Comparitive Study On Multiplier Algorithms using Verilog HDL

by

NUR SYAHADAH BINTI MOHD SAPLI 5983

Final Report submitted in partial fulfilment of the requirements for the Bachelor of Engineering (Hons) (Electrical & Electronics Engineering)

JUNE 2008

Universiti Teknologi PETRONAS Bandar Seri Iskandar 31750 Tronoh Perak Darul Ridzuan

Dedicated to

My parents who always give encouraging words Che Abas bin Hj Hamid Sarimah binti Hj Md. Ali

My siblings who have always been my pride and joy Mohd Shuufi bin Mohd Sapli Nur Syahida binti Mohd Sapli Mohd Shulhi bin Mohd Sapli Mohd Shubhi bin Mohd Sapli Muhammad Aliff Ali bin Che Abs

My friend & other half whom I always rely on Mohd Shahadan bin Mokhtar

My best friends with whom I share five years of my life Noor Fadhilah Mohd Raes Nor Hafizah Abdul Malek Shahrinima Sharifuddin

> My dear friend who always listens Ms Siti Hawa Hj Tahir

Thank you all for the great gifts that each of you have bestowed upon me

CERTIFICATION OF APPROVAL

Comparison Study on Multiplier Algorithms Using Verilog HDL

by

Nur Syahadah binti Mohd Sapli 5983

A project dissertation submitted to the Electrical & Electronics Engineering Programme Universiti Teknologi PETRONAS in partial fulfilment of the requirement for the BACHELOR OF ENGINEERING (Hons) (ELECTRICAL & ELECTRONICS ENGINEERING)

Approved by,

ha

(AP DR MOHAMMAD BIN AWAN) Final Year Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

June 2008

ii

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

(NUR SYAHADAH BINTI MOHD SAPLI) 5983

ABSTRACT

Multipliers are used in many applications especially in computers. A personal computer (PC) utilizes multipliers to perform calculations. Thus, having a multiplier with great speed will definitely boost the performance of a PC. Based on this, the purpose of the Final Year Project is to perform a comparison study on multiplier algorithms using Verilog HDL. Four multipliers have been selected to be the subject of study. The multipliers are Ripple Carry multiplier, Carry Save multiplier, Wallace multiplier and finally the Dadda multiplier. The propagation delay of each multiplier is determined to check their performance in terms of speed. The outcome of this project has showed that, among these four multipliers, Carry Save multiplier has exhibited the smallest amount of propagation. Therefore, it is the fastest multiplier out of the four that are studied whereas Dadda multiplier shows the least number of logic elements used up until 6-bit multiplication process.

ACKNOWLEDGEMENT

I would like to thank my supervisor, AP DR Mohammad bin Awan for the constant guidance and coaching throughout the entire duration of this project. He has kept me motivated and encouraged during the time that I am under his supervision. Thank you, Sir. I would also like to extend my gratitude to Mr Patrick Sebastian, who has never refused to answer lengthy questions on Verilog syntax. Thank you so much for the numerous Verilog coding that you have corrected as well as tips on good programming style. My deepest gratitude also goes to Mr Lo Hai Hiung who explains a lot about Quartus II software. Without the knowledge, I would not be able to do the simulation. To EE Lab Technologist, Ms Siti Hawa Hj Mohd Tahir, thank you for the moral support that you have given during the entire time of knowing you. Finally, to my parents and friends, thank you for the advices which I will always treasure.

TABLE OF CONTENTS

CERTIFICATI	ON O	F APPROVAL	ii
CERTIFICATI	ON O	F ORIGINALITY	iii
ABSTRACT	•••••		iv
ACKNOWLED	GME	NT	v
LIST OF FIGU	RES	······	ix
LIST OF TABL	ÆS		xi
CHAPTER 1 :	INT	RODUCTION	1
	1.1	Background of Study	1
	1.2	Problem Statement	1
	1.3	Objective and Scope of Study	2
CHAPTER 2 :	LIT	ERATURE REVIEW / THEORY	3
	2.0	Propagation Delay	3
	2.1	Basic Components of a Multiplier	3
		2.1.1 Half Adder	4
		2.1.2 Full adder	4
	2.2	Types of Adder	6
		2.2.1 Ripple Carry Adder	6
		2.2.2 Carry Save Adder	9
	2.3	Types of Multiplier	11
		2.3.1 Shift-Add Multiplier	11
		2.3.1.1 Ripple Carry Multiplier	12
		2.3.1.2 Carry Save Array Multiplier	13
		2.3.2 Wallace Multiplier	14
		2.3.2.1 2-bit Wallace Multiplier	15
		2.3.2.2 4-bit Wallace Multiplier	15
		2.3.2.3 6-bit Wallace Multiplier	16
		2.3.2.4 8-bit Wallace Multiplier	16

		2.3.3	Dadda Multiplier	17
			2.3.3.1 2-bit Dadda Multiplier	18
			2.3.3.2 4-bit Dadda Multiplier	18
			2.3.3.3 6-bit Dadda Multiplier	19
			2.3.3.4 8-bit Dadda Multiplier	19
CHAPTER 3 :	ME	THOD	OLOGY / PROJECT WORK	21
	3.1	Projec	t Flow Overview	21
	3.2	Metho	odology	22
	3.3	CAD	Tool	28
CHAPTER 4 :	RES	SULT A	AND DISCUSSION	29
	4.1	Result	t	29
		4.1.1	Ripple Carry Multiplier	29
			4.1.1.1 2-bit Ripple Carry Multiplier	29
			4.1.1.2 4-bit Ripple Carry Multiplier	31
			4.1.1.3 6-bit Ripple Carry Multiplier	31
			4.1.1.4 8-bit Ripple Carry Multiplier	31
		4.1.2	Carry Save Multiplier	32
			4.1.2.1 2-bit Carry Save Multiplier	32
			4.1.2.2 4-bit Carry Save Multiplier	33
			4.1.2.3 6-bit Carry Save Multiplier	33
			4.1.2.4 8-bit Carry Save Multiplier	34
		4.1.3	Wallace Multiplier	35
			4.1.3.1 2-bit Wallace Multiplier	35
			4.1.3.2 4-bit Wallace Multiplier	36
			4.1.3.3 6-bit Wallace Multiplier	36
			4.1.3.4 8-bit Wallace Multiplier	37
		4.1.4	Dadda Multiplier	38
			4.1.4.1 2-bit Dadda Multiplier	38
			4.1.4.2 4-bit Dadda Multiplier	39
			4.1.4.3 6-bit Dadda Multiplier	39
			4.1.4.4 8-bit Dadda Multiplier	40
		4.1.5	Implementation on FPGA	40

	4.2	Discussion	41
		4.2.1 Summary of Results	41
CHAPTER 5 :	CO	NCLUSIONS AND RECOMMENDATION	44
	5.1	Conclusions	44
	5.2	Recommendation	44
REFERENCES.			45
APPENDIX			46

LIST OF FIGURES

Figure 2.1 : Propagation Delay	3
Figure 2.2 : Half- Adder Schematic	4
Figure 2.3 : Full-adder schematic	4
Figure 2.4(a) : 2-bit full adder block diagram	6
Figure 2.4(b) : 4-bit full adder block diagram	7
Figure 2.5 : Schematic of 4 bit ripple carry adder with Carry bit highlighted	7
Figure 2.6 : Design Hierarchy of a 4-bit Ripple Adder	8
Figure 2.7 : A Full Adder (FA) and a Carry Save Adder (CSA)	9
Figure 2.8 : Design Hierarchy for Carry Save Adder	9
Figure 2.9(a) : 2-bit Carry Save Adder	10
Figure 2.9(b) : 4-bit Carry Save Adder	10
Figure 2.10 : Normal Multiplication	11
Figure 2.11 : Block Diagram of 4-by-4 bits Ripple Carry Multiplier	12
Figure 2.12 : Block Diagram of 4-by-4 bits Carry Cave Multiplier	13
Figure 2.13 : 2-bit Wallace Multiplication	15
Figure 2.14 : Dot Diagram for a 4-bit Wallace Multiplier	15
Figure 2.15 : Dot Diagram for a 6-bit Wallace Multiplier	16
Figure 2.16 : Dot Diagram for 8-bits Wallace Multiplier	17
Figure 2.17 : 2-bit Dadda Multiplier	18
Figure 2.18 : Dot Diagram for a 4-bit Dadda Multiplier	18
Figure 2.19 : Dot Diagram for a 6-bit Dadda Multiplier	19
Figure 2.20 : Dot Diagram for an 8-bit Dadda Multiplier	20
Figure 3.1 : General Project Flow	22
Figure 3.2 : Creating a New Design File	23
Figure 3.3 : Compilation Window	23
Figure 3.4 : Functional Simulation Window	24
Figure 3.5 : Example of Functional Simulation Output	25
Figure 3.6 : Assigning Device to Design	25
Figure 3.7 : Timing Simulation Window	26
Figure 3.8 : An Example of Timing Simulation Output	27
Figure 3.9 : Detailed Project Flow	28

Figure 4.1 : Timing Simulation for 2-bit Ripple Carry Multiplier	29
Figure 4.2 : Worst-Case Propagation Delay Time	30
Figure 4.3 : Propagation Delay Time for the Design	30
Figure 4.4 : Timing Simulation for 2-bit CS multiplier	32
Figure 4.5 : Worst-Case Propagation Delay	32
Figure 4.6 : Propagation Delays for the Design	33
Figure 4.7 : Timing Simulation for 2-bit Wallace multiplier	35
Figure 4.8 : Worst-Case Propagation Delay Time	35
Figure 4.9 : Total Propagation Delay for the Design	36
Figure 4.10 : Timing Simulation for a 2-bit Dadda multiplier	38
Figure 4.11 : Worst-Case Propagation Delay Time	38
Figure 4.12 : Total Propagation Delay for the Design	39
Figure 4.13 : Propagation Delay Chart	41
Figure 4.14 : Number of Logic Elements Chart	43

LIST OF TABLES

Table 2.1 : Truth table for half adder	4
Table 2.2 : Truth table for full adder	5
Table 2.3 : The Bit Product	11
Table 2.4 : Table of Reduction Stages for Wallace and Dadda Multipliers	14
Table 4.1 : Propagation Delay	41
Table 4.2 : Number of Logic Elements	43

CHAPTER 1

INTRODUCTION

1.1 Background of Study

Multipliers are used in many applications especially in computers. A personal computer (PC) utilizes multipliers to perform calculations. Thus, having a multiplier with great speed will definitely boost the performance of a PC. Therefore, this study will focus mainly on the speed performance of digital multipliers implemented in different ways. The purpose is to develop a comparison of speeds between several digital multipliers that are implemented in different methods. This is important as speed is a crucial factor in any digital design especially when the gates are connected in series. This can lead to a major time delay that will of course, affect the speed performance.

1.2 Problem Statement

In digital design area, there are many kinds of multipliers. Since there are so many ways to implement a multiplier, the issue of speed arises. Multipliers are commonly found in the computer systems area whereby it is used in the basic structure of the computer itself. Therefore, if the delay time is too great, it could and would affect the whole computer performance. Thus, the issue of speed is considered as a major issue in digital design.

1.2.1 Objective and Scope of Study

The objective of this project is to determine the propagation delay of each multiplier. Therefore, a multiplier needs to be designed and simulated to ensure that it is giving the expected correct output. The study covers several multipliers such as Ripple Carry Multiplier, Carry Save Multiplier, Wallace Multiplier and finally Dadda Multiplier.

CHAPTER 2

LITERATURE REVIEW / THEORY

2.0 Propagation Delay

Propagation delay occurs between the time that an input changes and the time taken by the output to change accordingly [1]. As shown in Figure 2.1, propagation delay is divided into two which are, the propagation delay high-low and propagation delay low-high [2]. Propagation delay high-low occurs because a change in the input from the logic state '1' to the logic state '0' is detected and the output signal takes some time to change accordingly. Propagation delay low-high is the exact reverse of propagation delay high-low. In this project, the worst-case propagation delay is determined so that the maximum frequency that the design can run on can be calculated.



Figure 2.1 : Propagation Delay

2.1 Basic Components of a Multiplier

Below are the basic components of a multiplier; a half adder and a full adder. Both components are used in a multiplication process because a multiplier needs adder to sum up the partial products.

2.1.1 Half Adder

A half-adder is one of the two types of adder. The basic concept is that, a half-adder accepts two binary inputs and produces a sum output and a carry output. Below is a logic circuit diagram of a half-adder and the corresponding truth table as shown in Figure 2.2 and Table 2.1 [3].



Α	В	Cout	Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Table 2.1 : Truth Table for Half Adder

Figure 2.2 : Half- Adder Schematic

2.1.2 Full adder

On the other hand, a full adder is able to accept one more bit which is the input carry bit, C_{in} . Therefore, the logic circuit diagram for the full adder looks like as shown below in Figure 2.3 and its truth table in Table 2.2 [1].



Figure 2.3 : Full Adder Schematic

Table 2.2 shows the two outputs; Carry-out bit (C_{out}) and Sum-bit (\sum) given certain inputs which are A, B and Carry-in bits.

A	В	C _{in}	Cout	Σ
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

Table 2.2 : Truth Table for Full Adder

2.2 Types of Adder

This section will discuss two types of commonly used adders which are the Ripple Carry Adder and Carry Save Adder.

2.2.1 Ripple Carry Adder

The block diagram for a ripple carry adder is shown in Figure 2.4. A ripple carry is a type of parallel adders found in digital world. A ripple carry adder is where the carry output of one stage is being used as input to a full adder in the next higher stage [1]. The carry output of a lower stage adder is connected to the carry input of the next higher adder stage. A 4-bit ripple carry adder can be formed by cascading four 1-bit full adders in a chain where the carry generated by one unit is then being passed forward to the next full adder via the carry input port of that adder. Figure 2.4(a) below shows the block diagram of a 2-bit Ripple Carry Adder.



(a)

Figure 2.4 (b) below depicts the block diagram for a 4-bit ripple carry adder.



Figure 2.4: (a) 2-bit full adder block diagram (b) 4-bit full adder block diagram

Figure 2.5 shows the logic gate diagram of 4-bit ripple carry adder with the Carry bit highlighted. Theoretically, this path is the longest path that causes the delay in the ripple carry adder [2].



Figure 2.5 : Schematic of 4-bit ripple carry adder with Carry bit highlighted

Shown below in Figure 2.6 is the design hierarchy of the ripple adder [2]. A half-adder consists of logic gates 'XOR' and 'AND'. By using two half-adders, a full adder is then constructed. In order to create a 4-bit ripple adder, four 1-bit full adders are used.



Figure 2.6 : Design Hierarchy of a 4-bit Ripple Adder

2.2.2 Carry Save Adder

If a carry save adder is briefly looked at, it does not look any different from a typical full adder. A carry save adder still accepts three inputs and produces two outputs. However, when this adder is used in a circuit, it obviously differs in the carry bit aspect. The figure below, Figure 2.7 demonstrates clearly the difference between a carry save adder and a typical full adder [4].



Figure 2.7 : A Full Adder (FA) and a Carry Save Adder (CSA) [4]

A Carry Save adder also has the same basic components as the previously discussed Ripple Carry adder. The design hierarchy for the Carry Save adder is shown in Figure 2.8 below.



Figure 2.8 : Design Hierarchy for Carry Save Adder

The Carry Save Adder usually consists of n number of full adders. The full adders are connected in such a way that the carry bit is propagated to the next layer of addition process. This means that, each layer of adders can perform addition without waiting for the carry bit input from the previous stage. Shown below in Figure 2.9 are the block diagrams for a 2-bit and a 4-bit carry save adder.





Figure 2.9 : (a) 2-bit Carry Save Adder (b) 4-bit Carry Save Adder

2.3 Types of Multiplier

This section discusses four types of multipliers which are Ripple Carry Multiplier, Carry Save Multiplier, Wallace Multiplier and Dadda Multiplier.

2.3.1 Shift-Add Multiplier

As the name suggests, a shift-add multiplier shifts the product bits before adding them together much like a normal multiplication method. This is further illustrated below.

				Table	2.3 : The	Bit Produc
		A 1	A0	Α	B	AxB
	Х	B1 A1B0	Bo AoBo	0	0	0
+	A 1 B 1	A 0 B 1		0	1	0
C 3	C2	C 1	Co	1	0	0
igure	2.10 : Nor	mal Multi	plication	1	1	1

There are a few types of shift-add multiplier. Each one differs from one another in terms of the adder component used in the array.

2.3.1.1 Ripple Carry Multiplier

This type of multiplier is constructed from several half adders and full adders. Shown below in Figure 2.11 is the block diagram of the shift-add multiplier using ripple carry adder to sum up the partial product bits. The partial product bits are the logical 'and' from each input [5].



Figure 2.11 : Block Diagram of 4-by-4 bits Ripple Carry Multiplier

Since ripple carry adder is used to sum up the bit products, it is expected to have a maximum delay from the Least Significant Bit (LSB) to the Most Significant Bit (MSB).

2.3.1.2 Carry Save Array Multiplier

This type of multiplier fundamentally produces the same gate delay as the ripple carry array multiplier shown previously [6]. However, the difference between the two multipliers is that a Carry Save Array Multiplier does not perform the carry chain. Instead, it passes the carry bit to the next layer of adder to be added together with other bit products. Due to this matter, the multiplier is called "Carry Save" Array Multiplier. Shown below in Figure 2.12 is block diagram of a Carry Save Array Multiplier.



Figure 2.12 : Block Diagram of 4-by-4 bits Carry Cave Multiplier

The last layer of the addition part uses ripple carries technique. To yield a smaller propagation delay, this process can be substituted with a faster carry tree adder.

2.3.2 Wallace Multiplier

For this type of multiplier, the partial products are generated in the same manner as the shift-add multiplier whereby n number of AND logic gates is used. Wallace multiplier tries to reduce as many partial products as possible in a single reduction layer. These partial products are reduced to a final level with a height of two. Then, ripple carry adders are used to complete the reduction. The length of the adder, m depends on the number of bits, n.

RCA length,
$$m = 2(n) - 2$$
 [7]

Each sub-section below will discuss in length on several *n*-bits multiplication process. The number of reduction level is done according to Table 2.4 [8].

Number of bits, n	Number of Reduction Level
2	0
3	1
4	2
5-6	3
7-9	4
10-13	5
14-19	6
20-28	7
29-42	8
43-63	9
64-94	10

Table 2.4 : Table of Reduction Stages for Wallace and Dadda Multipliers

2.3.2.1 2-bit Wallace Multiplier

Based on the concept on Wallace Multiplier mentioned earlier, it is required that the partial products are reduced to a level with a height of two. For two-by-two multiplication process, the partial products generated are already in a level with a height of two. Therefore, it needs no further reduction process. The diagram will give a better understanding.

2.3.2.2 4-bit Wallace Multiplier

Based on the Table 2.4, the number of reduction performed is two. The first reduction is to a height of three as shown in Figure 2.14. Then, the partial products are further reduced to a height of two. The reduction process uses several counters. There are two types of counters which are three-to-two (3, 2) and two-to-two (2, 2). The last stage is to complete the multiplication using ripple carry adder of length *m*.



Figure 2.14 : Dot Diagram for a 4-bit Wallace Multiplier [7]

2.3.2.3 6-bit Wallace Multiplier

As described in Table 2.4, the number of reduction stages of a 6-bit multiplier is three stages. For a 6-bit multiplier, the partial products generated initially are of a height six. Since it requires three stages of reduction process, the first stage will be to reduce the partial products to a height of four, then three and finally to a height of two. The multiplication process is completed by using ripple carry adders. Figure 2.15 shows the dot diagram for a 6-bit Wallace Multiplier.



Figure 2.15 : Dot Diagram for a 6-bit Wallace Multiplier

2.3.2.4 8-bit Wallace Multiplier

For an 8-bit Wallace multiplier, the number of reduction required is four stages. The first stage is to reduce to height of six, then four, three and finally two. The multiplication process is also completed using ripple carry adder. The figure below, Figure 2.16 shows the dot diagram of the multiplier. The 'crossed diagonal' line represents (2,2) counters and the single diagonal line represents (3,2) counters.



Figure 2.16 : Dot Diagram for 8-bits Wallace Multiplier

2.3.3 Dadda Multiplier

Dadda multiplier works in similar manner as the Wallace multiplier. It also consists of three stages which are; (1) partial products generation using AND logic gates, (2) partial product reduction and (3) using adders to complete the multiplication. However, unlike Wallace multiplier, Dadda multiplier does not try to reduce the partial products all at once. Instead, it only reduces partial products that exceed the required reduction. The sub-sections below will discuss this multiplier in length [7].

2.3.3.1 2-bit Dadda Multiplier

Same as Wallace multiplier, a 2-bit Dadda multiplier requires no reduction because the partial products are already in the final height of two. Therefore, the output of the multiplication process can easily be obtained straight from the output of each adder components.



Figure 2.17 : 2-bit Dadda Multiplier

2.3.3.2 4-bit Dadda Multiplier

A 4-bit Dadda multiplier also requires two reduction stages as shown in Table4. The first reduction is to height of three and finally to a height of two. The multiplication process is completed using ripple carry adder to sum up the reduced partial products. The dot diagram in Figure 2.18 shows the partial products that are reduced. Notice that in the second layer partial product, only a few are reduced.



Figure 2.18 : Dot Diagram for a 4-bit Dadda Multiplier [6]

2.3.3.3 6-bit Dadda Multiplier

A 6-bit Dadda multiplier requires three reduction stages. The first one is to reduce the partial products generated from the AND logic gates to a height of four, then to a height of three. Next, it will be reduced further to achieve the final height of two before adders are applied to complete the multiplication process. The reduction concept of the previously discussed 4-bit Dadda multiplier is applied in this multiplication process. Figure 2.19 shows the dot diagram for a 6bit Dadda Multiplier.



Figure 2.19 : Dot Diagram for a 6-bit Dadda Multiplier

2.3.3.4 8-bit Dadda Multiplier

As the number of bits, n increases, the number of reduction stages also increases. For an 8-bit Dadda multiplier, the required reduction stage is four stages. The first stage reduced the partial products of height eight to a height of six and the second stage reduces them further to a height of four. The process continues with the partial products being reduced to a height of three and finally to a height of two. The multiplication process is completed using ripple carry adders. Shown in Figure 2.20 is the 8-by-8 bits dot diagram for Dadda multiplier. As can be seen, only certain partial products are reduced to achieve the required reduction.



Figure 2.20 : Dot Diagram for an 8-bit Dadda Multiplier

CHAPTER 3

METHODOLOGY / PROJECT WORK

3.1 **Project Flow Overview**

The first stage of the project is to conduct research work. Research work is done to select the types of adders that will be used in the project. By doing research, a better understanding of the operations of the adders is achieved. Source of information is not only limited to books but also includes group discussion.

The second phase of the project is to understand the programming language used for the project which is the Verilog HDL. Verilog is a widely accepted language for VLSI design [5]. Therefore, the time allocated to be familiarized with Verilog is used to understanding the constructs in Verilog such as the basic of the language, its convention and so on.

With the second phase of the project is still on-going, the coding for adders begins. By using trial-and-error method, many codes are generated and simulated to find the eliminate errors in the coding of the adders. A lot of time has been spent in trying to convert the understanding of the adder operation into a workable Verilog code.



The figure below, Figure 3.1 shows the general project flow.

Figure 3.1 : General Project Flow

3.1 Methodology

As mentioned in the previous section, before being able to design the multiplier, concept of the multiplier operation must be first studied. This is to ensure that the result obtained during the simulation phase is the same as the calculated result. Therefore, as shown in the detailed project flow, understanding the design concept is the first stage.

The next stage of the project is to transfer the understanding gained on the multiplier design into a block diagram. Working with block diagrams is much

easier than complicated texts.

Using the block diagram created earlier, the designing stage of the project can begin. Assigning variables in the design becomes a lot easier with the help of the block diagram. Since this project uses Verilog HDL as the programming language, thus Verilog is selected in the pop up window of the Quartus II software as shown in Figure 3.2. Compiling the design is the next process that must be done. Compilation is done to check syntax errors. Figure 3.3 is the compiler tool.

	Other Files	
AHDL File Block Diagram/Sch	ematic File	
EDIF File SOPC Builder System	n	
Verilog HDL File		
TIDETIIS		

Figure 3.2 : Creating a New Design File

Analysis & Synthesis 0 % 00:00:00	Fitter 0 % 00:00:00	Assembler 0 % 00:00	Timing Analyzer 0 % 00:00:00
	1 A A A	1	►6 🧐 🤮
	lo	lle	
	0	% 0.00	Real and the second field of the
	00:0	10:00	
Start			A Repo

Figure 3.3 : Compilation Window

If the design is found to contain error, the design file needs to be checked again based on the error message that will normally appear after the compilation process. The next part of the project, which is to perform functional simulation, can only be started after successful compilation process. Functional simulation is one of the three simulation options that are available in Quartus II. It assumes that the logic elements and interconnection wires are perfect. Therefore, there is no delay in propagating the signals in the design circuit. Functional simulation can be said as verifying the functionality or correctness of the circuit as designed. In simple words, it is done to help designers check whether the design file is giving expected output.

Figure 3.4 shows the Simulator tool with Functional simulation mode selected.

Simulator T	ool	
Simulation mode:	Functional	Generate Functional Simulation Netlist
Simulation input:	Functional Timing Timing using Fast Timing Model	
Simulation perio	d	
Run simulat	ion until all vector stimuli are used	
C End simulat	ion at: 100 ns 📼	
Simulation optio	ns	
Automatical	ly add pins to simulation output wavefor	ms
Check outp	uts Wavelorn Comparison Selbings	
☐ Setup and 1		
F Glick dated	ton 1.0 ns 💌	
Verwrite si	mulation input file with simulation results	
🖵 Generate S	gnel/kolivity File	
	0 %	
	00:00:00	
E Start	1 Stop	🕑 Open 🥢 🚇 Report

Figure 3.4 : Functional Simulation Window

Figure 3.5 shows an example of the Functional simulation mode. Notice that the output is obtained as the same time as the input.

Mast	er Time Bar:	0 ps	4 1	Pointer:	3.7 ns	Intervat	3.7 ns	Start		End	
		0 ps		10.0 ns		20.0 ns		30.0 ns		40.0 ns	
	Name	0 ps									
	+ a		0001	X	0010	X	0011	X	0100	X	0101
X	€ b		0001	X	0010	X	0011	X	0100	X	0101
	e prod	0	0000001	X	00000100	X	00001001	X	00010000	X	00011001

Figure 3.5 : Example of Functional Simulation Output

All designs in this project are simulated on EPF10K70RC240-4 which is available in Quartus II as shown in Figure 3.6.

Family: FLEX10K			*	Show in 'Availa	able devic	ces' list
Davias & Dia Onlines				Package:	Any	-
Device & Pin Uptions				Pin count:	240	
Target device					210	-
C Auto device selecte	d by the Fitter		Speed grade:	4	-	
Specific device sele	-linu At ai bata			-		
	STREET IN LAYAR	Ine devices is		Core voltage:	5 IV	
0.0	cteu in Avalia	idie devices iisi	ł	Core voltage:	5.0V	
C Other n/a	icted in Avalia	IDIE DEVICES IISI		Core voltage:	5.0V anced de	vices
C Other n/a Available devices:	CLEU IN AVAIR	IDIE GEVICES IISI		Core voltage:	5.0V anced de	vices
C Other of a Available devices: Name	LEs	Memor		Core voltage:	5.0V anced de	vices
C Other of a Available devices: Name EPF10K20RC240-4	LEs 1152	Memor		Core voltage:	5.0V anced de	vices
Available devices: Name EPF10K20RC240-4 EPF10K20RI240-4	LEs 1152 1152	Memor 12288 12288		Core voltage:	5.0V anced de	vices
Available devices: Name EPF10K20RC240-4 EPF10K20RI240-4 EPF10K30RC240-4	LEs 1152 1728	Memor 12288 12288 12288		Core voltage:	5.0V anced de	vices
C Other m/a Available devices: Name EPF10K20RC240-4 EPF10K20RI240-4 EPF10K30RC240-4 EPF10K30RI240-4	LEs 1152 1728 1728	Memor 12288 12288 12288 12288 12288		Core voltage: ✓ Show adv	5.0V anced de	vices
Available devices: Name EPF10K20RC240-4 EPF10K20RI240-4 EPF10K30RC240-4 EPF10K30RI240-4 EPF10K30RI240-4 EPF10K40RC240-4	LEs 1152 1728 1728 2304	Memor 12288 12288 12288 12288 12288 12288 16384		Core voltage:	5.0V anced de	vices
Available devices: Name EPF10K20RC240-4 EPF10K20RI240-4 EPF10K30RC240-4 EPF10K30RI240-4 EPF10K30RI240-4 EPF10K40RC240-4 EPF10K50RC240-4	LEs 1152 1728 1728 2304 2880	Memor 12288 12288 12288 12288 12288 12288 16384 20480		Core voltage:	5.0V anced de	vices
Available devices: Name EPF10K20RC240-4 EPF10K20RI240-4 EPF10K30RC240-4 EPF10K30RI240-4 EPF10K30RI240-4 EPF10K50RI240-4 EPF10K50RI240-4	LEs 1152 1152 1728 1728 2304 2880 2880	Memor 12288 12288 12288 12288 12288 12288 16384 20480 20480		Core voltage: ✓ Show adv	5.0V anced de	vices

Figure 3.6 : Assigning Device to Design

When the output of the functional simulation is checked to be correct, then timing simulation can proceed. Timing simulation is to determine how well a circuit performs in terms of speed. Therefore, in this type of simulation, it will consider the delay in propagating all the signals. After timing simulation is done, propagation delay for the simulated design can be determined from the simulation report.

Shown in the diagram below, Figure 3.7, is the Simulator tool window with the timing simulation mode selected.

Simulator T	ool	
Simulation mode:	Timing 💌	Generate Functional Simulation Netlist
Simulation input:	Functional Timing	
Simulation period		
Run simulati	on until all vector stimuli are used	
C End simulati	on at 100 na 🚽	
Simulation option	18	
Automatical	ly add pins to simulation output wavefo	rms
Check outp	uts Wavelom Comparison Settings	
Setup and h	old time violation detection	
Glitch detec	tion: 1.0 m -	
Verwrite si	mulation input file with simulation result	8
📕 Generate Si	gnal Activity File	
	0%	
	00:00:00	
Start	10 Stop	Doen Benort

Figure 3.7 : Timing Simulation Window

Figure 3.8 is an example of timing simulation output. Notice that the output is delayed for a while.

D :	c_arraymult	.vwf	- 1.1-									
Mast	er Time Bar:	20.0 ns	4 • Po	ointer.	133.1 ns	Interval	113.1	ns	Start		End	
	Name		110,0 ns		120,0 ns		130,0 ns		140 _, 0 ns		150,0 ns	
	eresteresteresteresteresteresteresteres	101 101 X10X X	o X o X	1011 1011 0))(10))		1100 1100 X0101 X(X)	X X 1)(x)(01)(500	1101 1101	X	1110 1110 ())(00)()		1111 1111 XXXIXXXIIIIXXXX
<		> <								-11-1	intintia e tre	() () () () () () () () () () () () () (

Figure 3.8 : An Example of Timing Simulation Output

The flow chart below, Figure 3.9, summarizes the detailed project flow.



Figure 3.9 : Detailed Project Flow

3.2 CAD Tool

The CAD tool used in this project is Quartus II Web Edition Version 6.

CHAPTER 4 RESULT AND DISCUSSION

4.1 Result

In this section, the simulation results of the multipliers under study are analyzed.

4.1.1 Ripple Carry Multiplier

This section shows the simulation results for 2-bit, 4-bit, 6-bit and 8-bit Ripple Carry Multiplier.

4.1.1.1 2-bit Ripple Carry Multiplier

Shown below, Figure 4.1 is the simulation result of a 2-bit Ripple Carry Multiplier and the worst-case propagation delay time window, Figure 4.2 along with all the propagation delays in the design, Figure 4.3.

Mast	er Time Bar:	20.0 ns	4 3	Poin	ter:	41.4	15	Interv	at	21.4	ns !	Start			End		
	Name	Valu 20.	0 ps	10.01	ns 2 2	0.0 ns	30.p r	is 40.	pins	50.0 ni	s 60.0	ns 70.	Dins 1	80.0 ns	90.0 ns	100,0 r	18
1	a	8	00	X	01	10	X	11)	00	X	01 X	10)	11	X	<u>х о</u>	01 X	10
	⊞ b	B	00	X	01	10		11	00	X	01 X	10	(11	X	X 00	01 X	10
63.	El and	8.0		-	0000		Y	0001	010	0 Y	1001 ¥	0000	000	1 ¥ 0	100 ¥ 1	001	0000

Figure 4.1 : Timing Simulation for 2-bit Ripple Carry Multiplier

The diagram below, Figure 4.2 shows the worst case propagation delay which is 20.7 ns. The propagation delay occurs from input a[1] to output prod[3].

Compilation Report	Ti	ming Analyzer Summary					
Elegal Notice		Туре	Slack	Required Time	Actual Time	From	To
Flow Settings	1	Worst-case tpd	N/A	None	20.700 ns	a[1]	prod[3
Flow Non-Default Global Setting	2	Total number of failed paths					

Figure 4.2 : Worst-Case Propagation Delay Time

Compilation Report - tpd						
Compilation Report	tpo					alion and a second
Legal Notice		Slack	Required P2P Time	Actual P2P Time	From	То
Flow Settings	1	N/A	None	20.700 ns	a[0]	prod[3]
Flow Non-Default Global Settings	2	N/A	None	20.700 ns	b[1]	prod[3]
Flow Elapsed Time	3	N/A	None	20.700 ns	a[1]	prod[3]
	4	N/A	None	20.500 ns	Ь[0]	prod[2]
+ Analysis & Synthesis	5	N/A	None	20.500 ns	Ь[1]	prod[2]
+ Assembler	6	N/A	None	20.500 ns	a[1]	prod[2]
- Analyzer	7	N/A	None	20.500 ns	a[0]	prod[1]
Summary	8	N/A	None	20.500 ns	Ь[0]	prod[1]
Settings	9	N/A	None	20.500 ns	b[1]	prod[1]
tpd	10	N/A	None	20.500 ns	a[0]	prod[0]
Messages	11	N/A	None	20.400 ns	b[0]	prod[3]
	12	N/A	None	20.200 ns	a[0]	prod[2]
	13	N/A	None	20.200 ns	a[1]	prod[1]
	14	N/A	None	20.200 ns	b[0]	prod[0]

Figure 4.3 : Propagation Delay Time for the Design

Besides the propagation delay time, the number of logic elements for the design can also be obtained from the compilation report. For 2-by-2 Ripple Carry Multiplier, the number of logic elements is determined to be four.

4.1.1.2 4-bit Ripple Carry Multiplier

Based on the timing analyzer report, the worst-case propagation delay is 51.8 ns. It occurs between the time a signal is propagated from input b[0] to output prod[7]. Attached in Appendix 1A is the simulation result for the 4-by-4 bits multiplication process. From the compilation report, the number of logic elements used in this design is 29 which is less than 1% of the total logic elements available.

4.1.1.3 6-bit Ripple Carry Multiplier

From the timing analyzer report, the worst-case propagation delay time is 90.8 ns. It occurs from input a[0] to output prod[10]. Attached in Appendix 1B is the simulation result for the multiplication process. For 6-bit Ripple Carry multiplier, the number of logic elements is 69 out of 3744 logic elements. This is about 2% of the total logic elements available.

4.1.1.4 8-bit Ripple Carry Multiplier

For 8-by-8 bits multiplication process, the worst-case propagation delay is 124.0 ns. This occurs during the time a signal is sent from input a[0] to output prod[15]. Attached in Appendix 1C is the simulation result for the multiplication process. In 8-bit Ripple Carry multiplication process, the number of logic elements used is 130 which is 3% of the total logic elements available in EPF10K70RC240-4.

4.1.2 Carry Save Multiplier

This section shows the simulation results for 2-bit, 4-bit, 6-bit and finally 8-bit Carry Save Multiplier.

4.1.2.1 2-bit Carry Save Multiplier

The figure below, Figure 4.4 shows the result of the timing simulation for a 2-by-2 bit multiplication process which adopts the carry save multiplication method. There are 4 logic elements used in this design as determined in the compilation report.

Mast	er Time Bar:	20.0 ns	4 1	Point	er:	41.4 ni	8	Interval	2	1.4 ns	Start			End	
	Name	Valu 20.	0 ps	10.p n	a 20.0	ns :	en ().08	40.p	ns 50	Qins	60.0 ns	70.0 ns	80.0 ns	90.0 na	100,0 ns
2	+ a	В	00	X	01	10	X	11 X	00	X 01	X 10	χ1	1 X 0	o x c	11 X 10
S.	🛨 b	В	00	X	01	10	X	11 X	00	01	X 10	χ 1	1 X 0	IO X C	01 X 10
62	F prod	BC	-		0000		YO	001 Y	0100	¥ 100	1 1 000	00 Y 00	01 ¥ 0	100 ¥ 10	001 0000

Figure 4.4 : Timing Simulation for 2-bit CS multiplier

Figure 4.5 shows the worst propagation delay for the multiplication process which is 20.7 ns. The delay occurs between input a[1] to output prod[3].

Compilation Report - Ti	mi Ti	ng Analyzer Summary ming Analyzer Summary					
E Legal Notice		Туре	Slack	Required Time	Actual Time	From	То
Flow Settings	1	Worst-case tpd	N/A	None	20.700 ns	a[1]	prod[3]
Flow Non-Default	2	Total number of failed paths					
< President President >	<						>

Figure 2.5 : Worst-Case Propagation Delay

Compilation Report	tpo					
Legal Notice		Slack	Required P2P Time	Actual P2P Time	From	То
Flow Settings	1	N/A	None	20.700 ns	a[0]	prod[3]
Flow Non-Default Global Settings	2	N/A	None	20.700 ns	b[1]	prod[3]
Flow Elapsed Time	3	N/A	None	20.700 ns	a[1]	prod[3]
	4	N/A	None	20.500 ns	b[0]	prod[2]
Analysis & Synthesis	5	N/A	None	20.500 ns	b[1]	prod[2]
	6	N/A	None	20.500 ns	a[1]	prod[2]
- A Timing Analyzer	7	N/A	None	20.500 ns	a[0]	prod[1]
Summary	8	N/A	None	20.500 ns	Ь[0]	prod[1]
Settings	9	N/A	None	20.500 ns	Ь[1]	prod[1]
tpd 💭	10	N/A	None	20.500 ns	a[0]	prod[0]
(i) Messages	11	N/A	None	20.400 ns	b[0]	prod[3]
	12	N/A	None	20.200 ns	a[0]	prod[2]
	13	N/A	None	20.200 ns	a[1]	prod[1]
	14	N/A	None	20,200 ns	ыот	[0]borg

Figure 4.6 displays all the propagation delay in the design.

Figure 4.6 : Propagation Delays for the Design

4.1.2.2 4-bit Carry Save Multiplier

The worst-case propagation delay for a 4-by-4 bit Carry Save Multiplier is 47.6 ns. The delay happens between input b[0] and output prod[7]. The simulation result is attached in Appendix 2A. The design has 28 logic elements. Since EPF10K70RC240-4 contains 3744 logic elements, this design uses less than 1% of the total logic elements.

4.1.2.3 6-bit Carry Save Multiplier

For the 6-by-6 bits Carry Save multiplication process, the longest propagation delay time is 70.9 ns. The delay occurs between input a[4] and output prod[7]. The simulation result is attached in Appendix 2B. For 6-bit Carry Save multiplier, the total number of logic elements used in the design is 71 which is about 2% of the total logic elements available in EPF10K70RC240-4.

4.1.2.4 8-bit Carry Save Multiplier

As for the 8-by-8 Carry Save multiplication process, the worst-case propagation delay associated with the design is determined to be 89.2 ns. The propagation delay takes place between input a[5] and output prod[15]. Attached in Appendix 2C is the simulation result. From the compilation report, the number of logic elements used in this design is 130 logics. This is about 3% of the total logic elements.

4.1.3 Wallace Multiplier

This section shows the simulation result of 2-bit, 4-bit, 6-bit and 8-bit Wallace multiplication process.

4.1.3.1 2-bit Wallace Multiplier

Figure 4.7 shows the output from the timing simulation process. In Figure 4.8, the propagation delay is determined to be 20.7 ns from input a[1] to output prod[3]. This design contains 4 logic elements as obtained from the compilation report.

Simulation Report	Sum	ulation Wavel										
E Legal Notice	Sim	lation mode: Tim	sing									
Flow Settings Simulator	Mast	er Time Bar:	18.55	15	+ + Pointer:	63.6 ns	Interval	45.05 ns Start	End	Endt		
Settings Settings Simulation Wavef		Name		Valu 18.5	0 ps	20.0 ns 18.55 ns	40. 0 ns	60.p.ns	80.0 ns	100 _, 0 ns		
+ Simulation Covera	1	€ a		B	00 X 0	1 X 10	(11 X 00	X D1 X 10 X	11 X 00 X	01 X 1		
Messages	2	æ þ		В	00 00	1 X 10	(11) 00	X 01 X 10 X	11 X 00 X	01 1		
		E orod		BC	00	00	X 0001 X 010	0 X 1001 X 0000)	0001 1 0100	1001 00		

Figure 4.7 : Timing Simulation for 2-bit Wallace multiplier

Figure 4.8 below shows the worst-case propagation delay in the design.

		ming Analyzer Summary					
Legal Notice		Туре	Slack	Required Time	Actual Time	From	To
Flow Settings	1	Worst-case tpd	N/A	None	20.700 ns	a[1]	prod[3
Flow Non-Default	2	Total number of failed paths					

Figure 4.8 : Worst-Case Propagation Delay Time

Figure 4.9 display all the propagation delays that occur during the simulation of the design.

Compilation Report - tp	bd					
Compilation Report	tpo					
Legal Notice		Slack	Required P2P Time	Actual P2P Time	From	To
Flow Settings	1	N/A	None	20.700 ns	a[0]	prod[3]
Flow Non-Default Glo	2	N/A	None	20.700 ns	Ь[1]	prod[3]
Flow Elapsed Time	3	N/A	None	20.700 ns	a[1]	prod[3]
Flow Log	4	N/A	None	20.500 ns	Ь[0]	prod[2]
	5	N/A	None	20.500 ns	Ь[1]	prod[2]
+ Assembler	6	N/A	None	20.500 ns	a[1]	prod[2]
- Analyzer	7	N/A	None	20.500 ns	a[0]	prod[1]
Summary	8	N/A	None	20.500 ns	b[0]	prod[1]
Settings	9	N/A	None	20.500 ns	b[1]	prod[1]
tpd	10	N/A	None	20.500 ns	a[0]	prod[0]
(i) Messages	11	N/A	None	20.400 ns	Ь[0]	prod[3]
	12	N/A	None	20.200 ns	a[0]	prod[2]
	13	N/A	None	20.200 ns	a[1]	prod[1]
()	14	N/A	None	20.200 ns	b[0]	prod[0]

Figure 4.9 : Total Propagation Delay for the Design

4.1.3.2 4-bit Wallace Multiplier

For the 4-by-4 bit Wallace Multiplier, the worst-case propagation delay is determined to be 48.0 ns from input b[0] to output prod[6]. The simulation result is attached in Appendix 3A. The 4-bit Wallace multiplication process yields 28 logic elements in the design which is less than 1% of the available total logic elements.

4.1.3.3 6-bit Wallace Multiplier

For 6-by-6 bits Wallace multiplication process, the worst-case propagation delay is 71.4 ns. The delay occurs between input b[4] and output prod[10]. The simulation result is attached in Appendix 3B. For this design, the

number of total logic elements is 76. This is 2% of the total logic elements available in EPF10K70RC240-4.

4.1.3.4 8-bit Wallace Multiplier

For the multiplication process of an 8-by-8 bit Wallace multiplier, it is determined from the report that the worst-case propagation delay is 87.8 ns. The delay occurs from input b[1] to output prod[10]. Attached in Appendix 3C is the simulation results. The number of logic elements used in this design is 153 which are 4% of the available total logic elements.

4.1.4 Dadda Multiplier

This section shows the simulation result for 2-bit, 4-bit, 6-bit and 8-bit Dadda multiplier.

4.1.4.1 2-bit Dadda Multiplier

Figure 4.10 shows the timing simulation result. Notice that the output is delayed for a while. For 2-bit Dadda multiplier, the design uses 4 logic elements.



Figure 4.10 : Timing Simulation for a 2-bit Dadda multiplier

Figure 4.11 shows the worst-case propagation delay in the design. From the report, the delay is determined to be 20.7 ns from input a[1] to output prod[3].

+ 🕘 🗋 Fitter	~	Tir	ning Analyzer Summary					
+ Assembler - Assembler Timing Analyzer	_		Туре	Slack	Required Time	Actual Time	From	To
Summary	_	1	Worst-case tpd	N/A	None	20.700 ns	a[1]	prod[3]
Settings	=	2	Total number of failed paths					
tpo								

Figure 4.11 : Worst-Case Propagation Delay Time

Figure 4.12 shows all the propagation delays that occurred during the design simulation.

Compliation Report - 4						
Compilation Report	tpo					
E Legal Notice		Slack	Required P2P Time	Actual P2P Time	From	То
Flow Settings	1	N/A	None	20.700 ns	a[0]	prod[3]
Flow Non-Default Glo	2	N/A	None	20.700 ns	Ь[1]	prod[3]
Flow Elapsed Time	3	N/A	None	20.700 ns	a[1]	prod[3]
Flow Log	4	N/A	None	20.500 ns	b[0]	prod[2]
Eitter	5	N/A	None	20.500 ns	b[1]	prod[2]
Assembler	6	N/A	None	20.500 ns	a[1]	prod[2]
Timing Analyzer	7	N/A	None	20.500 ns	a[0]	prod[1]
Summary	8	N/A	None	20.500 ns	b[0]	prod[1]
Settings	9	N/A	None	20.500 ns	b[1]	prod[1]
tpd	10	N/A	None	20.500 ns	a[0]	prod[0]
(A) Messages	11	N/A	None	20.400 ns	b[0]	prod[3]
	12	N/A	None	20.200 ns	a[0]	prod[2]
	13	N/A	None	20.200 ns	a[1]	prod[1]
	14	N/A	None	20.200 ns	b[0]	prod[0]

Figure 4.12 : Total Propagation Delay for the Design

4.1.4.2 4-bit Dadda Multiplier

Simulation of the 4-by-4 bits Dadda multiplier yields a worst-case propagation delay of 50.3 ns. The delay happens between input b[0] to output prod[6]. The simulation results are attached in Appendix 4A. From the compilation report, the number of logic element in the design is determined to be 27 which is less than 1% of the total logic elements.

4.1.4.3 6-bit Dadda Multiplier

For 6-by-6 bits Dadda Multiplier, the worst-case propagation delay is determined to be 73.0 ns. The delay occurs between input a[4] and output prod[10]. The simulation results are attached in Appendix 4B. For 6-bit Dadda multiplication process, the number of logic elements used in the design is 61 which is about 2% of the total logic elements.

4.1.4.4 8-bit Dadda Multiplier

The simulation process of an 8-by-8 bits Dadda multiplier yields a worstcase propagation delay of 88.5 ns. The delay occurs between input b[0] to output prod[10]. The simulation results are attached in Appendix 4C. In the 8-bit Dadda multiplication process, 157 logic elements are used. This is 4% of the total logic elements in EPF10K70RC240-4.

4.1.5 Implementation on FPGA

The implementation on FPGA part of the project is successful. The UP2 board which contains the FPGA EPF10K70RC240-4 is connected to the computer via ByteBlaster II cable. The board is also connected to the power supply. Designs that have been created are downloaded onto the EPF10K70RC240-4 via the ByteBlaster II cable and the resulting outputs are observed. The inputs of the multiplier are user-defined whereby the inputs are assigned to the DIP switches available on the UP2 board. The DIP switches give logic '0' when they are pressed down and vice versa. The output of the multiplier can be observed through the seven-segment LEDs. The LEDs of the seven segment are active low which means that the LEDs light up when the output is a logic '0' and turns off when the output is logic '1'. To observe the output waveforms that result from the FPGA, a monitor is connected to the board via VGA port. However, even after proper settings have been done, no waveforms can be observed despite the correct outputs observed on the seven-segment. Appendix 5 shows one of the implementation results obtained.

4.2 DISCUSSION

This section discusses the result shown in the previous section.

4.2.1 Summary of Results

Table 4.1 summarizes the results obtained from the simulation process.

N bits	Ripple Carry	Carry Save	Wallace	Dadda
2	20.7 ns	20.7 ns	20.7 ns	20.7 ns
4	51.8 ns	47.6 ns	48.8 ns	50.3 ns
6	90.8 ns	70.9 ns	71.4 ns	73.0 ns
8	124.4 ns	89.2 ns	87.8 ns	88.5 ns

Table 4.1 : Propagation Delay



Figure 4.13 : Propagation Delay Chart

For the 2-bit results of each multiplier, it yields the same amount of propagation delay. This is because all 2-bit binary multipliers have the same adder architecture.

As the number of bits, n increases, propagation delays for all multipliers also increase as expected. Based on the table above, Ripple Carry multiplier has the worst performance among all four multipliers. This is due to the fact that one adder cannot begin the addition process until it receives the necessary carry in bit from the adder of the previous stage.

Carry save multiplier shows the best performance of all four multipliers analyzed. As mentioned in Chapter 2, carry save multiplier does not have to wait to begin the next partial product summation process as the carry bits are forwarded to the next layer of adders, which explains the small propagation delay experienced by this particular multiplier.

As for both Wallace and Dadda multipliers, in order to compare their performance, the level of reduction must also be taken into account. However, during the designing stage, care has already been taken to ensure that both are reduced with the same number of reduction stages. From the results shown, the Wallace multiplier has a smaller propagation delay compared to Dadda multiplier although the difference is not really huge.

Table 4.2 shows the number of logic elements used in the designs of 2-bit, 4-bit, 6-bit and 8-bit multipliers. As shown in Table 4.2, Dadda multiplier uses the least logics out of the four multipliers studied up until the 6-bit multiplication process. For 8-bit multiplication process, both Ripple Carry multiplier and Carry Save multiplier have the least number of logic gates which is 130 logic elements. However, between the two multipliers, Carry Save wil provide better performance because it has the smallest propagation delay although same number of logic elements.

N bits	Ripple Carry	Carry Save	Wallace	Dadda
2	4	4	4	4
4	29	28	28	27
6	69	71	76	61
8	130	130	153	157

Table 4.2 : Number of Logic Elements



Figure 4.14 : Number of Logic Elements Chart

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

To conclude the report, the importance of adder speed is again stressed. Speed is a major contributor towards the successful implementation of a digital multiplier. As such, the speed factor must be addressed properly. A smaller propagation delay means the shorter time response for the multiplication. In this study, ripple carry multiplier has given the largest propagation delay and carry save multiplier has shown the smallest delay among all multiplers studied whereas Dadda multiplier has the least amount of logic elements up until 6-bit multiplication process. For 8-bit multiplication process, both ripple carry multiplier and carry save multiplier use the same number of logic elements which is 130 logics.

5.2 Recommendation

This project can further be enhanced by adding several types of multipliers. The comparison between many types of multipliers will enable the industry to select the most reliable multiplier. Besides that, this project can also be expanded to compare these multipliers in terms of area and power consumption. Most multipliers usually come with advantages and disadvantages. One particular multiplier is able to yield small propagation delay but it can also be at a disadvantage in terms of area whereby it needs more space to accommodate a lot of logic elements. Therefore, it would be a great enhancement to the project if all aspects (speed, area and power) are taken into account. Another enhancement that can be done is to increase the number of bits compared to at least 64 bits to really be able to see the propagation delay of each multiplier.

44

REFERENCES

- [1] T. L. Floyd, 2003, Digital Fundamentals, New Jersey, Prentice Hall
- [2] D. Ciletti, 2002, Advanced Digital Design with Verilog HDL, New Jersey, Prentice Hall.
- [3] <http://www.cs.umd.edu/class/spring2003/cmsc311/Notes/Comb/ adder.html> *Retrieved 1 November 2007*
- [4] Prof. Loh, Processor Design, February 2005
- [5] <http://www.fpga-guru.com/multipli.htm> Retrieved 13 December 2007
- [6] <http://cag-www.lcs.mit.edu/6.004/Lectures/lect17/sld015.htm> Retrieved 12 December 2007
- [7] W. J. Townsend, E.E. Swartzlander, Jr., J.A. Abraham," A Comparison of Dadda and Wallace Multiplier Delays", 2003
- [8] K.C. Bickerstaff, E. E. Swartzlander, Jr."Analysis of Column Compression Multipliers", 2001

APPENDIX

.

APPENDIX A1

Simulation Result for a 4-bit Ripple Carry Multiplier

D 1	c_arraymult.	vwf		U B						W. R.		1 June	
Mast	er Time Bar	20.0 ns	4	+ Pointer:	92.2	ns	Intervat	72.2 ns	Start			End:	
	Name	Va 20	0 ps	10.0 ns	20.0 ns	30.0 n	s 40.0 ns	50 0 ns	60.0 ns	70 0 ns	80.0 ns	90.0 ns	100 _, 0 ns
	t a t b t prod	B B B OC		00 X 00 00 X 00 00	01 00	10 χ 10 χ	0011 X 01 0011 X 01 X00(\$K)X0	00 X 010 00 X 010 001,20,000	01 X 011 01 X 011 DOOMOOMO	0 X 01 0 X 01 0 X 01	11 X 10 11 X 10 5 X 10 X X		001 X 1010 001 X 1010
<		\$	<		l)

Timing Simulation for 4-bit RC Multiplier

Compilation Report	Ti	ming Analyzer Summary					
Legal Notice		Туре	Slack	Required Time	Actual Time	From	To
Flow Settings	1	Worst-case tpd	N/A	None	51.800 ns	Ь[0]	prod
Flow Non-Default Global Setting	2	Total number of failed paths					

Worst Case Propagation Delay Time for 4-bit RC Multiplier

🗢 Compilation Report - tpd			S. Contraction	and the second			
Compilation Report	tpo					And the second second	
Elegal Notice		Slack	Required P2P Time	Actual P2P Time	From	То	~
Flow Settings	1	N/A	None	51.800 ns	b[0]	prod[7]	
Flow Non-Default Global Settings	2	N/A	None	50.500 ns	b[0]	prod[6]	
Flow Elapsed Time	3	N/A	None	48.000 ns	b[0]	prod[5]	
Flow Log	4	N/A	None	45.000 ns	a[1]	prod[7]	
Analysis & Synthesis	5	N/A	None	45.000 ns	a[2]	prod[7]	
+ Assembler	6	N/A	None	44.800 ns	a[3]	prod[7]	
- Timing Analyzer	7	N/A	None	44.700 ns	a[0]	prod[7]	
Summary	8	N/A	None	44.400 ns	b[1]	prod[7]	Levi
Settings	9	N/A	None	43.700 ns	a[1]	prod[6]	a di Carl
tpd	10	N/A	None	43.700 ns	a[2]	prod[6]	
(A) Messages	11	N/A	None	43.500 ns	a[3]	prod[6]	
	12	N/A	None	43.400 ns	a[0]	prod[6]	
	13	N/A	None	43.100 ns	b[1]	prod[6]	
	14	N/A	None	42.200 ns	b[0]	prod[4]	
	15	N/A	None	41.400 ns	b[2]	prod[7]	
	16	N/A	None	40.900 ns	a[0]	prod[5]	
	17	N/A	None	40.900 ns	a[1]	prod[5]	
	18	N/A	None	40.900 ns	a[2]	prod[5]	
	19	N/A	None	40.700 ns	a[3]	prod[5]	
	20	N/A	None	40.600 ns	b[1]	pred(5)	
	21	N/A	None	40.100 ns	b[2]	prod[6]	
	22	N/A	None	39.600 ns	b[0]	pred[3]	
	23	N/A	None	38.200 ns	b[3]	prod[7]	
	24	N/A	None	37.600 ns	b[2]	prod[5]	
	25	N/A	None	36.900 ns	P[3]	prod[6]	
	26	N/A	None	35.100 ns	a[0]	prod[4]	~

Propagation Delay Time in the design for 4-bit RC Multiplier

APPENDIX 1B

Simulation Result for a 6-bit Ripple Carry Multiplier

🖬 s	ixbit_rcarra	ymult.vwf				1.1		Charles and	_ 🗆 🛛
Mast	er Time Bar:	47.5 ns	• Pointer:	37.6 ns	Interval	-9.9 ns	Start:	End	
			0 ps	20.0 ns	40.0	ns	60.0 ns	80.0 ns	100.0 ns
	Name					47.5 ns			
	æ a	E	(000000 X 000	0001 X 00001	0 X 000011 X	000100 X 00	0101 000110	000111 001000	X 001001 X10
D)	⊞ b	٤	000000 000	0001 00001	0 X 000011 X	000100 X 00	00101 0000110	000111 001000	X 001001 X10
1	🛨 prod	B 00	00	000000000000000000000000000000000000000)\$00	000000000000000000000000000000000000000	XOMXXXXXXXXXX		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
<		>	< 1 mm						>

Timing Simulation for 6-bit RC Multiplier

Compilation Report - Ti	mi Ti	ng Analyzer Summary ming Analyzer Summary				-	
Legal Notice		Туре	Slack	Required Time	Actual Time	From	To
Flow Settings	1	Worst-case tpd	N/A	None	90.800 ns	a[0]	prod[10]
Flow Non-Default	2 <	Total number of failed paths		1			>

Worst Case Propagation Delay Time for 6-bit RC Multiplier

· compliation report tp							<u>ماركانك</u>
Compilation Report	tpd		In the second second				
E Legal Notice		Slack	Required P2P Time	Actual P2P Time	From	То	^
Flow Settings	1	N/A	None	90.800 ns	a[0]	prod[10]	
Flow Non-Default Glo	2	N/A	None	90.700 ns	a[0]	prod[11]	
Flow Elapsed Time	3	N/A	None	88.100 ns	b[0]	prod[10]	
Flow Log	4	N/A	None	88.000 ns	b[0]	prod[11]	
+ Analysis & Synthesis	5	N/A	None	87.500 ns	a[0]	prod[9]	
+ Assembler	6	N/A	None	84.800 ns	b[0]	prod[9]	
- A Timing Analyzer	7	N/A	None	82.900 ns	a[0]	prod[8]	
Summary	8	N/A	None	82.300 ns	a[1]	prod[10]	
Settings	9	N/A	None	82.200 ns	a[1]	prod[11]	
bot 50	10	N/A	None	82.000 ns	Ь[1]	prod[10]	
(A) Messages	11	N/A	None	81.900 ns	b[1]	prod[11]	
	12	N/A	None	81.900 ns	a[4]	prod[10]	
	13	N/A	None	81.800 ns	a[4]	prod[11]	
	14	N/A	None	80.300 ns	e[2]	prod[10]	
	15	N/A	None	80.200 ns	a[2]	prod[11]	
	16	N/A	None	90.200 ns	b[0]	prod[8]	
	17	N/A	None	79.000 ns	a[1]	prod[9]	
	18	N/A	Nene	78.700 ns	b[1]	pred[9]	
	19	N/A	None	78.600 ns	a[4]	prod[9]	
< > >	20	N/A	None	78.200 ns	a[3]	prod[10]	~

Propagation Delay Time in the design for 6-bit RC Multiplier

APPENDIX 1C

Simulation Result for a 8-bit Ripple Carry Multiplier

Simulation Report	Sim	ulation Waveforms		
Legal Notice Flow Summary Flow Settings Simulator	Sim	ulation mode: Timing		
Settings	Mast	er Time Bar: 17.425	ó ns	Pointer: 120.4 ns Interval 102.98 ns Start: End:
Settings Simulation Wavef Simulation Covera INI Usage Simulation Covera		Name	Γ	0 ps 10.0 ns 20.0 ns 30.0 ns 40.0 ns 50.0 ns 60.0 ns 70.0 ns 80.0 ns 90.0 ns 100,0 n 17.425 ns
C A Messages		et a		280808062008888942088881020000132088881920888102020888112208881122088811322888918920888811122088811122088811052

Timing Simulation for 8-bit RC Multiplier

Compilation Report - Timing Analy	/ze	r Summary					
Compilation Report	Ti	ming Analyzer Summary					
Legal Notice		Туре	Slack	Required Time	Actual Time	From	To
Flow Settings	1	Worst-case tpd	N/A	None	124,400 ns	a[0]	prod[15]
Flow Non-Default Global Setting	2	Total number of failed paths				and frankline star	
Flow Log	<			1			>

Worst Case Propagation Delay Time for 8-bit RC Multiplier

Compilation Report	tpd						
E Legal Notice		Slack	Required P2P Time	Actual P2P Time	From	To	1
Flow Settings	1	N/A	None	124.400 ns	a[0]	prod[15]	1
Flow Non-Default Global Settings	2	N/A	None	122.800 ns	6[0]	prod[15]	
Flow Elapsed Time	3	N/A	None	121,900 ns	a[0]	prod[14]	
E Flow Log	4	N/A	None	120.300 ns	b[0]	prod[14]	
Analysis & Synthesis	5	N/A	None	120.000 ns	a[0]	prod[13]	
Assembler	6	N/A	None	118.400 ns	b[0]	prod[13]	
Timing Analyzer	7	N/A	None	117.000 ns	a[1]	prod[15]	
Summary	8	N/A	None	116.900 ns	b[1]	prod[15]	
Settings	9	N/A	None	116.500 ns	a[2]	prod[15]	
tpd	10	N/A	None	115.100 ns	a[4]	prod[15]	
(I) Messages	11	N/A	None	114.500 ns	a[1]	prod[14]	
	12	N/A	None	114.400 ns	b[1]	prod[14]	
	13	N/A	None	114.400 ns	a[0]	prod[12]	
	14	N/A	None	114.100 ns	a[3]	prod[15]	
	15	N/A	None	114.000 ns	a[2]	prod[14]	
	16	N/A	None	112.800 ns	b[0]	prod[12]	
	17	N/A	None	112.600 ns	a[4]	prod[14]	
	18	N/A	None	112.600 ns	a[1]	prod[13]	1

Propagation Delay Time in the design for 8-bit RC Multiplier

APPENDIX 2A

Simulation Result for a 4-bit Carry Save Multiplier

Simulation Report	Sim	ulation Waveforms			
Legal Notice	Sim	lation mode: Timing			1.00
Flow Settings	Mast	er Time Bar: 17.4	25 ns	◆ Pointer: 40.0 ns Intervat 22.58 ns Start: 0 ps End:	0 ps
Summary Settings Simulation Wavef		Name	Va 17	0 ps 10.0 ms 20.0 ms 30.0 ms 40.0 ms 50.0 ms 60.0 ms 70.0 ms 80.0 ms 90.0 ms 17.425 ms	100 _, 0 na
+ Simulation Covera		E a	8	0000 X 0001 X 0010 X 0011 X 0100 X 0101 X 0110 X 0101 X 1000 X 100	1 X 1
(A) Messages		🛨 b	B	C0000 X 0001 X 0010 X 0011 X 0100 X 0101 X 0110 X 0111 X 1000 X 100	1 X 1
		e prod	B 00		01) (/

Timing Simulation for a 4-bit CS multiplier

🗣 Compilation Report - T	imi	ng Analyzer Summary					
🞒 Compilation Report 🛛 💦	Ti	ming Analyzer Summary					
E Legal Notice		Туре	Slack	Required Time	Actual Time	From	To
Flow Settings	1	Worst-case tpd	N/A	None	47.600 ns	Ь[0]	prod[7]
Flow Non-Default	2	Total number of failed paths					
Flow Log	<						>

Worst-Case Propagation Delay Time

Compilation Report - tp	bd							
Compilation Report	tpo							
Elegal Notice		Slack	Required P2P Time	Actual P2P Time	From	To	^	
Flow Settings	1	N/A	N/A	None	47.600 ns	b[0]	prod[7]	
Flow Non-Default Glo	2	N/A	None	47.000 ns	b[0]	prod[6]		
Flow Elapsed Time	3	N/A	None	44.200 ns	a[3]	prod[7]		
Flow Log	4	N/A	None	43.800 ns	a[1]	prod[7]		
	5	N/A	None	43.700 ns	a[2]	prod[7]		
+ Assembler	6	N/A	None	43.600 ns	a[3]	prod[6]		
E A Timing Analyzer	7	N/A	None	43.400 ns	b[1]	prod[7]		
Summary	8	N/A	None	43.200 ns	a[1]	prod[6]		
Settings	9	N/A	None	43.100 ns	a[2]	prod[6]		
tpd	10	N/A	None	42.800 ns	b[1]	prod[6]		
Messages	11	N/A	None	42.300 ns	b[0]	prod[5]		
	12	N/A	None	42.300 ns	b [0]	prod[4]		
	13	N/A	None	40.100 ns	a[0]	prod[7]		
	14	N/A	None	39.500 ns	a[0]	prod[6]		
	15	N/A	None	39.500 ns	a[3]	prod[5]		
	16	N/A	None	38.500 ns	a[1]	prod[5]		
	17	N/A	None	38.500 ns	a[1]	prod[4]		
	18	N/A	None	38.400 ns	a[2]	prod[5]		
	19	N/A	None	38.400 ns	a[2]	prod[4]		
<	20	N/A	None	38.100 ns	b[1]	prod[5]	~	

Propagation Delay Time in the design for 8-bit Carry Save Multiplier

APPENDIX 2B

Simulation Result for a 6-bit Carry Save Multiplier

Simulation Report	Sim Sim	ulation Wavefo dation mode: Timi	ng			1				
Flow Settings	Mas	er Time Bar:	0 ps	• Pointer	58.2 ns	Interval	58.2 ns	Start	En	d:
Summary Settings Simulation Wavef		Name	0 ps 0 ps	20.0 ns		40.0 ns	60.pr	is	80.0 ns	100,0 ns
+ Jimulation Covera INI Usage S Messages		te a te b te prod	0000	00 X 000001 X 00 00 X 000001 X 00 0000000000	0010 X 000 0010 X 000 00	011 X 000100 011 X 000100 X00000000	X 000101 X 0 X 000101 X 0 X 000X 00X 0		01111 X 001000 X 01111 X 001000 X 00/00/0001X////	001001 X 0010 001001 X 0010

Timing Simulation for a 4-bit CS multiplier

Compilation Report - Ti	mi	ng Analyzer Summary					
Scompilation Report	Ti	ming Analyzer Summary					
E Legal Notice		Туре	Slack	Required Time	Actual Time	From	To
Flow Settings	1	Worst-case tpd	N/A	None	70.900 ns	a[4]	prod[11]
Flow Non-Default	2	Total number of failed paths					
	<						>

Worst-Case Propagation Delay Time

Compilation Report	tpd						
Legal Notice		Slack	Required P2P Time	Actual P2P Time	From	То	1
Flow Settings	1	N/A	None	70.900 ns	a[4]	prod[11]	
Flow Non-Default Glo	2	N/A	None	69.600 ns	a[4]	prod[10]	
Flow Elapsed Time	3	N/A	None	69.400 ns	a[5]	prod[11]	
Flow Log	4	N/A	None	68.100 ns	a[5]	prod[10]	
Analysis & Synthesis	5	N/A	None	66.300 ns	b[0]	prod[11]	
Assembler	6	N/A	None	66.100 ns	b[1]	prod[11]	
Timing Analyzer	7	N/A	None	66.100 ns	a[2]	prod[11]	
Summary Settings Settings tpd Messages	8	N/A	None	65.000 ns	b[0]	prod[10]	
	9	N/A	None	64.800 ns	b[1]	prod[10]	
	10	N/A	None	64.800 ns	a[2]	prod[10]	
	11	N/A	None	64.400 ns	a[4]	prod[9]	
	12	N/A	None	63.900 ns	a[0]	prod[11]	
	13	N/A	None	62.900 ns	a[5]	prod[9]	
	14	N/A	None	62.600 ns	a[0]	prod[10]	
	15	N/A	None	61.600 ns	a[3]	prod[11]	
	16	N/A	None	61.500 ns	a[4]	[8]betq	
	17	N/A	None	60.400 ns	b[2]	prod[11]	
	18	N/A	None	60.300 ns	a[3]	prod[10]	
	19	N/A	None	60.000 ns	a[5]	prod[8]	
	20	N/A	None	59.800 ns	b[0]	prod(9)	3

Propagation Delay Time in the design for 8-bit Carry Save Multiplier

APPENDIX 2C

Simulation Result for a 8-bit Carry Save Multiplier

Report	Sim	ulation Wave	form														
lotice ummary	Şim	ulation mode: Ti	iming														
ettings or	Mas	ter Time Bar:		18.55 ns	4	Pointer:	ę	55.8 ns	Interv	ət	37.25 ne	Start	1		End	t	
nmar y			0 ps		10.0 ns	1	20.0 m	5	30.0 m	-	40.0 ns		50.0 ns		60.0 ns		70.0 ns
ulation V		Name				18	55 ns										
ulation C	1	E a		00000000	X	00000001	Tx	00000010	X	8000001	X	00000100	X	00000101	X	00000110	2000001
Usage	10	æ b		00000000	X	0000001	TX	00000010	X	00000011	X	00000100	X	00000101	X	00000110	000001
114444	1	El prod				0000000	000000	986		Y	M980088	53/53/0/0655	CHOYOGE	00000 Micke	SY05YD	AND TO A TO A TO A	100000000

Timing Simulation for 8-bit Carry Save multiplier

😃 Compilation Report -	Tir	ning Analyzer Summary				-						
Compilation Report	Ti	Timing Analyzer Summary										
Legal Notice		Туре	Slack	Required Time	Actual Time	From	To					
Flow Settings	1	Worst-case tpd	N/A	None	89.200 ns	a[5]	prod[15]					
Flow Non-Defat Flow Elapsed Tir	2	Total number of failed paths										
< >	<						2					

Worst-Case	Propagation	Delay Time	ŝ
------------	--------------------	------------	---

Compilation Report -	tpd	-	15.30	Second Second	-		
Compilation Report	tpd						
Legal Notice		Slack	Required P2P Time	Actual P2P Time	From	To	^
Flow Settings	1	N/A	None	89.200 ns	a[5]	prod[15]	
Flow Non-Default G	2	N/A	None	88.600 ns	a[7]	prod[15]	
Flow Elapsed Time	3	N/A	None	87.400 ns	a[5]	prod[14]	
Flow Log	4	N/A	None	86.800 ns	b[0]	prod[15]	
Analysis & Synthesi	5	N/A	None	86.800 ns	a[7]	prod[14]	
+ Assembler	6	N/A	None	86.000 ns	a[5]	prod[13]	
- A Timing Analyzer	7	N/A	None	85.900 ns	a[3]	prod[15]	
Summary	8	N/A	None	85.900 ns	a[6]	prod[15]	
Settings	9	N/A	None	85.400 ns	a[7]	prod[13]	
tpd	10	N/A	None	85.000 ns	b[0]	prod[14]	
Messages	11	N/A	None	84.100 ns	a[4]	prod[15]	
	12	N/A	None	84.100 ns	a[3]	prod[14]	
	13	N/A	None	84.100 ns	a[6]	prod[14]	
	14	N/A	None	83.600 ns	b[0]	prod[13]	
	15	N/A	None	82.700 ns	a[3]	prod[13]	
	16	N/A	None	82.700 ns	a[6]	prod[13]	
	17	N/A	None	82.500 ns	b[1]	prod[15]	
	18	N/A	None	82.300 ns	a[4]	prod[14]	
	19	N/A	None	81.200 ns	a[5]	prod[12]	
< 1 Mit >	20	N/A	None	80.900 ns	a[4]	prod[13]	~

Total Propagation Delays in the Design

APPENDIX 3A

Simulation Result for a 4-bit Wallace Multiplier

Simulation Report	Sim	ulation Waveform	15										
Legal Notice	Sim	Simulation mode: Timing											
Flow Settings	Mast	er Time Bar: 21	0.0 ns	+ → Pointer: 19.2 ns Intervat -800 ps Start: 0 ps End: 0	ps								
Summary		Name	Va 20	0 ps 10.0 ms 20.0 ms 30.0 ms 40.0 ms 50.0 ms 60.0 ms 70.0 ms 80.0 ms 90.0 ms 10 20.0 ms	0,0 198								
* Simulation Covera	1	🕀 a	B	0000 X 0001 X 0010 X 0011 X 0100 X 0101 X 0110 X 0111 X 1000 X 1001	X 10								
(A) Messages	ELV?	🛨 b	B	<u>0000 X 0001 X 0010 X 0011 X 0100 X 0101 X 0110 X 0111 X 1000 X 1001</u>	X 10								

Timing Simulation for 4-bit Wallace multiplier

Compilation Report - T	imi Ti	ng Analyzer Summary					
Legal Notice		Туре	Slack	Required Time	Actual Time	From	To
Flow Settings	1	Worst-case tpd Total number of failed paths	N/A	None	48.800 ns	Ь[0]	prod[6]
Flow Log	<	TE - ME					>

Worst-Case Propagation Delay Time

Compilation Report - tp	bd						
Compilation Report	tpo	1					
E Legal Notice		Slack	Required P2P Time	Actual P2P Time	From	To	^
Flow Settings	1	N/A	None	48.800 ns	b[0]	prod[7]	
Flow Non-Default Glo	2	N/A	None	48.800 ns	b[0]	prod[6]	
Flow Elapsed Time	3	N/A	None	48.100 ns	a[3]	prod[7]	
Flow Log	4	N/A	None	48.100 ns	a[3]	prod[6]	
Analysis & Synthesis	5	N/A	None	45.600 ns	Ь[0]	prod[5]	
	6	N/A	None	44.900 ns	a[3]	prod[5]	
	7	N/A	None	41.500 ns	a[1]	prod[7]	
Summary	8	N/A	None	41.500 ns	a[1]	prod[6]	
Settings	9	N/A	None	41.400 ns	b[1]	prod[7]	
tpd	10	N/A	None	41.400 ns	a[2]	prod[7]	
Messages	11	N/A	None	41.400 ns	b[1]	prod[6]	
	12	N/A	None	41.400 ns	a[2]	prod[6]	
	13	N/A	None	38.900 ns	Ь[0]	prod[4]	
	14	N/A	None	38.900 ns	b[0]	prod[3]	
	15	N/A	None	38.300 ns	a[1]	prod[5]	
	16	N/A	None	38.200 ns	b[2]	prod[7]	
	17	N/A	None	38.200 ns	b[2]	prod[6]	
	18	N/A	None	38.200 ns	b[1]	prod[5]	
	19	N/A	None	38.200 ns	a[2]	prod[5]	
<	20	N/A	None	38.200 ns	a[3]	prod[4]	~

Total Propagation Delay for the Design

APPENDIX 3B

Simulation Result for a 6-bit Wallace Multiplier

	Sim	lation mode:	Timin	10				_												
	Mast	er Time Bar		18.55	i na	+ + F	Pointer	30.61	ris	Interve	al.	12.06	198	Starl	1			End		
		Nan	0.04		10.0 ng	18.5	20.0 na 55 na	30.pr	ą.	40.0 na	1	50. Q ne	<u>, </u>	60.Q ni	i i	70.0 n	1	80.0 m	i .	90.0 ne
	1	⊞a		000000	X	000001	X 000	010 X	000011	X	000100	X	000101	X	000110	X	000111	X	001000	X 00
9	1	i b i brod	F	000000	X	000001	X 000	010 X	000011	X 0000000	000100	X 000000	000101	X 100010	000110	X doradov	000111	X 00 XXX	001000	

Timing Simulation for a 6-bit Wallace multiplier

Compilation Report - T	Compilation Report - Timing Analyzer Summary												
Compilation Report	Ti	Timing Analyzer Summary											
Flow Summary		Туре	Slack	Required Time	Actual Time	From	To						
Flow Settings	1	Worst-case tpd	N/A	None	71.400 ns	b[4]	prod[10]						
Flow Non-Default (2	Total number of failed paths											
Flow Log	<	1					>						

Worst-Case Propagation Delay Time

Compilation Report - tr	bd	and the		1.00			
Compilation Report	tpd						
E Legal Notice		Slack	Required P2P Time	Actual P2P Time	From	To	^
Flow Settings	1	N/A	None	71.400 ns	b[4]	prod[11]	
Flow Non-Default Glob	2	N/A	None	71.400 ns	b[4]	prod[10]	
Flow Elapsed Time	3	N/A	None	70.500 ns	b[3]	prod(11)	
Flow Log	4	N/A	None	70.500 ns	b[3]	prod[10]	
+ Analysis & Synthesis	5	N/A	None	68.800 ns	b[4]	prod[9]	
+ Assembler	6	N/A	None	68.700 ns	a[4]	prod[11]	
- Analyzer	7	N/A	None	68.700 ns	a[4]	prod[10]	
Summary	8	N/A	None	67.900 ns	b[0]	prod[11]	
Settings	9	N/A	None	67.900 ns	a[5]	prod[11]	
tpd	10	N/A	None	67.900 ns	b[0]	prod[10]	
A Messages	11	N/A	None	67.900 ns	a[5]	prod[10]	
	12	N/A	None	67.900 ns	b[3]	prod[9]	
	13	N/A	None	66.900 ns	b[5]	prod[11]	
	14	N/A	None	66.900 ns	b[5]	prod[10]	
×	15	N/A	None	66.100 ns	a[4]	prod[9]	
	16	N/A	None	65.300 ns	b[0]	prod[9]	
	17	N/A	None	65.300 ns	a[5]	prod[9]	
	18	N/A	None	64.300 ns	b[5]	prod[9]	
	19	N/A	None	63.400 ns	b[1]	prod[11]	
< >	20	N/A	None	63.400 ns	b[1]	prod[10]	~

Total Propagation Delay in the Design

APPENDIX 3C

Simulation Result for a 8-bit Wallace Multiplier

s e	ightbit_wal	lace_mult	.vwf	- Hereit		- The second	4-69-2-2		
Mast	er Time Bar	18.55 ns	• Pointer:	68.4 ns	Interval:	49.85 ns	Start	End	
	Name		Ops	20.0 ns 18.55 ns	40 p	ns	60.0 ns	80.0 ns	100.0 ps
	● a ● b ● prod	B 00	000000000000000000000000000000000000000		1(<u>x0000001</u>) 1(<u>x0000001</u>) (00)	0000010(200 0000010(200 0000(00)(00)	0010120000110 0010120000110 0040020000110	20000011 2000100 20000011 2000100	C/0001001/01 C/0001001/01 D///0///00//X
<	40	>	<	1					>

Timing Simulation for a 8-bit Wallace multiplier

Compilation Re	port -	Tin	ning Analyzer Summary				-					
Compilation Rep	ort 🔨	Timing Analyzer Summary										
E Legal Notice	ary		Туре	Slack	Required Time	Actual Time	From	To				
Flow Setting	as in the second s	1	Worst-case tpd	N/A	None	87.800 ns	b[1]	prod[15]				
Flow Non-De	efau d Tir	2	Total number of failed paths									
<	>	<						>				

Worst-Case Propagation Delay Time

Compilation Report - 1	tpd	1 1 10	Contraction of the	1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-			
Compilation Report	tpd						
Legal Notice		Slack	Required P2P Time	Actual P2P Time	From	To	^
Flow Settings	1	N/A	None	87.800 ns	b[1]	prod[15]	
Flow Non-Default G	2	N/A	None	87.600 ns	b[1]	prod[14]	
Flow Elapsed Time	3	N/A	None	85.800 ns	b[0]	prod[15]	
Flow Log	4	N/A	None	85.600 ns	b[0]	prod[14]	
+ Analysis & Synthesi	5	N/A	None	84.600 ns	b[1]	prod[13]	
+ Assembler	6	N/A	None	82.600 ns	b[0]	prod[13]	
Timing Analyzer	7	N/A	None	82.400 ns	a[5]	prod[15]	
Summary	8	N/A	None	82.200 ns	a[5]	prod[14]	
Settings	9	N/A	None	82.100 ns	a[2]	prod[15]	
tpd	10	N/A	None	81.900 ns	a[2]	prod[14]	
(I) Messages	11	N/A	None	81.500 ns	b[3]	prod[15]	
	12	N/A	None	81.500 ns	b[4]	prod[15]	
	13	N/A	None	81.300 ns	b[3]	prod[14]	
	14	N/A	None	81.300 ns	b[4]	prod[14]	
	15	N/A	None	79.800 ns	a[3]	prod[15]	
	16	N/A	None	79.600 ns	a[3]	prod[14]	
	17	N/A	None	79.500 ns	a[4]	prod[15]	
	18	N/A	None	79.500 ns	a[6]	prod[15]	
	19	N/A	None	79.300 ns	a[4]	prod[14]	
<	20	N/A	None	79.300 ns	a[6]	prod[14]	~

Total Propagation Delay in the Design

APPENDIX 4A

Simulation Result for a 4-bit Dadda Multiplier

D d	ladda_mult.v	wf								(.)[=)[
Mast	er Time Bar.	0 ps	Pointer:	18.2 ns	Interval	18.2 ns	Start		E	ndt	
	Name	Va	0 ps 10.0 ns 2 0 ps	0.0 ns 30.() ns 40.0 ns	50.0 ns	60.0 ns	70.0 ns	80. <mark>0</mark> ns	90.0 ns	100 _, 0 ns
	t a t b t prod	B B B 00	0000 X 0001 0000 X 0001 0000	(0010) (0010) (0000	0011 X 01 0011 X 01 X00 000	00 X 01 00 X 01 00 X 01	01 X 011 01 X 011 0000000000000000000000	0 X 01 0 X 01 11)((0101)	11 X 10 11 X 10 (10)(10)	00 X 10 00 X 10	01 X 1010 01 X 1010
<		>	<								>

Timing Simulation for a 4-bit Dadda multiplier

Compilation Report	Ti	Timing Analyzer Summary										
Legal Notice		Туре	Slack	Required Time	Actual Time	From	To					
Flow Settings	1	Worst-case tpd	N/A	None	50.300 ns	b[0]	prod[6]					
Flow Non-Default (2	Total number of failed paths										
Flow Log	<			1			>					

Worst-Case Propagation Delay Time

Compilation Report - tp	bd						
Compilation Report	tpo						
E Legal Notice		Slack	Required P2P Time	Actual P2P Time	From	To	^
Flow Settings	1	N/A	None	50.300 ns	Ь[0]	prod[6]	
Flow Non-Default Glob	2	N/A	None	49.300 ns	b[0]	prod[7]	
Flow Elapsed Time	3	N/A	None	47.300 ns	b[3]	prod[6]	
Flow Log	4	N/A	None	46.300 ns	b[3]	prod[7]	
+ Analysis & Synthesis	5	N/A	None	46.100 ns	Ь[0]	prod[5]	
+ Assembler	6	N/A	None	44.100 ns	b[1]	prod[6]	
- A Timing Analyzer	7	N/A	None	44.000 ns	a[1]	prod[6]	
Summary	8	N/A	None	43.900 ns	a[0]	prod[6]	
Settings	9	N/A	None	43.800 ns	a[2]	prod[6]	
tpd	10	N/A	None	43.100 ns	b[1]	prod[7]	
Messages	11	N/A	None	43.100 ns	b[3]	prod[5]	
	12	N/A	None	43.000 ns	a[1]	prod[7]	
	13	N/A	None	42.900 ns	a[0]	prod[7]	
	14	N/A	None	42.800 ns	a[2]	prod[7]	
	15	N/A	None	41.800 ns	b[0]	prod[4]	
	16	N/A	None	40.500 ns	b[2]	prod[6]	
	17	N/A	None	39.900 ns	b[1]	prod[5]	
	18	N/A	None	39.800 ns	a[1]	prod[5]	
	19	N/A	None	39.700 ns	a[0]	prod[5]	
< >	20	N/A	None	39.600 ns	a[2]	prod[5]	*

Total Propagation Delay for the Design

APPENDIX 4B

Simulation Result for a 6-bit Dadda Multiplier

Simulation Report	Sim	simulation Waveforms													
Legal Notice Flow Summary Flow Settings	Sim.	imulation mode: Timing													
	Master Time Bar		18.55 n	1 1	I Pointer 87.		7.15 na Interval		68.6 ma		Start: E		End		
Summary Settings Simulation Wave		Name	0 ps	10.0 ns	20.0 ns 18.55 ns	30	pns	40.0 ns	50.pr	15	60.0 ns	70.0 ns	80.p.ns		
Simulation Cove INI Usage () Messages		a b f prod	(000000 (000000 ((X 000 X 000	001 X 001 X 0000000000	000010 000010 68	X 00001 X 00001		00100 X 00100 X 6666660	000101 000101 000101	X 00 X 00	0110 X 000 0110 X 000 56601X(66)((666)	11 X 0010 11 X 0010 5661 52 00 /66		

Timing Simulation for a 6-bit Dadda multiplier

٠	😃 Compilation Report - Timing Analyzer Summary												
8	Compilation Report	^	Timing Analyzer Summary										
	Legal Notice			Туре	Slack	Required Time	Actual Time	From	To				
-	Flow Settings		1	Worst-case tpd	N/A	None	73.000 ns	a[4]	prod[10]				
Flow Non-Defau Flow Elapsed Tir		~	2	Total number of failed paths									
<			<						>				

Worst-Case Propagation Delay Time

Compilation Report - 1	tpd						×
Compilation Report	tpd						
Legal Notice		Slack	Required P2P Time	Actual P2P Time	From	To	^
Flow Settings	1	N/A	None	73.000 ns	a[4]	prod[11]	
Flow Non-Default G	2	N/A	None	73.000 ns	a[4]	prod[10]	
Flow Elapsed Time	3	N/A	None	72.800 ns	Ь[0]	prod[11]	
Flow Log	4	N/A	None	72.800 ns	Ь[0]	prod[10]	
+ Analysis & Synthesi	5	N/A	None	71.400 ns	a[4]	prod[9]	
+ Assembler	6	N/A	None	71.200 ns	b[0]	prod[9]	
Timing Analyzer	7	N/A	None	68.700 ns	a[0]	prod[11]	
Summary	8	N/A	None	68.700 ns	a[0]	prod[10]	
Settings	9	N/A	None	67.500 ns	a[4]	[8]borq	
tpd	10	N/A	None	67.300 ns	b[0]	prod[8]	
Messages	11	N/A	None	67.100 ns	b[3]	prod[11]	
	12	N/A	None	67.100 ns	b[3]	prod[10]	
	13	N/A	None	67.100 ns	a[0]	prod[9]	
	14	N/A	None	66.500 ns	a[3]	prod[11]	
	15	N/A	None	66.500 ns	a[3]	prod[10]	
	16	N/A	None	66.400 ns	b[1]	prod[11]	
	17	N/A	None	66.400 ns	Ь[1]	prod[10]	
	18	N/A	None	65.500 ns	b[3]	prod[9]	
	19	N/A	None	64.900 ns	a[3]	prod[9]	
<	20	N/A	None	64.800 ns	b[1]	prod[9]	×

Total Propagation Delay in the Design

APPENDIX 4C

Simulation Result for a 8-bit Dadda Multiplier

I e	ightbit_dadd	a_mult.v	wf						
Mast	er Time Bar	18.55 ns	+ + Pointer	34.2 ns	Interval	15.65 ns	Start	End	
	Name		0 ps	20.0 ns 18.55 ns	40.0	ns	60.0 ns	80.0 ns	100.0 ns
	t a		(00000000 (0000	000 X 00000 10	X00000011X	00000100 00	000101 200000110	00000111 0000100	0 (00001001)
	te b ∎ prod	B 00	(00000000) (0000000) (0000	0001 X 00000010	X00000011 X00	00000100 X 00 00)XX00X00000	000101 X 00000110	00000111 X 0000100	0 (00001001)
<	10	>	<	1					>

Timing Simulation for a 6-bit Dadda multiplier

Compilation Report -	Tin	ning Analyzer Summary	1									
🞒 Compilation Report 🔥	Ti	Timing Analyzer Summary										
Legal Notice		Туре	Slack	Required Time	Actual Time	From	To					
Flow Settings	1	Worst-case tpd	N/A	None	88.500 ns	Ь[0]	prod[15]					
Flow Non-Defau Flow Elapsed Tir	2	Total number of failed paths										
< >	<		1	1	77		>					

Worst-Case Propagation Delay Time

🗢 Compilation Report - tpd										
Compilation Report	tpd	tpd								
Legal Notice		Slack	Required P2P Time	Actual P2P Time	From	To	^			
Flow Settings	1	N/A	None	88.500 ns	b[0]	prod[15]				
Flow Non-Default G	2	N/A	None	88.300 ns	b[1]	prod[15]				
Flow Elapsed Time	3	N/A	None	87.700 ns	a[3]	prod[15]				
Flow Log	4	N/A	None	87.100 ns	Ь[0]	prod[14]				
+ Analysis & Synthesi	5	N/A	None	86.900 ns	b[1]	prod[14]				
+ Assembler	6	N/A	None	86.300 ns	a[4]	prod[15]				
- Analyzer	7	N/A	None	86.300 ns	a[3]	prod[14]				
Summary	8	N/A	None	84.900 ns	a[4]	prod[14]				
Settings	9	N/A	None	84.100 ns	b [0]	prod[13]				
tpd	10	N/A	None	84.000 ns	a[2]	prod[15]				
Messages	11	N/A	None	83.900 ns	b[1]	prod[13]				
	12	N/A	None	83.300 ns	a[5]	prod[15]				
	13	N/A	None	83.300 ns	a[6]	prod[15]				
	14	N/A	None	83.300 ns	a[7]	prod[15]				
	15	N/A	None	83.300 ns	a[3]	prod[13]				
	16	N/A	None	82.600 ns	a[2]	prod[14]				
	17	N/A	None	81.900 ns	a[5]	prod[14]				
	18	N/A	None	81.900 ns	a[6]	prod[14]				
	19	N/A	None	81.900 ns	a[7]	prod[14]				
< · · · · · · · · · · · · · · · · · · ·	20	N/A	None	81.900 ns	a[4]	prod[13]	~			

Total Propagation Delay in the Design

APPENDIX 5 IMPLEMENTATION ON EPF10K70RC240-4

The figure below shows the output of the multiplication process of input bits of 0011 and 1100, observed on the seven-segment LEDs. The seven-segment LEDs are active low. The seven-segment shows the output to be 00100100 which is correct.



4-bit Ripple Carry Multiplier