shigematsu, Y. Matsuya, H. Fukuda, T. Kaneko, and J. Yamada, "A 1-V multithreshold-voltage CMOS digital signal processor for mobile phone applications," *IEEE Journal of Solid-State Circuits*, pp. 1795–1802, 1996.

[Ono01] A. Ono and et al., "A 100nm node CMOS technology for pratical SOC application requirement," In *Proceedings of the International Electron Devices Meeting*, pp. 511–514, 2001.

[Pier96] R. F. Pierret, *Semiconductor Device Fundamentals*, Addison-Wesley, Reading, MA, 1996.

[Raba96] J. M. Rabaey, *Digital Integrated Circuits*, Prentice Hall, NJ, 1996.

[Ragh96] A. Raghunathan, S. Dey, and N. K. Jha, "Glitch analysis and reduction in register transfer level power optimization," In *Proceedings of the 33rd Design Automation Conference*, pp. 331–336, 1996.

[Rard98] R. Rardin, *Optimization in Operations Research*, Prentice Hall, Boston, 1998.

[Roy00] K. Roy and S. C. Prasad, *Low-Power CMOS VLSI Circuit Design*, Wiley Interscience, New York, 2000.

[Rush98] A. Rushton, *VHDL for Logic Synthesis*, Wiley, New York, 1998.

[Shah91] K. Shahookar and P. Mazumder, "VLSI Cell Placement Techniques," *ACM Computing Surveys*, vol. 23, No. 2, pp. 143–220, 1991.

[Shig97] S. Shigematsu, S. Mutah, Y. Matsuya, Y. Tanabe, and J. Yamada, "A 1-V High-Speed MTCMOS Circuit Scheme for Power-Down Application Circuits," *IEEE Journal of Solid-State Circuits*, vol. 32, No. 6, pp. 861–869, 1997.

[Siri99] S. Sirichotiyakul, T. Edwards, O. Chanhee, Z. Jingyan, A. D. Harchoudhury, R. Panada, and D. Blaauw, "Stand-by Power Minimization through Simultaneous Threshold Voltage Selection and Circuit Sizing," In *Proceedings of the 36th Design Automation Conference*, pp. 436–441, New Orleans, LA, 1999.

[Sysw89] G. Syswerda, "Uniform crossover in genetic algorithms," In *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 2–9, Morgan Kaufmann, 1989.

[Veen84] H. 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, No. 4, , 1984.

[Ye98] Y. Ye, S. Borkar, and V. De, "A New Technique for Standby Leakage Reduction in High-Performance Circuits," In *Proceedings of the 1998 Symposium on VLSI Circuits*, pp. 40–41, 1998.

[Yeo00] Y. C. Yeo, "Direct tunneling gate leakage current in transistors with ultra thin silicon nitride gate dielectric," *IEEE Transactions on Electron Devices*, pp. 540–542, 2000.