

Ethan Longley - Assignment ACDA6001

by Ethan Longley

Submission date: 13-Apr-2025 03:00PM (UTC+0100)

Submission ID: 255609473

File name: 50234_Ethan_Longley_Ethan_Longley_-_Assignment_ACDA6001_654553_1620002854.docx (6.17M)

Word count: 4497

Character count: 25036



apply innovation™



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Critical Evaluation of Brauns Multiplier and Wallace Tree Multiplier using VHDL

ELECTRONICS 2 (ACDA6001)

ETHAN LONGLEY
2318473

Contents

Abstract	3
1.0 - Introduction	3
2.0 - Research.....	4
2.1 - Logic Gates	4
2.2 - Multipliers	4
2.2.1 - Brauns Multiplier	5
2.2.2 - Wallace Tree Multiplier	5
2.3 - Adders	6
2.3.1 - Half Adder	6
2.3.2 - Full Adder	6
2.4 - Applications.....	7
2.5- Advantages.....	7
3.0 – Multiplier Flowchart.....	8
4.0 – Multiplier Design.....	9
4.1 – Partial Product Array	9
4.2 – Logic Circuit Design	9
5.0 – 8x8 Brauns Multiplier	10
5.1 – Partial Product Array for 8x8 Brauns Multiplier	10
5.2 – Logic Circuit Design for 8x8 Brauns Multiplier	11
5.3 – VHDL and Simulation for 8x8 Brauns Multiplier.....	11
5.3.1 – VHDL of Brauns Multiplier.....	11
5.3.2 – Vivado Simulation of 8x8 Brauns Multiplier.....	12
6.0 – 8x8 Wallace Tree Multiplier	17
6.1 – 8x8 Wallace Tree Multiplier Dot Diagram	17
6.2 - Partial Product Array for 8x8 Wallace Tree Multiplier	18
6.3 – Logic Circuit Design for 8x8 Wallace Tree Multiplier	18
6.4 – VHDL and Simulation for Wallace Tree Multiplier.....	19
6.4.1 – VHDL for 8x8 Wallace Tree Multiplier	19
6.4.2 – Vivado Simulation of 8x8 Wallace Tree Multiplier	19
7.0 – Simulation Analysis	24
7.1 – Timing Analysis.....	24
7.2 – Power Consumption.....	25
7.2.1 – Methods to improve power efficiency	26
8.0 - Conclusion.....	27
References.....	28

Appendix A.....	30
Appendix B.....	30
Appendix C.....	31

Abstract

This report details the workings of multipliers within electronics, discussing how a multiplier works, how they can be designed and what results are shown after simulation. Wallace Tree and Brauns Multipliers were used to create partial product arrays and logic circuit designs to allow for the VHDL program to be created. Using Vivado software to simulate the program, it was determined upon comparison that the Wallace Tree was faster in operation however the Brauns was more efficient.

1.0- Introduction

Multipliers are important components in digital signal processing, computer arithmetic, and embedded systems. Their efficiency significantly impacts the overall performance of digital circuits, particularly in applications requiring high-speed computation, such as image processing, artificial intelligence, and cryptography. Two widely used multiplier architectures, Braun's Multiplier and Wallace Tree Multiplier, are commonly implemented to optimize speed, power consumption, and hardware utilization.

The purpose of this paper is to use VHDL to compare, synthesise, and critically assess the performance of the Wallace Tree and Braun's multipliers. The study involves designing an 8-bit Braun's Multiplier and an 8x8 Wallace Tree Multiplier, implementing them using appropriate computational techniques, and synthesizing them for FPGA deployment. Furthermore, a detailed performance analysis will be conducted using Xilinx Vivado software to compute propagation delays, analyse power consumption, and assess the efficiency of both multipliers.

The analysis will apply engineering and mathematical principles to determine the critical path delays, highlighting the advantages and limitations of each technique. Additionally, this report will explore potential improvements in multiplier efficiency, offering insights into optimizing speed and power consumption. By evaluating these two architectures, this study aims to contribute to the development of more efficient digital systems in modern computing applications.

2.0- Research

2.1- Logic Gates

Logic gates are electrical circuits that are designed using resistors, diodes, transistors, and other electrical components. It is used to do logical operations based on the inputs that are presented to it and can produce logical output that is either high (1) or low (0). The basis for the operation of logic gates is the Boolean algebra or mathematics. Many parts of our daily life, such as the construction of memory devices, computers, tablets, and telephones, require logic gates [1].

Name	AND	OR	Inverter	Buffer	NAND	NOR	Exclusive-OR (XOR)	Exclusive-NOR or equivalence																																																																																																						
Graphic Symbol																																																																																																														
Algebraic Function	$F = x \cdot y$	$F = x + y$	$F = x^1$	$F = x$	$F = (xy)^1$	$F = (x+y)^1$	$F = xy^1 + x^1y = x \oplus y$	$F = xy + x^1y^1 = (x \oplus y)^1$																																																																																																						
Truth Table	<table><tr><th>x</th><th>y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x	y	F	0	0	0	0	1	0	1	0	0	1	1	1	<table><tr><th>x</th><th>y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x	y	F	0	0	0	0	1	1	1	0	1	1	1	1	<table><tr><th>x</th><th>F</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	x	F	0	1	1	0	<table><tr><th>x</th><th>F</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	x	F	0	0	1	1	<table><tr><th>x</th><th>y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	x	y	F	0	0	1	0	1	1	1	0	1	1	1	0	<table><tr><th>x</th><th>y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	x	y	F	0	0	1	0	1	0	1	0	0	1	1	0	<table><tr><th>x</th><th>y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	x	y	F	0	0	0	0	1	1	1	0	1	1	1	0	<table><tr><th>x</th><th>y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x	y	F	0	0	1	0	1	0	1	0	0	1	1	1
x	y	F																																																																																																												
0	0	0																																																																																																												
0	1	0																																																																																																												
1	0	0																																																																																																												
1	1	1																																																																																																												
x	y	F																																																																																																												
0	0	0																																																																																																												
0	1	1																																																																																																												
1	0	1																																																																																																												
1	1	1																																																																																																												
x	F																																																																																																													
0	1																																																																																																													
1	0																																																																																																													
x	F																																																																																																													
0	0																																																																																																													
1	1																																																																																																													
x	y	F																																																																																																												
0	0	1																																																																																																												
0	1	1																																																																																																												
1	0	1																																																																																																												
1	1	0																																																																																																												
x	y	F																																																																																																												
0	0	1																																																																																																												
0	1	0																																																																																																												
1	0	0																																																																																																												
1	1	0																																																																																																												
x	y	F																																																																																																												
0	0	0																																																																																																												
0	1	1																																																																																																												
1	0	1																																																																																																												
1	1	0																																																																																																												
x	y	F																																																																																																												
0	0	1																																																																																																												
0	1	0																																																																																																												
1	0	0																																																																																																												
1	1	1																																																																																																												

Figure 1 - Table showing different types of Logic Gates, Algebraic Function and Truth Tables [2].

Logic gates are essential components in digital multipliers, enabling arithmetic operations on binary signals. AND gates create partial products by multiplying each bit of one operand with each bit of the other. For example, if $A = 1010$ and $B = 1101$, their bits are ANDed to form partial product rows. These partial products are then summed using Half Adders (HA) and Full Adders (FA)—HAs add two binary numbers without a carry input, while FAs handle cases with a carry. Braun's Multiplier uses a carry propagate adder, whereas the Wallace Tree Multiplier employs parallel addition with reduction stages for faster computation [3].

2.2- Multipliers

A multiplier is a digital circuit that performs multiplication of two numbers. AND gates are used to create partial products, which are then summed together using half and full adders. The two primary multiplier types are the Wallace Tree Multiplier, which employs parallel reduction for speedier calculation, and Braun's Multiplier, which adds partial products sequentially.

In applications requiring quick arithmetic operations, such as digital signal processing (DSP), cryptography, image processing, and microprocessors, multipliers are crucial. They are extensively used in embedded systems, FPGAs, and ASICs to maximise speed and efficiency in a range of computing applications [4].

2.2.1- Brauns Multiplier

A Braun multiplier is an unsigned binary parallel multiplier or carry-save multiplier. It's made up using an array of AND gates and adders in a recurring array, in which the bits of the multiplicand are ANDed with the bits of the multiplier to produce partial products, which are summed by the adders to produce the product. Although the Braun multiplier has a straightforward and area-conserving structure for low bit-widths, it has some disadvantages such as potential problems of glitching in the last ripple carry adder stages, which can affect speed and power [5].

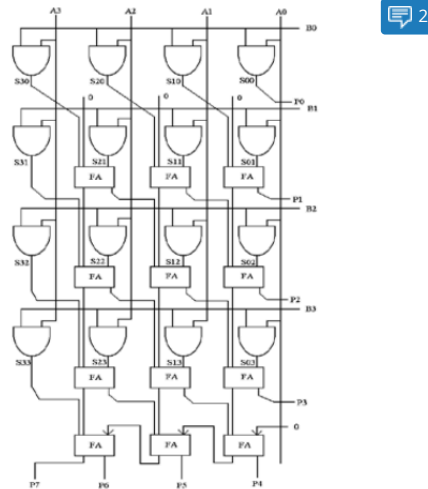


Figure 2 - Braun Multiplier [6].

A Braun multiplier is made up of adders that are set up repeatedly to sum partial products and AND gates that create them. Carry bits are relocated and appropriately handled, and a row of adders is used to add a partial product to the calculated sum. A ripple carry adder (RCA), which adds delay and affects speed, is used in the last step. While, although Braun multipliers are effective with inputs up to 16 bits in size, carry propagation in the last stage causes glitching. Row and column bypass techniques can optimise both power and area efficiency. Despite its extreme simplicity, the Braun's multiplier is frequently utilised in low-power FPGA implementations [7].

2.2.2- Wallace Tree Multiplier

A digital circuit that effectively multiplies one integer by another is represented by a hardware device called a Wallace Tree Multiplier. Using a combination of half adders and full adders, the Wallace tree, also known as the Wallace reduction, progressively gathers partial products until only two remain, which are then added to get the outcome. This reduces the amount of successive addition stages, which speeds up multiplication when compared to simpler approaches [8].

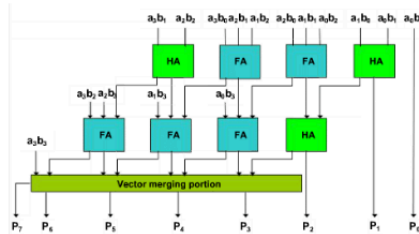


Figure 3 - Wallace Tree Multiplier [9]

The first part of a Wallace Tree Multiplier's operation is to create partial products by multiplying each bit of an operand by each bit of the second operand. The second is that successive layers of full and

half adders decrease the partial products to two numbers to minimise the amount of adds that need to be made. The final product of multiplication is then obtained by adding the final two numbers using a standard adder. By reducing the depth of the critical path over the adder network, this structured procedure greatly speeds up multiplication processes [10].

2.3- Adders

An adder is a simple digital circuit used in computers and other digital devices that performs the addition arithmetic function. The two primary types of adders are half adders and complete adders. The sum and carry outputs are obtained by applying a half adder to two binary values of one bit each. It never considers any carry input from the previous additions, though. On the other hand, when three single-bit binary values are added—typically two significant bits and an entering carry bit from the previous less significant stage—a Full Adder produces a sum output as well as a carry output. This enables cascading Full Adders to add binary numbers longer than one bit [11].

2.3.1- Half Adder

A simple digital circuit known as a half adder can be used to add two one-bit binary integers. From two inputs, typically represented by the letters A and B, it produces two outputs: the sum (S) and the carry (C). The sum output is the least significant bit in the sum, but the carry output overflows into the next significant bit. In terms of the logic gates, the sum is provided by an Exclusive OR (XOR) gate, while the carry is provided by a while gate. This configuration allows the half adder to do basic binary addition without requiring any carry input from a previous step [12].

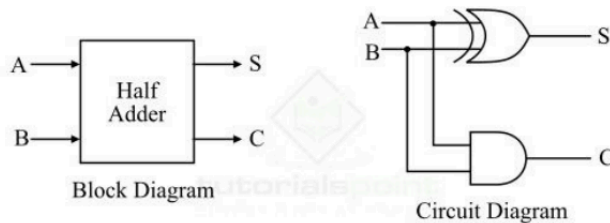


Figure 4 - Half Adder Diagrams [13]

2.3.2- Full Adder

The total of two bits and a carry-in from the previous addition of three single-bit binary integers is calculated by a digital logic gate called a full adder. In multi-bit addition computations, it generates two outputs: a sum and a carry-out, which are sent to the subsequent higher bit position. Usually, an XOR gate is used to create the total output, and a combination of AND and OR gates is used to generate the carry-out. This enables the construction of adders that can handle binary numbers with more than one bit by cascading numerous full adders [14].

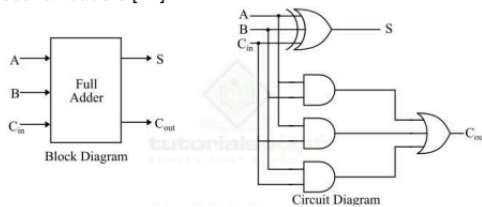


Figure 5 - Full Adder Diagrams [15]

2.4- Applications

Multipliers are essential components of many digital systems and are used in many different applications. They make it possible to perform operations like filtering, convolution, and Fourier transforms in digital signal processing (DSP), which serve as the basis for calculations in audio and image processing. Multipliers in computer graphics allow for the fast calculations needed to render images and animations. In addition, they play a key role in the arithmetic logic units (ALUs) of microprocessors, carrying out crucial mathematical calculations that power computer programs. Such multipliers play a crucial role in contemporary electronics since their speed and efficiency have a direct impact on the power and speed of digital devices [16].

2.5- Advantages

Electronic multipliers, such as, Braun and Wallace tree structures, have great benefits in digital signal processing and VLSI applications. The Braun multiplier has a regular structure and simplicity, which makes it a good choice for FPGA implementations. It is designed to have simple layout and scalability, which is advantageous in hardware design [17]. The Wallace tree multiplier achieves optimal speed and efficiency by minimizing sequential additions. Its tree-based architecture reduces critical path delay, facilitating faster computation. It can also be implemented for low power, suitable for high-performance, low-power designs [18].



3.0 – Multiplier Flowchart

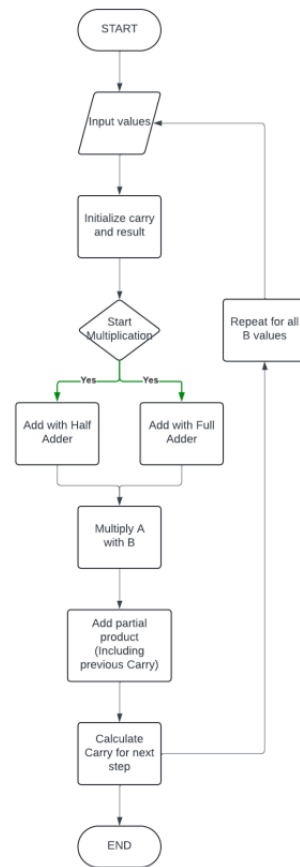


Figure 6 – Flowchart showing process of a Multiplier

Figure 6 shows a flowchart that shows the binary multiplication of two values and illustrates the entire multiplier operation. The process starts with the numbers being entered and the carry and sum being initialised. Each bit in the multiplier (B) is multiplied by the corresponding bit in the multiplicand (A), and adders are used to accumulate the partial product in the final result. The full adder adds three bits, whereas the half adder adds two bits. This updates the result and carry for every bit in the multiplier.

4.0 – Multiplier Design

To create a Multiplier, a partial product array must be designed, which will then be used to create the Logic Circuit Design. With this, it will allow for the VHDL program to be developed and simulated.

4.1 – Partial Product Array

In multiplier design, a partial product array is used to arrange and show the intermediate products that are produced when each multiplier bit is multiplied by each multiplicand bit. This array streamlines carry handling and subsequent summing. [19].

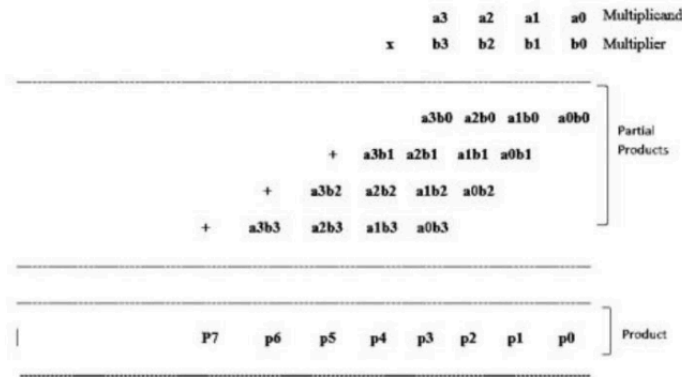


Figure 7 - Example of Partial Product Array for 4x4 Brauns Multiplier [20]

By arranging the intermediate multiplication results in a systematic manner, a partial product array makes it possible to construct a combinational circuit. The carry proceeds through different phases and the partial products are efficiently summed using full adders and half adders. Fast computing, less delay, and effective final product calculation are made possible by this parallelised structure, which eliminates the requirement for clock synchronisation [21].

4.2 – Logic Circuit Design

A digital circuit that adds several binary values together is called an logic circuit design. It allows for the parallel addition of partial sums and carries by arranging several rows of full or half adders in a structured manner. Because it expedites the addition process, this array layout is helpful for processor tasks like multiplication and large-scale arithmetic [22].

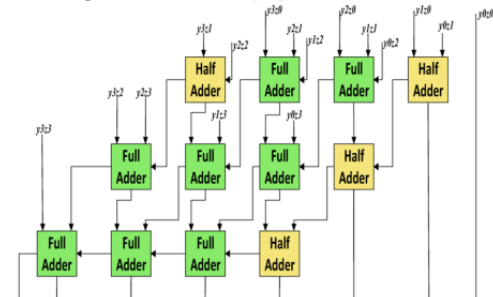


Figure 8 - Example of Adder Array for 4x4 Wallace Tree Multiplier [23]

An Logic Circuit Design can then be used to create the VHDL program by defining full and half adder components and arranging them in a structured array. VHDL's process or concurrent statements handle bit addition and carry propagation. Inputs are connected to the adders, and outputs generate the final result, which is tested using testbenches for accuracy.

5.0 – 8x8 Brauns Multiplier

To begin designing a Braun’s multiplier, a partial product array will be created through bitwise multiplication of the multiplicand and multiplier. These partial products are then summed using a logic circuit design, with full and half adders handling bitwise addition and carry propagation for efficient final product computation.

5.1 – Partial Product Array for 8x8 Brauns Multiplier

									A7	A6	A5	A4	A3	A2	A1	A0
								x								
									B7	B6	B5	B4	B3	B2	B1	B0
									A7B0	A6B0	A5B0	A4B0	A3B0	A2B0	A1B0	A0B0
									A7B1	A6B1	A5B1	A4B1	A3B1	A2B1	A1B1	A0B1
									A7B2	A6B2	A5B2	A4B2	A3B2	A2B2	A1B2	A0B2
									A7B3	A6B3	A5B3	A4B3	A3B3	A2B3	A1B3	A0B3
									A7B4	A6B4	A5B4	A4B4	A3B4	A2B4	A1B4	A0B4
									A7B5	A6B5	A5B5	A4B5	A3B5	A2B5	A1B5	A0B5
									A7B6	A6B6	A5B6	A4B6	A3B6	A2B6	A1B6	A0B6
									A7B7	A6B7	A5B7	A4B7	A3B7	A2B7	A1B7	A0B7
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0	

Figure 9 - 8x8 Partial Product Array for a Brauns Multiplier

The Partial Product Array in Figure 9 shows the Multiplicand (A) and the Multiplier (B). From the research above, it is known that a half adder includes two inputs and a SUM and CARRY outputs, and a full adder has three inputs as well as the SUM and CARRY outputs [24]. It is visible that in the column with A0B0, there is only one input, this means that there is no SUM or CARRY. In the column to the left of it, with A1B0 and A0B1, there are two inputs meaning they will produce a SUM and a CARRY.



5.2 – Logic Circuit Design for 8x8 Brauns Multiplier

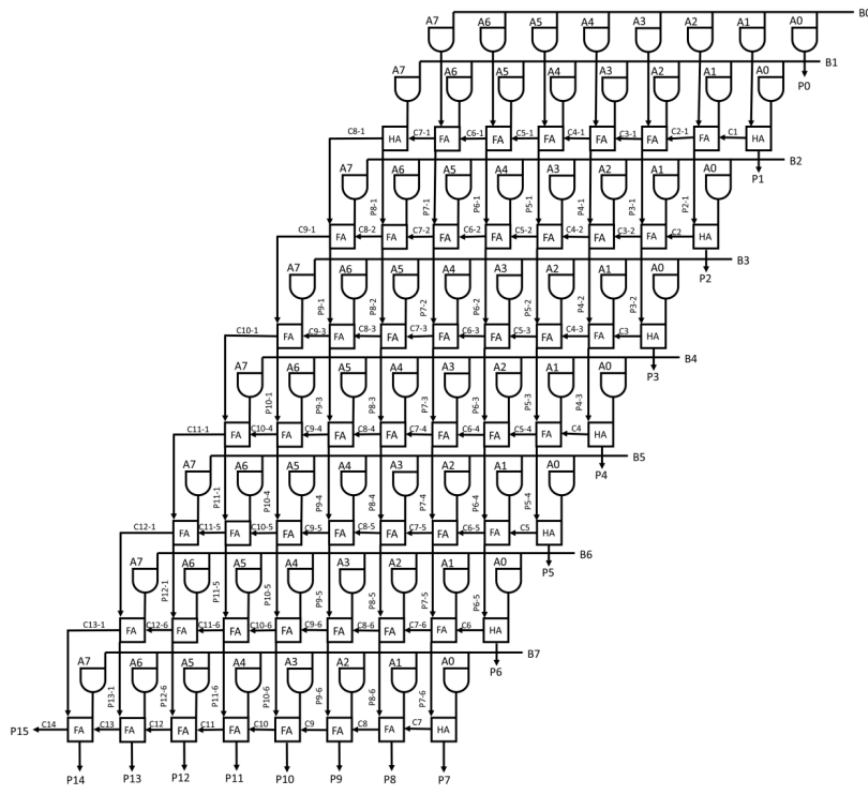


Figure 10 – Logic Circuit Design for 8x8 Brauns Multiplier

As shown in Figure 10, the logic circuit design lays out the foundation for the 8x8 Brauns Multiplier. It visually shows where the SUM and the CARRY end up after the adder. Using the Adder Array, the values can be input into the VHDL program to create and simulate the multiplier.

5.3 – VHDL and Simulation for 8x8 Brauns Multiplier

This section presents the VHDL program developed for an 8x8 Braun's Multiplier. The design is based on a structured array of partial products and a logic circuit design to compute the final 16-bit result. The full VHDL for the 8x8 Brauns Multiplier can be seen in **Appendix A**. Using the software VIVADO, the program can be simulated and analysed. This will provide a detailed analysis of the speed and efficiency of the multiplier.

5.3.1 – VHDL of Brauns Multiplier

An 8-bit Braun multiplier, a combinational digital circuit for multiplying two 8-bit binary values (A and B), is implemented in this VHDL software. The architecture computes the partial products and sums them correctly to generate a 16-bit output product (P) using an array of AND gates, half adders, and full adders. By creating partial products (via AND operations between bits of A and B) and adding them

in a structured pipeline with adders, each bit of the result is computed. The approach is appropriate for high-speed multiplication in digital systems and places an emphasis on hardware parallelism.

5.3.1.1 – Port Maps in VHDL for 8x8 Brauns Multiplier

```
P0      : P(0) ≤ A(0) and B(0) ;

P1_0    : half_adder port map (A⇒A1B0 , B⇒A0B1 , S⇒ P(1) , Cout⇒C1);

P21     : full_adder port map (A⇒A2B0 , B⇒A1B1 , Cin⇒C1 , S⇒P2_1 , Cout⇒C2_1);
P2_0    : half_adder port map (A⇒A0B2 , B⇒P2_1 , S⇒P(2) , Cout⇒C2);

P31     : full_adder port map (A⇒A3B0 , B⇒A2B1 , Cin⇒C2_1 , S⇒P3_1 , Cout⇒C3_1);
P32     : full_adder port map (A⇒A1B2 , B⇒P3_1 , Cin⇒C2 , S⇒P3_2 , Cout⇒C3_2);
P3_0    : half_adder port map (A⇒A0B3 , B⇒P3_2 , S⇒P(3) , Cout⇒C3);
```

Figure 11 - Partial Product Generator with Port Maps for Multiplier

This area of the VHDL program, shown in Figure 11, implements the calculation of partial products for an 8-bit multiplier by utilizing AND gates and both half adders and full adders in a structured hierarchy. Each line of the program uses port maps to instantiate different components, connecting specific input signals to the inputs of those components and defining the outputs to correspond with the desired product bits.

The application computes partial products for an 8-bit multiplier using port mappings. It begins with P0, which translates the AND operation between A(0) and B(0) directly. To add partial products and handle carries, it then employs a combination of half and full adders. P1_0, for instance, adds A1B0 and A0B1, resulting in a sum and carry that are sent to the following step. Following port maps are used in this operation, each of which handles carry propagation and adds partial products before producing the product bits.

5.3.2 – Vivado Simulation of 8x8 Brauns Multiplier

5.3.2.1 – Hexadecimal Calculations

To begin, the program was implemented into the software and simulated using a hexadecimal calculator. The first equation being.

$$03 \times A1 = 01E3$$

As shown in Figure 12 below, the 8x8 Brauns Multiplier has been able to successfully calculate this equation.

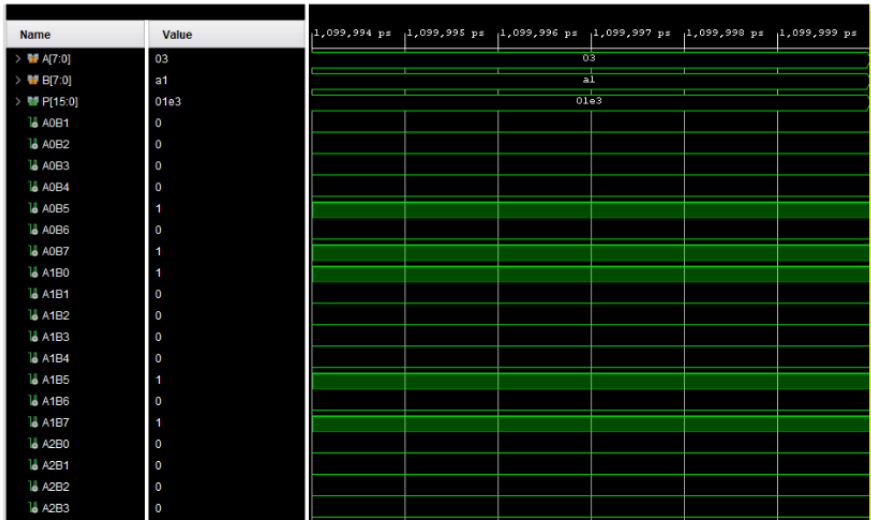


Figure 12 - 03xA1 Hexadecimal Calculation using 8x8 Brauns Multiplier

The following calculations shown in Figure 13, Figure 14 and Figure 15 were also carried out during the simulation.

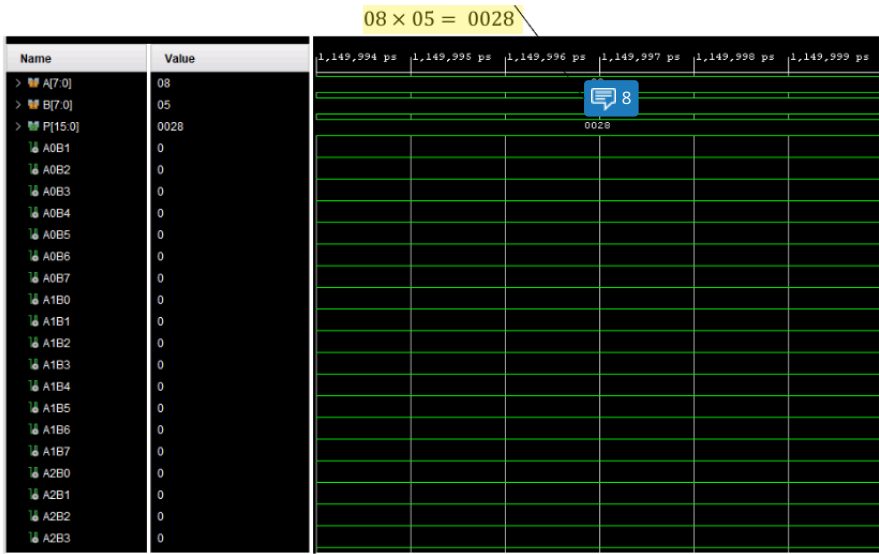


Figure 13 - 08x05 Hexadecimal Calculation using 8x8 Brauns Multiplier

$$0A \times B1 = 06EA$$

Name	Value	1,199,994 ps	1,199,995 ps	1,199,996 ps	1,199,997 ps	1,199,998 ps	1,199,999 ps
> A[7:0]	0a				0a		
> B[7:0]	b1				b1		
> P[15:0]	06ea				06ea		
A0B1	0						
A0B2	0						
A0B3	0						
A0B4	0						
A0B5	0						
A0B6	0						
A0B7	0						
A1B0	1						
A1B1	0						
A1B2	0						
A1B3	0						
A1B4	1						
A1B5	1						
A1B6	0						
A1B7	1						
A2B0	0						
A2B1	0						
A2B2	0						
A2B3	0						

Figure 14 - 0AxB1 Hexadecimal Calculation using 8x8 Brauns Multiplier

$$FF \times FF = FE01$$

Name	Value	1,049,995 ps	1,049,996 ps	1,049,997 ps	1,049,998 ps	1,049,999 ps
> A[7:0]	ff				ff	
> B[7:0]	ff				ff	
> P[15:0]	fe01				fe01	
A0B1	1					
A0B2	1					
A0B3	1					
A0B4	1					
A0B5	1					
A0B6	1					
A0B7	1					
A1B0	1					
A1B1	1					
A1B2	1					
A1B3	1					
A1B4	1					
A1B5	1					
A1B6	1					
A1B7	1					
A2B0	1					
A2B1	1					
A2B2	1					
A2B3	1					

Figure 15 - FFxFF Hexadecimal Calculation using 8x8 Brauns Multiplier

As shown, each of the calculations are correct. This proves that the VHDL program is multiplying correctly.

5.3.2.2 – RTL Design of 8x8 Brauns Multiplier

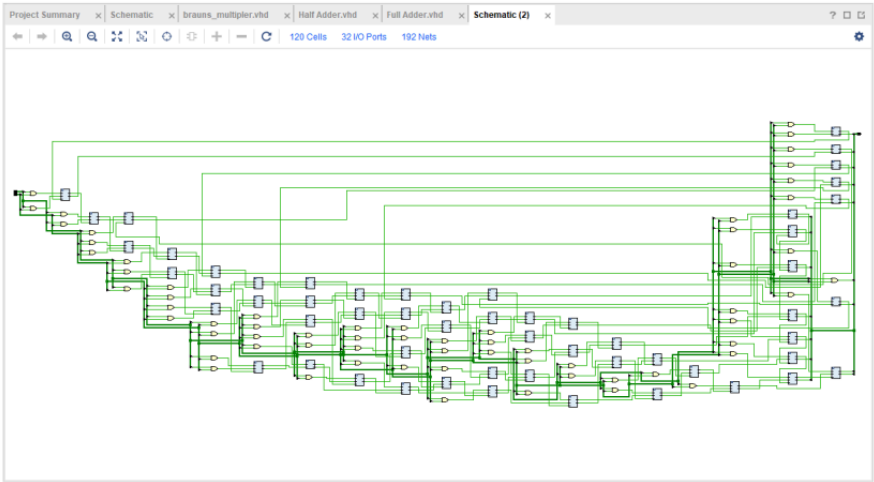


Figure 16 - 8x8 Brauns Multiplier RTL Design

5.3.2.3 – Implementation Schematics of 8x8 Brauns Multiplier

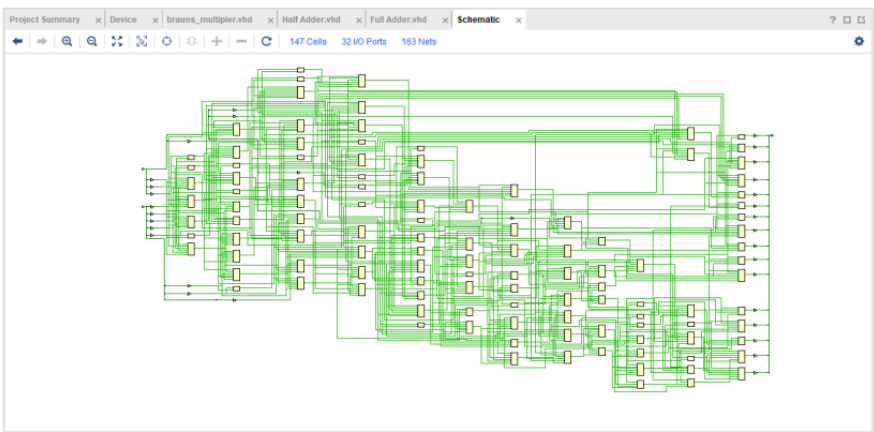


Figure 17 - 8x8 Brauns Multiplier Implementation Schematic

5.3.2.4 – 8x8 Brauns Multiplier Device

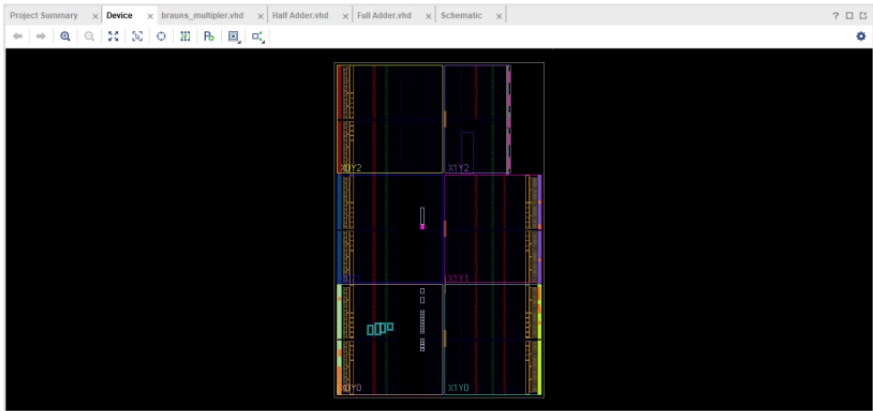


Figure 18 - 8x8 Brauns Multiplier Device

5.3.2.5 – I/O Planning for 8x8 Brauns Multiplier



Figure 19 - I/O Plan for 8x8 Brauns Multiplier

6.0 – 8x8 Wallace Tree Multiplier

A Wallace Tree Multiplier is a fast hardware multiplier that effectively reduces partial products by using a tree structure. The multiplicands are initially used to create partial products, which are subsequently reduced in stages using half adders and full adders that gradually merge and remove carries. A network of adders arranged in layers is created as part of the design; each layer lowers the number of partial products until there are just two numbers left, which are then added together to create the final product [25].

6.1 – 8x8 Wallace Tree Multiplier Dot Diagram

A Dot Diagram in a Wallace tree multiplier visually shows how partial products are generated and reduced during multiplication. It helps organize the bits from the multiplication and illustrates how they are added together in stages using adders. This diagram is important for optimizing the design, ensuring faster multiplication with fewer logic gates [26].

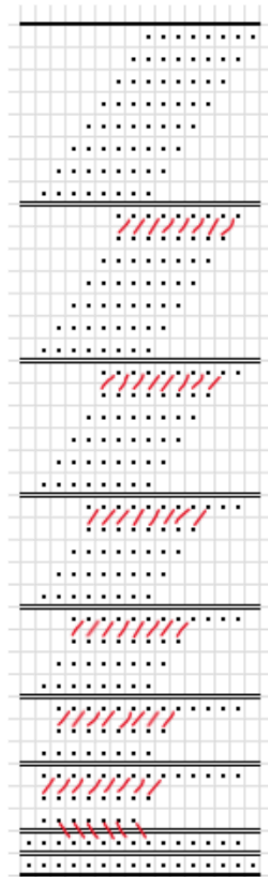
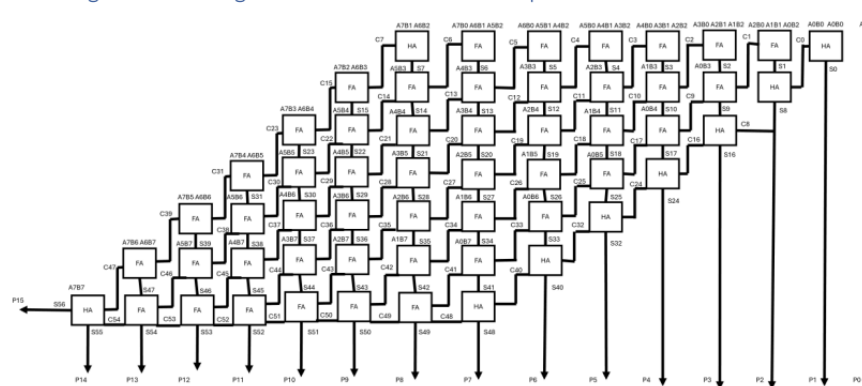


Figure 20 - Dot Diagram for 8x8
Wallace Tree Multiplier

[illegible]

6.3 – Logic Circuit Design for 8x8 Wallace Tree Multiplier



Page 18

6.4 – VHDL and Simulation for Wallace Tree Multiplier

Throughout this section, images will be shown of the simulation results taken from uploading the VHDL program into the Vivado simulation software.

6.4.1 – VHDL for 8x8 Wallace Tree Multiplier

```
-- Stage 1
P0 : P(0) ≤ A(0) and B(0) ;

ha11 : half_adder port map(A1B0,A0B1,P(1),C1);
fa12 : full_adder port map(A2B0,A1B1,A0B2,S2,C2);
fa13 : full_adder port map(A3B0,A2B1,A1B2,S3,C3);
fa14 : full_adder port map(A4B0,A3B1,A2B2,S4,C4);
fa15 : full_adder port map(A5B0,A4B1,A3B2,S5,C5);
fa16 : full_adder port map(A6B0,A5B1,A4B2,S6,C6);
fa17 : full_adder port map(A7B0,A6B1,A5B2,S7,C7);
ha18 : half_adder port map(A7B1,A6B2,S8,C8);

-- Stage 2
ha21 : half_adder port map(S2,C1,P(2),C9);
fa22 : full_adder port map(S3,C2,A0B3,S10,C10);
fa23 : full_adder port map(S4,C3,A1B3,S11,C11);
fa24 : full_adder port map(S5,C4,A2B3,S12,C12);
fa25 : full_adder port map(S6,C5,A3B3,S13,C13);
fa26 : full_adder port map(S7,C6,A4B3,S14,C14);
fa27 : full_adder port map(S8,C7,A5B3,S15,C15);
fa28 : full_adder port map(A7B2,C8,A6B3,S16,C16);
```

Figure 23 - Section of VHDL program for 8x8 Wallace Tree Multiplier

Shown in Figure 23 is a section of the VHDL program used to simulate the 8x8 Wallace Tree Multiplier. The entire program can be seen in **Appendix B**.

6.4.2 – Vivado Simulation of 8x8 Wallace Tree Multiplier

6.4.2.1 – Hexadecimal Calculation

08 × 05 = 0028

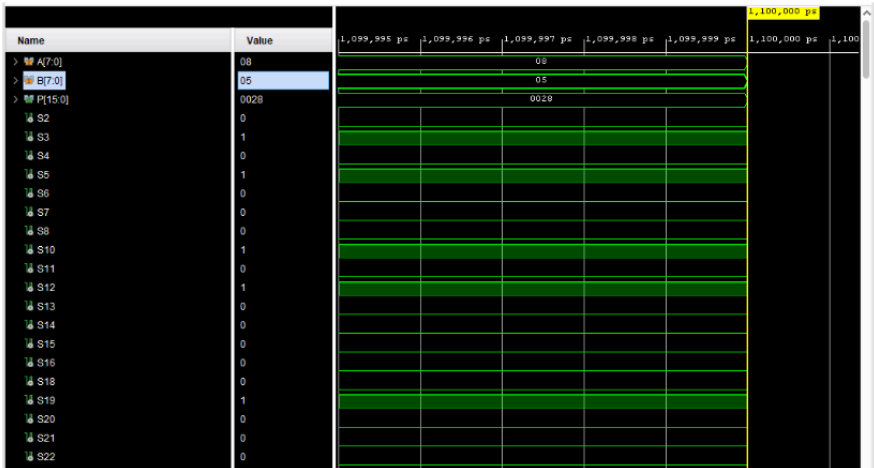


Figure 24 - 08x05 Hexadecimal Calculation

$$B1 \times 0A = 06EA$$

Name	Value	1,149,995 ps	1,149,996 ps	1,149,997 ps	1,149,998 ps	1,149,999 ps	1,150,000 ps	1,150,001 ps
> A[7:0]	b1							
> B[7:0]	0a							
> F[15:0]	06ea							
S2	0							
S3	0							
S4	0							
S5	1							
S6	1							
S7	0							
S8	1							
S10	1							
S11	0							
S12	1							
S13	1							
S14	1							
S15	0							
S16	0							
S18	0							
S19	1							
S20	1							
S21	1							
S22	0							

Figure 25 - B1 x 0A Hexadecimal Calculation

$$A1 \times 03 = 01E3$$

Name	Value	1,249,995 ps	1,249,996 ps	1,249,997 ps	1,249,998 ps	1,249,999 ps	1,250,000 ps	1,250,001 ps
> A[7:0]	a1							
> B[7:0]	03							
> F[15:0]	01e3							
S2	0							
S3	0							
S4	0							
S5	1							
S6	1							
S7	1							
S8	1							
S10	0							
S11	0							
S12	1							
S13	1							
S14	1							
S15	1							
S16	0							
S18	0							
S19	1							
S20	1							
S21	1							
S22	1							

Figure 26 - A1x03 Hexadecimal Calculation

$$FF \times FF = FE01$$

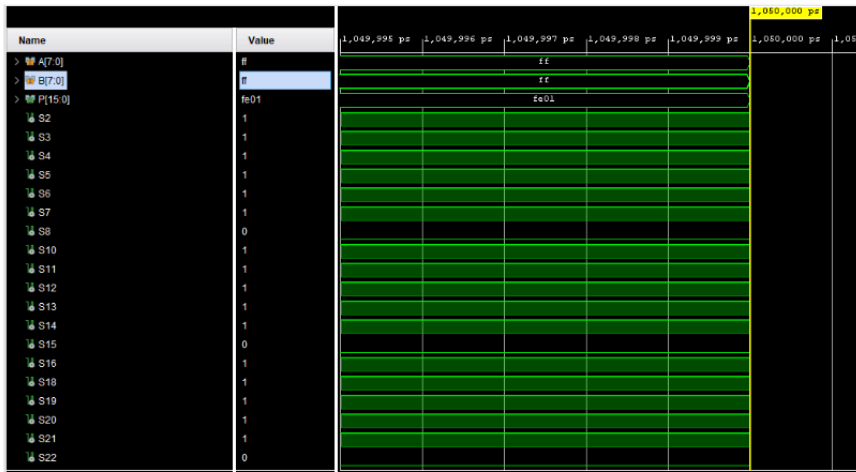


Figure 27 - FFxFF Hexadecimal Calculation

6.4.2.2 – RTL Design for 8x8 Wallace Tree Multiplier

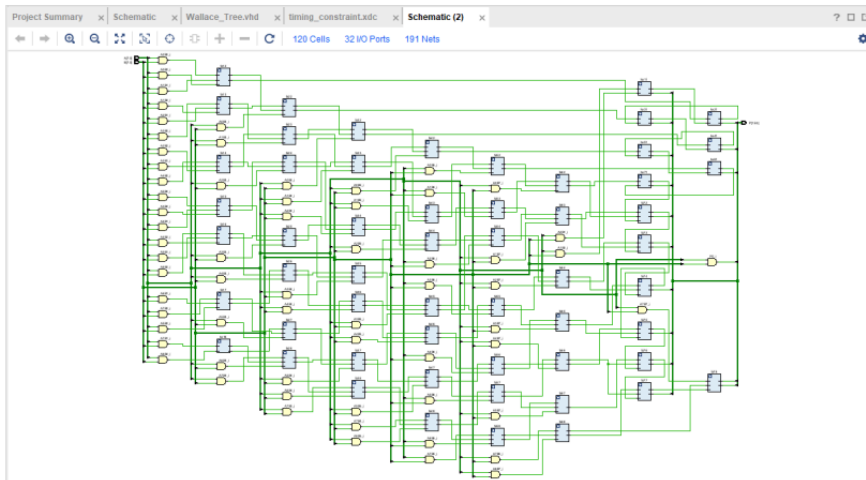


Figure 28 - 8x8 Wallace Tree Multiplier RTL Design

6.4.2.3 – Implementation Schematics of 8x8 Wallace Tree Multiplier



Figure 29 - 8x8 Wallace Tree Multiplier Implementation Schematic

6.4.2.4 – 8x8 Wallace Tree Multiplier Device

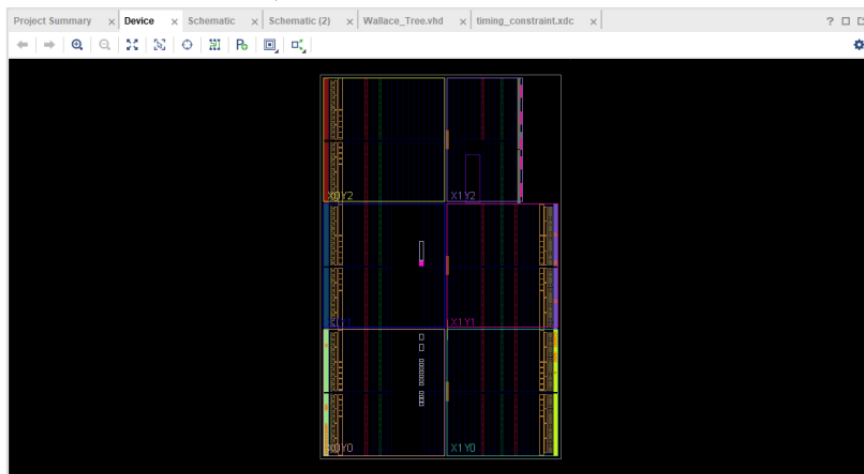


Figure 30 - Device for 8x8 Wallace Tree Multiplier

6.4.2.5 – I/O Planning for 8x8 Wallace Tree Multiplier

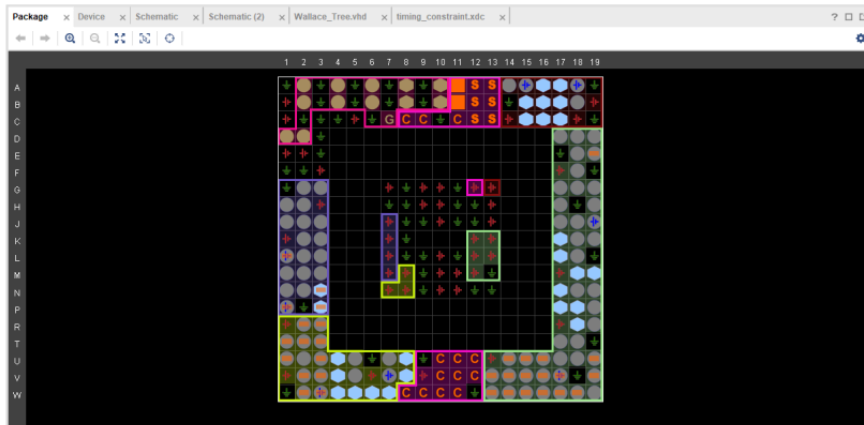


Figure 31 - 8x8 Wallace Tree Multiplier I/O Planning

12

7.0 – Simulation Analysis

With both multipliers successfully simulated, the results can now be compared to determine the differences between each.

7.1 – Timing Analysis

To begin, by calculating the longest delay by looking at both sections, 5.2 – Logic Circuit Design for 8x8 Brauns Multiplier and 6.3 – Logic Circuit Design for 8x8 Wallace Tree Multiplier. It can be determined that for the Brauns Multiplier, Figure 10 shows 20 adders in the longest route, therefore the longest delay is expected to be.

$$20 \times 2 + 1 = 41nS$$

13

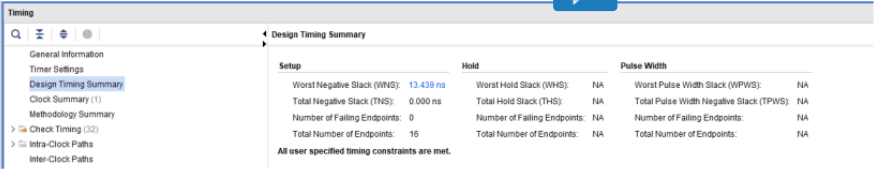


Figure 32 - Report Timing Summary of 8x8 Brauns Multiplier

Figure 22, shows that the Wallace Tree Multiplier has only 14 adders on its longest route, therefore the longest delay is expected to be.

$$14 \times 2 + 1 = 29nS$$

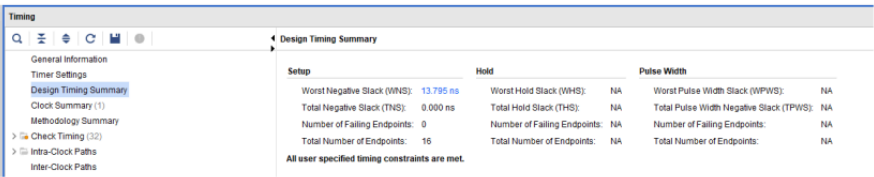


Figure 33 - Report Timing Summary of 8x8 Wallace Tree Multiplier

With these equations, it will be expected that the Wallace Tree is quickest.

From Port	To Port	M a	Max Process Corner
A[3]	P[14]	23.839	SLOW
A[0]	P[15]	23.816	SLOW
B[1]	P[15]	23.733	SLOW
A[0]	P[13]	23.661	SLOW
B[1]	P[13]	23.578	SLOW
A[1]	P[14]	23.451	SLOW
A[0]	P[14]	23.259	SLOW
B[0]	P[15]	23.233	SLOW
B[1]	P[14]	23.176	SLOW
B[0]	P[13]	23.078	SLOW
A[2]	P[12]	23.035	SLOW
A[4]	P[15]	22.909	SLOW
A[3]	P[12]	22.895	SLOW
A[4]	P[13]	22.754	SLOW
B[0]	P[14]	22.676	SLOW
A[1]	P[12]	22.507	SLOW
B[2]	P[15]	22.417	SLOW
A[6]	P[15]	22.394	SLOW
A[4]	P[14]	22.352	SLOW
A[0]	P[12]	22.315	SLOW
A[2]	P[11]	22.273	SLOW
B[2]	P[13]	22.262	SLOW
A[6]	P[13]	22.238	SLOW
B[1]	P[12]	22.232	SLOW
A[3]	P[11]	22.133	SLOW
A[5]	P[15]	21.989	SLOW
B[2]	P[14]	21.880	SLOW
A[6]	P[14]	21.836	SLOW
A[5]	P[13]	21.834	SLOW
A[1]	P[11]	21.745	SLOW
B[0]	P[12]	21.732	SLOW
A[0]	P[11]	21.553	SLOW
B[1]	P[11]	21.470	SLOW
A[5]	P[14]	21.432	SLOW
A[4]	P[12]	21.408	SLOW
A[7]	P[15]	21.103	SLOW

From Port	To Port	M a	Max Process Corner
A[4]	P[15]	12.180	SLOW
A[3]	P[15]	12.175	SLOW
B[2]	P[15]	12.175	SLOW
A[4]	P[14]	12.174	SLOW
B[1]	P[15]	12.172	SLOW
A[2]	P[15]	12.169	SLOW
A[3]	P[14]	12.169	SLOW
B[2]	P[14]	12.169	SLOW
B[1]	P[14]	12.166	SLOW
B[0]	P[15]	12.164	SLOW
A[2]	P[14]	12.163	SLOW
A[5]	P[15]	12.162	SLOW
B[0]	P[14]	12.158	SLOW
A[5]	P[14]	12.156	SLOW
A[4]	P[10]	12.058	SLOW
A[3]	P[10]	12.054	SLOW
B[2]	P[10]	12.054	SLOW
B[1]	P[10]	12.051	SLOW
A[2]	P[10]	12.048	SLOW
B[0]	P[10]	12.043	SLOW
A[5]	P[10]	12.041	SLOW
A[7]	P[15]	12.028	SLOW
A[6]	P[15]	12.027	SLOW
A[7]	P[14]	12.022	SLOW
A[6]	P[14]	12.021	SLOW
A[1]	P[15]	11.938	SLOW
A[1]	P[14]	11.932	SLOW
A[1]	P[10]	11.817	SLOW
B[5]	P[15]	11.787	SLOW
B[5]	P[14]	11.781	SLOW
A[7]	P[10]	11.771	SLOW
A[6]	P[10]	11.770	SLOW
B[5]	P[10]	11.666	SLOW
B[4]	P[15]	11.631	SLOW
B[4]	P[14]	11.625	SLOW
B[3]	P[15]	11.605	SLOW
B[3]	P[14]	11.599	SLOW
A[4]	P[12]	11.586	SLOW

Figure 34 - Longest Delays for 8x8 Brauns Multiplier (Left) and 8x8 Wallace Tree Multiplier (Right)

The equations above are confirmed by the fact that the Wallace Tree Multiplier is far faster than the Brauns Multiplier, as seen in Figure 32, which shows that it takes almost half as long.

7.2 – Power Consumption

Due to their different structures, Braun and Wallace Tree multipliers use varying amounts of electricity. Due to their regular and straightforward structure, Braun multipliers often use less dynamic power, but for big bit-widths, they have a larger area and slower performance. On the other hand, Wallace Tree multipliers, which use parallel partial product reduction to boost speed, typically use more power because of their complicated wiring and higher switching activity. Wallace trees that use more power may produce more heat, which could compromise thermal stability and need the use of more reliable power and cooling systems. In the end, Wallace Trees provide faster speeds, but their power requirements may affect the overall effectiveness of the system, particularly in low-power or battery-operated designs [27-28]. Both multipliers were given a Power Budget of 1W.

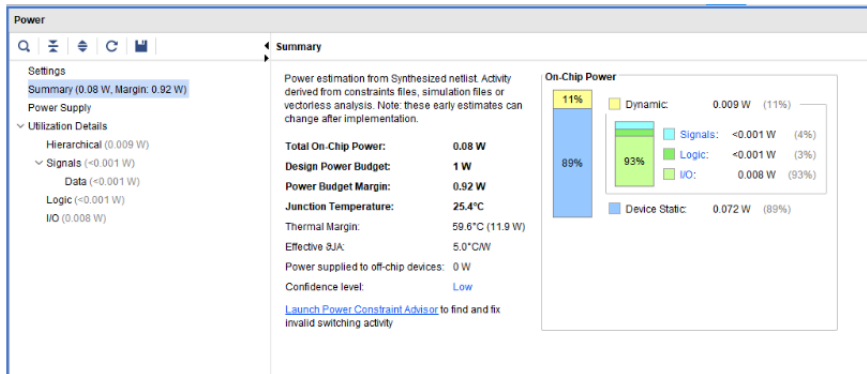


Figure 35 - Power Consumption of 8x8 Brauns Multiplier

Shown in Figure 35 is the power consumption of the Brauns Multiplier. As shown, the total On-Chip power is at 0.08W, which is the total estimated power used during operation. The Power Budget margin is 0.92W which means that only 8% power is being used, proving this is a very efficient result.

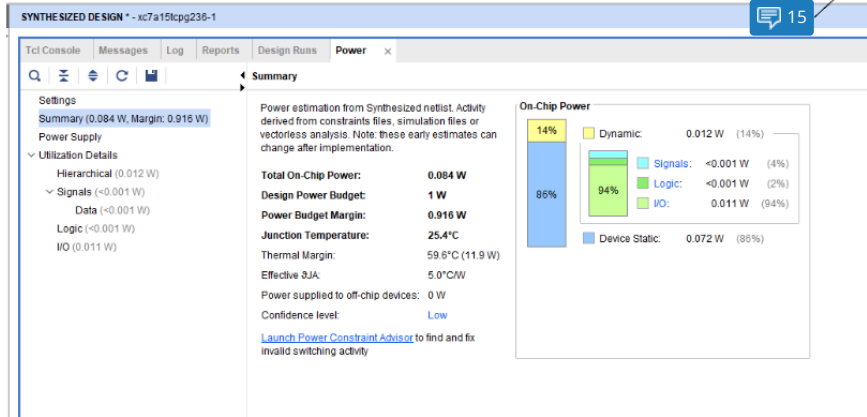


Figure 36 - Power Consumption of 8x8 Wallace Tree Multiplier

The Wallace Tree Multiplier's power usage is displayed in Figure 36. This allows for comparison to identify the most efficient multiplier.

From comparing the two, it is visible that the Brauns has a slightly lower total and dynamic power which means that it is more efficient in terms of power usage when in operation. The static power is identical between both. To conclude, the Wallace Tree is faster however it consumes more power to operate, confirming that Brauns multiplier is more efficient.

7.2.1 – Methods to improve power efficiency

One method to improve the efficiency of both, is to optimize the I/O switching. By using fewer toggles, or a lower toggle rate during its testing, it will reduce the load capacitance on the I/Os. A second method is power gating, this method completely turns off power to unused blocks during periods of no or low activity, this method will also help with power leakage.

Finally, a third method of improving efficiency is Clock Gating. This method simply shuts off the clock to idle sections of logic to reduce the power used during dynamic switching. Each of these can be implemented into the program to allow it to run more efficiently [29-30].

8.0- Conclusion

To conclude this report,  successful simulation of an 8x8 Brauns Multiplier and an 8x8 Wallace Tree Multiplier was designed using Partial Product Arrays and Logic Circuit Designs. This allowed for the multipliers pattern to be visualised and then transferred into a VHDL program which was then simulated using Vivado software.

The Wallace Tree Multiplier was clearly quicker than the Brauns during the simulation, but the Brauns were able to operate more effectively and consume less electricity. The simulation also allowed for Hexadecimal equations to be carried out to determine that the output and pattern of the Logic Circuit Design was correct, and the program was able to provide the correct output. Using this, the program was also input to a Basys 3 PCB where a sequence  output the correct answer depending on the input sequence which was determine by the swi  This can be viewed in Appendix C. Overall, the design methods and simulation were a complete success. With each method used successfully helping to create the final VHDL program.

References

Bibliography

- [1] geeksforgeeks, "Logic Gates - Definition, Types, Uses," *GeeksforGeeks*, Nov. 09, 2021. <https://www.geeksforgeeks.org/logic-gates/>
- [2] Geeksforgeeks.org, *Types of Logic Gates*. 2025.
- [3] S. D. Brown and Z. G. Vranesic, *Fundamentals of Digital Logic with VHDL Design*. McGraw Hill, 2005, p. 739.
- [4] Asim, "Multipliers," 2013. Available: https://users.ensc.concordia.ca/~asim/COEN_6501/Lecture_Notes/L3_Notes.pdf
- [5] M. Thakur, "Design of Braun Multiplier with Kogge Stone Adder & It's Implementation on FPGA," *Ijser.org*, 2025. <https://www.ijser.org/paper/Design-of-Braun-Multiplier-with-Kogge-Stone-Adder-Its-Implementation-on-FPGA.html> (accessed Mar. 20, 2025).
- [6] J. T. Yan and Z. W. Chen, *Braun Multiplier*. 2015. Available: https://engineeringresearch.org/index.php/GJRE/article/download/1223/4-Design-of-Low-Power_html?inline=1
- [7] A. R. Bagyaveereswaran, "Brauns Multiplier Implementation using FPGA with Bypassing Techniques," 2011.
- [8] S. Naga Leela, B. Manisha, P. Bharath, and E. Praneeth, "Design of Wallace tree multiplier circuit using high performance and low power full adder," *E3S Web of Conferences* 391, vol. 1, no. 1, 2023, doi: <https://doi.org/10.1051/e3sconf/202339101025>.
- [9] P. Mishra, *Wallace Tree Multiplier*. 2018. Available: https://www.researchgate.net/figure/Wallace-tree-multiplier-with-partial-product-and-various-stages_fig2_327209221
- [10] D. M Ravi, "Design and Implementation of Wallace Tree Multiplier using Higher Order Compressors DHANYA M RAVI," 2016. Available: <https://www.ijvdc.org/uploads/546213IJVDCS9984-85.pdf>
- [11] GeeksforGeeks, "Difference between Half Adder and Full Adder," *GeeksforGeeks*, Oct. 09, 2020. <https://www.geeksforgeeks.org/difference-between-half-adder-and-full-adder>
- [12] GeeksforGeeks, "Half Adder Using Verilog," *GeeksforGeeks*, Jul. 2023, doi: <https://doi.org/10.17125041/lnkedandkmap1-200x155>.
- [13] tutorialspoint, "Half Adder in Digital Electronics," *Tutorialspoint.com*, 2024. <https://www.tutorialspoint.com/digital-electronics/digital-electronics-half-adder.htm>
- [14] GeeksforGeeks, "Full Adder in Digital Logic," *GeeksforGeeks*, Mar. 21, 2017. <https://www.geeksforgeeks.org/full-adder-in-digital-logic/> (accessed Mar. 20, 2025).
- [15] tutorialspoint, "Full Adder in Digital Electronics," *Tutorialspoint.com*, 2024. <https://www.tutorialspoint.com/digital-electronics/digital-electronics-full-adder.htm>

- [16] V. Kaushik and H. Saini, "A Review on Comparative Performance Analysis of Different Digital Multipliers," *Advances in Computational Sciences and Technology*, vol. 10, no. 5, pp. 1257–1272, 2017, Accessed: Mar. 20, 2025. [Online]. Available: https://www.ripublication.com/acst17/acstv10n5_54.pdf?utm_source=chatgpt.com
- [17] R. Anitha and V. Bagyaveereswaran, "Comparative study of Braun's Multiplier Using FPGA Devices," Jun. 2011.
- [18] Dr. S. Vallem, G. Tejaswi, H. Sidharth, and S. Reddy, "High Performance, Low Power Wallace Tree Multiplier," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 12, no. 2, pp. 20–25, Jul. 2023, doi: <https://doi.org/10.35940/ijrte.b7685.0712223>.
- [19] University of Pennsylvania, "HW 7: 4x4 Array Multiplier," 2019.
- [20] T. Agarwal, "4x4 Array Multiplier : Construction, Working and Applications," *ElProCus - Electronic Projects for Engineering Students*, Mar. 18, 2020. <https://www.elprocus.com/4x4-array-multiplier-and-its-working/>
- [21] T. Gopi, "Design of Efficient Braun Multiplier for Arithmetic Applications," *International Journal of Science and Research (IJSR)*, vol. 10, no. 7, pp. 1483–1500, Jul. 2021, doi: <https://doi.org/10.21275/sr21726074026>.
- [22] K. Mathew, S. Latha, T. Ravi, and E. Logashanmugam, "Design and Analysis of an Array Multiplier Using an Area Efficient Full Adder Cell in 32nm CMOS Technology," 2013.
- [23] V. Raja Bulu and S. M. Rezaul Hasan, *4X4 Array Multiplier*. 2019.
- [24] T. L. Floyd, *Digital fundamentals*. Pearson, 2011, pp. 286–287.
- [25] H. K N and S. B G, "Efficient Design of Compact 8-bit Wallace Tree Multiplier Using Reversible Logic," *International Journal of Engineering and Manufacturing*, vol. 12, no. 4, pp. 29–36, Aug. 2022, doi: <https://doi.org/10.5815/ijem.2022.04.03>.
- [26] ushendra's engineering tutorials, "4x4 WALLACE TREE MULTIPLIER | VLSI design," *YouTube*, Aug. 06, 2024. <https://www.youtube.com/watch?v=5VIKv6HtiZ4> (accessed Apr. 12, 2025).
- [27] B. Chennaiah, M. Omnamasivaya, V. Hemanth, M. Suresh, and C. H. Harika, "Performance Analysis of Different Multipliers," 2018.
- [28] K. Juneja, A. Jangra, and D. Khurana, "Design of a Quaternary Component and Wallace Tree Integrated Baugh-Wooley Multiplier," 2024.
- [29] Y. Shin, Jun Seomun, K.-M. Choi, and T. Sakurai, "Power gating," *ACM Transactions on Design Automation of Electronic Systems*, vol. 15, no. 4, pp. 1–37, Sep. 2010, doi: <https://doi.org/10.1145/1835420.1835421>.
- [30] S. Jairam, R. Madhusudan, J. Srinivas, P. Vishwanath, H. Udayakumar, and R. Jagdish, "Clock Gating for Power Optimization in ASIC Design Cycle: Theory & Practice," 2008.

Appendix A

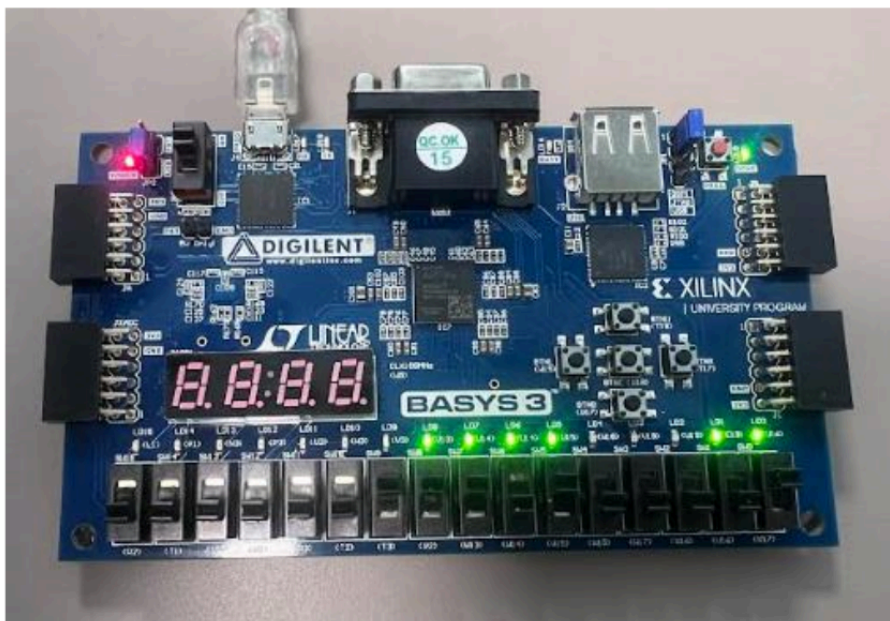
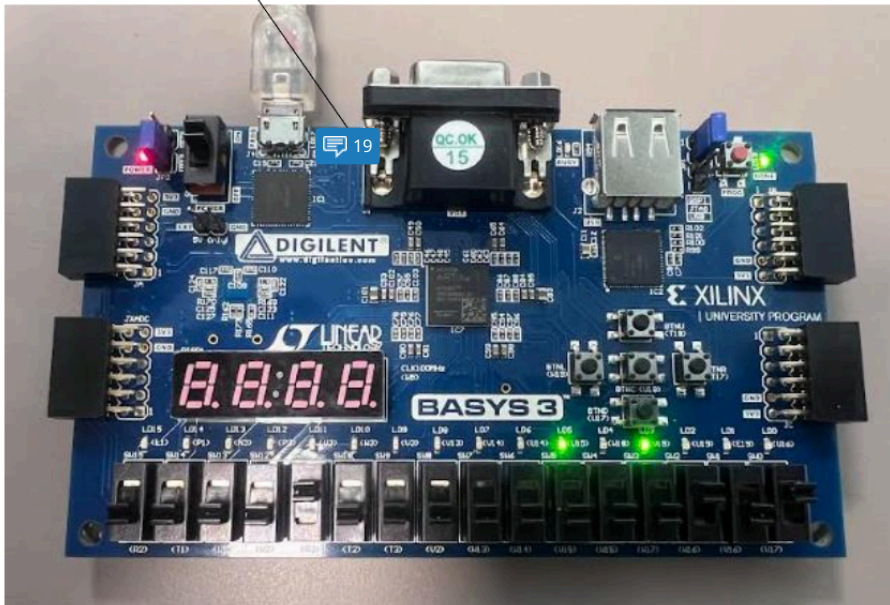
See 8x8 Brauns Multiplier CHDL Program in supporting documents file.

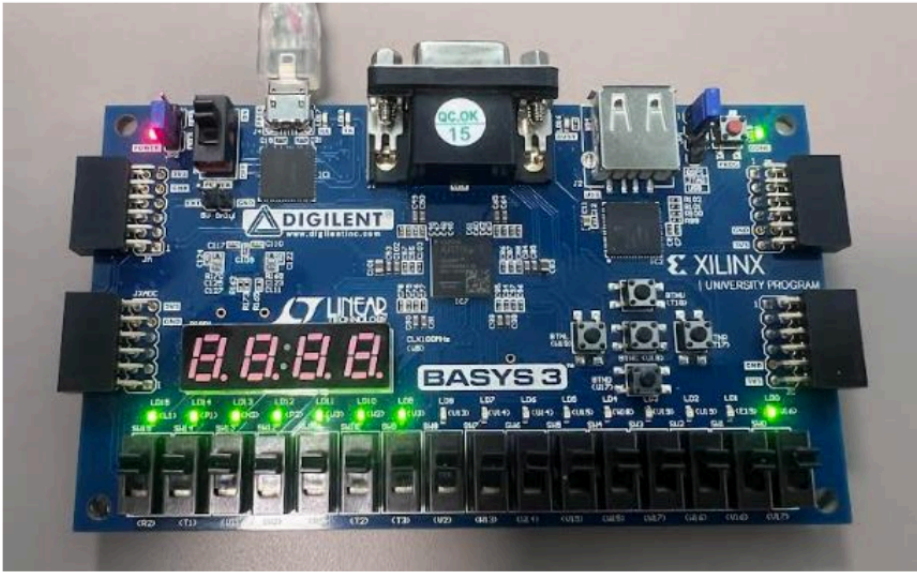
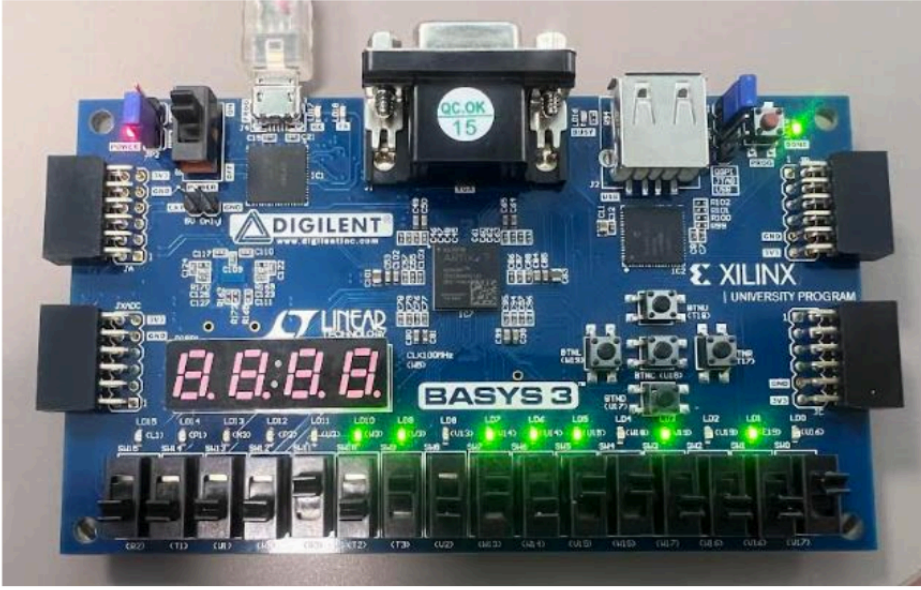
Appendix B

See 8x8 Wallace Tree Multiplier VHDL Program in supporting documents file.

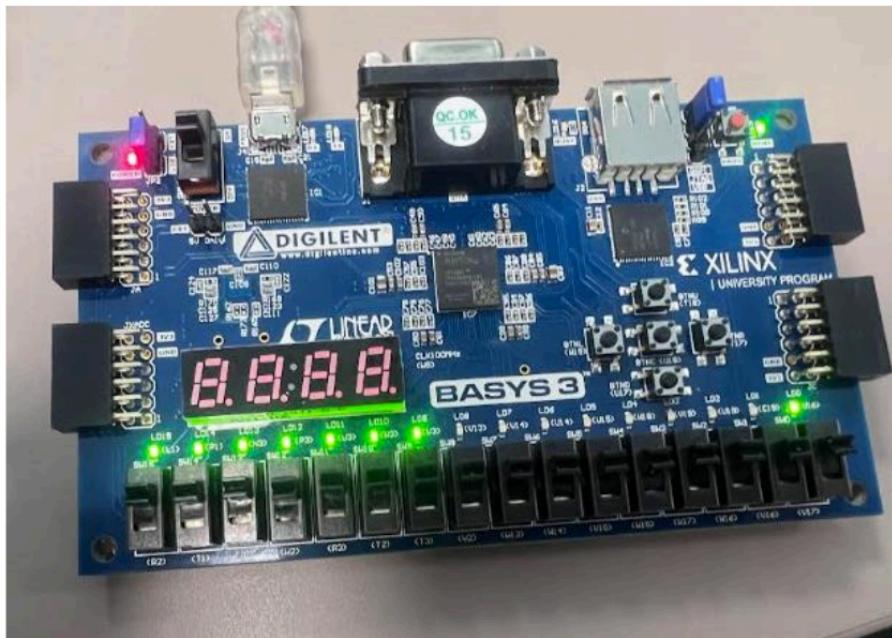
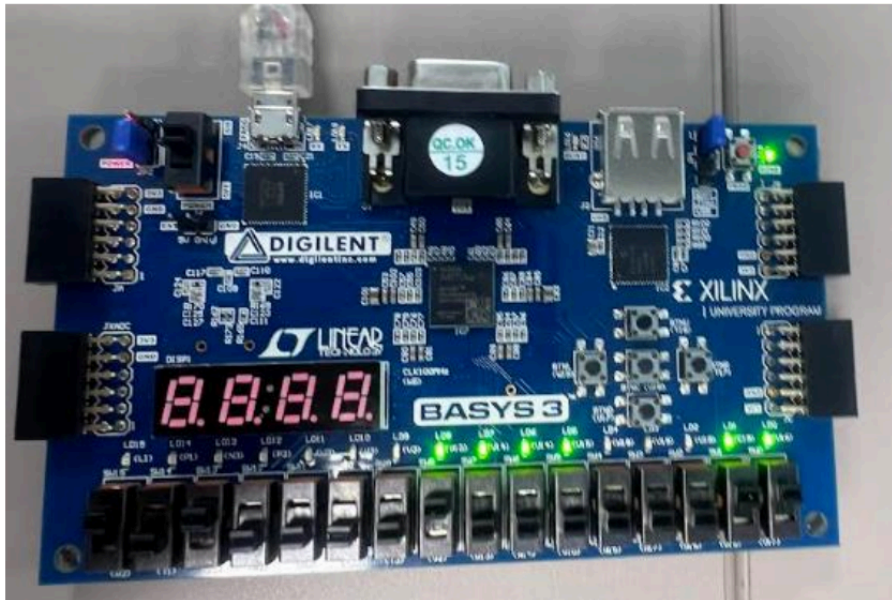
Appendix C

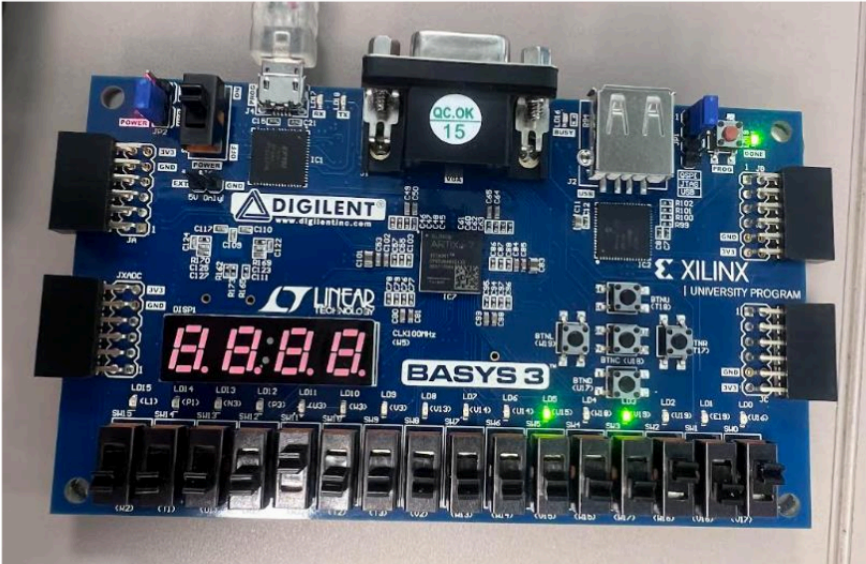
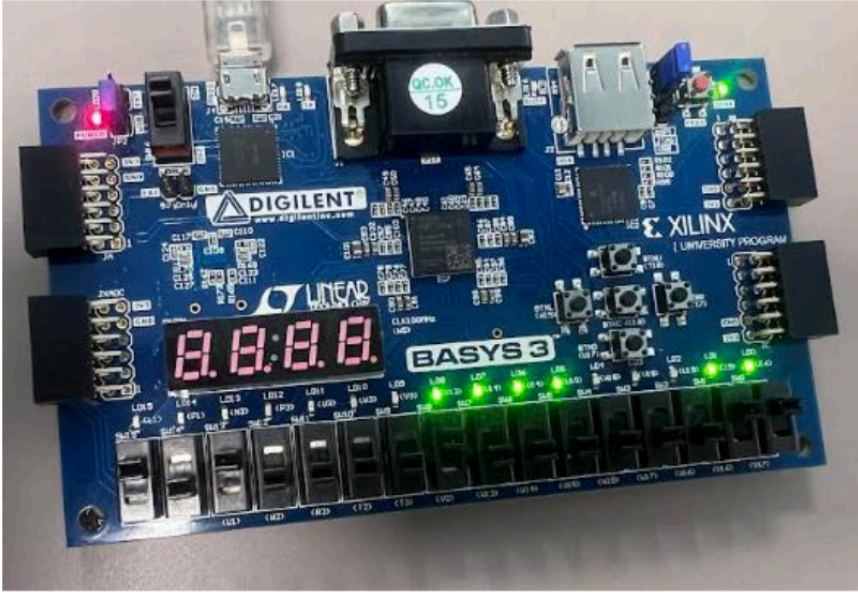
Wallace Tree Multiplier





Brauns Multiplier





Ethan Longley - Assignment ACDA6001

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

74/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5



Comment 1 |  Engagement with literature(20)

The concepts of logic gates are well covered, which is commendable.

PAGE 6



Comment 2 |  Knowledge & Understanding(20)

The technical explanation of Braun's and Wallace multipliers is satisfactorily presented, accompanied by relevant diagrams that illustrate additional stages and the ripple carry adder, which is a strong point.

PAGE 7



Comment 3 |  Engagement with literature(20)

The basics of half adders and full adders are also included, with a detailed explanation that is appreciable.

PAGE 8



Comment 4 |  Engagement with literature(20)

It is good to see that the advantages and applications are also addressed.



Comment 5 | Knowledge & Understanding(20)

An
example of partial product generation is included.



Comment 6 | Cognitive skills (20)

The Braun multiplier covers the algorithm and presents a clear architectural overview, which is excellent.



Comment 7 | Cognitive skills (20)

Port
mapping is clearly explained, which is appreciated.



Comment 8 | Practical Application (25)

Simulation
screenshots are clear, and the display of hexadecimal values is well done.



Comment 9 | Practical Application (25)

Other screenshots, such as
schematics and I/O mapping, are also provided.



Comment 10 | Cognitive skills (20)

The
stages of the dot representation in the Wallace multiplier are clearly shown,
and its architectural diagram is well captured.



Comment 11 | Practical Application (25)

Port
mapping is visible and effectively illustrated.



Comment 12 | Practical Application (25)

Additionally, the code for timing
constraints and pin assignments should be added.



Comment 13 | Cognitive skills (20)

Delay calculations are included and
are commendable. However, including a critical path diagram would further
enhance the report.



Comment 14 | Cognitive skills (20)

The
combinational delay and timing requirements are well presented.



Comment 15 | Cognitive skills (20)

There
is some comparison between the two multipliers, and power comparisons are also
included, which is clearly demonstrated.

To
further strengthen the report, a more detailed conclusion could be added.



Comment 16 | **Practical Application (25)**

The report would benefit from the inclusion of the Utility Report details (such as the number of LUTs) and FPGA pin assignment screenshots.



Comment 17 | **Cognitive skills (20)**

It should summarise key results, including actual values of delay, power, and the number of LUTs, to improve the quality of the conclusion.



Comment 18 | **Transferable skills (15)**

The report is well-organised and flows logically. Citations are properly included, and all software procedures appear to have been correctly followed. Only minor flaws are observed.



Comment 19 | **Practical Application (25)**

The hardware outputs are included, which is commendable.

GRADING FORM: UNTITLED

ENGAGEMENT WITH LITERATURE(20)

Assesses clarity of definition, technical explanation, advantages, disadvantages, applications, and use of relevant references. Also includes critical analysis, research gaps, and proper citation and referencing.



Shows very good use of appropriate and up-to-date literature. References are mostly relevant and well connected to the topic. Critical analysis is well reasoned, and research gaps are addressed with clarity. Citations are mostly accurate, and formatting is consistent. Good range of sources.

14.4**KNOWLEDGE & UNDERSTANDING(20)**

This section evaluates understanding of binary multiplication, multiplier architectures, logic design, and adders. It also includes key concepts like power efficiency, delay path, LUTs, and slack, reflecting application of core engineering principles.



Very good understanding of core concepts. Technical explanations are mostly clear and correct. Binary multiplication and multiplier structures are well explained. Delay, slack, and low-power concepts are addressed with minor gaps. Demonstrates confident application of engineering principles across tasks.

14.7**COGNITIVE SKILLS (20)**

This section includes analysis of design structures, performance interpretation, and comparison of results using appropriate

techniques. Emphasis is placed on understanding technical relationships, interpreting data, and drawing meaningful conclusions with supporting evidence such as calculations, graphs, and recommendation



Shows strong analytical ability. Most design analyses are accurate, with logical interpretation of delay and performance parameters. Graphs or comparisons support conclusions well. Minor details may lack depth, but overall recommendations and insights are valid and technically sound.

14.8

PRACTICAL APPLICATION (25)

This section assesses the practical skills in VHDL implementation, port mapping, FPGA pin assignment, simulation, and hardware proof. It also includes synthesis, applying XDC constraints, and interpreting reports on LUTs, power, delay, and timing, reflecting effective tool usage.



Very good execution of practical tasks. VHDL code is functionally sound; port mapping and pin assignments are mostly correct. Simulation and hardware outputs are included with minor gaps. FPGA synthesis and report handling are well managed. Constraints are applied with general accuracy.

17.8

TRANSFERABLE SKILLS (15)

This section evaluates the ability to communicate work clearly and effectively. It covers report writing, clarity of diagrams, structured presentation, and appropriate use of screenshots. It also assesses how well findings are explained, the accuracy with which procedures are followed, and how results are summarised. Communication should be suited to both technical and non-technical audiences, demonstrating professionalism and clarity.



Very good communication. Report is coherent and includes clear diagrams and screenshots. Findings are explained well and structure is mostly consistent. Procedural accuracy and summary are evident. Minor improvements could be made in tailoring language for diverse audiences.

11.5