P-3476

DESIGN OF LOW-POWER HIGH-SPEED 32-BIT TRUNCATION-ERROR-TOLERANT ADDER

By

MANEESHA.V.P

Reg. No. 1020106012

of

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution affiliated to Anna University, Coimbatore)

COIMBATORE - 641006

A MINI PROJECT REPORT

Submitted to the

FACULTY OF ELECTRONICS AND COMMUNICATION ENGINEERING

In partial fulfillment of the requirements

for the award of the degree

of

MASTER OF ENGINEERING

IN

APPLIED ELECTRONICS

MAY 2011

BONAFIDE CERTIFICATE

Certified that this project report entitled "**DESIGN OF LOW-POWER HIGH-SPEED 32-BIT TRUNCATION-ERROR-TOLERANT ADDER**" is the bonafide work of Mrs.Maneesha.V.P [Reg. no. 1020106012] who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Project Guide

Prof.K.Ramprakash

Head of the Department

Dr. (Mrs.) Rajeswari Mariappan

The candidate with university Register no. 1020106012 is examined by us in the project viva-voce examination held on 23-5-2011

Internal Examiner

External Examiner

ACKNOWLEDGEMENT

I express my profound gratitude to our director **J.Shanmugham.** for giving this opportunity to pursue this course

At this pleasing moment of having successfully completed the project work, I wish to acknowledge my sincere gratitude and heartfelt thanks to our beloved Principal **Prof.Ramachandran,** for having given me the adequate support and opportunity for completing this project work successfully.

I extend my heartfelt thanks to my internal guide **Prof.K.Ramprakash,** for his ideas and suggestion, which have been very helpful for the completion of this project work. His careful supervision has ensured me in the attaining perfection of work.

I express my sincere thanks to **Dr.Rajeswari Mariyappan Ph.D.,** the ever active, Head of the Department of Electronics and Communication Engineering, who rendering us all the time by helps throughout this project

In particular, I wish to thank and everlasting gratitude to the project coordinator **Asst.Prof.R.Hemlatha**, Department of Electronics and Communication Engineering for her expert counseling and guidance to make this project to a great deal of success.

Last, but not the least, I would like to express my gratitude to my family members, friends and to all my staff members of Electronics and Communication Engineering department for their encouragement and support throughout the course of this project.

ABSTRACT

Error tolerance is an emerging concept in VLSI design and testing. Error tolerant system contains a circuit with some internal or external errors and which produces acceptable results with accuracy greater than the minimum acceptable accuracy. The threshold value of accuracy should be selected in such a way so that the output should meet the requirements of the whole system. Based on error tolerance, an error tolerant adder has been designed. Error Tolerant Adder (ETA) is able to ease the strict restriction on accuracy at the same time achieve tremendous improvement in both power dissipation and speed performance. When compared to conventional counterparts, Error Tolerant adder is able to attain 65% improvement in Power-Delay product. One important potential application of Error tolerant adder is in digital signal processing that can tolerate certain amount of errors. Error tolerant adder has been simulated in Microwind and the performance is compared with that of the conventional adders.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

VLSI	-------	Very Large Scale Integration
ET	-------	Error Tolerance
RCA	-------	Ripple Carry Adder
CSK	-------	Carry Skip Adder
ETA	-------	Error Tolerant Adder
FPGA	-------	Field-Programmable Gate Arrays
FFT	-------	Fast Fourier Transform
DSP	-------	Digital Signal Processing

CHAPTER 1
INTRODUCTION

Adders are key components of many high performance systems such as FIR filters, microprocessors, digital signal processors, etc. A system's performance is greatly determined by the performance of the adder because many numbers of adders will be required. So if the delay of each adder is high then this may greatly increase the delay of the system. Hence, optimizing the speed and power consumption of the adder is a major design issue. However, there are always trade-offs between speed and power. Based on the characteristic of digital VLSI design, some novel concepts and design techniques have been proposed. The concept of error tolerance (ET) is one among them. In conventional digital VLSI design, one usually assumes that a usable circuit/system should always provide definite and accurate results. But in fact, such perfect operations are seldom needed in non-digital applications. "Analog computation," requires "good enough" results rather than totally accurate results and the4 concept of error tolerance is based on this.

1.1 Error Tolerant Adder

Increasingly huge data sets and the need for instant response require the adder to be large and fast. The traditional ripple-carry adder (RCA) is therefore no longer suitable for large adders because of its slow-speed performance. Many different types of fast adders, such as the carry-skip adder (CSK), carry-select adder (CSL), and carry-look-ahead adder (CLA) have been developed. Also, there are many low-power adder design techniques that have been proposed. However, there are always trade-offs between speed and power. The error-tolerant design can be a potential solution to this problem. By sacrificing some accuracy, the ETA can attain great improvement in both the power consumption and speed performance.

1.2 Design of 32-bit Adder

The aim of this project is to design a 32 bit error tolerant adder and to compare its performance in terms of delay, power dissipation and power delay product with that of the conventional adders. Also the accuracy of the adder has to be calculated for the set of random inputs.

1.3 Overview

The basic steps involved in the implementation of the project are summarized.

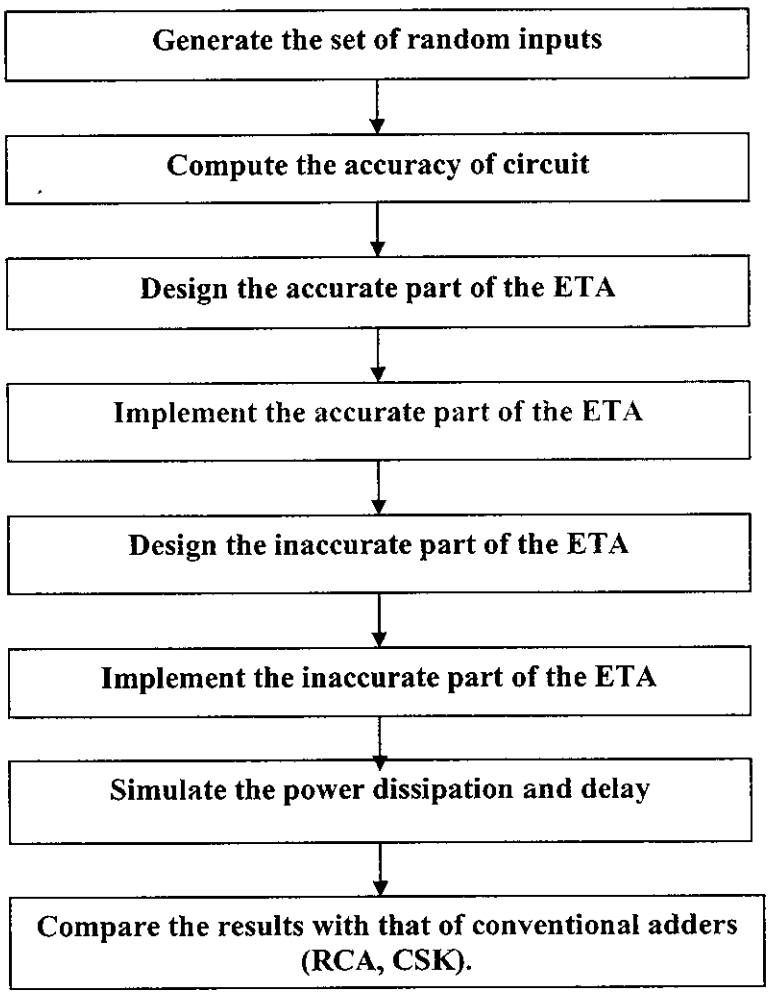

Figure 1.1: Implementation Flow

1.4 Softwares used

- Microwind 3.1.7
- DSCH version 3.1.10
- Microsoft Visual C++

1.5 Organization of the report

- **Chapter 2** discusses about the error tolerant adder
- **Chapter 3** deals with the design of 32 bit error tolerant adder.
- **Chapter 4** shows the experimental results
- **Chapter 5** shows the Conclusion and Future scope of the project.

CHAPTER 2
ERROR TOLERANT ADDER

Before detailing the ETA, the definitions of some commonly used terminologies shown in this paper are given as follows.

- Overall error (*OE*): $OE = |Rc - Re|$, where R_e is the result obtained by the adder, and R_c denotes the correct result (all the results are represented as decimal numbers).
- Accuracy *(ACC)*: In the scenario of the error-tolerant design, the accuracy of an adder is used to indicate how "correct" the output of an adder is for a particular input. It is defined as: $ACC = (1 - (OE/Rc)) \times 100\%$.Its value ranges from 0% to 100%.
- Minimum acceptable accuracy *(MAA):* Although some errors are allowed to exist at the output of an ETA, the accuracy of an acceptable output should be "high enough" (higher than a threshold value) to meet the requirement of the whole system. Minimum acceptable accuracy is just that threshold value. The result obtained whose accuracy is higher than the minimum acceptable accuracy is called acceptable result.
- Acceptance probability *(AP):* Acceptance probability is the probability that the accuracy of an adder is higher than the minimum acceptable accuracy. It can be expressed as $AP = P(ACC > MAA)$, with its value ranging from 0 to 1.

2.1. Proposed addition Arithmetic

In a conventional adder circuit, the delay is mainly attributed to the carry propagation chain along the critical path, from the least significant bit (LSB) to the most significant bit (MSB). Meanwhile, a significant proportion of the power consumption of an adder is due to the glitches that are caused by the carry propagation. Therefore, if the carry propagation can be eliminated or curtailed, a great improvement in speed performance and power consumption can be achieved.

We first split the input operands into two parts: an accurate part that includes several higher order bits and the inaccurate part that is made up of the remaining lower order bits. The length of each part need not necessary be equal. The addition process starts from the middle (joining point of the two parts) toward the two opposite directions simultaneously.

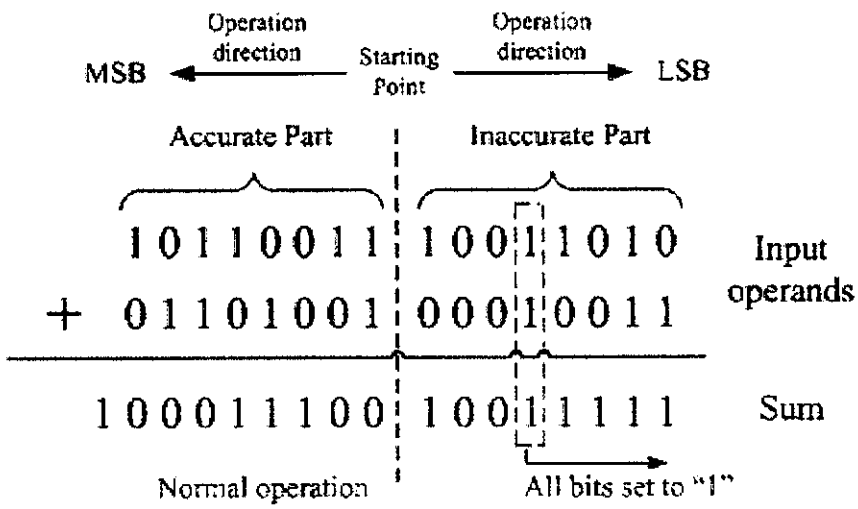

Figure 2.1:Proposed addition arithmetic

In the example of Fig. 1, the two 16-bit input operands, A = "1011001110011010" (45978) and B = "0110100100010011" (26899), are divided equally into 8 bits each for the accurate and inaccurate parts.

The addition of the higher order bits (accurate part) of the input operands is performed from right to left (LSB to MSB) and normal addition method is applied. This is to preserve its correctness since the higher order bits play a more important role than the lower order bits. The lower order bits of the input operands (inaccurate part) require a special addition mechanism. No carry signal will be generated or taken in at any bit position to eliminate the carry propagation path. To minimize the overall error due to the elimination of the carry chain, a special strategy is adapted, and can be described as follow: 1) check every bit position from left to right (MSB to LSB); 2) if both input bits are "0" or different, normal one-bit addition is performed and the operation proceeds to next bit position; 3) if both input bits are "1," the checking process stopped and from this

bit onward, all sum bits to the right are set to "1." The addition mechanism described can be easily understood from the example given in Fig. 1 with a final result of "10001110010011111" (72863).

The example given in Fig. 1 should actually yield "10001110010101101" (72877) if normal arithmetic has been applied. The overall error generated can be computed $OE = 72877-72863 = 14$. The accuracy of the adder with respect to two input operands is these $ACC =(1-(14/72877)) X100\% = 99.98\%$ .By eliminating the carry propagation path in the inaccurate part and performing the addition in two separate parts simultaneously, the overall delay time is greatly reduced, so is the power consumption.

2.2. Relationships Between Minimum Acceptable Accuracy, Acceptance Probability, Dividing Strategy, and Size of Adder

The accuracy of the adder is closely related to the input pattern. Assume that the input of an adder is random; there exists a probability that we can obtain an acceptable result .The accuracy attribute of an ETA is determined by the dividing strategy and size of adder. In this subsection, the relationships between the minimum acceptable accuracy the dividing strategy, and the size of adder are investigated. We first consider the extreme situation where we accept only the perfectly correct result. The minimum acceptable accuracy in this "perfect" situation is 100%. According to the proposed addition arithmetic, we can obtain correct results only when the two input bits on every position in the inaccurate part are not equal to "1" at the same time.

In situations where the requirement on accuracy can be somewhat relaxed are investigated, the result will be different. C program is engaged to simulate a 32-bit adder that had adopted the proposed addition mechanism. By checking the output results, we can derive the relationship between the minimum acceptable accuracy and acceptance probability. For the input patterns, we randomly selected certain inputs from all possible input patterns (i.e., 0–65 535). It can be deduced that the lower the minimum acceptable accuracy set, the higher the acceptance probability for the adder.

As modern VLSI technology advances, the size of the adder has to increase to cater to the application need. The trend of the accuracy performance of an ETA is therefore investigated by following the same dividing strategy whereby the inaccurate part is three times larger than that of the accurate part. Since small numbers will be calculated at the Inaccurate part of the adder, the proposed ETA is best suited for large input patterns.

2.3. Hardware Implementation

The block diagram of the hardware implementation of such an ETA that adopts our proposed addition arithmetic is provided in Fig. 3. This straightforward structure consists of two parts: an accurate part and an inaccurate part. The accurate part is constructed using a conventional adder such as the RCA, CSK, CSL, or CLA. The carry-in of this adder is connected to ground. The inaccurate part constitutes two blocks: a carry-free addition block and a control block. The control block is used to generate the control signals, to determine the working mode of the carry-free addition block. In the next section, a 32-bit adder is used as an example for our illustration of the design methodology and circuit implementation of an ETA.

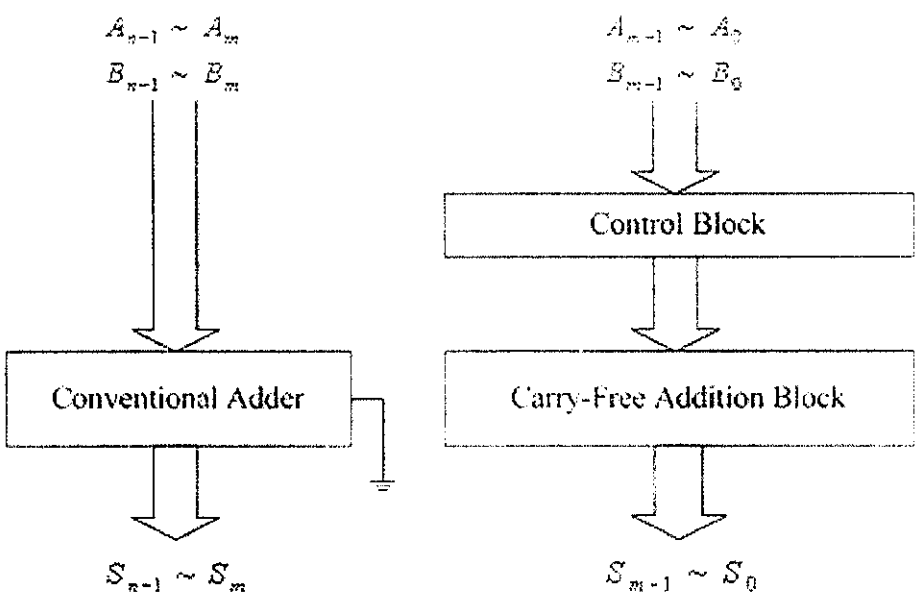

Figure 2.2: Hardware implementation of the ETA

2.4. Applications

In image processing and many other DSP applications, fast Fourier transformation (FFT) is a very important function. The computational process of FFT involves a large number of additions and multiplications.

It is therefore a good platform for embedding the ETA.ETA can be replaced for all adders involved in a normal FFT algorithm. However, for many digital signal processing (DSP) systems that process signals relating to human senses such as hearing, sight, smell, and touch, e.g., the image processing and speech processing systems, the error-tolerant circuits may be used. The potential applications of the ETA fall mainly in areas where there is no strict requirement on accuracy or where super low power consumption and high-speed performance are more important than accuracy. One example of such applications is in the DSP application for portable devices such as cell phones and laptops.

Of course, not all digital systems can engage the error-tolerant concept. In digital systems such as control systems, the correctness of the output signal is extremely important, and this denies the use of the errortolerant circuit.

CHAPTER 3
DESIGN OF 32 BIT ERROR TOLERANT ADDER

3.1 Strategy of Dividing the Adder

The first step of designing a proposed ETA is to divide the adder into two parts in a specific manner. The dividing strategy is based on a guess-and-verify stratagem, depending on the requirements, such as accuracy, speed, and power. First, we define the delay of the proposed adder *as* $T_d = max(T_h, T_l)$,where T_h is the delay in the accurate part and T_l is the delay in the inaccurate part. With the proper dividing strategy, we can make T_h approximately equal to T_l and hence achieve an optimal time delay.

With this partition method, we can check whether the accuracy performance of the adder meets the requirements preset by designer/ customer. This can be checked very quickly via some software programs. (Here this is done using C++ program). For example, for a specific application, we require the minimum acceptable accuracy to be 95% and the acceptance probability to be 98%. The proposed partition method must therefore have at least 98% of all possible inputs reaching an accuracy of better than 95%. If this requirement is not met, then one bit should be shifted from the inaccurate part to the accurate part and have the checking process repeated. Also, due to the simplified circuit structure and the elimination of switching activities in the inaccurate part, putting more bits in this part yields more power saving.

Having considered the above, we divided the 32-bit adder by putting 12 bits in the accurate part and 20 bits in the inaccurate part.

3.2 Design of Accurate Part of the Adder

In the designed 32-bit ETA, the inaccurate part has 20 bits as opposed to the 12 bits used in the accurate part. The overall delay is determined by the inaccurate part, and so the accurate part need not be a fast adder. The ripple-carry adder, which is the most power-saving conventional adder, has been chosen for the accurate part of the circuit.

3.3 Design of Inaccurate Part of the adder

The inaccurate part is the most critical section in the proposed ETA as it determines the accuracy, speed performance, and power consumption of the adder. The inaccurate part consists of two blocks: the carryfree addition block and the control block. The carry-free addition block is made up of 20 modified XOR gates, and each of which is used to generate a sum bit. The block diagram of the carry-free addition block and the schematic implementation of the modified XOR gate are presented in Fig. 4. In the modified XOR gate, three extra transistors, M1, M2, and M3, are added to a conventional XOR gate. CTL is the control signal coming from the control block of Fig. 5 and is used to set the operational mode of the circuit. When CTL = 0 , M1 and M2 are turned on, while M3 is turned off, leaving the circuit to operate in the normal XOR mode. When CTL=1, M1 and M2 are both turned off, while M3 is turned on, connecting the output node to VDD, and hence setting the sum output to "1".

The function of the control block is to detect the first bit position when both input bits are "1," and to set the control signal on this position as well as those on its right to high. It is made up of 20 control signal generating cells (CSGCs) and each cell generates a control signal for the modified XOR gate at the corresponding bit position in the carry-free addition block. Instead of a long chain of 20 cascaded GSGCs, the control block is arranged into five equal-sized groups, with additional connections between every two neighboring groups. Two types of CSGC, labeled as type I and II in Fig. 5(a), are designed, and the schematic implementations of these two types of CSGC are provided in Fig. 5(b). The control signal generated by the leftmost cell of each group is connected to the input of the leftmost cell in next group. The extra connections allow the propagated high control signal to "jump" from one group to another instead of passing through all the 20 cells. Hence, the worst case propagation path [shaded in gray in Fig. 5(a)] consists of only ten cells.

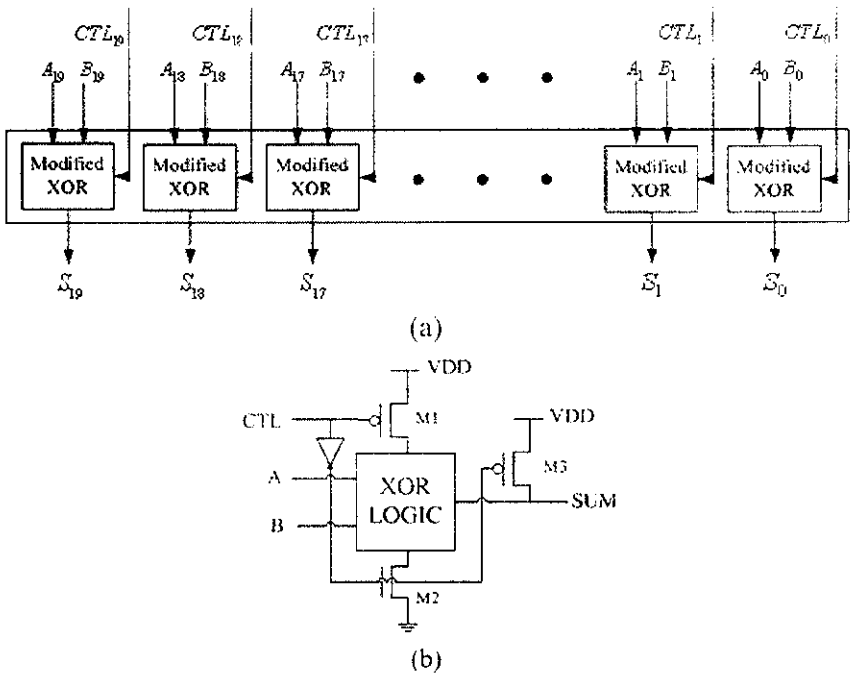

Figure 3.1:Carry free addition block.(a) Overall architecture (b)Schematic diagram of the modified xor gate

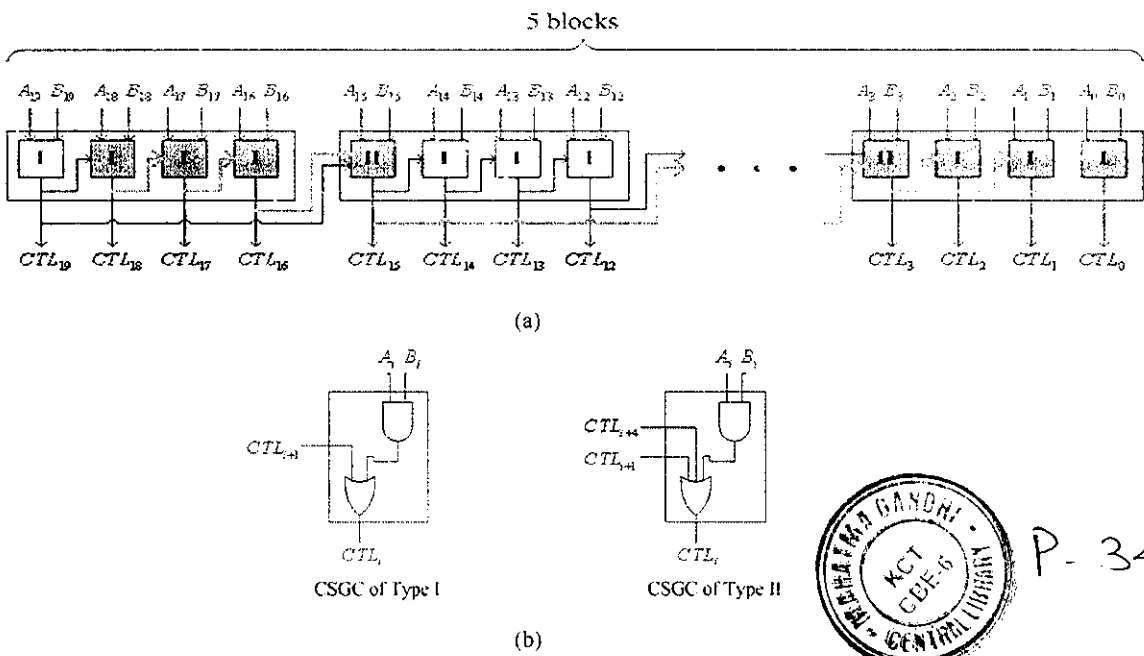

Figure 3.2: Control block.(a) Overall architecture (b)Schematic diagram of CSGC

CHAPTER 4
EXPERIMENTAL RESULTS

To demonstrate the advantages of the error tolerant adder we have simulated the ETA along with the other two conventional adders the ripple carry adder (RCA) and the carry skip adder (CSK).The simulation is done using Microwind3.1.7.

MICROWIND is truly integrated EDA software that allows the designer to simulate and design an integrated circuit at physical descriptionlevel.Microwind3 unifies schematic entry, pattern based simulator, SPICE extraction of schematic, Verilog extractor, layout compilation, on layout mix-signal circuit simulation, cross sectional & 3D viewer, netlist extraction, BSIM4 tutorial on MOS devices and sign-off correlation to deliver unmatched design performance and designer productivity. The package contains a library of common logic and analog ICs to view and simulate. The MICROWIND software has the segment DCH.DCH is a schematic editor and a simulator. DSCH provides fast simulation with delay analysis, which allows the design and validation of complex logic structures

- All the circuits were simulated using the 0.18μm CMOS process
- The input frequency was set as 100 Mhz.

The simulation results at the various stages of implementation are attached below.

4.1 Delay Simulation

4.1.1 Error Tolerant Adder

Verilog, Hierarchy and Netlist

Verilog | Hierarchy | Netlist | Critical path

Path n°	Symbol	Pin	Node	Delay (ns)
1	light(50)	cout(1)	39	0.202
2	fulladdmodule(9)	Cout(5)	39	0.202
3	fulladdmodule(9)	Cin(3)	35	0.188
4	fulladdmodule(8)	Cout(5)	35	0.188
5	fulladdmodule(8)	Cin(3)	33	0.172
6	fulladdmodule(1	Cout(5)	33	0.172
7	fulladdmodule(1	Cin(3)	42	0.156
8	fulladdmodule(11	Cout(5)	42	0.156
9	fulladdmodule(11	Cin(3)	6	0.140
10	fulladdmodule(1)	Cout(5)	6	0.140
11	fulladdmodule(1)	Cin(3)	4	0.124
12	fulladdmodule(1	Cout(5)	4	0.124
13	fulladdmodule(1	Cin(3)	30	0.108
14	fulladdmodule(7)	Cout(5)	30	0.108
15	fulladdmodule(7)	Cin(3)	26	0.092
16	fulladdmodule(6)	Cout(5)	26	0.092
17	fulladdmodule(6)	Cin(3)	22	0.076
18	fulladdmodule(5)	Cout(5)	22	0.076
19	fulladdmodule(5)	Cin(3)	15	0.060
20	fulladdmodule(3)	Cout(5)	15	0.060
21	fulladdmodule(3)	Cin(3)	13	0.044
22	fulladdmodule(4)	Cout(5)	13	0.044

Information

Module name (8 char. max)

accuratepart

☑ Add gate delay info

☑ Append simul. infomations

☐ Add labels as comments

The Verilog file has 134 lines

The design includes 110 symbols

The circuit has 85 nodes

Misc.

Time scale : 1.00

Update Verilog

OK

Figure 4.1: Delay of accurate part of ETA

Verilog, Hierarchy and Netlist

Verilog | Hierarchy | Netlist | Critical path

Path n°	Symbol	Pin	Node	Delay (ns)
1	light(4)	S0(1)	5	0.157
2	carryfree_modul	SUM(4)	5	0.157
3	carryfree_modul	CTRL(1)	6	0.144
4	CSGC1(9)	ctli(4)	6	0.144
5	CSGC1(9)	ctlip1(3)	9	0.135
6	CSGC1(10)	ctli(4)	9	0.135
7	CSGC1(10)	ctlip1(3)	12	0.124
8	CSGC1(11)	ctli(4)	12	0.124
9	CSGC1(11)	ctlip1(3)	15	0.113
10	CSGC2(12)	CTLi(5)	15	0.146
11	CSGC2(12)	CTLip1(4)	19	0.133
12	CSGC1(21)	ctli(4)	19	0.133
13	CSGC1(21)	ctlip1(3)	35	0.122
14	CSGC1(20)	ctli(4)	35	0.122
15	CSGC1(20)	ctlip1(3)	32	0.111
16	CSGC1(19)	ctli(4)	32	0.111
17	CSGC1(19)	ctlip1(3)	18	0.100
18	CSGC2(18)	CTLi(5)	18	0.133
19	CSGC2(18)	CTLip1(4)	29	0.117
20	CSGC1(33)	ctli(4)	29	0.117
21	CSGC1(33)	ctlip1(3)	40	0.106
22	CSGC1(34)	ctli(4)	40	0.106

Information

Module name (8 char. max)

inaccurateparta

☑ Add gate delay info

☑ Append simul. informations

☐ Add labels as comments

The Verilog file has 259 lines

The design includes 241 symbols

The circuit has 160 nodes

Misc.

Time scale : 1.00

Update Verilog

OK

Figure 4.2 : Delay of inaccurate part of ETA

Delay of Error tolerant adder is $Td = max(Th,Tl)$,where Th is the delay in the accurate part and Tl is the delay in the inaccurate part.

Delay is Max(0.202ns,0.157ns) => **0.202ns**

4.1.2 Ripple Carry Adder

Verilog, Hierarchy and Netlist

Verilog | Hierarchy | Netlist | Critical path

Path n°	Symbol	Pin	Node	Delay (ns)
1	light(50)	cout(1)	40	10.355
2	fulladdmodule(9)	Cout(5)	40	10.355
3	fulladdmodule(9)	Cin(3)	36	10.135
4	fulladdmodule(8)	Cout(5)	36	10.135
5	fulladdmodule(8)	Cin(3)	34	9.815
6	fulladdmodule(10	Cout(5)	34	9.815
7	fulladdmodule(10	Cin(3)	43	9.495
8	fulladdmodule(11	Cout(5)	43	9.495
9	fulladdmodule(11	Cin(3)	6	9.175
10	fulladdmodule(1)	Cout(5)	6	9.175
11	fulladdmodule(1)	Cin(3)	4	8.855
12	fulladdmodule(12	Cout(5)	4	8.855
13	fulladdmodule(12	Cin(3)	31	8.535
14	fulladdmodule(7)	Cout(5)	31	8.535
15	fulladdmodule(7)	Cin(3)	27	8.215
16	fulladdmodule(6)	Cout(5)	27	8.215
17	fulladdmodule(6)	Cin(3)	23	7.895
18	fulladdmodule(5)	Cout(5)	23	7.895
19	fulladdmodule(5)	Cin(3)	16	7.575
20	fulladdmodule(3)	Cout(5)	16	7.575
21	fulladdmodule(3)	Cin(3)	14	7.255
22	fulladdmodule(4)	Cout(5)	14	7.255

Information

Module name (8 char. max)

ripplecarryadde

☑ Add gate delay info

☑ Append simul. infomations

☐ Add labels as comments

The Verilog file has 340 lines

The design includes 290 symbols

The circuit has 225 nodes

Misc.

Time scale : 1.00

Update Verilog

OK

Figure 4.3 : Delay of ripple carry adder

4.1.3Carry Skip adder

Verilog, Hierarchy and Netlist

Verilog | Hierarchy | Netlist | Critical path

Path n°	Symbol	Pin	Node	Delay (ns)
1	light(25)	Carry(1)	122	18.420
2	carryskip_mux(2	C3(4)	122	18.420
3	carryskip_mux(2	Carry(2)	79	18.070
4	carryskip_Fullad	Carry(11)	79	18.070
5	carryskip_Fullad	Cin(3)	71	15.370
6	carryskip_mux(2	C3(4)	71	15.370
7	carryskip_mux(2	Carry(2)	92	14.820
8	carryskip_Fullad	Carry(11)	92	14.820
9	carryskip_Fullad	Cin(3)	65	12.120
10	carryskip_mux(1	C3(4)	65	14.820
11	carryskip_mux(1	Carry(2)	63	14.270
12	carryskip_Fullad	Carry(11)	63	14.270
13	carryskip_Fullad	Cin(3)	64	11.570
14	carryskip_mux(1	C3(4)	64	14.270
15	carryskip_mux(1	Carry(2)	67	13.720
16	carryskip_Fullad	Carry(11)	67	13.720
17	carryskip_Fullad	Cin(3)	68	11.020
18	carryskip_mux(2	C3(4)	68	11.020
19	carryskip_mux(2	Carry(2)	58	10.470
20	carryskip_Fullad	Carry(11)	58	10.470
21	carryskip_Fullad	Cin(3)	56	7.770
22	carryskip_mux(1	C3(4)	56	7.770

Information

Module name (8 char. max)

carrySkip_fullckt

☑ Add gate delay info

☑ Append simul. infomations

☐ Add labels as comments

The Verilog file has 547 lines

The design includes 466 symbols

The circuit has 410 nodes

Misc.

Time scale : 1.00

Update Verilog

OK

Figure 4.4 : Delay of carry skip adder

4.2 Simulation of Power Dissipation

4.2.1Error Tolerant Adder

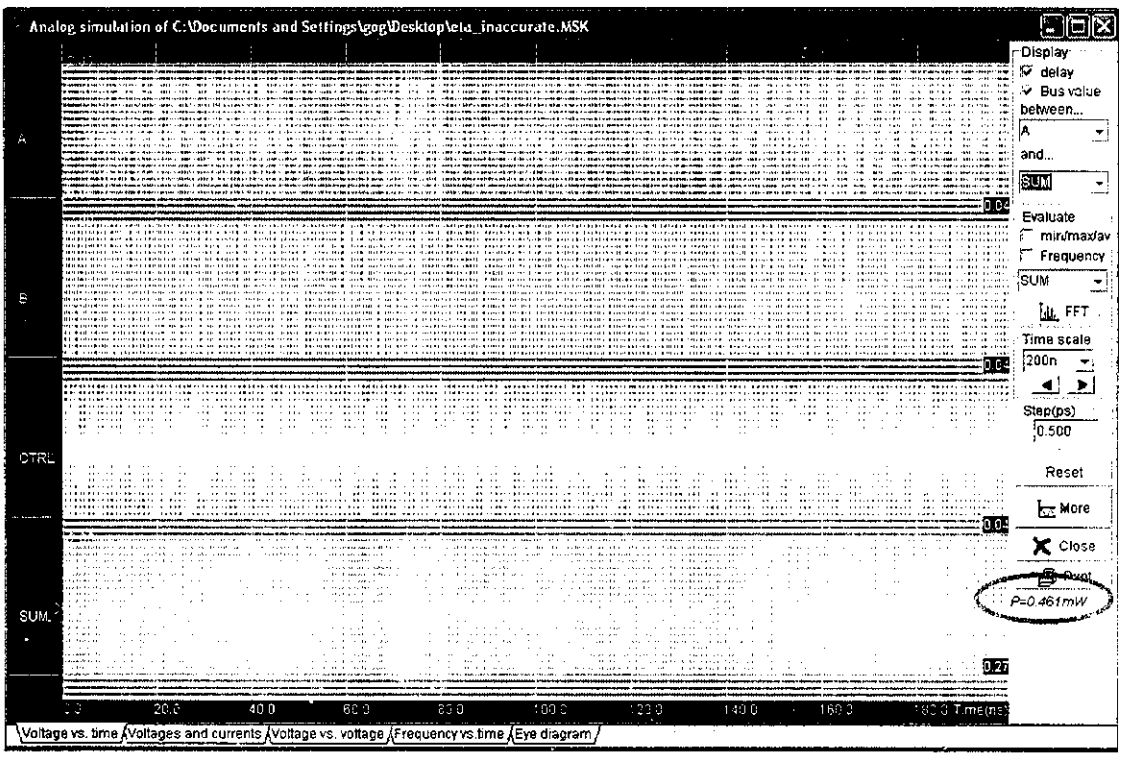

Figure 4.5 : Power dissipation of ETA

4.2.2 Ripple Carry Adder

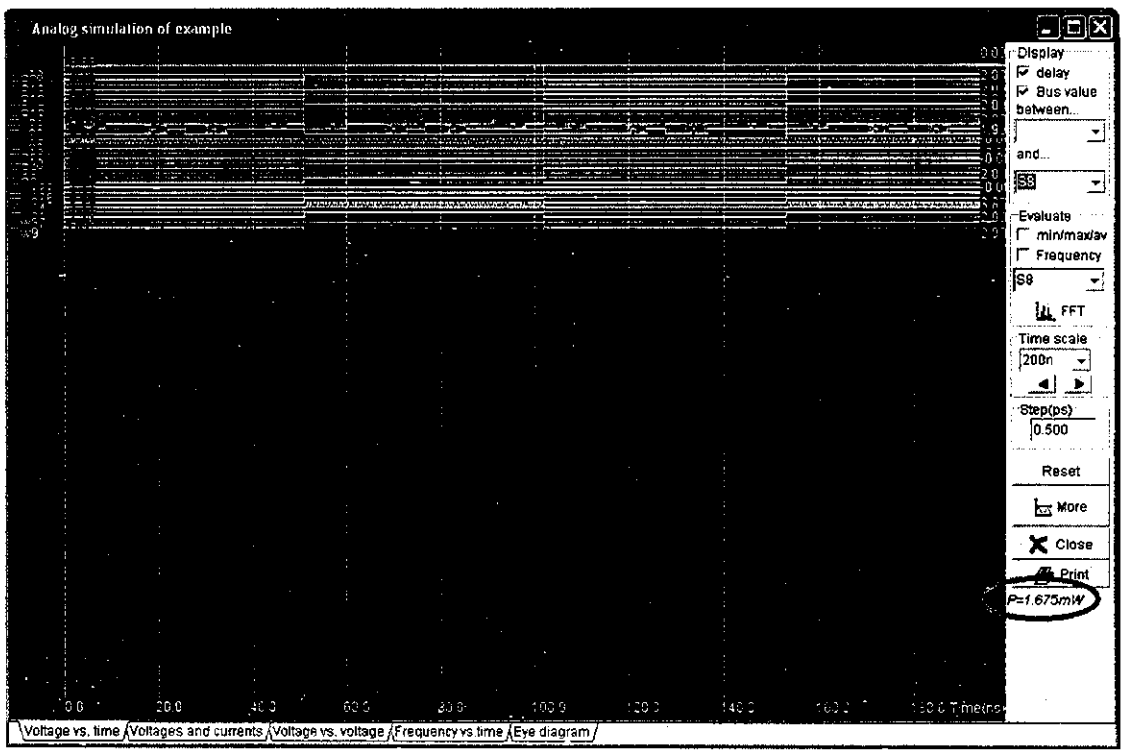

Figure 4.6 : Power dissipation of RCA