NTNU

Norwegian University of
Science and Technology

Switching in multipliers

Jakub Jerzy Kalis

Master of Science in Electronics
Submission date: June 2009
Supervisor: Per Gunnar Kjeldsberg, IET
Co-supervisor: Johnny Pihl, Atmel Norway

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Problem Description

An inherent characteristic of a multiplier implementation is that most of its power consumption is caused by spurious toggling on internal nodes (up to 75% have been cited). This master thesis assignment builds on results from a project assignment and aims at developing techniques for estimation of the functional and spurious switching in multipliers. A number of different candidate multiplier implementations will be provided. It involves work at the gate and layout levels using advanced circuit layout and simulation tools.

Assignment given: 15. January 2009
Supervisor: Per Gunnar Kjeldsberg, IET

Abstract

Digital multipliers are an important part of most of digital computation systems, such as microcontrollers and microprocessors. Multiplication operation is a quite complex task, thus there is many different solution varying in area, speed and power consumption. An important notice is that multipliers often are a part of critical path of a system which makes them especially important for these factors. During last decade, power efficiency has become an important issue in digital design and a lot of design methods has been created and investigated to meet this subject.

It is a known fact that most of power consumed by arithmetic circuit is dissipated by hazards and toggles (up to 75%), that do not bring any information to final result. The method of evaluating the amount of spurious switching and its effect on power dissipation is investigated here.

This thesis aims to find a method to estimate switching characteristics and its effect on power dissipation of eight supplied multipliers given in form of HDL net-list with some software overhead. As switching generally stands for majority of power consumption in digital CMOS circuits, this effect gives also good indication of overall power dissipation. One of the difficulties in estimating average power and transition density is pattern dependency problem. The method based on Monte Carlo technique is used where an adequate accuracy is obtained within moderate time and resource usage.

Three of investigated multipliers are net-lists created by using methodology developed in [21]. These are synthesized and laid out in the technology used by Atmel Norway. The amount of logical state changes is compared from pre- and post- synthesis net-lists. The technology mapped net-lists are also examined for power consumption to see the connection between switching and dynamic power dissipation. The fan-out delay model used to estimate total toggling gives a good approximation of circuit properties; it is however too simple to give a good estimate of spurious toggling inside the circuit and its effect on power consumption.

The same estimation technique is used to investigate a DesignWare circuit (DW02) which is an industrial approach of building fast and power efficient multipliers. The results show that this is the most power effective solution among the examined circuits

(45-47% less than the most power efficient circuit from [21]) It is also a solution with smallest amount of hazards during a multiplication operation (38-52%).

A circuit generated by module generation software (ModGen) is also investigated. This solution is quite power efficient, it has however largest amount of power dissipated by the spurious toggling (62-68%).

It is also noticed that transition density and what follows the power dissipation in strongly dependent on the process, temperature and voltage variation. In fact the higher temperature gives reduction in power consumption.

Preface

This thesis is submitted as a result of my work done in the period from January to June 2009 as the last step of my master degree studies at The Norwegian University of Science and Technology (NTNU). The main supervisor of this work was Professor Per Gunnar Kjeldsberg at the Department of Electronics and Telecommunications. The co supervisor and a company representative was Johnny Phil from Atmel Norway who supported the data necessary for the experiments.

The assignment was originally created for two students. The scope of the task had to be limited because I was the only one who accepted this assignment.

Acknowlegments

I want to thank my supervisor Per Gunnar Kjeldsberg for his help and comments on my work. I am also thankful to Johnny Phil for the support and information necessary to carry out this assignment. For help with the diverse software problems I want also to thank Saeeid Tamasbi Oskuii. I am grateful for all help from my friends and teachers here at the Department of Electronics and Telecommunications.

Jakub Kalis
Trondheim, 18. June 2009

Contents

List of Tables

List of Figures

List of Abbreviations

CLA	Carry Look-Ahead Adder
CMOS	Complementary Metal Oxide Semiconductor
CPA	Carry Propagate Adder
FA	Full Adder
HA	Half Adder
I/O	Input/Output
PP	Partial Product
PPRT	Partial Product Reduction Tree
RCA	Ripple Carry Adder
SAIF	Switching Activity Interchange Format
SDF	Standard Delay Format
VCD	Value Change Dump
VMA	Vector Merge Adder
-	-

Chapter 1

Introduction

Recent trend in portable computing and wireless communication makes power consumption a critical concern in VLSI circuit and system design. The decreasing size of electronic devices makes it possible to place several units with different tasks on one chip. This leads to larger power density [30]. These devices must meet demands of high speed computation and complex functionality with low battery power consumption. All these factors are taken into consideration while designing various digital signal processing chips or microcontrollers.

Traditionally, the main priority has been given to area and speed, while the latest trends exhibit more consideration on parameters like flexibility, testability, reliability. Power optimization is also presented as a design goal in its own right in digital circuit design [21]. There are a lot of reasons for that. High-speed circuits consume a lot of energy in a short amount of time, generating a great deal of heat. This is an undesired bi-product which has to be removed by otherwise unnecessary hardware overhead. Another consideration is battery driven products. Batteries must last longer for devices with higher and faster computation possibility.

One of the main considerations in power aware design is constriction and examination of a basic arithmetic circuit where multipliers are the dominating building blocks. Literature studies show that multiplication has been an important research area in the recent years [11], [21], [22].

The multiplication process occurs in most of digital computation systems like microprocessors and microcontrollers. Multipliers are in fact among the main contributions of area and power consumption in digital signal processing systems [22]. An important fact is that they are usually placed in critical paths of such systems. This makes the multiplication a significant process with regard to design possibilities.

Calculating a product of two input data require a lot of switching activities in CMOS designed multipliers due to many partial products accumulation operations. As the switching activities in a multiplier account for the majority of its power con-

sumption, minimization of this activity can effectively reduce the power dissipation of the whole circuit [24]. The way in which some circuits are built makes it sensitive to problem of spurious glitching. This switching activity can stand for up to 75% of power consumption in a digital circuit [11], [14].

To minimize the power consumption a good estimation technique of switching activities is needed.

1.1 Problem Description

An inherent characteristic of a multiplier implementation is that most of its power consumption is caused by spurious toggling on internal nodes. This thesis aims at estimating the functional and spurious switching in multipliers and its effect on power dissipation.

The objectives of the thesis are to investigate 32 bit combinatorial multiplier given in form of a HDL net-list together with the necessary overhead like delay models, post synthesis information etc.

Three of these are generated by software created as a work behind the Ph.D thesis by Saeeid T. Oskuii [21]. This software is used to create a HDL net-lists which represent three different power optimization levels. One is a maximally optimized structure for lowest power dissipation, one represents worst-case power characteristic, while the last one is a random generated multiplier net-list. All three multipliers are supplied with a delay model created for the purposes of the above named thesis.

A model of a multiplier used by Atmel Norway is also supplied. The circuit is representative for the layout and synthesis outcomes commercially used for 32 bit multipliers in the digital IC industry. The delay model of synthesized net-list is also provided.

The three multipliers from [21] are synthesized and laid out in a synthesis tools used by Atmel Norway. They are laid out with the same technology constrains as the multipliers supplied by Atmel Norway. This way they can be compared with attention to power and switching characteristic

Last multiplier type, inspected in this thesis is a net-list generated by Arithmetic Module Generator (ModGen) available at http://modgen.dnsalias.com. The net-list generated there is then laid out in the same way technology as previews multipliers. This is done to verify how good the ModGen tool produces compared to the other circuits results.

The goals of this thesis are:

- Get better understanding of power aware design of digital integrated circuits.

- Develop a methodology for obtaining switching activity from HDL simulation on the net-list level.
- Compute the spurious switching in the supplied circuits.
- Compare the amount of switching in for different cases, both the theoretical and technology dependent cases.
- Compare the amount of power consumed by the supplied multipliers in the given technology.
- Propose an eventual improvement of the fan-out delay model used in [21].
- Produce a tutorial which describes step-by-step methodology of obtaining the switching characteristics of the circuit.

1.2 Outline of the Thesis

Chapter 1 has introduced the significance of low power design and importance of good estimate of this power during the design phase. The mail goals and contributions are mentioned here as well.

In Chapter 2 the main idea behind multiplication process is discussed together with the relevant information about creating the low power digital multipliers.

Chapter 3 presents the sources of power dissipation in digital circuits. The power estimation methods with special emphasis on the switching power dissipation are discussed here as well.

Proposition of a switching measuring technique is given in Chapter 4. Also a method of measuring the average power is described in this chapter.

Results of the experiments carried out in this thesis are described in Chapter 5. Results are then discuss in Chapter 6 together with conclusions drawn by the author.

Tables with more detailed result presentation are presented in Appendix A. Tutorial on toggling and power measurement method is presented in Appendix B. Appendix C contains scripts and parts of code that are developed and used to obtain presented results.

1.3 Contributions

This thesis has led to some interesting results where the main contributions are listed below:

- Method for recording the switching activity in a net-list of combinational arithmetic circuit with both timing and the zero-delay model to find the spurious toggling density of the circuit.
- Collection of the switching activity of supplied multipliers in terms of average toggling per executed operation.
- Collection of average power used by the multipliers during their operational mode.
- Comparison of power dissipation together with the switching characteristics of the given multipliers to find out the improvement in the different optimization mode of the circuits.
- Comparison of switching characteristics for the technology independent low-power reduction-tree multipliers before and after mapping into technology library used by Atmel Norway.
- Developing a step-by-step tutorial on measuring total glitching characteristics of a combinational circuit together with a technique of extracting the spurious glitches of internal nodes.
- Discussion on how the results can be used by designers and developers of digital integrated circuits.

Chapter 2

Multiplication

This chapter presents the theory behind digital multiplication process. The main problems regarding low power multiplication are presented here together with the most common solution for low power digital multipliers.

2.1 Multiplier

A multiplier is a digital logic circuit built with a purpose of computing the result of multiplying two data inputs. Like every other digital component, it manipulates digital signals in some hardware components and have a capability of operating on a mathematical data represented by these signals.

There are several ways to obtain the desired result inside a logic circuit. Output of a *combinational circuit* at any time depends only on the present inputs, with total disregard to the past state of the circuit [15]. The function of this circuit type is fully defined by a set of Boolean expressions. Another type of logic circuit are the *sequential circuits* which includes also memory elements, such as flip-flops and latches. This makes the output of such circuit dependent on past states as well as input values. In this thesis just the combinational circuits will be considered.

2.1.1 Microcontroller

The digital multiplier is a core component of computation circuits like microprocessors or microcontrollers. A microcontroller is a small microcomputer system fitted on a single integrated circuit. It has a quite simple processing unit combined with some support functions like oscillators, timers, I/O support etc. They have a program memory often based on flash or Masked ROM [32]. In contrast to microprocessors used in personal computers, the simplicity is an important factor. They are usually designed for small applications like automatically controlled devices, but there are

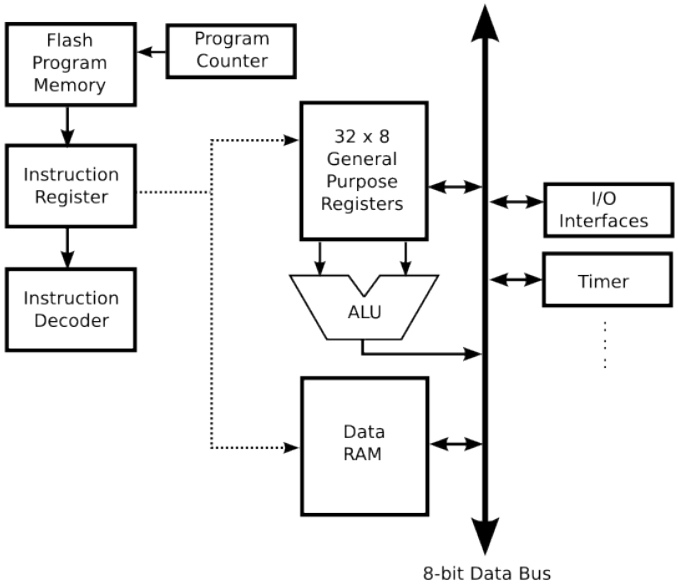

Figure 2.1: Microcontroller Architecture (after AVR ATmega32 datasheet p. 6)

also multipurpose microcontroller designs available.

A heart of a microcomputer system is a programmable device which accepts binary data from some kind of an input device and processes this data to produce a desired output. This way the microcontroller executes programs stored in the memory registers and transfers data to and from I/O ports in the central bus.

An example of microcontroller structure is shown on Figure 2.1 in form of a block diagram. A core component of most microcontrollers is an arithmetic logic unit (ALU) which performs the entire integer arithmetic and bit-wise logical operations. It includes logical addition and subtraction and some may also perform *multiplication* or even the division operations. Fixed-point multiplication is an important issue in the arithmetic circuit.

The other blocks are often support units that store the results or help to determine the next step of computation. The control unit and instruction register governs and coordinates the activities of different processor sections and I/O devices. The register file comprises different registers used to store data, addresses and other information during the program execution.

2.1.2 Data Representation

Digital circuits have to work on data represented in binary number system in several ways. Fixed-point binary number system is based on radix-2 with the digit set {0, 1} where the number consists of a fixed number of fractional and whole part digits. Natural numbers also referred to as the unsigned integers can be viewed as the fixed-

point numbers without the fractional part.

The sign-and-magnitude format is used to represent both positive and negative numbers by letting the first bit represent the sign extension (usually 1 denote negative sign while 0 a positive sign). A two's complement number system encodes positive and negative numbers in a binary number representation in such what that the addition and subtraction circuitry do not need to examine the signs of the operands to determine whether to add or subtract the number [23]. Although there are many advantages of signed-magnitude representation, like simplicity and intuitive appeal, the main drawback is that arithmetic operations of numbers with unlike sign must be handled differently than the same-sign operations, while the two's complement numbers needs to be decoded to obtain the final result.

Different multiplier architecture operates with binary data represented in many different ways dependent on specifications and design constrains.

2.2 Multiplication Schemes

A multiplier computes by manipulating two input data to generate the result. This is done by generating many partial products (PP) for successive accumulation operations. The accumulation, often implemented as addition, require many switching activities in the functional units of multipliers and that is why they account for most of the power dissipation in a multiplier.

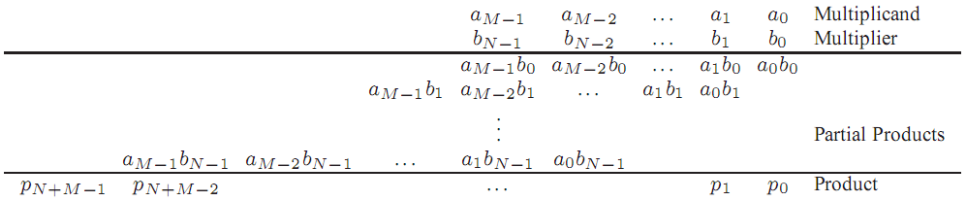

Figure 2.2: Multiplication of binary numbers

		Binary representation
Multiplicand	$A = \sum_{i=0}^{M-1} a_i 2^i$	$(a_{M-1}a_{M-2}\cdots a_1a_0)_2$
Multiplier	$B = \sum_{j=0}^{N-1} b_j 2^j$	$(b_{N-1}b_{N-2}\cdots b_1b_0)_2$
Product	$P = A \cdot B = \sum_{k=0}^{M+N-1} p_k 2^k$	$(p_{M+N-1}p_{M+N-2}\cdots p_1p_0)_2$

Table 2.1: Notation used to discus multiplication algorithms

Generation and summing of PPs for an unsigned $M \times N$-bit multiplier is illustrated on Figure 2.2. This is in fact one of the simplest multiplication schemes, known as shift-and-add method which consists of cycles of shifting and adding inside a control loop. The block diagram of shift-and-add multiplier is presented on Figure 2.3. The

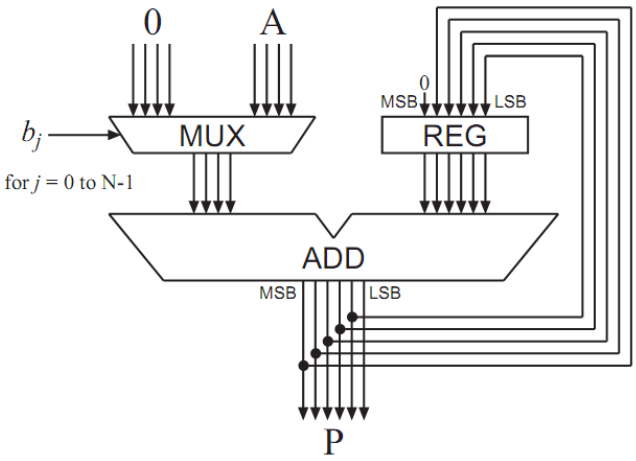

Figure 2.3: Block diagram of shift-and-add multiplier

PPs reduction can be implemented using multiplexer or logical AND gates. After N cycles the product is determined as shown in Equation 2.1.

$$P = \sum_{k=0}^{M+N-1} p_k 2^k = (\sum_{i=0}^{M-1} a_i 2^i)(\sum_{j=0}^{N-1} b_j 2^j) = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} a_i b_j 2^{i+j} \tag{2.1}$$

In other words the multiplication is performed by simply multiplying (or taking a logical AND) the multiplicand A with each single bit of multiplier B. Thus, the multiplication can be considered as an adding of set of numbers, all-zero vector or a shifted version of multiplicand A [23]. PPs can be viewed as either a sum of previously accumulated PPs or a shifted version of A.

The more convenient method of illustrating the PPs is a dot-diagram as shown on Figure 2.4. This representation shows just the position and alignment of bits and not their values.

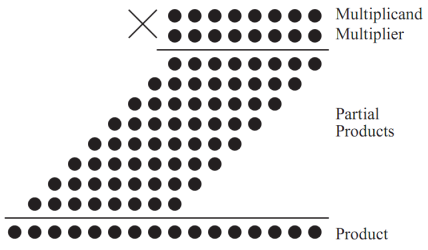

Figure 2.4: A dot-diagram representation of a 8x8 multiplier [21]

.

Thus the multiplication of two fixed-point numbers can be divided into two basic steps, generation and accumulation of partial. To get the desired characteristic of multiplier, these two steps need to be investigated. The accumulation procedure

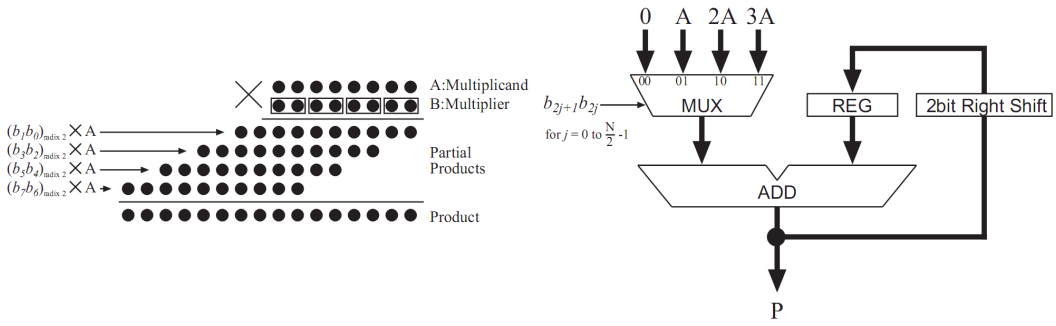

Figure 2.5: 8×8 Radix-4 multiplication, a dot-diagram and its possible structure

is often narrowed down to a process of successive multi-operand addition with the number of PP being equal to the number of bits in the multiplicand. These additions require many switching activities and that is why they account for most of the power dissipation in a multiplier.

The multiplication schemes can be classified in three general types [21]. The sequential multipliers successively add the generated PPs to the previous accumulated sum. Multipliers of this type are very slow devices used only in applications where the speed is not a critical factor. Shift-and-Add is an example of this multiplication method.

Array multipliers generate and accumulate PPs simultaneously. Therefore the same circuit is used to both PPs generation and accumulation. This way the overhead caused by separate control of these steps is avoided.

Parallel multipliers are preferred in high speed applications where all PPs are generated in parallel and then accumulated by fast multi-operand adders. They are often implemented as combinational circuits.

In next Sections the PPs generation and accumulation methods are presented. The most efficient ways of PPs handling are tree structures that consist of carefully design addition chains. The way in which they are built is presented here as well.

2.2.1 Partial Products Generation

Different ways of representing data makes slight differences in the PP generation process. Multiplexers and AND gates are used in an unsigned radix-2 shift-and-add multiplication. For sign-magnitude numbers the circuitry may be more complex because of the overhead managing the sign extension bits. The one way is to complement the negative operands, multiply the unsigned values and complement the result if needed. It is however quite complicate technique for the 2's complement numbers. For these the sign extension to the width of the final product is needed [23].

Reducing the number of generated PP reduce the complexity of the accumulation

step. Higher radix representation leads to fewer digits by examination of two or more bits at the time. For example the radix-4 multiplication assumes digits of values 0, 1 2 and 3, therefore A, 2A and 3A are needed, where A is multiplicand. This is illustrated in Figure 2.5. This gives reduction in number of PPs by a factor of 2. However the complexity of the PP accumulation can be moved to generation step where the 3A multiple requires some additional overhead [21].

Many different methods of PP generating procedures has been introduced and developed. A way of dealing with binary multiplication is to use Booth recoding technique [3]. When a zero vector is to be added and shifted the addition step can be skipped. Shifting along is much faster than addition followed by shifting, which makes the multiplication process faster. Booth observed that multiplication will be faster whenever there is a large number of consecutive 1's. This is done by replacing the equivalent addition sequence with one subtraction and one addition. This way process became faster and more efficient.

2.2.2 Partial Product Accumulation

After the PPs are generated they must be accumulated (summed) to achieve the final result. This reduction process can be performed either by rows using adders or by columns using counters [21].

The simplest method is reduction by rows with use of numerous two-operand Carry-Propagate Adders (CPA). Addition circuit is the primary building block of arithmetic operation but the addition operation is also the most time consuming process in parallel multipliers [21]. The carry propagation is the main cause of speed restriction in arithmetic circuits.

A		**B**		**Sum**
0	+	0	=	0
0	+	1	=	1
1	+	0	=	1
1	+	1	=	10

Table 2.2: Addition of binary numbers

Adding binary numbers is a straightforward operation as shown in Table 2.2. This operation can be viewed as Boolean function and presented in form of half-adder (HA) boolean function with a truth-table illustrated in Table 2.3. When addition is performed on larger numbers, the carry have to be propagated, thus the full-adder (FA) function is develop as shown in Table 2.4. Half and full adders can be implemented in many ways, the most common gate level implementation is shown in Figure 2.6.

One of simplest adder circuits to understand is a ripple-carry adder (RCA) which is build of a number of full and half-adders connected in chain as in Figure 2.7 (a). The latency of k-bit RCA is $O(k)$ thus accumulation of n words would use computation time equal to [21]:

$$T_{RCA} = O(n + logk) \tag{2.2}$$

This makes the RCA undesirable for high speed arithmetic units.

When analyzing the carry propagation it has been notice that the key to fast addition is a low-latency carry network [23]. Instead of propagating, Carry Look-Ahead Adder (CLA) calculates, for each position, whether that position is going to propagate a carry if one comes in from the right. Doing that the speed of addition gets severely improved ($O(logk)$) and that is why CLA is most widely used design for high speed solutions. However it is usually expensive and unaffordable solution.

There is also many alternative designs that have some other advantages over CLA like Carry-Skip Adders or Carry-Select Adder, wider described in [23].

A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$S = A \oplus B$
$C_{in} = A \cdot B$

Table 2.3: Truth-table of half-adder

A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$S = A \oplus B \oplus C_{in}$
$C_{in} = A \cdot B + B \cdot C_{in} + C_{in} \cdot A$

Table 2.4: Truth-table of full-adder

Carry-Save Adder

In cases that involve addition of three or more operands, such as PP accumulation in multipliers, the carry propagation is not necessary in each cycle. Carries can be instead saved and added in some next operand cycle. This is the idea behind a Carry-Save Adder (CSA).

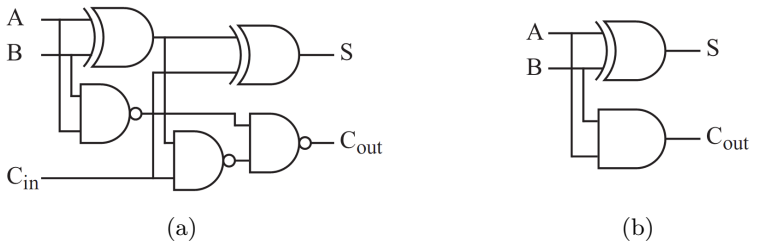

Figure 2.6: Gate-level implementation of full-adder (a) and half-adder (b)

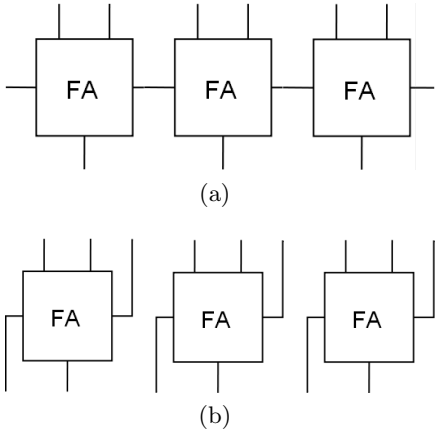

Figure 2.7: Relationship between ripple-carry adder (a) and carry-save adder (b)

The CSA is made with the same building blocks as RCA, full- and half-adders. The difference is that carry is saved and not propagated through the addition chain (Figure 2.7). The row of FAs reduces three numbers (C_{in}, A and B) into two numbers (C_{out} and S), hence CSA is often referred to as a $[3:2]$ adder.

If X, Y and Z are considered inputs to a three-operand CSA then the output can be viewed as a partial sum and shifted carry vector, S and C respectively.

$$X + Y + Z = S + 2C^1 \tag{2.3}$$

Producing a partial sum and partial carry makes the bits in the same row independent of each other, and the addition may be carried out in parallel, thus it makes the whole multiplication process faster.

The total PPs reduction can be perform performed as a $[p:2]$ adder where the p bit-vectors is reduced to 2 vectors, partial carry and partial sum. Nevertheless, the partial sum and carry have to be merged together to give correct final result. This final addition is described in Section 2.2.2

CSA can be implemented serially, using one CSA and some sum and carry registers, more preferable way is however tree structure which makes addition faster

[1]Multiplying radix-2 number with two corresponds to binary left shift.

[23].

Tree Multipliers

Multipliers using high performance CSA trees followed by a fast final adder, make logarithmic time multiplications possible. Various multiples of the multiplicand is formed at the top and the added in a combinational partial product tree. They produce sums in a redundant way to be converted to standard binary output at the bottom.

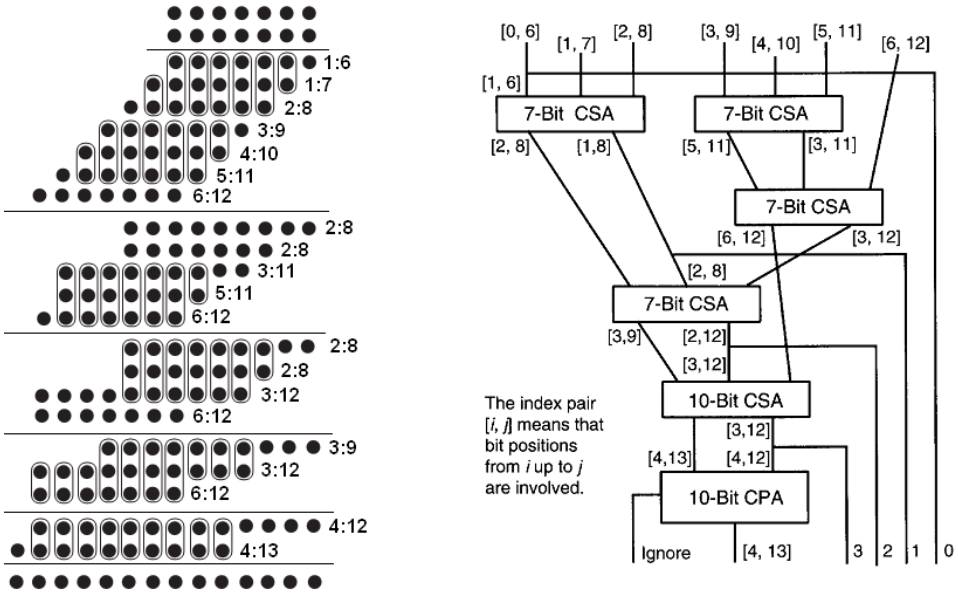

Figure 2.8: Possible CSA tree for a 7×7 multiplier with the corresponding dot-diagram [23]

In [37] a partial product reduction tree (PPRT) called Wallace tree has been proposed. In each stage of reduction, Wallace tree performs a preliminary grouping of rows into sets, which reduces the number of operand by a factor of 1.5 [23]. Figure 2.8 present a possible outcome of a tree structured multiplier.

The Dadda Tree is a similar method introduced by Dadda [8] but reducing PPs by columns. By combining these two methods the faster and more power efficient multipliers can be built.

The benefit of the tree structures that the logic depth is reduced as well as the propagation delay. The multiplication latency is only $O(logk)$, not much slower than addition. Adding PP with regular carry propagate adders would require $O(logk)^2$ time. On the other hand, PPRT can have irregular structure making its design and layout difficult. Variation in signal path length gives and connections may have implications on both performance and power consumption.

The Final Adder

At the last step of addition the partial carry and sum have to be merged together to give correct result, thus the final adder is often referred to as the Vector Merge Adder (VMA) [21]. There are several different alternatives to construct the final CPA varying in speed, area and complexity. RCA or a Manchester adder is an example of worst-case delay proportional to the length, while the CLA can offer the logarithmic delay growth. The CPA must be carefully investigated for optimal solution possibilities [23].

Such adders must be designed with special consideration since the inputs (output of the reduction tree) not always arrive at the same time. In fact the timing characteristic of the reduction stage can be used to optimize the VMA. The bits that come last can be merged by a fast but larger adder while the parts of input that comes first can be computed by slower but smaller device.

Careful design of the final adder can save power consumption of the circuit as well as its area. Use of structural components which compute as late as possible may reduce spurious toggling.

Chapter 3

Power Dissipation

Power estimation is an important part of digital circuit design. Nowadays, it is important to construct integrated circuits that are able to perform in portable computing and communication devices. They demand combining high-speed computation and complex functionality with low power consumption. That is why power estimation is essential on every step of digital IC design. Power estimation is especially important in arithmetic circuit design, since they stands for major of power dissipation in digital systems [22].

3.0.3 Sources of Power Dissipation

There are four different sources of power consumption in digital CMOS circuits [24]. Leakage current is determined by fabrication process and consists of two types currents, reverse biased diode current and the subthreshold current (see Figure 3.1). The first sort occurs when the transistor is turned off and another active transistor charges the drain with respect to formers bulk potential. Subthreshold leakage is due to the carrier inversion charge that exists at the gate voltages below the threshold voltage. Next type of power consuming effect is given by standby current which is a direct current (DC) drawn continuously from voltage supply to the ground.

The sum of these two power sources is often referred to as the static power dissipation. Leakage currents can be minimized with proper device technology choice. Standby currents are an important issue in design styles like memory cores or pseudo-NMOS, but they are insignificant in pure CMOS technology. Gate and subthreshold leakage currents becomes however more and more important in modern circuit production technologies (60 and 45 nm). In technologies below 0,13um static power dissipation becomes an important element of total power dissipation [27].

The short circuit current also called rush-through current comes about due to the DC path between supply voltages during the output transitions. It is often a case that

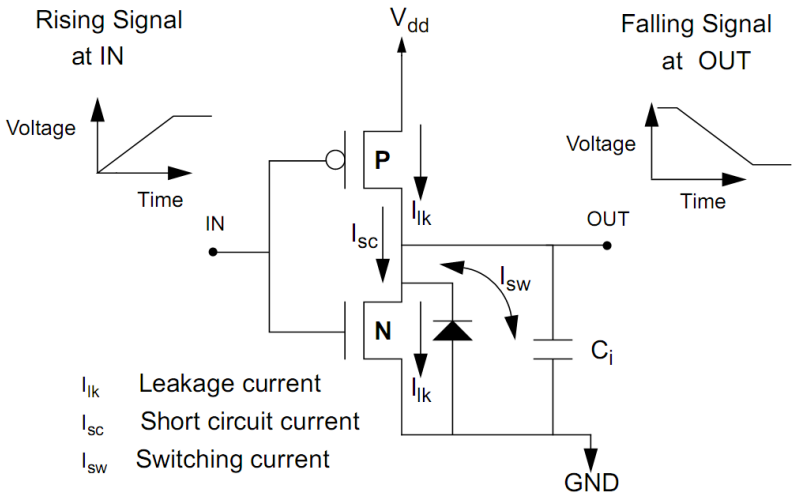

Figure 3.1: Components of power dissipation [31]

both PMOS and NMOS transistors are simultaneously active during a state change. Then the current is conducted directly between supplies. This effect is proportional to the input ramp time and the load and transistor gate size. The most important part of power characteristics is the capacitance current flowing to charge and discharge capacitive loads while logical changes occurs. This is closer described in Section 3.1.

The term dynamic power dissipation refers to the sum of short circuit and capacitive dissipations. The short circuit current can be made small by right designs techniques. That is why the capacitive dissipation is the dominant source of power dissipation in CMOS circuits.

3.1 Switching Power

Switching power element of total power dissipation is the most important aspect of power-awareness in digital circuits design. It is established that the switching power comprises 70% or even up to 90% of the power consumption of an active CMOS circuit [11], [14].

Dynamic power dissipation is when the load capacitance C_i of a CMOS gate is charged through the PMOS transistor during voltage transition from 0 to higher voltage level (V_{dd}). A zero to one transition draws $C_i V_{dd}^2$ energy from voltage supply in a CMOS circuit such as the inverter on Figure 3.1. This energy is equally divided between PMOS transistor and an output capacitor. On the other hand the output transition from V_{dd} to zero does not draw any charge from voltage supply, but the energy stored in the capacitor goes to the pull-down NMOS transistor (See Equation 3.1).

$$\text{Energy per transition} = \frac{1}{2}V_{dd}^2 C_i \tag{3.1}$$

The transitions at the node i may happened at a clock rate, f_{clk}, in most cases however switching occurs at some other rate. This can be described probabilistically by a transition density factor D_i.

$$P_i^{sw} = \frac{1}{2}V_{dd}^2 D_i C_i \tag{3.2}$$

The D_i is defined as average number of times in each clock cycle that node with the physical capacitance, C_i will make a transition. The notation of transition density is introduced in [18]. For a circuit consisting of N nodes the total power dissipation can be defined as in Equation 3.3.

$$P_{total}^{sw} = \sum_{i=1}^{N} P_i^{sw} = \frac{1}{2}V_{dd}^2 \sum_{i=1}^{N} D_i C_i \tag{3.3}$$

3.1.1 Transition Density Factor

The transition density factor shows the average switching rate of the gate per unit of time. As Equation 3.3 shows, the dynamic power consumption of a gate is directly proportional to this factor. [18] defines transition density factor as shown on Equation 3.4 where the $n_i(T)$ is the number of transitions at node in a time interval T.

$$D_i = \lim_{T \to \infty} \frac{n_i(T)}{T} \tag{3.4}$$

When the circuit is working at some frequency, the average numbers of transitions in one clock cycle can be defined as:

$$\overline{n_i} = \frac{D_i}{f_{clk}} \tag{3.5}$$

Equation 3.6 shows an alternative way of calculating the average power consumed by a combinational circuit of N nodes when knowing the average number of transitions of all the nodes of the circuit.

$$P_{total}^{sw} = \frac{1}{2}V_{dd}^2 \sum_{i=1}^{N} C_i \overline{n_i} \tag{3.6}$$

In combinatorial circuits it is sometimes desirable to extract the activity factor as a number of all transitions that happens during the circuit operation. Since this kind of circuit carries out a previously defined function, it is desired to extract the total amount of switching needed to execute this operation. This way several circuits with

the same function can be coopered in terms of switching activity. Since switching is the main contributor of circuit power usage, its amount is a good indicator for total power dissipation of the circuit.

The switching activity factor is difficult to calculate because it is strongly dependent on a number of circuit parameters and technology factors. The activity at the output of a gate is strongly dependent on the inputs activity as well. It is also strongly dependent on Boolean function of the circuit, as well as the logic style used to implement the circuit. The working conditions influence it as well. This makes the straight-forward technique of estimating power in a logic simulator a very complicated task due to dependence problem. The activity factor can however be found by direct simulation of the circuit. All this has to be taken into consideration when computing the power usage of a digital circuit.

3.1.2 Spurious Switching

Not all the signal transitions at a gate output in a combinational circuit are useful for overall result. Gates and other circuit elements introduce some kind of delay to signals they propagate. In complex circuits with a lot of fan-out and fan-in signal paths the signal arrival times to internal gates can vary. Such nodes can have multiple transitions in one clock cycle before they settle to the correct logical value.

The undesired transitions, which do not bring any information, are called glitches or hazards [29]. They cause only unnecessary switching and short-circuit power dissipation. Glitches can have to origins; they are either generated or propagated in the circuit. In the worst case the level of glitching transitions can grow as $O(N^2)$, where N is the logic depth of the circuit [14].

A glitch is generated when arrival times of different signals into a gate are greater than the internal delay of the gate itself. More about gate delay is presented in Section 3.5.4. In addition glitches can occur only when an input pattern makes glitching possible. When this two conditions meet, a probability of glitch generation is given in Equation 3.7.

$$P(G) = P_{pattern} \cdot P_{prop} \tag{3.7}$$

The pattern probability P_{patt} is a fixed value of a gate. As an example we can look at 2 input NAND gate. From the truth table (Table 3.1) it can be concluded that a glitch is generated only when the transition of input goes through the 11 input vector, for example transition from 10 to 01, presented on left side of Figure 3.2.

The value of factor P_{prop} is a dependent on the circuit architecture, it is a number of possible pair of paths that may cause a glitch, compared to all possible path pairs leading to the gate input.

A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

Table 3.1: Truth table of at NAND gate

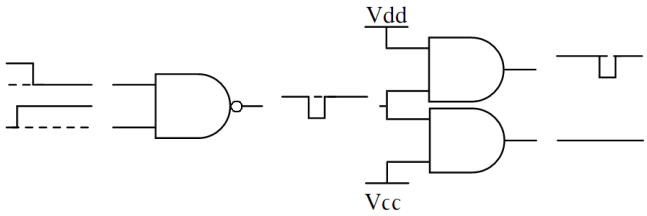

Figure 3.2: Glitch is generated and filtered or propagated

A glitch is propagated when the characteristic of glitch arriving at the gate makes it possible to change the output state. Otherwise glitch arriving at the input is suppressed or filtered at the output as shown on Figure 3.2. Glitches can be also filtered when the delay through the gate is greater that the length of a glitch, even thou the gate function make propagation possible. All this makes the glitching estimation a difficult task.

Glitching cause unnecessary power consumption and that is why estimating and reducing this phenomenon is an important factor in digital design.

3.2 Power Estimation

Power estimation is defined as the process of calculating power and energy dissipated at the different phases of the circuit design process. This is mostly referred to as a problem of estimating the average power dissipation of a digital circuit [19]. In other cases we can talk about estimating worst case power, so called voltage drop problem. A lot of work about the average power estimation is devoted to, so called logic-level methods for CMOS circuits.

As mentioned in Section 3.1 it can be assumed that the digital chip components draw power only during the logical transitions. As shown in Equation 3.3 the power dissipation is highly dependent on the switching activity inside the circuit. Because of that the power estimation problem becomes more complicated due to the pattern dependency problem. That is why the good method of power usage and switching activity factor computation is an important issue in digital design.

3.2.1 Simulation based estimates

The most straight-forward method of reporting power is to simulate a circuit in some kind of circuit simulator, compute and report the power dissipation for a given set of inputs or all possible input values [19]. However, circuits are very complicated nowadays, with large amount of gates and inputs, so it is practically impossible to simulate circuit for all the input patterns.

Simulation with typical input pattern can also be difficult due to the fact that the input signals are generally unknown during the design phase [30]. They depend on the variety of factors like application specifications and the system in which the circuit will be used. Some circuits, like microcontroller cores are designed to work in variety types of systems and for diverse applications. Often the different parts of a system are designed separately or for different purposes. All this makes the complete and specific information about the inputs almost impossible to obtain.

Simulation based technique can be quite expensive and in order to improve the efficiency many different simulation-based methods were proposed on different level of design [28]. Their main advantage is that they are capable of handling various device models, different design styles, multi-phase clocking etc. The results are however strongly dependent on input signals used in the simulation. Due to the memory and execution time constrains, they are not suitable for large cell-based designs.

3.2.2 Probabilistic methods

To overcome the problem of pattern dependency the probabilistic techniques has been proposed ([2], [10], [12]). When based on the zero delay model symbolic simulation, they offers a fast solution of power dissipation estimation [30]. They relies on the probabilistic information about the circuit, like signal and activity probabilities, directly propagated through the circuit.

For different logical functions the different static probabilities yields. Looking at the NAND gate from section 3.1.2 we can assume that input has uniform distribution. The truth table (Table 3.1) shows that the probability of the output being 0 is $\frac{1}{4}$ while the probability of 1 is $\frac{3}{4}$. The probability of a 0 to 1 transition which actually is a power consuming transition for a 2 input NAND gate is as shown in Equation 3.8

$$p(0 \rightarrow 1) = p(0) \cdot p(1) = \frac{1}{4} \cdot \frac{3}{4} = \frac{3}{16} = 0,19 \tag{3.8}$$

When the probability of each node in the circuit is calculated, the average power consumption can be obtained using the Equation 3.3.

Calculating of the symbolic probability is however NP-hard and grows exponentially with the number of inputs [30]. The probabilistic methods suffer from the

speed/accuracy trade off due to correlations between the internal circuit nodes. These methods are quite accurate but computationally very expensive, when the correlations are taken into account. The main estimation error in the probabilistic power estimation methods is the glitch filtering and the dependency issues inside the circuits.

In [20] the concept of probability waveforms is introduced. This proposal consists of a compact signal probability and a sequence of events happening in different time instances. The simple waveform set is used in estimating power of tree multipliers in [21].

3.2.3 Monte Carlo Techniques

Another way to estimate power is by using statistical methods which try to combine the speed of probabilistic techniques with the accuracy of simulative methods. Estimating power by using the Monte Carlo approach is widely used since it has been introduced in [4]

Idea behind this approach is to simulate the circuit repeatedly for some typical or random input streams. The result will eventually converge to the average power dissipation of the circuit.

$$P_{total}^{sw} = \sum_{i=1}^{N} P_i{}^s w = \frac{1}{2} V_{dd}^2 \sum_{i=1}^{N} C_i \lim_{T \to \infty} \frac{n_i(T)}{T} \tag{3.9}$$

The main problem is that it is desired that the power is estimated for an infinite time period T. This can be done by calculating power corresponding to infinite T as a mean of several measurements of power dissipated in the circuit in a finite time interval. This is a well known mean estimation problem [36]

By considering a random representation of logic signals the stochastic process $\mathbf{x}_i(t)$ can be constructed. Then the power sample $\mathbf{P}_T$ corresponding to random power of $\mathbf{x}_i(t)$ over the time interval of T. When the $\mathbf{x}_i(t)$ is stationary the expected average number of transitions per second is a constant [4].

Figure 3.3 shows the overview of the technique. The setup region is an important part of this method. In the beginning of simulation run, the circuits does not work at its typical rate. Thus the circuit should be simulated until all the nets are switching at the stable rate. Main purpose of this phase is to make sure that the typical values are measured.

There are two main issues with this method; how to select right patterns to be applied in the simulation and how to choose when the power is converges close enough to the actual average power.

In order to guarantee that the length T of the sampling region is correct two factors is considered. This value can affect the error in normality approximation and

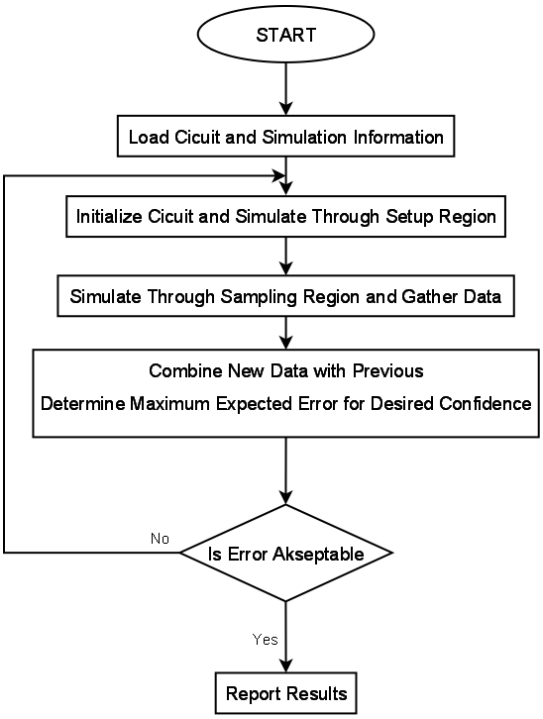

Figure 3.3: Block-diagram overview of a Monte Carlo technique

is heavily dependent on upon the circuit. Another factor is the simulation time is also dependent on circuit parameters as well as on simulation hardware. These two factors point to the conclusion that there are no optimal solutions for choosing of interval T. Best way is to decide it experimentally for the given hardware set [4].

In [4] two distinct advantages of Monte Carlo approach are described. It achieves desired accuracy in reasonable amount of time. The speed/accuracy trade off known from probabilistic techniques is avoided. The algorithm is simple to implement and use with existing logic or timing simulation environment.

3.3 Alternative Ways of Power Measurement

One of the most accurate way of estimating power consumption by direct circuit simulation is by using circuit simulators like SPICE [28]. They are also the most computationally expensive with long execution times. That is why they are most suitable for small but critical parts of the design.

PowerMill is a transistor level simulator from Synopsys that uses an event driven algorithm to increase the speed by to or three orders of SPICE magnitude [6].

When the Monte Carlo technique is to be used a McPower tool may be applied [28].

3.4 Statistical Method

The output of a statistical experiment is recorded as a numerical value. This values, or so called random samples are expected to vary somehow from sample to sample [36]. If X_1, X_2 ... X_n represent a random sample of size n, then the sample mean is defined by Equation 3.10.

$$\overline{X} = \frac{\sum_{i=1}^{n} X_i}{n} \tag{3.10}$$

The variability in the sample displays also how the observations are spread out from the average. Two different observations can have the same mean but it is quite possible that they differ considerably in the variability of their measurements. To determine this, sample variance is defined in Equation 3.11

$$S^2 = \frac{\sum_{i=1}^{n} (X_i - \overline{X})^2}{n-1} \tag{3.11}$$

The results from experiments are treated as the samples of a statistic process, where several samples are gathered as independent measurements.

3.5 Simulation

3.5.1 Levels of modeling

The type of primitive components used in a model of digital circuit decides the level of modeling. Model which contains only gates is called gate-level model. The register-transfer level provides models for systems at the register and instruction set level [1]. Usually at the lowest level we have components which cannot be decomposed into simpler components like gates and transistors. Non-primitive components, often referred to as modules or functional blocks, are described using functional operators.

3.5.2 Logic simulation

Logic simulation is a part of design verification that uses a model of designed system. The simulation program takes the representation of input stimuli and determinate the time evolution of the signals in the model [1]. The logical simulation ascertains that the expected results are obtained as well as that the circuit works as designed in term of power, speed, accuracy etc.

3.5.3 Type of Simulation

Simulators are classified after the type of internal model of the circuit as well as the level of abstraction. A simulator that executes the compiled HDL code, generated

from the RTL model is often referred to as the compiler-driven simulator. It is mainly oriented toward functional verification and the timing information is not considered. A simulator that takes a model based on data structures and event activity is said to be table-driven.

An event-driven simulator uses a structural model of a circuit to propagate events where the values of the input signals are defined in a stimulus file. This is usually a table-driven simulator. When an event takes place, the simulation time-flow mechanism manipulates the events in a way that its visibility will occur at the internal signals in the right time. This is an important feature because it allows accurate simulation of non-synchronized events, such as interrupts or internal hazards. This way values of certain signals can be checked and the time behavior of the circuit can be observed as well as the time skew of different signals and its effect on final result.

Several types of simulation are often combined to get more accurate results. The event driven algorithm propagates events through the components described by compiled-code models.

3.5.4 Delay models

Signals get delayed while propagating through CMOS gates inside a circuit. By modeling the delay value, we can get more accurate simulation results that show time dependencies inside the circuit. Delay is introduced by every gate to the signal propagated through this gate and it is dependent on a logic function of the gate as well as the technological aspects.

CMOS transistors are the main building blocks of logic gates in a digital circuit. Transistors together with other physical components introduce time delay when propagating signals through the digital circuit.

Problem of physical delay is illustrated on Figure 3.1. When an input signal is applied to the inverter, it does not make an abrupt transition from 0 to 1 but rather changing its value in time period t_r. The output do not change its value rapidly either, but falls with the time t_f. The relationship between these two time periods is called fall time delay. If an opposite situation arises (1 to 0 transition at the input) the rise time delay is introduced [35]. These delays are due to the parasitic resistance and capacitance of the transistors, for example the load capacitance C_i. In a logic chain, every logic gate drives another gate or number of gates, called gate fan-out. The size of C_i is dependent on number of gate fan-out, while its charging/discharging time are closely related to rise and fall delays. On the other hand if the input pulse is too short, the gate load capacitor would not reach to charge/discharge fast enough. These way short pulses can neither be propagated nor created inside such gate.

In real networks the perturbations of circuit parameters due to process or tem-

perature variation may change the propagation delays. That is why it is assumed that component of power dissipation crated by signal delay effects is strongly dependent on the parameter fluctuations. Hence the information about layout constrains and technology helps to create better and more genuine delay models to use in logic simulation.

Exact calculation of each gate delay is a quite complex task. That is why a good gate delay model is needed for simulation purposes. When modeling a gate behavior the function of a gate is evaluated, then the delay computation can be performed.

Process, Voltage and Temperature

The properties of a CMOS gate are strongly dependent on its working conditions. Its performance characterizations can be represented by the technology library. Temperature influences both internal and load capacitances as well as the carrier mobility inside the circuits [9]. Both threshold voltage and carrier mobility decreases with increasing temperature, which again makes the current and speed of a CMOS gate dependent on these factors. To model the delay of a gate all this factors has to be considered. This way, the simulation of the circuit at the gate level with the delay model supplied by the technology library gives quite accurate results.

Gate Delay Model

Transport delay is a basic delay model of a gate. It specifies the interval d separating the input and output change. This is usually simplified in such manner that delay values are integers with common divider. Depending on the type of delay value several types delay models were developed [1].

The most simple delay model is so called zero-delay model (Figure 3.4a), where all the delay values are set to zero. In this model all the events happens simultaneously. This way it can be quickly evaluated if the circuit produces the correct result.

Under the assumption that all the gate delays are know we have a nominal delay models. When all the transport delays are considered equal, they can be all scaled to 1 unit, thus this model is called a unit-delay model (Figure 3.4b). While the zero-delay model can be enormously effective in diagnosing and fixing design problems, the unit-delay model allows one to detect unbalanced paths and hazards. It is more accurate than the zero-delay model but still fairly simple without suffering from performance issues.

For devices like MOS gates, the output signal rise and fall transition delay are greatly different [1]. To imitate this fact the different rise and fall delays may be used in the simulation creating a transition-dependent delay model. The result of

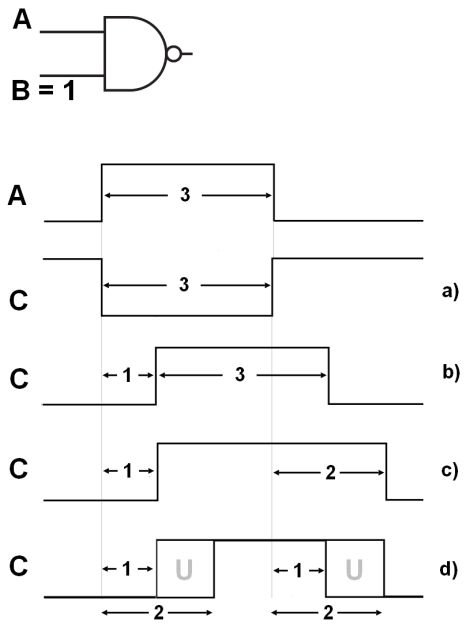

Figure 3.4: Delay models (a) Zero delay model, (b) Nominal transition-independent unit delay, (c) Rise and fall model with $d_{rise} = 1$ and $d_{fall} = 2$ (d) Ambiguous delay with $d_{min} = 1$ and $d_{max} = 2$.

this phenomenon is change in pulse width of signals propagating through the gates (Figure 3.4c).

Often the exact delay of a gate is unknown or specified to vary between two values depending on the conditions the circuit is working in or some other factors. To model this uncertainty in simulation an ambiguity interval is defined by minimum and maximum delays for every gate, creating an ambiguous delay model (Figure 3.4d). This model may be combined with the transition-dependent models.

Circuits use energy to switch states and produce results. The energy which a signal transfers is a function of its duration and amplitude. The signal will not force a gate to switch if its energy is too low. The minimum pulse width on the input, necessary to switch the gate output, is called the input inertial delay. Pulse shorter than the inertial delay are filtered (or suppressed) by the gate while the wider pulses propagate according to nominal transport model of the gate. Sometimes gates cannot generate pulses that are too short and an output inertial delay model is defined.

Delays introduced by the signal propagation along the wires become a significant part of delay modeling in modern high speed circuits. This may also be taken into consideration while simulating with timing models.

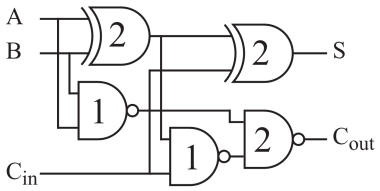

Figure 3.5: Fan-out model of a FA gate

Fan-out Delay Model

The full adder based PPRT shows unique properties of glitch generation and filtration. The FA and HA for this structures are connected in such way that they input capacitance is similar. The maximal fan-out of such nodes is two and only gates that can generate glitch are two NAND gates of the FA with unit delay [21].

The delay model proposed by [21] is called a fan-out model. Figure 3.5 shows a FA circuit with the corresponding delay unit marked for each gate. It is a fairly simple way of delay representation in a net-list of a digital circuit. Every gate is assumed to have a delay and the rejection limit equal to the number of its fan-outs. For example if a gate has 2 fan-outs, the propagation delay is assumed to be equal to 2 time units. The rejection limit is also sat up to 2 making all the pulses that are shorter than rejection limit will not appear at the output of the gate. In other words they will be filtered. The gates at the output have delay of 2 because they it is assumed that S and C_{out} will be connected to inputs of a similar adder in the next stage of reduction tree.

This model suffers however from some drawbacks due to its simplicity. In real networks the delay would not be of nominal values. Type of gate, type of inputs and outputs are not taken into consideration in this model. The layout information in terms of wire length or internal construction is not considered either. This way the XOR with two fan-outs gate has the same delay properties as for example NAND gate with two fan-outs. Process and technology information is not considered either.

This way of representing gate delay gives simple but fast way of dealing with glitch filtering and generation inside the circuit.

Chapter 4

Tools and Methods

This chapter presents the methodology developed and used in the thesis to measure toggle characteristics together with average power dissipation of a combinational circuit. This method gives total switching properties of the simulated circuit including spurious toggles and hazards. To get the most accurate results first the minimal period of simulation time at which the power and toggle count per multiplication are approximate at their average value, is decided. Then the exact average value within some error margin is computed by repetitive simulating and annotating the result from both simulation and power measuring tool.

The described method is possible under certain circumstances. First the supplied net-list have to come with some kind of delay model. To measure power technology library and parasitic information has to be given.

4.1 Choice of Estimation Method

High speed and accuracy is main concern while estimating circuit parameters. Among the large number of existing methods of gate level power estimations, the method based on Monte Carlo technique is chosen.

Simulated circuits have two 32 bit wide inputs and consist of over 5000 nets. This makes it virtually impossible to simulate for all possible inputs. To overcome this problem the technique where several shorter simulations are carried out is developed. Results from these simulations are then used as observations in a statistical experiment. This makes it possible to find average power consumption and switching activity per operation in terms of mean and standard deviation factor.

It is said that Monte Carlo based techniques are generally better than the probabilistic methods because they achieve greater accuracy with equivalent speed [4]. The pattern dependency that simulation techniques suffer from is avoided as well. The most important drawback noticed in work on this thesis was that the measurements

cannot be totally atomized. The simulation length has to be decided individually for a circuit type and the measurements obtained from software tool has to be adapt to extract the final value. That is why a noticeable part of this thesis is script development to edit and adapt simulation results.

Simulation method is chosen on basis of format the data is delivered in. The synthesized Verilog and VHDL net-lists may be simulated in programs such as Mentor Graphics ModelSim or Adlec ActiveHDL Simulator. The challenge is to create a high-quality testbench and input vector set to use under a simulation shift.

4.2 Measuring Method

To measure switching characteristic and its power consumption method shown on Figure 4.1 is developed. This method flow is suitable for circuits given as a HDL net-list with some kind of delay model. When the specified net-list is made technology dependent, power dissipated can be measured.

Two sets of simulations of one circuit net-list are performed, one using the zero delay model, and one with a fixed delay model. The zero delay models make all the signal transitions happened simultaneously without any delay through the internal gates. Assumption is made that the unnecessary switches are not taken into consideration under this type of simulation and only the productive transitions of the circuit nets are annotated.

Internal gate delay can be modeled in several ways, as a continuous time model, unify delay model etc. To obtain the accurate results the Standard Delay Data Format (SDF) is used. This contains the representation and interpretation of timing data such as path and interconnection delays, timing constrains and technology parameters for use in simulation process [33]. Since the switching characteristics depend on the external conditions as well as the technological aspects of design, the SDF file is created on the basis of technology library of the circuit.

The gate level net-list is simulated in some kind of simulation tool as described in Section 4.3. Chosen tool have to be able to report switching activities in some way. En example is a Value Change Dump file (VCD) used to carry out the switching information. This file contains header information, variable definitions, and value changes for the user specified variables. There is also a Switching Activity Interchange Format (SAIF) which has the same purpose as the VCD file. They save information about transition changes to use in some other software environment, or to save the waveform as a simple text file.

The net-state information can be also saved in a TSSI list file. TSSI format consist of a text list file and signal definition file. The list file contains the net values at the value change time periods, where each line of a file represents a time when one

or more signals changed value. It can include state change values at user specified times.

To report power a power measuring tool is needed. This tool has to be able to read the design together with the technology library information. Annotating the switching activity from the simulation tool is also necessary. Information about parasitic in the given technology is needed as well. This way the accurate power consumption can be reported by the power tool.

By comparing circuit switching and power characteristics of those two models, the power dissipated by spurious switching can be computed. The exact results are collected when the zero time and the timing simulation are carried out under same conditions. That involves simulation with the same input set, as well as under the same process, voltage and temperature situation.

In the following Sections, the experiment techniques developed during this thesis work are presented.

4.3 Logic Simulation

The simulation of digital circuits is an important tool in the design process. It is used to avoid serious design errors and to verify whether the design fulfills its specifications [16]. On each level of design, the outcome is tested and checked for errors and faults. A good simulation tool can help to create high-quality and reliable circuit.

Time behavior of a logic circuit can be modeled in many ways as described in Section 3.5.4. To run a complete simulation the proper data input have to be supplied, either randomly generated or obtained from the circuit specifications. The simulation tool can captures the state- and path dependent- switching in form of dump files or waveform formats

Mentor Graphics ModelSim is a GUI based simulation and debug environment for HDL designers. The tool provides simulation support for latest standards of SystemC, SystemVerilog, Verilog 2001 standard and VHDL [17]. It is a complete environment for circuit simulation and verification.

ModelSim has some built in functions which can count the switching of each net in the design. Switching activity is closely related to estimating the power consumption the power functions are used to monitor the nets or signals activities.

Another software tool considered is Adlec Active-HDL software. It is a completely integrated HDL design and verification solution that gives verification and debugging tools for design of digital circuits. However the ModelSim environment is preferred because this tool is well known by the author of this thesis.

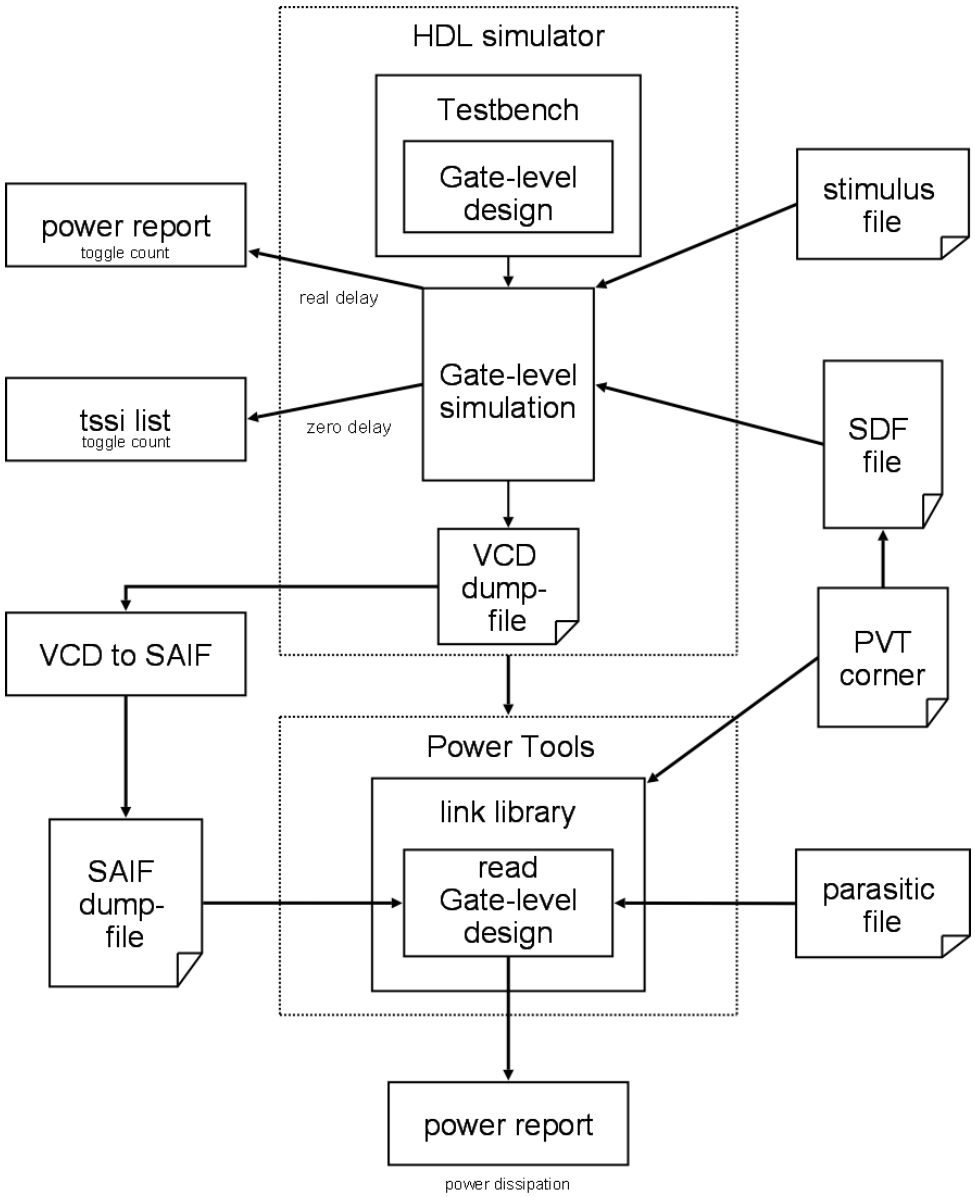

Figure 4.1: Block diagram of the methodology used to measure toggle characteristics and power dissipation of a gate level design

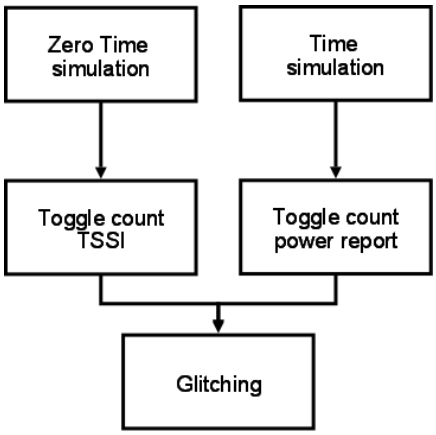

Figure 4.2: Obtaining switching properties

4.3.1 Stimulus

To estimate the average switching activity of the circuits a good stimulus is needed. The circuits are designed to manage a variety of tasks in different situations in a microcontroller. This means that the random generated stimulus is a good choice when finding the average power a multiplier.

To generate numbers to be multiplied by the circuits, a python script is developed (see Section C.4.1). The random function implements a pseudo-random number generator in which the 0 and 1 are uniformly selected to create a sequences consisted of 32 bits. The arbitrary number of these vectors can be created and saved in stimulus files. This way the different circuits can be simulated with the same vector set which make it easier to compare the switching characteristics of the circuits obtained during the logic simulation.

4.3.2 Toggle Count

The circuits are simulated under two modes; a zero delay model, and a time model. The difference between results of these two simulation sets gives the unnecessary transition count (See Figure 4.2).

When timing information is annotated in the simulation tool, the number of state changes (0 to 1 and 1 to 0) may be extracted for each node inside the circuit. To do that the power report function is invoked in ModelSim. This function annotates the number of transitions of an arbitrary node or nodes in the circuit under test. When summing the transition number of all circuit nodes the total toggle count is achieved (Power report on Figure 4.1). A Python script is created to extract the total count from the power report file as shown in Section C.4.3.

There are a lot of methods of counting the net state changes in the simulation

with time modeled in any way. The introduced here way of using the power command in ModelSim is just one of them. There are many ways in which toggle information is passed by simulation tools, like VCD dump files, SAIF or TSSI list format. They can be evoked when using some other HDL simulator that does not support the power commands of ModelSim. An extraction script can be created to extract the switching information in form of total toggle count.

Method of straightforward state change counting gives however incorrect results when simulating with the zero delay model. The reason of that is due to the way in HDL languages handles the circuits without explicitly prescribed delay. VHDL simulator deals with concurrent assignments by using an infinite short time unit called delta delay. In Verilog simulation a similar concept exists in terms of Non-Blocking Assignment with infinite short delay unit [7]. This way the simulators make sure that all the parallel statement will be able to produce the correct result, hence signals toggle with zero width time pulses before settling to the final value. In zero delay model those toggles seems to happen simultaneously but are counted as a usual time transitions by both dumping procedure and the power report tool.

To overcome this problem, the counting in the zero time model simulations is carried out in a different way. Many solutions of this problem have been examined. It is among others possible to export the simulation waveforms and analyze them externally to extract the zero time switching information. One of inspected solution required a software script development that would count the state changes only once in a time period, thus just the necessary transitions. The information of state of each net can be however annotated by the simulation tool at selected points in time as and dump the information into a TSSI list file. This way the state of each node is measured after it gains it stable value and just the necessary transitions are measured and accounted for. This solution of the problem is chosen because it gives the desired result without unnecessary software overhead.

To count the total number of signal transitions a simple Python script is developed. It reads a list file in TSSI format and returns the total number of transitions recorded in this file (See Section C.4.2).

The zero-time delay method is also applicable to the time modeled circuit and gives the same results as power report function. To verify that a simple experiment have been carried out. For a short input vector sequence the circuit toggle information, simulated under time model was gathered by these two methods. It showed that they give same total toggle count for each simulation. The zero delay method produces however, far too huge list files for the timing simulations, since there is thousands transition in different points of time, each given own line it the file. Script executing time would decrease the need for both hardware power and time.

4.4 Power Measurement Tool

Estimation and measurement of power dissipation of a design is important on every level of digital circuit construction. Synopsys DesignPower tools are a complete set of methodology for low power design [31]. They can analyze the design for switching power, internal cell power and leakage power dissipation. Power analysis of gate-level design is done by looking on activity of nets in the design. Design Power has a dedicated command-line interface called dc_shell which allows the user to access the DesignPower's power analysis features.

The switching activity can originate either from Register-Transfer Level (RTL) or Gate-Level simulation of the design, performed in some kind of HDL simulator. Some or all of the nets of a design can annotate their switching activity from full-timing gate-level simulation. To facilitate the exchange of information between the simulators and Synopsys Power an ASCI format is applied. The SAIF format is used in representing the forward- and/or back-annotation file (See bottom part of Figure 4.1)

Design Power calculates switching power for each gate, as shown on Equation 3.3, using the switching activity information from the simulation flow. The C_i of each net is obtained from a wire load model, parasitic file and from the technology library information for the gates connected to the net. Technology library characterizes also the information needed to compute the internal power of each cell. Cells consumes different amount of power depending on transistor input or state of the cell. Based on the input toggle rates, input transition times and output load capacitances, DesignPower access the lookup table for deriving the consumed power.

Sum of switching power consumed by each net, together with internal power of every net (power dissipated by internal capacitances of a gate [31]), is reported as dynamic power dissipation. The report gives power consumption expressed in mW (milli Watt).

As shown on Figure 4.1, the simulation tool dumps the data generated under each logic simulation into a VCD file. Synopsys Power cannot however read VCD files, just the SAIF files. Because these two file formats are equivalent, VCD files can be converted to SAIF in dc_shell using a utility called vcd2saif. The SAIF files are then used in the power analysis of the net list in the Synopsys Power.

Design Power is able to read the technology library, the net-list, the wire loads and annotation files and report the power consumption by evoking the power report command. Power is reported as the cell internal power and net switching power. These two numbers stands for the dynamic power dissipation of the simulated and annotated circuit.

4.5 Script Development

Simulation results gathered during the experiments had to be adapted to extract the wanted information. To do that some scripts were developed. Python was chosen because it has its own built-in memory management and good facilities for cooperating with other programs. It is quite simple syntax makes script development en easy task with no place for errors.

Programming scripts in C language has been considered as well. However it is more complicated language and Python is preferred. Scripts developed in this thesis work are presented in Appendix C.

Chapter 5

Results

Results of the experiments executed for purposes of this thesis are presented in this chapter. The experiments are carried out in sequences to fulfill the general principles of the Monte Carlo technique as described in Section 3.2.3. Abbreviations used in the tables of this chapter and later in the discussion, are presented in Table 5.1.

Each circuit presented in next section is simulated eight times under each PVT condition with input vector sets of different size. The stimulus sets are identical for all simulations and all the input vectors used in the experiments are random.

The average switching characteristics and power consumption of the circuits are dependent on the process, voltage and temperature parameters (PVT). Measurements are carried out under two different PVT circumstances, called PVT-MIN and PVT-MAX.

The PVT-MIN corner presents faster process with working temperature of -40°C and supply voltage of 1.95V while the PVT-MAX corner stands for 100°C with supply voltage of 1.6V. This are the parameter settings used when the technology library and timing characteristics were created.

The theoretical multipliers (TM-) are simulated with the zero-delay model and with the fan-out delay model. The technology dependent multipliers (MM-) are simulated with zero-delay models and with real-delay model. The real-delay model is supplied by Atmel Norway in form of a SDF files, one for each PVT corner.

The total number of switches on the internal nets of the circuit, under both delay modes, is recorded by the simulation tool. The extracted outcome of these experiments is shown in Tables A.2, A.3 and A.4 in Appendix A.

The more interesting information is however how many times a net shifts its value during the execution of multiplier operation under given circumstances. Results are shown in Table A.5, A.6 and A.7 in Appendix A.

For each simulation the power dissipated by circuit is recorded. Results are exposed in the Table A.8 and A.9 in Appendix A

Tables in this chapter show the mean values experiments outcome, together with the standard deviation. It is assumed that these values represent the average power and toggle count of each of the investigated circuit in the given PVT corner. The standard deviation indicate the accuracy of the presented result.

5.1 Circuits Under Test

The circuits investigated in this thesis are 32-bit multipliers. All the circuits have tree structure which is considered the most power efficient kind for construction for this type arithmetic circuit [21]. Totally 8 different circuit candidates are studied here. The abbreviations used in the experiment presentation and discussion are presented in Table 5.1.

Three of the investigated multipliers are supplied by Saeeid T. Oskuii who has been working with low power multiplier design in his Ph.D. thesis *Design of Low-Power Reduction-Tree in Parallel Multipliers* [21]. His method produces parallel multiplier with power optimized partial product reduction tree (PPRT). These multipliers are represented by technology independent net-lists with a fan-out delay model. Three candidates are represented in this thesis, a power optimized circuit built to consume least possible power, random generated circuit and a circuit that correspond to worst case power scenario.

Another set consist of five multipliers supplied by Atmel Norway. All of the circuits are mapped into technology used by Atmel Norway, a 0,18/0,15u CMOS process. One of multipliers is a Synopsys DesignWare (DW02) multiplier with a pparch tree structure, investigated in [13]. This multiplier represents the industrial approach of arithmetic circuit design for the given technology. It is tested for the same premises as all the other multipliers to compare the research outcomes with the industrial approach.

Three of above mentioned multipliers are built on the basis of theoretical multipliers from [21]. They are synthesized and laid in the same process as the DW02 multiplier from [13]. Same component library is used here as well to make the competition of experimental results easier to comprehend.

Last candidate is represented by a multiplier generated by Module Generator. This is software that generates HDL net-lists of high performance digital arithmetic circuits. It has been developed by a Master's student, Espen Sand under supervision of Johnny Pihl. The net-list is generated and then mapped to the same technology as the other multipliers.

All the circuits supplied by Atmel Norway are represented by a Verilog net-list with corresponding technology library. Wire loads and all the parasitic needed for the power estimation are provided to make computation of dynamic power dissipation

1	TM-MIN	Power optimized multiplier
2	TM-MAX	Worst case power optimization
3	TM-RAND	Random built multiplier
4	MM-MIN	Power optimized multiplier synthesized by Atmel Norway
5	MM-MAX	Worst case power optimization synthesized by Atmel Norway
6	MM-RAND	Random built multiplier synthesized by Atmel Norway
7	SMP	Multiplier used by Atmel Norway in the industry
8	MG	Multiplier built by ModGen software tool

Table 5.1: Abbreviations used in result presentation

possible. All the circuits are also equipped with the matching timing characteristics corresponding to each PVT for timing simulation requirements.

5.2 Simulation Length

Measurement of the power and switching characteristics are very sensitive to the input pattern of the simulation, despite the delay model. Regardless of this, when working for some amount of time the average power and toggling characteristics are obtained.

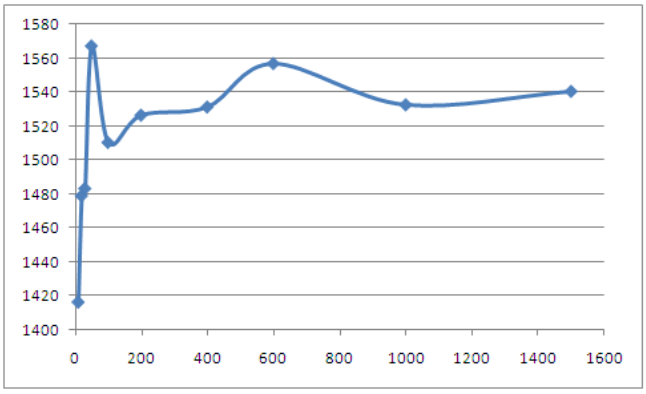

Figure 5.1: Number of consecutive operations in Zero Time mode

Figures 5.1 and 5.2 show how the switching activity in terms of toggles per multiplication operation, depends on the length of work period. The X-axes shows the number of consecutive operations of the circuit while the Y-axes represent the measured total toggling per operation. This way the length of simulation period that gives the average result, can be decided. The figures are based on measurements of the SMP multiplier presented in Table A.1 in Appendix A.

As given on the figures, when simulating with random vector sequences the circuit switching activity decreases, if sequence length increase, for the zero time simulation.

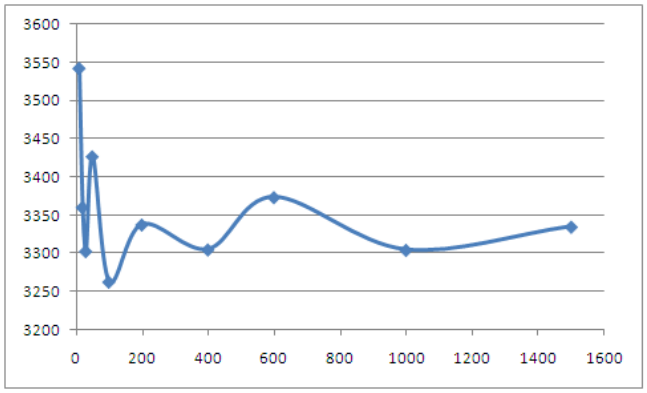

Figure 5.2: Number of consecutive operations in Real Time mode

In case of time modeled simulation the toggling activity per multiplication is much lower than the average, and converges to the average toggle count when sequence length decreases. In both cases the line flattens out when length of simulation overcomes 200 consecutive vectors. This shows that when simulating for too short amount of time, the toggling count would be fault and give too low number of toggles per operation in zero time delay, and too high in timing simulation.

Knowing the minimal length sampling region is an important result for Monte Carlo technique requirement. The further experiment can be carried out with simulation time above this limit. This way the average toggle count can be obtain within approvable error limit.

The experiment shows that it is adequate to simulate the circuits with sequence of about 200 or more vectors. This way the simulation time as well as the computation requirement is reduced, while the result lies within acceptable discrepancy.

To do the further experiments the lengths of individual samples, as well as number of these samples, have to be chosen. It has been decided that sample lengths would be 100, 200, 400 and 600 samples to see if there is difference in power and toggling activities for different lengths, as well as four samples of 300 vectors to investigate the pattern variation problem. Totally, each circuit is simulated eight times under each PVT condition, both in real delay mode and in zero delay mode. Set with 100 vectors is used to ensure that the sampling limit is correct for all the circuits, the measurements are however not taken into consideration when reporting the circuit characteristic in terms of average switching and power. As shown on Figure 5.3 the outcome of these eight simulations of all the given multipliers gives a steady toggle count per multiplication.

Another effect visible on these figures is that the input sequence influence on the number of toggling despite the delay model or even the circuit. Curves on all the figures in this section follow the change in the amount of toggling. That proves that

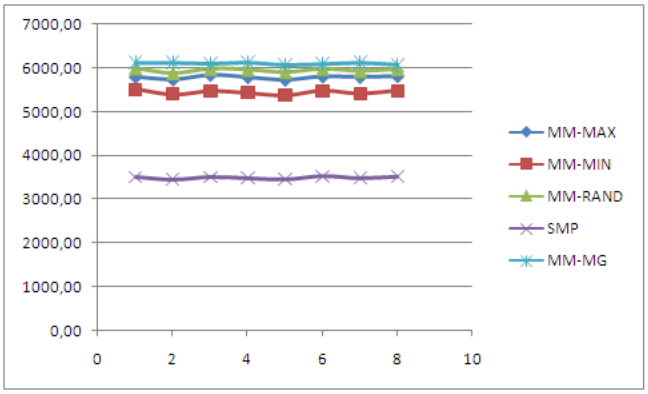

Figure 5.3: The average toggle count of all the MM multipliers in proportion to simulation order based on Table A.7 (PVT-MAX)

the amount of spurious toggling is in fact dependent on the input sequence, but the percentage of unnecessary transitions is assumed to be a quite constant number.

5.3 Switching Characteristics

TM- multipliers

Table 5.2 shows switching of theoretical circuits (TM) supplied by Saeeid T. Oskuii. Circuits are simulated with the fan-out timing model to estimate the total switching of internal circuit nets.

		TM-MIN	TM-MAX	TM-RAND
Timing model	Mean.	8579.22	9773.5	9689.07
	Std. Dev.	59.01	63.52	81.79
Zero-Time model	mean	2161.89	2206.66	2189.5
	Std. Dev.	13.80	10.38	12.61
Spurious Toggling	Mean.	6417.33	7566.84	7499.57
	Std. Dev.	45.21	53.43	70.49
% Spurious Toggling		74.8%	77.4%	77.4%

Table 5.2: Average toggling per multiplication in theoretic multipliers

This simulation results in Table 5.2 shows that the gain of proposed power optimization is about 12.12% comparing the best case circuit (TM-MIN) with the worst case multiplier (TM-MAX) and 11.49% when comparing with the random generated circuit (TM-RAND).

The given delay model shows however that the spurious switching stands for well over 70% of overall switching activity in the circuits. As expected the TM-MIN

has the best switching properties, including the spurious toggling, while in the other multipliers this effect is a constant number.

The fan-out model gives the approximate indication of the timing properties of the circuit. The next section presents the resulting the switching activity after mapping these circuits into industrial technology and how this stands comparing to the fan-out model.

MM-multipliers

Tables in this section show the switching activity of technology dependent multipliers supplied by Atmel Norway. Table 5.4 shows the results of the experiments performed for the MAX PVT corner, while the 5.3 shows the MIN PVT corner.

		MM-MAX	**MM-MIN**	**MM-RAND**	**SMP**	**MG**
Timing model	Mean	5777.07	5440.68	5943.84	3495.54	6095.57
	Std. Dev.	30.56	33.32	33.95	21.44	21.94
Zero-Time model	Mean	2064.04	1967.55	2057.64	1685.49	1519.64
	Std. Dev.	10.61	10.55	9.92	8.74	3.92
Spurious Toggling	Mean	3713.03	3473.13	3886.20	1810.04	4575.93
	Std. Dev.	21.54	23.07	24.83	14.25	18.65
% Spurious Toggling		64.27%	63.84%	65.38%	51.78%	75.07%

Table 5.3: Average toggling per multiplication of technology mapped multipliers (PVT MIN)

		MM-MAX	**MM-MIN**	**MM-RAND**	**SMP**	**MG**
Timing model	Mean	4760.14	4440.39	4994.47	2707.44	5022.73
	Std. Dev.	27.06	25.24	27.44	13.88	24.87
Zero-Time model	Mean	2064.04	1967.55	2057.64	1685.49	1519.64
	Std. Dev.	10.61	10.55	9.92	8.74	3.92
Spurious Toggling	Mean	2696.10	2472.84	2936.83	1021.94	3503.09
	Std. Dev.	17.36	14.90	18.32	6.97	21.58
% Spurious Toggling		56.64%	55.69%	58.80%	37.75%	69.74%

Table 5.4: Average toggling per multiplication of technology mapped multipliers (PVT MAX)

The effect of different PVT corners is visible when comparing these two tables. All the circuits both switch least and have the best spurious toggling properties in the PVT MAX corner. This corner represents much higher temperature (100°C) but at the lower supply voltage (V=1.6V). Comparing with the other corner which

represents much lower temperature (-40°C) the total switching is around 16-18% higher for the MIN conditions, thus the spurious toggling also.

As the circuits built in the CMOS technology work slower at the high temperatures, the signals travel time is slower. The charging/discharging times of the capacitive loads increase in higher temperature. This may cause that several glitches are filtered and less is generated, hence less unnecessary transitions is present during the working period. As the result, the lower switching activity is recorded in this PVT corner.

This means that the spurious switching is in fact reduced in higher working temperature for the same circuit. This is an important observation because even thou high temperature makes CMOS circuit work at slower ratios, the switching activity is clearly reduced.

The difference between the corners is not a fast number for all the investigated multipliers (See Table 5.5). This means that the way in which the circuit is constructed makes switching more or less fragile to the parameters like temperature differences. This is especially visible when comparing the MM multipliers with the SMP candidate. The three MM circuits are built in the same way, while the SMP multipliers have some different construction approach.

	MM-MAX	**MM-MIN**	**MM-RAND**	**SMP**	**MG**
$\frac{PVTMAX}{PVTMIN}$	0.82	0.82	0.84	0.78	0.83

Table 5.5: The proportion between PVT MIN and MAX

When comparing the circuit versions with each other, it is clear that the MM-MIN multiplier switches least among MM circuits. It records also best spurious switching characteristic. This is an expected result since it is built on a base of the multiplier which has been constructed to be power efficient.

The switching reduction gain in building circuit by method given in [21] before technology mapping, lie in around 11% (PVT MAX) and 8,5% (PVT MIN) comparing to the MM-RAND. Again the higher temperature gives better switching properties of the same circuit. The amount of spurious toggling is also reduced for the MM-MIN multiplier. The power optimized circuit have 5% (PVT MAX) to 2% (PVT MIN) decrease in amount of spurious toggling.

Results show that the power optimization proposed by [21] gives in fact reduction in switching activity and amount of unwanted hazards, compared to non-optimized circuit. Experiments show that this technique gives promising results but still more research are needed.

The circuit that stands out is multiplier supplied by Atmel Norway (SMP). Reason

for that is that this circuit net-list is built by a different tool than the rest of the circuits. It is possible that it takes parameters such as temperature into consideration when synthesizing the circuit, while the other net-lists do not take these factors into consideration. SMP circuit has also the best toggle characteristics among the investigated circuits. The amount of spurious toggling lies in the range of 38% (PVT MAX) and 52% (PVT MIN) of the total toggling needed to execute the multiplication operation, much less that for the other candidates. Several things can lead to this. The multiplier supplied by Atmel might have a different partial product generation method. It can use some kind of Booth recoding or similar method. The number of partial product generation steps is significantly smaller for Booth based multipliers (up to 50%). The difference in PP reduction technique have also influence on power consumption and switching characteristics. Unfortunately the documentation of the DW2 multiplier does not go into details of circuit construction. It is however steel desirable to scale down the amount of transitions without any useful application to make this circuit switch even less.

Last investigated circuit is the MG multiplier. It has the poorest switching properties in timing simulation among the investigated gate-level designs. It switches about 11% more than the MM-MIN multiplier despite the PVT corner. This shows that the amount of its switching is more dependent on the internal construction rather than the PVT fluctuations.

When looking at the results from circuit simulations with the zero delay model, the switching activity per operation is a quite constant number, ling within a ten percent range for both technology depended (Table 5.4 and 5.3) and independent case (Table 5.2). The difference between the MM- and TM- versions of multipliers simulated in zero delay mode, are shown in Table 5.6.

	MIN	MAX	RAND
MM- vs TM-	8.99%	6.46%	6.02%

Table 5.6: Difference in switching activity between the technology dependent and independent version in zero delay mode

This is an expected result since the most of toggling happens due to unbalance paths inside the circuit. This is not a case in zero delay mode since all the signals go through the nets in the same time and all the paths have same length, thou there is no difference in arrival times.

The interesting result is given by the MG multiplier in the zero delay simulation. The experiments show that it needs least logic state changes necessary to produce the multiplication result. This means that the ModGen software is able to create circuits that has very good switching characteristics when timing dependencies are

not taken into consideration.

Differences between the amount of switching in TM- and MM- are mostly due to the different timing models used under timing simulation, but also the way in which the MM circuit were synthesized. To make the circuits better suited for the technological circumstances, some registers have been introduced on the input and output of the circuits. Although these registers are bypassed during the simulations, their presents can still have some effect on the final result. The registers consist of gates switching together with other nets and are an integral part of the circuit, hence they can introduce some different path delay.

Net activity is strongly dependent on the model used to represent the timing properties of the circuit. Fan-out model is quite simple and it does not take all the timing properties of the circuit elements into consideration. This is main reason of the difference between the same type TM and MM multiplier versions, the technology mapped circuits use much more complicated and advanced circuit timing model.

5.4 Power Consumption

		MM-MAX	**MM-MIN**	**MM-RAND**	**SMP**	**MG**
Timing Model	mean	21.28	20.07	21.76	10.96	18.65
	std.dev.	0.11	0.12	0.12	0.06	0.06
Zero Time Model	mean	7.38	7.6	7.57	6.12	5.98
	std.dev.	0.03	0.04	0.03	0.03	0.02
Spurious Toggling	mean	13.9	12.48	14.19	4.84	12.67
	std.dev.	0.08	0.09	0.09	0.04	0.04
% Spurious Toggling		65.32%	62.16%	65.20%	44.19%	67.95%

Table 5.7: Power consumed by the multipliers (PVT MIN)

		MM-MAX	**MM-MIN**	**MM-RAND**	**SMP**	**MG**
Timing Model	mean	11.63	10.93	12.15	5.84	10.37
	std.dev.	0.06	0.06	0.06	0.03	0.04
Zero Time Model	mean	5.00	4.87	5.00	4.06	3.92
	std.dev.	0.02	0.02	0.02	0.02	0.01
Spurious Toggling	mean	6.63	6.06	7.15	1.78	6.45
	std.dev.	0.04	0.04	0.04	0.01	0.04
% Spurious Toggling		56.99%	55.47%	58.87%	30.44%	62.22%

Table 5.8: Power consumed by the multipliers (PVT MAX)

The analysis of power dissipation confirms that the power consumption is considerably different for the zero delay and real delay simulation model. This difference is

in fact the power consumed by unnecessary transitions discussed in previews section.

Table 5.7 and 5.8 show the average power consumption of the circuits in different PVT corners. The power collected in the both corners is measured with the switching activity annotated during the gate level simulations. In fact, toggling characteristics from the previews section are used to report power dissipated by the investigated multipliers. As expected, power dissipated by unnecessary transitions is a significant part of total dynamic power consumption inside the circuits.

Power dissipated by each circuit is strongly dependent on internal and external circumstance. Equation 3.3 shows that the power is proportional to the square of supply voltage and the net activity. So these two factors influence the power dissipation of the circuits, thus the differences between corners. As shown in previews section the net-activity is strongly dependent on the PVT factors, which also influences the power dissipation.

The difference between the corners is much bigger than the switching activity implies as shown in Table 5.9. The difference is in fact quite a constant number when looking at power usage.

	MM-MAX	**MM-MIN**	**MM-RAND**	**SMP**	**MG**
$\frac{PVTMAX}{PVTMIN}$	0.55	0.54	0.56	0.53	0.56

Table 5.9: The proportion between PVT MIN and PVT MAX

Table A.4 shows that power consumption is different for different vector sets at the input of the circuit. Standard deviation in the experiments lies within 1% off the mean value of the power consumed by the circuit. In consequence the result of combining the sufficient number of vectors (over 200 in this case) with several different approaches shows a good approximation of average power consumption.

The circuit that uses least power is the SMP multiplier. It dissipates considerably less power than the others multipliers, in fact almost half the amount. This is an expected result since this circuit has the smallest amount of toggles, shown in previous section.

The number of internal nets is actually smallest in the power optimized multiplier (MM-MIN) as shown in Table 5.10. It is in fact smaller than in the industrial multiplier from Atmel. This shows that power dissipation is dependent on how the circuit is built rather than number of cells and nets. Most power reduction is obtained by carefully designing the circuit rather than making it smaller.

Most of the power properties are preserved in the technology dependent version of the multipliers (MM) comparing to the theoretical assumptions as well as the switching characteristics. There is about 10% (PVT MAX) and 7,5% (PVT MIN) less power

	MIN	**MAX**	**RAND**	**MG**	**SMP**
Number of ports:	128	128	128	129	316
Number of nets:	5052	5236	5227	3663	5086
Number of cells:	4988	5172	5163	3597	4083

Table 5.10: Multipliers area information

dissipated by the power optimized multiplier (MM-MIN) that by the random generated version (MM-RAND). This is a significant improvement in power consumption attributes for the given design method.

This result corresponds to the switching characteristics of the circuits. In fact the percentage of spurious toggling corresponds with the power consumed by this effect, with exception of SMP and MG multiplier. This shows that the most dynamic power dissipated in these circuits comes from internal switching.

Power dissipation is strongly dependent on external factors like process and supply voltage, and the percentage of glitching power is closely related to the switching properties of the circuit despite some small differences. (Comparing Tables 5.4 and 5.3 with 5.8 and 5.7)

Nevertheless, the multiplier that has worst switching properties (MG) according to previews section dissipates less power than MM circuits. It has however more power dissipated in the spurious switching. These differences may be due to that not all the annotated transitions consumes same amount of power. The diverse gate types consume different amount of energy during a switch, thus differences in amount of gate types can also have influence on the overall result. Some of the dynamic power usage may be because of the circuit layout. Dynamic power is strongly dependent on the switching activities of the circuit. But the amount of the power may vary because of the technological aspects, as well as the way in which circuit is constructed. The layout and interconnection of the circuit can vary the result considerably as well as the fan-out characteristics of internal nets.

As expected the ModGen generated net-list uses least power to compute the result of multiplication, when the spurious toggling effects are not taken into consideration. One of reasons may be that it is the circuit with smallest amount of internal cells and nets as shown in Table 5.10. This tool has potential of creating multiplier net-lists that are in the same area of the total power properties as the TM circuits. To achieve that the spurious switching and its effect on power consumption has to be reduced, probably on expense of area of the design.

Chapter 6

Disscutions and Conclusions

6.1 Counting Method

The work of this thesis involves development of a method that performs characterization of toggling properties of a digital circuit given as a gate-level net-list. The developed method gives the total toggle count from timing logic simulation. When the counting of just the necessary transitions is performed as well, the amount of spurious toggling can be extracted.

The developed method can be used to get the total switching characteristic of an arbitrary circuit net-list, thus it became a practical tool for hardware engineers who want to decide the switching characteristics of their design. The software tools used for purposes of this thesis work are a commonly use in digital design, hence it is easy to implement purposed method. The attached step-by-step tutorial together with the counting scripts can be used to carry out a set of experiments, similar to these presented in Chapter 5.

6.2 Switching Characteristics

The goal of this thesis was to simulate and observe the effect of switching activities in a digital arithmetic circuit. In almost every digital circuit some amount of switching activity is considered unnecessary because it does not have any logical or functional purpose.

This report shows that theoretic model of circuit delay introduced in [21] can give a quite good indication of circuit switching characteristics. The experiment illustrates that the theoretical optimization does not necessarily mean that the outcome of technology mapping has the same properties. The way in which the circuit toggles depends on the timing model used in the simulation, thus the fan-out model is too simple to give the exact approximation of final switching properties. The outcome of

experiments executed in this thesis shows that more research is needed in that field.

The switching activity is also dependent on factors like temperature or supply voltage. That is why it is difficult to estimate the timing properties of a circuit when the final production technology is unknown. As some of the circuits, such as multipliers used in microcontrollers, are created before the final design application area is known, the model of timing is virtually impossible to approximate.

The results show that well over 60% of total dynamic power consumption in multipliers goes to this type of state transition called glitching. Amount of useful transition is independent on the external conditions, while the total toggling more influence by such factors. Thus the glitching effect is also dependent on PVT conditions. This means that these factors also must be taken into consideration when designing an arithmetic circuits.

The experiments show also that the multipliers have poor characteristic of avoiding spurious switching. This is partly explained by a structure of this kind of circuit. Much partial product calculation that are dependent on each other makes the internal nodes toggle couple of times before settling on the final value. The different path leading signals to the outputs makes the switching an unavoidable effect, thus it can be decreased by careful deign. The effects of lay-out influence has should also be included when designing digital multipliers.

The net-lists of theoretical multipliers from [21] are created to be technology independent. By making the reduction tree generator sensitive to the technological aspects can grant better post-synthesis results. The multiplier could also be created to aim at a previously defined technology library, thus measurements would not give so big differences in post and pre synthesis activity.

The experiment shows that digital IC industry has access to tools that give much better outcome in terms of general switching properties as well as spurious toggling. The synthesis tools used by Atmel Norway produce very good circuit net-lists with excellent switching characteristics. The example given in this thesis shows almost half as many switching as the other candidates. It is partially explained by the different construction and partial product generation method (like Both recoding).

It has been noticed that the multiplier generated by the ModGen tool uses much less necessary necessary necessary logical state changes then the other multipliers to compute the result. It suffers however from effect of spurious switching. This tool is clearly able to produce good quality net-lists, but more study is needed on its toggling characteristics handling.

6.3 Power Dissipation

Power consumed in the digital circuit is a strong function of its switching activity. This makes this area of research an important part of circuit design.

In the circuits built on basis of method from [21] the power consumed by transitions that do not bring any information to the computation, is about 60% of total power dissipation. Method of reducing this effect is an important part of research in low power circuit design. The results illustrates that the amount of power dissipated by this effect is very closely related to the switching characteristic.

As expected the SMP multiplier uses least power to perform a multiplication operation. The circuit that is switching most during the experiments (MG), uses however considerably less power in the proportion to other circuits. It uses also least power to compute the result in zero delay mode, as the switching characteristic imply. This means that the ModGen software is able to produce a good quality and power efficient multipliers, yet the amount of spurious switching should be deceased. Since it is the circuit that consists of fewest gates, thus it gives possibility of balancing the spurious toggling by increasing the circuit area. More research is needed in this area.

Method of creating a low power multipliers introduced in [21] gives decrease in overall power consumption. However, the circuit design tools used in the IC industry give more power efficient solutions. Models introduced in this thesis do not take all the circuit layout effects into account. Some of the problems may be avoided by proper path balancing on the physical level of the design.

The percentage of power dissipated by glitching is in accordance with the amount of spurious switching. This shows that circuit hazards are in fact main contributor to the overall dynamic power consumption. Clearly reducing internal glitching would reduce the overall power consumption of a digital circuit.

Reduction tree models (TM) consider only the effect of cell delay. On the layout level the paths can be balanced by proper place and route algorithms to avoid different signal arriving times. The interconnection length is important when minimizing the total physical capacitance of the circuit. The differences between physical and switching capacitances have to be considered. The wiring geometries and layer assignments are an important issue as well. Various optimization techniques can be used to partition, place, resize and route the design depending on design style, technology etc [26]. This can reduce glitching and its effect on power consumption.

6.4 Future Work

Delay model used in [21] shows the approximate value of internal switching. The more complex method of estimating how internal signals are propagated through

the circuit is needed. As the results show the zero-delay mode gives approximately the same amount of switching activities inside the theoretical as well as synthesized circuit. Difference in the timing mode indicates that the fan-out model is too simple.

Elaborating more realistic model of delay through the circuit is an important task for future work. This model would have to include more complex representation of gate delay that includes ambiguity model or a transition dependent model. This may give better approximation of switching activities in these circuits.

More research is also needed in gathering information about the synthesis tool used to create the technology dependent net-lists. Understanding of the aspect of optimization during synthesis is an important task when exploring possibilities of modeling the delay in complex arithmetic circuits.

ModGen software produces quite good results in comparing to the other solutions in terms of power efficient arithmetic circuit. The further development of ModGen software can however give even better results in terms of avoiding hazards and spurious switching.

Appendix A

Tables

A.1 Switching

	Zero time		Real Time	
Vectors	Total	Per Operation	Total	Per Operation
8	11330	1416.25	28341	20.01
18	26620	1478.89	60493	40.9
28	41529	1483.18	92490	62.36
48	75224	1567.17	164504	104.97
98	148013	1510.34	319791	211.73
198	302221	1526.37	660902	432.99
398	609404	1531.17	1315522	859.16
598	930962	1556.79	2017466	1295.91
998	1529434	1532.5	3298194	2152.17
1498	2307558	1540.43	4996025	3243.28

Table A.1: Simulation length

Real time	**TM-MIN**	**TM-MAX**	**TM-RAND**
98	845341	953950	957358
198	1678129	1909986	1889656
398	3435287	3927451	3889238
598	5134964	5846766	5792245
298	2537698	2884939	2861284
298	2573222	2930523	2917869
298	2544909	2911692	2872316
298	2583711	2931474	2917426
Zero time	**TM-MIN**	**TM-MAX**	**TM-RAND**
98	210128	214876	213176
198	421947	432054	428335
398	866553	884228	877372
598	1296144	1322106	1312064
298	639576	654074	647864
298	648502	659422	656806
298	643708	657744	651727
298	648140	660683	655463

Table A.2: Total toggling measured during experiments on TM multipliers

vectors	**MM-MAX**	**MM-MIN**	**MM-RAND**	**SMP**	**MM-MG**
98	202033	191731	200256	164099	148315
198	404644	385029	402570	330082	301711
398	827976	788703	824590	674870	605468
598	1236639	1176766	1234067	1011167	910836
298	611018	582165	610119	499130	451681
298	617912	590278	614723	504274	453016
298	614159	586559	614013	501795	454011
298	617303	588870	615117	504755	449939

Table A.3: Toggling measured in zero-time mode in the MM multipliers

PVT MAX					
vectors	**MM-MAX**	**MM-MIN**	**MM-RAND**	**SMP**	**MG**
98	466805	439556	490395	266149	493495
198	932346	871232	980044	531072	1000218
398	1914373	1773814	2000687	1081509	1995948
598	2848281	2656564	2993544	1614916	3020764
298	1405875	1309600	1477782	802182	1484436
298	1424605	1333988	1495075	811598	1496353
298	1418237	1322535	1483633	806565	1505559
298	1424955	1333305	1497182	813552	1485911
PVT MIN					
vectors					
98	567027	539944	587205	344240	599665
198	1134112	1069728	1164088	685744	1211400
398	2320185	2177832	2381761	1397988	2425946
598	3455553	3249974	3563836	2085595	3662726
298	1704369	1604860	1759106	1032867	1806060
298	1727449	1633534	1781583	1052011	1812663
298	1723377	1616131	1767267	1040062	1823871
298	1729663	1634555	1779596	1048626	1807907

Table A.4: Total toggling of TM multipliers in opposite PVT corners

Real time	**TM-MIN**	**TM-MAX**	**TM-RAND**
98	8625.93	9734.18	9768.96
198	8475.40	9646.39	9543.72
398	8631.37	9867.97	9771.95
598	8586.90	9777.20	9686.03
298	8515.77	9681.00	9601.62
298	8634.97	9833.97	9791.51
298	8539.96	9770.78	9638.64
298	8670.17	9837.16	9790.02
Mean	**8579.22**	**9773.50**	**9689.07**
std.dev.	**59.01**	**63.52**	**81.79**
Zero time	**TM-MIN**	**TM-MAX**	**TM-RAND**
98	2144.16	2192.61	2175.27
198	2131.05	2182.09	2163.31
398	2177.27	2221.68	2204.45
598	2167.46	2210.88	2194.09
298	2146.23	2194.88	2174.04
298	2176.18	2212.83	2204.05
298	2160.09	2207.19	2187.00
298	2174.97	2217.06	2199.54
Mean	**2161.89**	**2206.66**	**2189.50**
std.dev.	**13.80**	**10.38**	**12.61**

Table A.5: Toggling per operation of TM multiplier

vectors	**MM-MAX**	**MM-MIN**	**MM-RAND**	**SMP**	**MG**
98	2061.56	1956.44	2043.43	1674.48	1513.42
198	2043.66	1944.59	2033.18	1667.08	1523.79
398	2080.34	1981.67	2071.83	1695.65	1521.28
598	2067.96	1967.84	2063.66	1690.91	1523.14
298	2050.4	1953.57	2047.38	1674.93	1515.71
298	2073.53	1980.8	2062.83	1692.19	1520.19
298	2060.94	1968.32	2060.45	1683.88	1523.53
298	2071.49	1976.07	2064.15	1693.81	1509.86
mean	**2064.04**	**1967.55**	**2057.64**	**1685.49**	**1519.64**
std.dev.	**10.61**	**10.55**	**9.92**	**8.74**	**3.92**

Table A.6: Toggles per operation of MM multipliers in zero-time model

PVT MAX					
vectors	**MM-MAX**	**MM-MIN**	**MM-RAND**	**SMP**	**MM-MG**
98	4763.32	4485.27	5004.03	2715.81	5035.66
198	4708.82	4400.16	4949.72	2682.18	5051.61
398	4809.98	4456.82	5026.85	2717.36	5014.94
598	4763.01	4442.41	5005.93	2700.53	5051.44
298	4717.7	4394.63	4959	2691.89	4981.33
298	4780.55	4476.47	5017.03	2723.48	5021.32
298	4759.18	4438.04	4978.63	2706.59	5052.21
298	4781.73	4474.18	5024.1	2730.04	4986.28
mean	**4760.14**	**4440.39**	**4994.47**	**2707.44**	**5022.73**
std.dev.	**27.06**	**25.24**	**27.44**	**13.88**	**24.87**
PVT MIN					
98	5785.99	5509.63	5991.89	3512.65	6119.03
198	5727.84	5402.67	5879.23	3463.35	6118.18
398	5829.61	5471.94	5984.32	3512.53	6095.34
598	5778.52	5434.74	5959.59	3487.62	6124.96
298	5719.36	5385.44	5903.04	3466	6060.6
298	5796.81	5481.66	5978.47	3530.24	6082.76
298	5783.14	5423.26	5930.43	3490.14	6120.37
298	5804.24	5485.08	5971.8	3518.88	6066.8
mean	**5777.07**	**5440.68**	**5943.84**	**3495.54**	**6095.57**
std.dev.	**30.56**	**33.32**	**33.95**	**21.44**	**21.94**

Table A.7: Toggling per operation of MM multiplier

A.2 Power

Real Time	mW				
vectors	**MM-MAX**	**MM-MIN**	**MM-RAND**	**SMP**	**MM-MG**
98	11.64	11.03	12.18	5.87	10.42
198	11.51	10.83	12.04	5.78	10.41
398	11.75	10.96	12.22	5.86	10.36
598	11.65	10.93	12.18	5.83	10.42
298	11.53	10.82	12.07	5.80	10.29
298	11.68	11.01	12.19	5.87	10.37
298	11.64	10.92	12.11	5.84	10.43
298	11.68	11.00	12.21	5.88	10.30
mean	**11.63**	**10.93**	**12.15**	**5.84**	**10.37**
std.dev	**0.06**	**0.06**	**0.06**	**0.03**	**0.04**
Zero Time	mW				
vectors	**MM-MAX**	**MM-MIN**	**MM-RAND**	**SMP**	**MM-MG**
98	5.00	4.84	4.97	4.05	3.92
198	4.96	4.82	4.94	4.02	3.92
398	5.04	4.90	5.03	4.09	3.92
598	5.02	4.87	5.01	4.07	3.93
298	4.97	4.84	4.97	4.03	3.90
298	5.03	4.89	5.00	4.07	3.92
298	5.00	4.87	5.00	4.07	3.93
298	5.02	4.88	5.01	4.08	3.90
mean	**5.00**	**4.87**	**5.00**	**4.06**	**3.92**
std.dev	**0.02**	**0.02**	**0.02**	**0.02**	**0.01**

Table A.8: Power dissipated in PVT MAX

PVT MIN					
Real Time	mW				
vectors	**MM-MAX**	**MM-MIN**	**MM-RAND**	**SMP**	**MM-MG**
98	21.32	20.32	21.95	11.03	18.75
198	21.09	19.93	21.55	10.85	18.69
398	21.46	20.18	21.90	11.02	18.66
598	21.29	20.05	21.83	10.94	18.74
298	21.08	19.89	21.62	10.86	18.56
298	21.35	20.21	21.88	11.06	18.62
298	21.31	20.01	21.72	10.96	18.72
298	21.36	20.24	21.86	11.02	18.57
mean	**21.28**	**20.07**	**21.76**	**10.96**	**18.65**
std.dev	**0.11**	**0.12**	**0.12**	**0.06**	**0.06**
Zero Time	mW				
vectors	**MM-MAX**	**MM-MIN**	**MM-RAND**	**SMP**	**MM-MG**
98	7.34	7.59	7.53	6.10	5.98
198	7.30	7.52	7.49	6.05	5.98
398	7.43	7.65	7.62	6.15	5.99
598	7.38	7.62	7.60	6.14	5.99
298	7.33	7.55	7.54	6.07	5.95
298	7.41	7.63	7.58	6.13	5.98
298	7.39	7.59	7.58	6.12	6.00
298	7.40	7.62	7.59	6.14	5.94
mean	**7.38**	**7.6**	**7.57**	**6.12**	**5.98**
std.dev	**0.03**	**0.04**	**0.03**	**0.03**	**0.02**

Table A.9: Power dissipated in PVT MIN

Appendix B

Tutorial

B.1 Switching Activity

Switching activity is an important factor in power estimation of digital CMOS circuit. Power consumed by spurious switching is a magnificent part of power consumption inside these circuits. That is why this method of measuring amount of spurious toggling is developed.

Spurious toggling is may be found by comparing switching activity obtained from simulations of the circuit under two delay models. When simulating with zero-delay model all signals propagate instantaneously through the internal gates and every gate will get its final logical value right away. This way the only transitions that are present are so called useful transitions.

When some kind of delay model is used in the simulation the unbalanced paths and other factor makes signal toggle before settling to final value. If the model represents a real component library the total amount of switching is achieved.

The difference of the toggling obtained from these two simulations stands for spurious switching of the circuit. Most representative results are gathered when the circuit is simulated with same stimulus under both delay models.

B.2 Power dissipation

The power consumed by the unnecessary transitions can be found in the same way. By annotation the activity under both delays the power dissipated by spurious toggling can be computed. In order to measure the power in Synopsys Power Tools some more information about design is needed.

The target library representing the conditions under which circuit work must be supplied. This is often referred to as a PVT (process, voltage and temperature) corner. Information about parasitic of the circuit or some other RC information is

also required.

B.3 Device Under Test

In this tutorial we operate on a Verlilog net-list of a digital combinatorial circuit. The net-list includes a library of components together with their delay characteristics contained in a standard delay file (SDF). The SDF may include max:typ:min delay modes. The target library in DB format together with SPEF (Standard Parasitic Extraction Format) file used to report power. The flow of toggle count purposed in this tutorial may be used to count state changes in a VHDL net-list as well.

A good testbench is required as well. Testbench examples that read input data from a file are given in Section C.1. A script that creates file with random data input is presented in Section C.4.1

B.4 Simulations

B.4.1 Timing Simulation

In the simulation with timing model defined by SDF file a **vsim** command has a extension that allows to include delay file into the simulation flow. The `-sdfmax` is used here, but the `-sdftyp` or `-sdfmin` can be used if SDF include this.

```
vsim
-sdfmax {instance_path/circuit_instance = file.sdf}
work.your_testbench
```

This can be done in the *Simulation* menu, *Start Simulation*, *SDF* mark.

It is desirable to run circuit for some time to make all the signals toggle couple of times. This way we are shore that circuit is running at the typical rate.

```
#F.ex two clock periods
run 40ns
```

ModelSim has some built in functions which can count the switching of each net in the design. Switching activity is closely related to estimating the power consumption the power add (CR-184) is used to specify the nets or signals to track.

```
power add sim:/test_bench/circuit_instance/nets_or_signals_to_track
```

We need to annotate the circuit activity needed by Synopsys Power Tools to report the power consumption of the circuit. This is done by invoking a VCD file in the ModelSim environment and adding the user decided signals to be accounted for. This way the power dissipated by chosen nets or signals may be estimated and reported.

```
vcd file dump.vcd
vcd add sim:/sim:/test_bench/circuit_instance/nets_or_signals
```

Next step is to run the simulation for some amount of time. After it finish the number of transitions of each signal may be extracted by the power report command (CR-185). This can be either displayed in the main window or saved to a text file.

```
#extract to power_report file
power report -all -noheader -file power_report.log
```

To save the dump information to a VCD file we need to notice the tool that simulation is over

```
vcd checkpoint
```

After executing the simulation in this order the two files are saved. The power report format shows the name of the gate, toggle count, hazard count and the times spent at each state. They are described inside the file when the `-noheader` command in not used. By creating a script or pasting the report into a spreadsheet program the total toggle count is obtained.

The VCD file needs to be saved for the power measurement process.

B.4.2 Zero Delay

To simulate a circuit under zero delay conditions it is sufficient to use a straight foreword `vsim` command (CR-298):

```
vsim work.your_testbench
```

The method described in previews section shows to be insufficient when dealing with zero delay model. The reason is the way in which logic simulations deal with concurrent expressions.

VHDL simulators use an infinite short delay called delta delay unit. This makes the simulated signal toggle with zero times pulses. Even though all signals gets it final value in after zero time, the signal toggles are taken into account by the power function in ModelSim.

In Verilog simulation a similar concept exists in terms of Non-Blocking Assignment infinitesimally short delays and Blocking Assignments where the assignments are immediate. All this makes the toggle count commands I ModelSim deliver incorrect results

To overcome this counting problem the methods of saving waveforms are used. ModelSim can produce a list of all signal values and changes by adding the selected signals or nets to the list.

The useful option is that the signal value can be measure with the probe at the user chosen rate. If this rate is set to be the clock period just the useful toggles are showed in the list window. This can be done in the *List* menu by setting *Strobe Period* to the clock rate, *First Strobe* to 0, uncheck *Signal Change* and check the *Strobe Box*, uncheck *Signal Change* and check the *Strobe Box*.

This can be also done using list `config list` command (CR-110) in the command line:

```
config list -strobeperiod {clock_periode_in_ns}
-strobestart {0 ns}
-usesignaltriggers 0
-usestrobe 1
```

Then the selected signals can be added to list

```
add list sim:/test_bench /circuit_instance/nets_or_signals
```

After the simulation is ready the list can be exported to a TSSI format, either by item *Export* in file menu or by command:

```
write tssi file_name.lst
```

The TSSI format saves signal changes at different time marks in columns of the file. To get toggle count a script which counts each time the signal changed its value has to be developed.

B.5 Spurious Toggling

After extracting toggle count with the time delay model and zero delay model these two numbers can be compared. The difference between these two numbers is the not useful transitions, also called spurious toggling.

B.6 Reporting Power

Power dissipation can be measured by Synopsys DesignPower tool. It computes average power consumption based on net activity of the gate level design. Synopsys tool has a graphical interface called Design Vision (DV) which can be used instead of command line.

First we have to create a working directory for example MyFolder. In this directory an empty text file is created with name .synopsys_dc.setup. The file can be edited with a simple text editor and it should consist of following lines

```
set search_path /path/to_library/directory
set link_library name_of_library.db
```

This way when the Synopsys DesignVision is started, the right technology library setup is created. The library represents a process, voltage and temperature corner of circuit working conditions.

It is recommended that the design net-list file (netlist.v) together with the corresponding parasitic file (netlist.spef.gz) is saved into source directory as well.

The VCD files obtained in simulation tool have to be copied to the source directory as well, for example to a VCD folder. Synopsys PowerCompiler cannot read VCD files but uses another file format called SAIF (This Switching Activity Interchange Format). Because these two file formats are equivalent, vcd files can be converted to saif in `dc_shell` (command line interface of DesignVision) by following command

```
dc_shell
vcd2saif -input VCD/vcd_file.vcd -output SAIF/saif_file.saif
```

This way the corresponding SAIF file is saved in SAIF folder in source directory. All the VCD files from simulations have to be converted to SAIF format to be used by power analysis tool.

When all the files are in place the power analysis can be started. First the program has to be started in the source directory. By typing following in the Linux command line the graphical environment is opened.

```
design_vision
```

Then the design file can be red into software environment. It is done either in menu *File*, *Read* or by command

```
read_file - format verilog {path/netlist.v}
```

The parasitic information has to be read as well.

```
read_parasitics path/netlist.spef.gz
```

Next step is to annotate the switching activity in form of SAIF file

```
read_saif -input path/SAIF/saif_file.saif
 -instance_name path/test_bench/circuit_instance/nets_or_signals
```

Where the circuit instance is the same as in the test-bench used to simulate the design. This instant name is saved in the SAIF annotation file.

Last step is to report measured power either in menu Design or by following command

```
report_power >path/Power_report/power_report_file.rpt
```

The power report is saved and can be opened by an arbitrary text editor. It consist of some design information as well as the dynamic and static power measurements.

References:

ModelSim SE Command Reference, Software Version 5.6d, August 1992
ModelSim SE User's Manual, Software Version 6.2a, June 2006
Synopsys Power Products Reference Manual, v1999.10, October 1999

Appendix C

Code

C.1 Testbench

C.1.1 VHDL

VHDL testbench that reads input values generated by the random number generator in Section C.4.1.

```
process (clk)

      FILE file_in  : TEXT IS IN "stimuli300\300_vectors1.txt";
      VARIABLE line_in  : LINE;
      VARIABLE inputA, inputB  : bit_vector(31 downto 0);

      if (clk'event and clk='1') then

        IF NOT (ENDFILE(file_in)) THEN
           READLINE(file_in, line_in);
           READ(line_in, inputA);
           READ (line_in, inputB);

           inA <= inputA;
           inB <= inputB;

        A <= to_stdlogicvector(inputA);
        B <= to_stdlogicvector(inputB);

        ELSE
           ASSERT FALSE
             REPORT "End of file!"
              SEVERITY NOTE;
         END IF;
```

```
    end if;
```

C.1.2 Verilog

Verilog testbench that reads input values generated by the random number generator in Section C.4.1.

```
file = $fopen("stimuli300/300_vectors3.txt","r");

// If error opening file
if (file == 0)
   begin
      // Just quit and display
      $display("Error file open");
   end

//when not end of file
while(!$feof(file)) begin
   @(posedge clock)
   //scan file for two bit vectors with white space between
   c = $fscanf(file,"%b %b\n",line_A, line_B);
   A = line_A;
   B = line_B;
   end //while not EOF
$fclose(file);
```

C.2 Script: Timing Simulation

Script to be used in ModelSim to report all the transitions of the circuit nets.

```
#timing_som.do
vsim
-sdfmax {/multmax_ins=D:/Documents and Settings/Jakub/Mine dokumenter/MASTER/Atmel/max/ex_mul_l
work.tb_mul_max

#Circuit is going into working mode (2 vectors)
run 60ns

#Create the VCD file
vcd file vcd/rt3_298vec.vcd

#Add signals and nets to the VCD dump
vcd add sim:/tb_mul_max/multmax_ins/U_MUL/*
```

```
#Add signals to report logic state changes
power add sim:/tb_mul_max/multmax_ins/U_MUL/*

#run rest of vectors (#vectors-2 * 20ns)
run 5960 ns

#dump values to file
vcd checkpoint

#report power consuming state changes
power report -all -noheader -file power_report/rt3_298vec.log

#end simulation
quit -sim
```

C.3 Script: Zero Time Simulation

Script to be used in ModelSim to report just the necessary transitions.

```
#zero_time.do
vsim work.tb_mul_max

#to count the switching without delta delay strobe is used
#strobeperiod is a clock period
config list -strobeperiod {20 ns} -strobestart {0 ns} -usesignaltriggers 0 -usestrobe 1

#add signals to annotate the switching
add list sim:/tb_mul_max/multmax_ins/U_MUL/*

#Circuit is going into working mode (2 vectors)
run 60ns

vcd file vcd/zt3_298vec.vcd
vcd add sim:/tb_mul_max/multmax_ins/U_MUL/*

#run rest of vectors (#vectors-2 * 20ns)
run 5960 ns

#dump values to file
vcd checkpoint

#save the switching into tssi file
write tssi tssi/zt3_298vec.lst
```

```
quit -sim
```

C.4 Python Scripts

C.4.1 Random Number Generator

Python script that generates the random stimulus.

```
#generate a (vec_nr) pair of random numbers
for i in range(0, vec_nr):
    for j in range(0, 32):
     #sets either 1 or 0 into a 32-bit vector
        ran = random.randint(0, 1)
        f.write("%d" %(ran) )

    #makes space between vectors
    f.write(" ")

#makes second vector in the same line
    for j in range(0, 32):
        ran = random.randint(0, 1)
        f.write("%d" %(ran) )
#jumps to next line
    f.write("\n")
```

Example of a stimulus file:

```
11000100010000100000000100001111 00001010100000000000110001000011
11000000111011100111010111011101 11001110001100011001000101100001
00000111000101100111001100000010 00001011010000010100100010010000
11001000010010111000101000000000 01000110000111110000001000101101
01100001110010000001001000001110 01000110010000011010010000001101
(...)
```

C.4.2 Toggle Count from TSSI List File

Counts the transitions in a TSSI list file

```
file1=open('try.lst', 'r')

#reads first three lines
file1.readline()
file1.readline()
file1.readline()

#Decide how many characters there is in one line
```

```
for line in file1.readline():
        # Splits the line into characters
        char = line.split(' ')
        # makes an array with each character of the line
        tmp1.append(char)
        tg.append(0)
        #number of character in one line
        i=i+1

#go through the file (max 1500 lines)
for k in range(1500):
        #read line next line
        i=0
        for line in file1.readline():
                char = line.split(' ')
                tmp2.append(char)

                i=i+1

        #checks if line has changed
        for j in range(i):
                #print j
                #checks if the number in same column are same.
                #if so, the number in this column is marked
                if tmp1[j]==tmp2[j]:
                        tg[j] = tg[j]
                else:
                        tg[j] = tg[j]+1

        tmp1 = tmp2
        tmp2 = []

#sums the signal changes for each node
#first 20 lines stands for input values
sum = 0
if len(tg) > 20:
        for cnt in range(20, len(tg)-1):
                sum = sum + tg[cnt]
#Prints total number of logic state transitions
print "sum = %d" % sum

file1.close()
```

C.4.3 Toggle Count from Power Report

Reads and sums the logic state changes in the power report from ModelSim

```
num = 0

#Opens file
file=open('min98vec.log', 'r')
#first line is bypassed
file.readline()

sum=0
for line in file.readlines():
      #splits line into array
      line_str = line.split()
      #if not end of file
      if line_str != []:
              num = int(line_str[1])
              #sums the switching
              sum = sum + num
print sum
file.close()
```

Bibliography

[1] Abramovici M., Breuer M. A., Friedman A. D. (1990) *Digital Systems Testing And Testable Design*, New York. IEEE Press.

[2] Bahanja S., Raghunatan A. (2003) Switching Activity Estimation of VLSI Circuits Using Bayesian Networks, *IEEE Trans. Very Large Scale Integr. Syst.*, 11(4), 558-567

[3] Booth A. D., (1951) A signed binary multiplication technoique, *Quarterly J. of Mechanics and Applied Mathematics*, 4(2), 236-240

[4] Burch R., Najm F., Yang P. and Trick T. (1993), A Monte Carlo Approach for POwer Estimation, *IEEE Trans. Very Large Scale Integr. Syst.*, 1(1) 63-71

[5] Chandrakasan A. P. and Brodersen R. W. (1995), *Low Power Digital CMOS Design* Boston: Kluwer Academic Publishers

[6] Chandrakasan A. P. and Brodersen R. W. (1995), Minimizing Power Consumption in Digital CMOS Circuits. *Proceedings of the IEEE*, 83(4), 498-523

[7] Churiwala S. (2009) *Delta Delay, Clock Data Race*, avalible at http://knol.google.com (http://knol.google.com/k/sanjay-churiwala/delta-delay/2h67vjqmw58bp/5#)

[8] Dadda L. (1965) Some shemes for parallal multipliers, *Alta Frequenza*, 45, 574-850

[9] Daga, J.M. Ottaviano E. Auvergne, D, (1998) Temperature effect on delay for low voltage applications [CMOS ICs], *Design, Automation and Test in Europe* 680-685

[10] Ding C., Tsui C. and Pedram M. (1998), Gate-Level Power Estimation Using Tagged Probabilistic Simulation, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems* 17(11) 1099-1107

[11] Ericsson H. Larsson-Edefors P. (2004) Glitch-Conscious Low-Power Design of Arithemtic circuit, *IEEE International Symposium on Circuits and Systems* 281-284

[12] Hu F. and Agrawal V. D., (2005), Dual Transition Glitch Filtrering in Probabilistic Waveform Power Estimation, *Proc. 15th Grat Lakes Symp on VLSI*, 357-360

[13] Kalis J. (2008) *Switching in Multipliers*, project report in TFE 4520, Depertament of Electronics And Telecomunication, Norwegian University of Sience and Technology (NTNU), Trondheim, Norway

[14] Leijten J., van Meerbergen J. and Jess J. (1995). Analysis and Reduction of Glitches in Synchronous Networks, *Proceedings og the 1995 European Design and Test Conference.*

[15] Maini A. K. (2007). *Digital Electronics: Principles, Devices and Applications*, New York, John Wiley & Sons, Inc.

[16] Meister G. (1993), *A Survey on Parallel Logic Simulation*, University of Saarland, Department of Computer Science, Misra J

[17] ModelSim SE User's Manual, Software Version 6.2a, June 2006

[18] Najm F. (1990) Transition density: A new measure if activity in digital circuits, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 12(2), 310-323

[19] Najm F. (1995) Power Estimation Techniques For Integrated Circuits, *IEEE/ACM International Conference on Computer Aided Design*, 492-499

[20] Najm F., Burch R., Yang P. and Hajj I. (1990) Probabilistic Simulation for Reliability Analysis of CMOS VLSI Circuits, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Sytsems*, 9(4), 439-450

[21] Oskuii S. T. (2008) *Design of Low-Power Reduction-Trees in Parallel Multipliers*, Depertament of Electronics And Telecomunication, Norwegian University of Sience and Technology (NTNU), Trondheim, Norway

[22] Chen O. T.-C., Wang S., and Wu Y.-W. (2003) Minimization of Switching Activities of Partial Products for Designing Low-Power Multipliers, *IEEE Trans. Very Large Scale Integr. Syst.*, 11(3), 2003

[23] Parhami B. (2000), *Computer Arithmetic, Algorithms and Hardware Design*, New York, Oxford University Press

[24] Pedram M. (1996). Power Minimization in IC Design: Principles and Applications, *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 1(1), 3-56

[25] Pedram M. (1999). *Power Simulation and estimation in VLSI circuits* W-K. Chen ed., CRC Press and IEEE Press, 18-1–18-27

[26] Massoud Pedram and Hirendu Vaishnav (1997). Power optimization in VLSI layout: A survey, *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, 15, 221-232

[27] Piguet C. (2007) Low Power Design in Deep Submicron 65 & 45 nm Technologies, *Electronics, Circuits and Systems, 2007. ICECS 2007. 14th IEEE International Conference*, 915-918

[28] Raghunatan A., Dey S. and Jha N. K. (2003) High-Level Modeling and Estimation Techniues for Switching Activity and Power Consumption, *IEEE Transactions on VLSI Systems*, 11(4), 538-557

[29] Sayed A., Al-Asaad H. (2007) A New Statistical Approach for Glitch Estimation in Combinational Circuits, *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, 1641-1644

[30] Soeleman H., Roy K. and Chou T. (2000). Estimating Circuit Activity in Combinational CMOS Digital Circuits. *IEEE Design & Test of Computers*, 17(2), 112-119

[31] Synpopsys (2008) *Power Products Reference Manual* Downloaded 20. October, 2008., from www.synopsys.com

[32] Svendsli O. J. (2003) *Atmel's Self Programming Flash Microcontrollers*, Atmel Corporation, San Jose, from www.atmel.com

[33] Standard Delay Format Specification, Version 3.0, May 1995

[34] ASIC World, (1998-2009) Deepak Kumar Tala, http://www.asic-world.com/

[35] Uyemura John P. (2002) *Introduction to VLSI Circuits and Systems*, New York, John Wiley & Sons, Inc.

[36] Walpole R. E., Myers H. R., Myers S. L., Ye K. (2002) *Probability & Statistisc for Engenieers & Scientists*, New York, Prientice Hall

[37] Wallace C. S. (1964) A suggestion for fast multiplyer. *IEEE Transactions on Electronioc Computers*, 12-17