

Electronics 2 Oliver Kucyj

by Oliver Kucyj

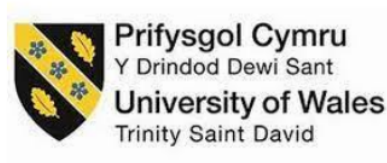
Submission date: 10-May-2023 01:11PM (UTC+0100)

Submission ID: 205570560

File name: 33333_Oliver_Kucyj_Electronics_2_Oliver_Kucyj_497727_783203735.docx (8.25M)

Word count: 4149

Character count: 23701



University of Wales Trinity Saint David
Electronics 2

Critical Evaluation of Braun's Multiplier and Wallace Tree Multiplier using VHDL

Author: Oliver Kucyj
Student number: 1910808
Course: B.Eng Electrical and Electronic Engineering
Module Code: ACDA6001
Lecturer: Jenifer Prince Devaraj
Date: 10th May 2023

Table of Contents

1	<i>Abstract</i>	3
2	<i>Introduction</i>	4
3	<i>Background research</i>	5
3.1	Field-programmable Gate Arrays	5
3.2	Multiplier Building blocks	7
3.3	Types of Multipliers	10
4	<i>Designing the multipliers</i>	14
4.1	8x8 bit Braun's multiplier	14
4.2	Wallace multiplier	30
5	<i>Comparison of the results</i>	42
6	<i>Conclusion</i>	56
7	<i>References</i>	57
	<i>Appendix A</i>	59
	<i>Appendix B</i>	64
	<i>Appendix C</i>	66

1 Abstract

This report investigates the Braun's and Wallace-Tree multiplier designs. Simulation results from Xilinx Vivado and background research on the multipliers are used to conclude that the Wallace-tree multiplier provides the better option in applications where speed is critical. The Wallace-Tree design uses 8.4% more LUTs, however the Braun's multiplier has a maximum delay which is 10.46% longer than the Wallace tree multiplier.

The report concludes that for most applications the Wallace tree multiplier is the most suitable options due to the superior processing speeds. In an application where hardware utilization is critical, the Braun's multiplier may be more suitable.

2 Introduction

This report will investigate, simulate and test two parallel multipliers designed to be implemented onto FPGAs (Field Programmable Gate Arrays) using VHDL coding. The Xilinx Vivado software will be used to do this. The parallel multipliers which will be used are the Braun's multiplier and the Wallace-Tree multiplier. They will be 8x8 bit multipliers, meaning there will be 16 inputs and 16 outputs.



The results of the investigation and simulation together should clearly demonstrate how the characteristics of each multiplier differ. This should allow a conclusion to be made on which of the designs provides the best solution for a parallel multiplier.

3 Background research

Section 3 will present research on FPGAs and different types of parallel multipliers.

3.1 Field-programmable Gate Arrays

2

FPGAs are integrated circuits sometimes containing millions of transistors making up logic blocks. Commonly, these logic blocks are LUTs (Lookup Tables). Lookup tables contain storage cells which can represent a logic function. They have multiple select inputs which select one of the storage cells to transmit to the single output. They are connected via vertical and horizontal routes. These can be enabled and disabled by programming interconnection switches. This allows many different combinations to be created via software. A typical FPGA is shown in figure 1.

3

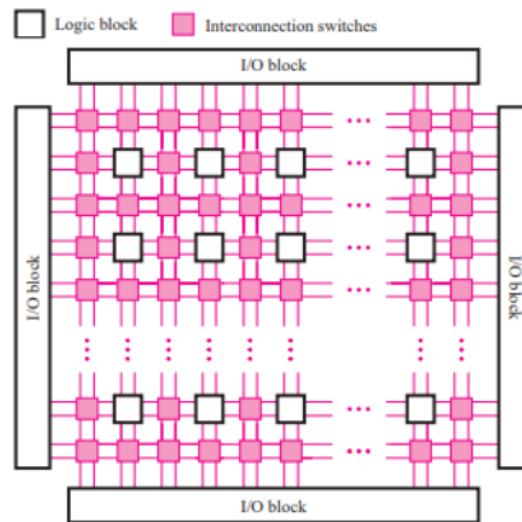


Figure 1 – typical FPGA diagram [1]

These circuits are not designed for any specific application. The designer can program the FPGA to suit almost any required application using a hardware description language such as VHDL. This allows complex parallel circuits which can achieve very high processing speeds to be implemented. Common applications include image and video processing, medical applications and telecoms.

FPGAs are Field programmable, meaning existing systems can be easily upgraded without the need to modify or rebuild any hardware. This is an advantage over Application Specific Integrated Circuits (ASICs) which are a common alternative to FPGAs.

This feature is useful in situations where regulations are continually updated, requiring updates to the system. An example of this is satellites. FPGAs which are radiation hardened and installed in satellites can be updated without accessing the device [2]. Another

advantage is that design errors can be easily rectified making them ideal for prototype designs before purchasing a specialized ASIC board once the design is finalized.

They are like microcontrollers in many ways. They are both integrated circuits which can be programmed to meet a requirement. Microcontrollers are cheap devices which can be easily programmed for most basic applications. FPGAs are expensive and complex devices which can be complicated to program. This makes microcontrollers well suited to simple applications, however FPGAs are ideal for high complexity applications thanks to their customizability and programmable hardware. FPGAs can also attain higher processing speeds thanks to the parallel arrangement, allowing multiple tasks to be carried out at the same time. Microcontrollers can only complete one line of code at a time, sometimes causing bottlenecks [3].

3.2 Multiplier Building blocks.

The multipliers investigated in this report will utilize 2-input AND gates, half adders, and full adders to multiply two 8-bit numbers resulting in a 16-bit output. The symbols and truth tables for these components are shown in figures 2-4.

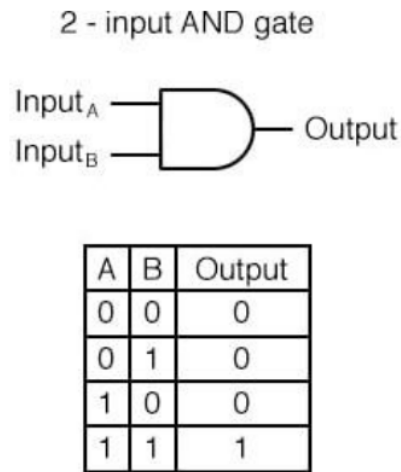


Figure 2 – AND gate truth table and logic diagram [4]

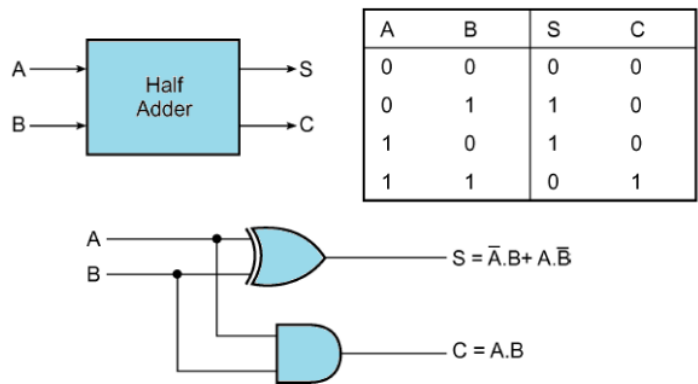


Figure 3 – Half adder truth table and logic diagram [5]

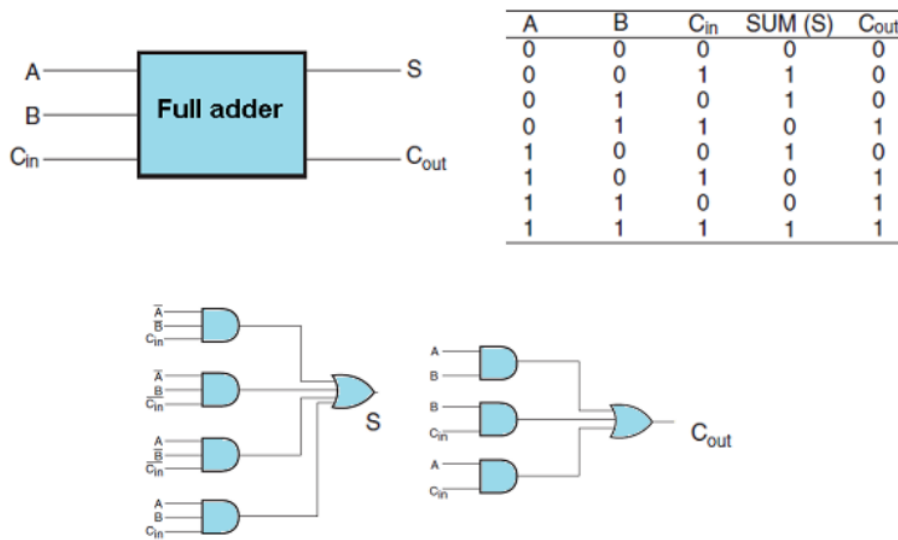


Figure 4 – Full adder truth table and logic diagram [5]

The final stage of both multipliers requires multiple bits to be added together. The simplest method of doing this is the ripple carry adder (RCA). This utilizes multiple full adders connected in series. The carry out from each full adder is connected to the carry in of the full adder to its left. The sum out of each full adder is a bit from the output. Figure 5 shows this.

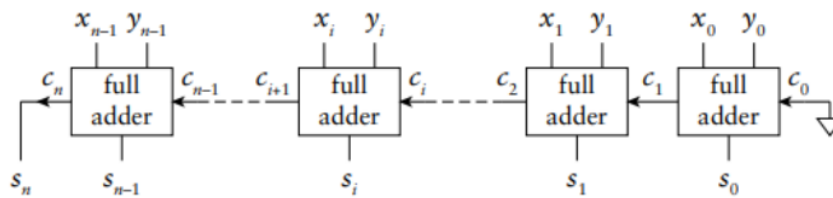


Figure 5 – 4-bit Ripple carry adder [6]

These adders are connected in series. Each adder must wait for the previous adder to generate the carry, leading to long delays. A more complex adder known as a carry lookahead adder (CLA) can be used. This adder calculates the carry beforehand to eliminate delays caused by waiting for the carry to be generated. Figure 6 shows the logic diagram.

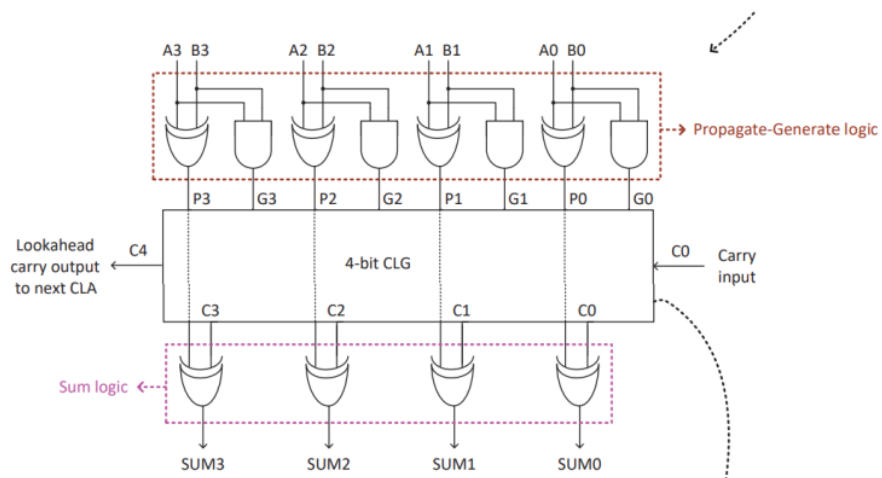


Figure 6 - 4-bit Carry lookahead adder [7]

3.3 Types of Multipliers

Multipliers play a key role in Very Large-Scale Integration (VLSI). There are many different multiplier designs available. Different multiplier designs will have different characteristics such as speed, power consumption and complexity. An ideal multiplier design should have maximum speeds with minimum power consumption [8].

This section will investigate some of these multiplier designs.

Braun's multiplier

The Braun's design is the simplest parallel multiplier. They utilize a shift and carry method. The partial products generated by the AND gates are added together using half adders and full adders. The sum from each half adder is added to the next partial product. The carry from each half adder is shifted left. The final stage consists of an RCA. As discussed in section 3.2, a CLA could be used to reduce delays. Figure 7 demonstrates the partial products of a 4x4 Braun multiplier and figure 8 demonstrates the hardware.

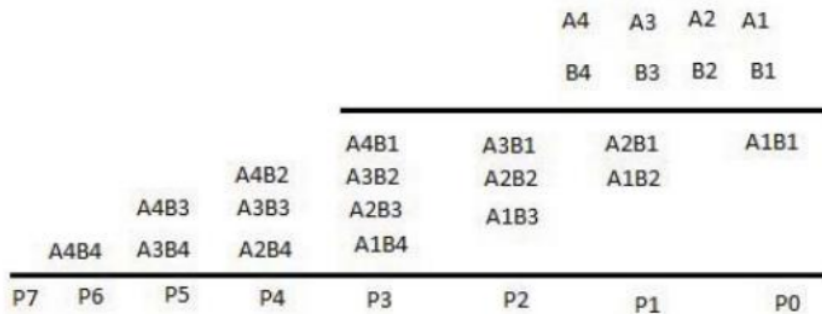


Figure 7 – 4x4 Braun's multiplier partial products [8]

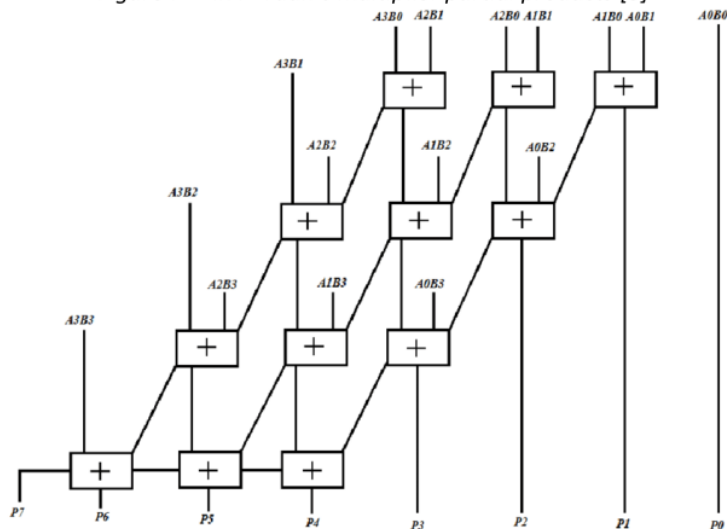


Figure 8 – 4x4 Braun's multiplier logic diagram [9]

Wallace tree multiplier

Wallace tree multipliers utilize carry and save addition. Its design is complex when compared to the Braun's multiplier. It also utilizes more hardware and consumes more power. The advantage of the Wallace tree multiplier is the improved speed performance when compared to the Braun's multiplier.

All tree type multipliers divide the multiplication into 3 stages. The generation of partial products, the reduction of partial products and then an adder. The partial products are generated using AND gates. The reduction stage uses half adders and full adders to add these partial products together in parallel arrangements. The final adder adds the remaining bits.

The partial products table, dot diagram and circuit diagram of an example 4x4 multiplier are shown in figure 9. Each partial product reduction step has a total delay equal to that of a full adder since that is the longest route in each step. All adders in each step are completed in parallel. This helps to minimize the delays.

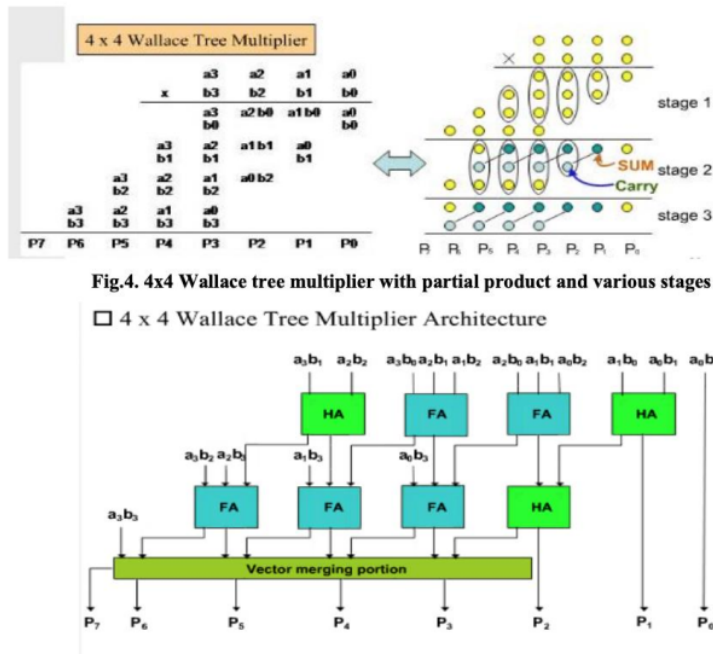


Fig.4. 4x4 Wallace tree multiplier with partial product and various stages

□ 4 x 4 Wallace Tree Multiplier Architecture

Figure 9 – 4x4 Wallace tree multiplier partial product, dot diagram and logic diagram [10]

Dadda multiplier

The Dadda multiplier utilizes the same tree style algorithm as the Wallace tree multiplier. The difference is in how the reduction is done. Whilst the Wallace tree multiplier aims to reduce the partial products as quickly as possible, the Dadda multiplier minimizes reduction to optimize the hardware usage. This means the stage 3 adder must add more bits than in

the Wallace-Tree. This presents even more reason to use a more complex adder such as the CLA.

The Dadda exhibits a low propagation delay compared to the Wallace-Tree multiplier [11] as shown in figure 10. The simulation results show that the Dadda multiplier has a slightly lower delay. Figure 11 shows the Dot diagram for a Dadda multiplier.

Table 1: Summary of parallel multipliers

Multiplier algorithm	Area (number of bit slices)	Delay (nsec)
Booth	95	3.2900
Wallace	107	14.6650
Dadda	111	12.8340

Figure 10 – Simulation results showing the delay of a Wallace, Dadda and Booth multipliers [12].

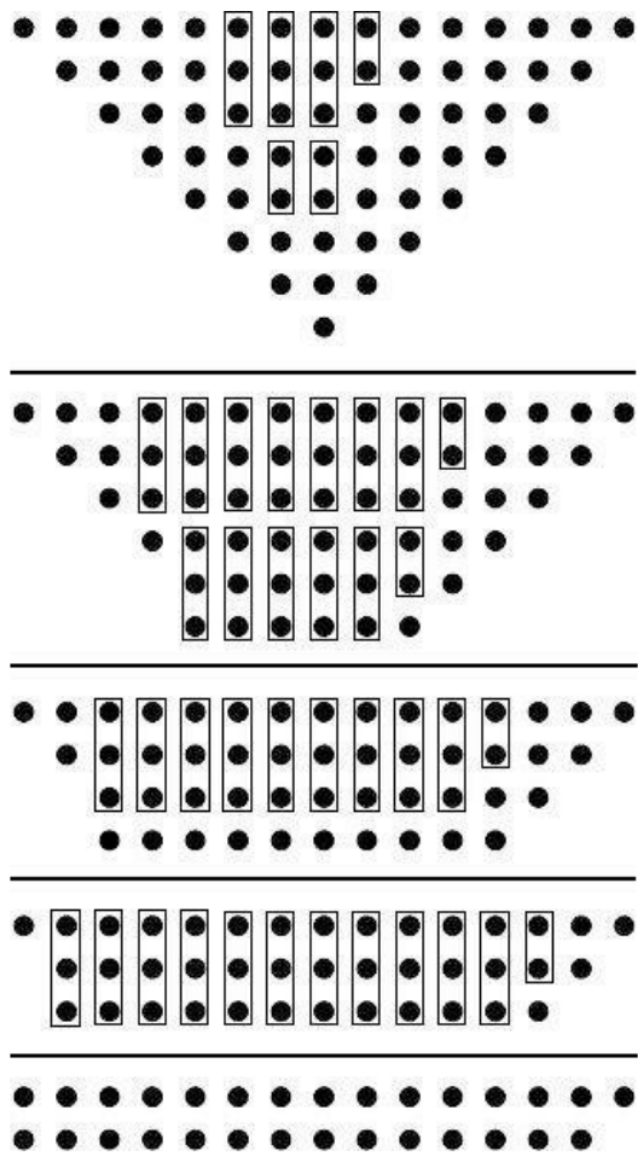


Figure 11 – Dadda multiplier dot diagram [13]

4 Designing the multipliers

This section of the report details the design of an 8x8 bit Braun multiplier and an 8x8 bit Wallace Tree multiplier.

4.1 8x8 bit Braun's multiplier

There are multiple design stages which must be complete to create a Braun's multiplier. These are the partial products, a basic flowchart showing the operation of the multiplier and the logic diagram. Figures 12-14 show this. Figures 15-17 show a closer view of the logic diagram stages. The logic diagram contains 64 AND gates, 8 half adders and 48 full adders.

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Figure 12 – Partial products of the Braun's multiplier [14]

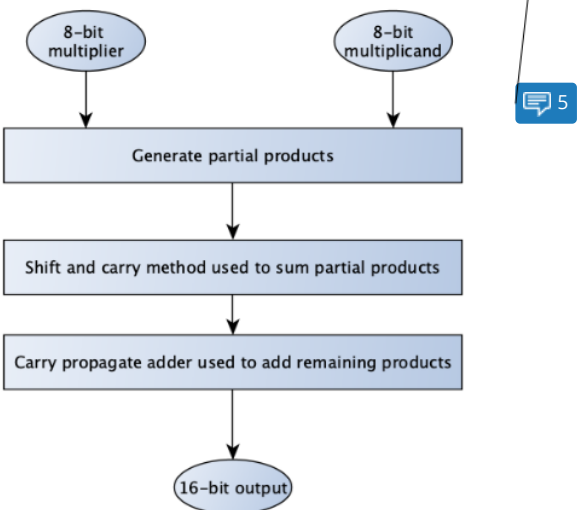


Figure 13 – Flowchart showing the basic operation

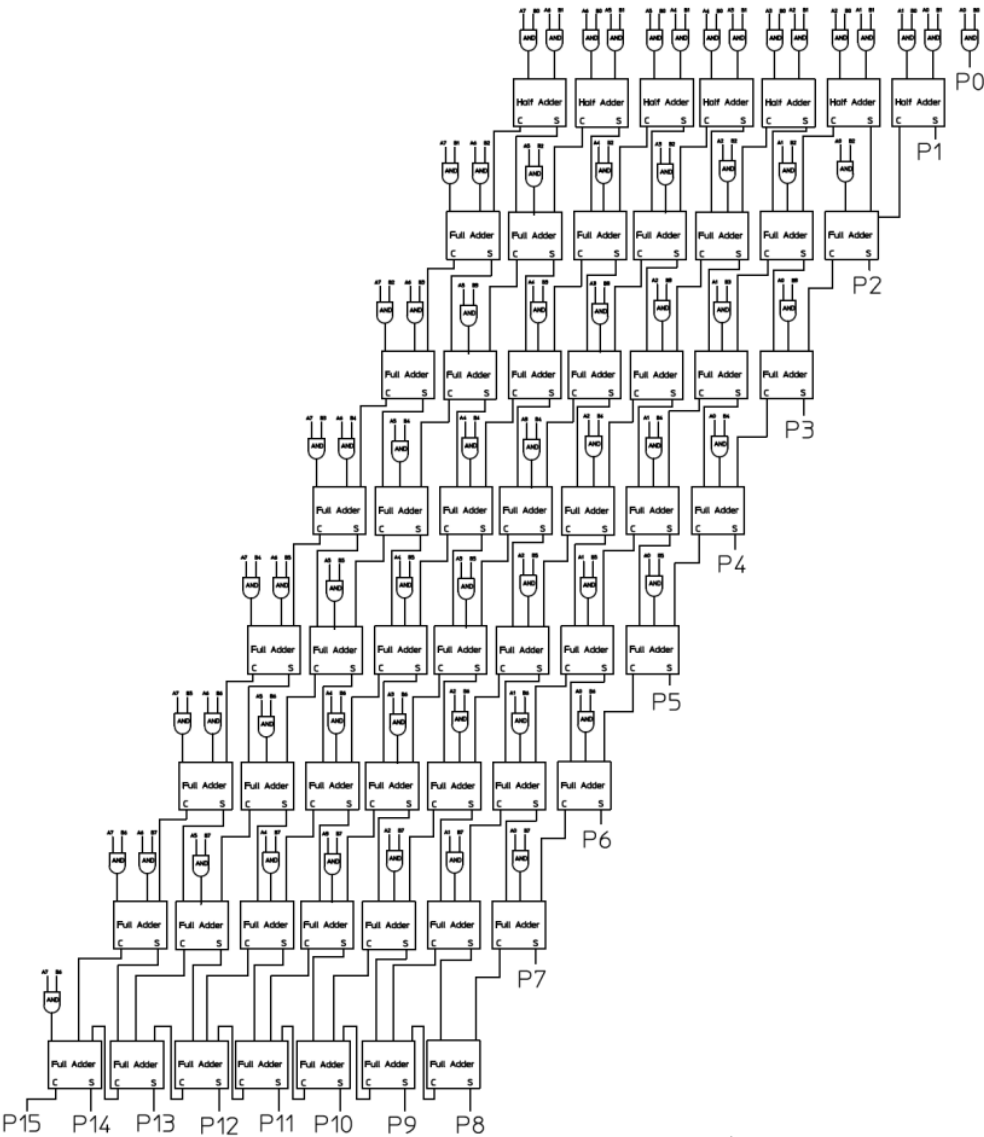


Figure 14 – Braun's multiplier logic circuit overview



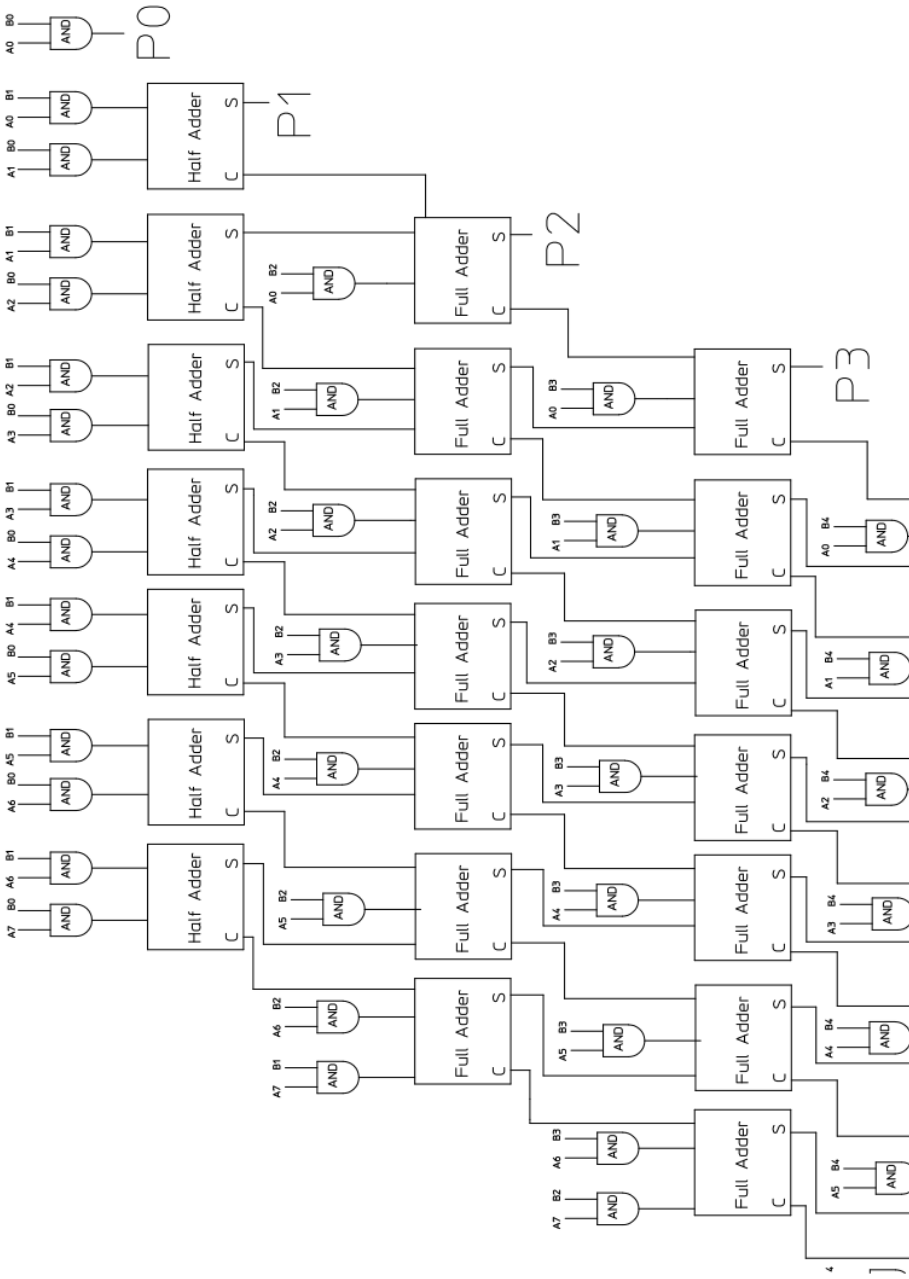


Figure 15 – Braun's logic diagram top section

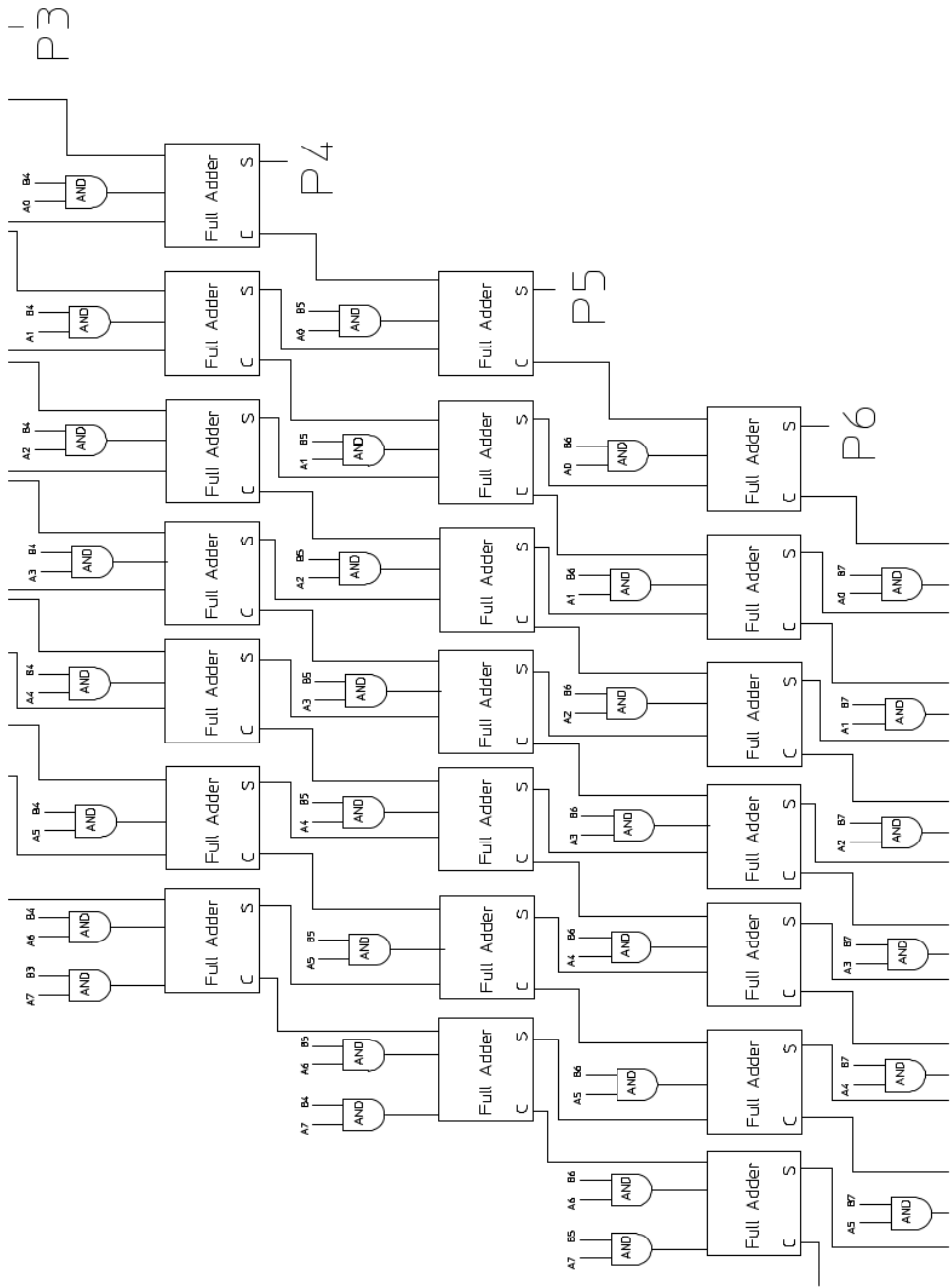


Figure 16 – Braun's logic diagram middle section

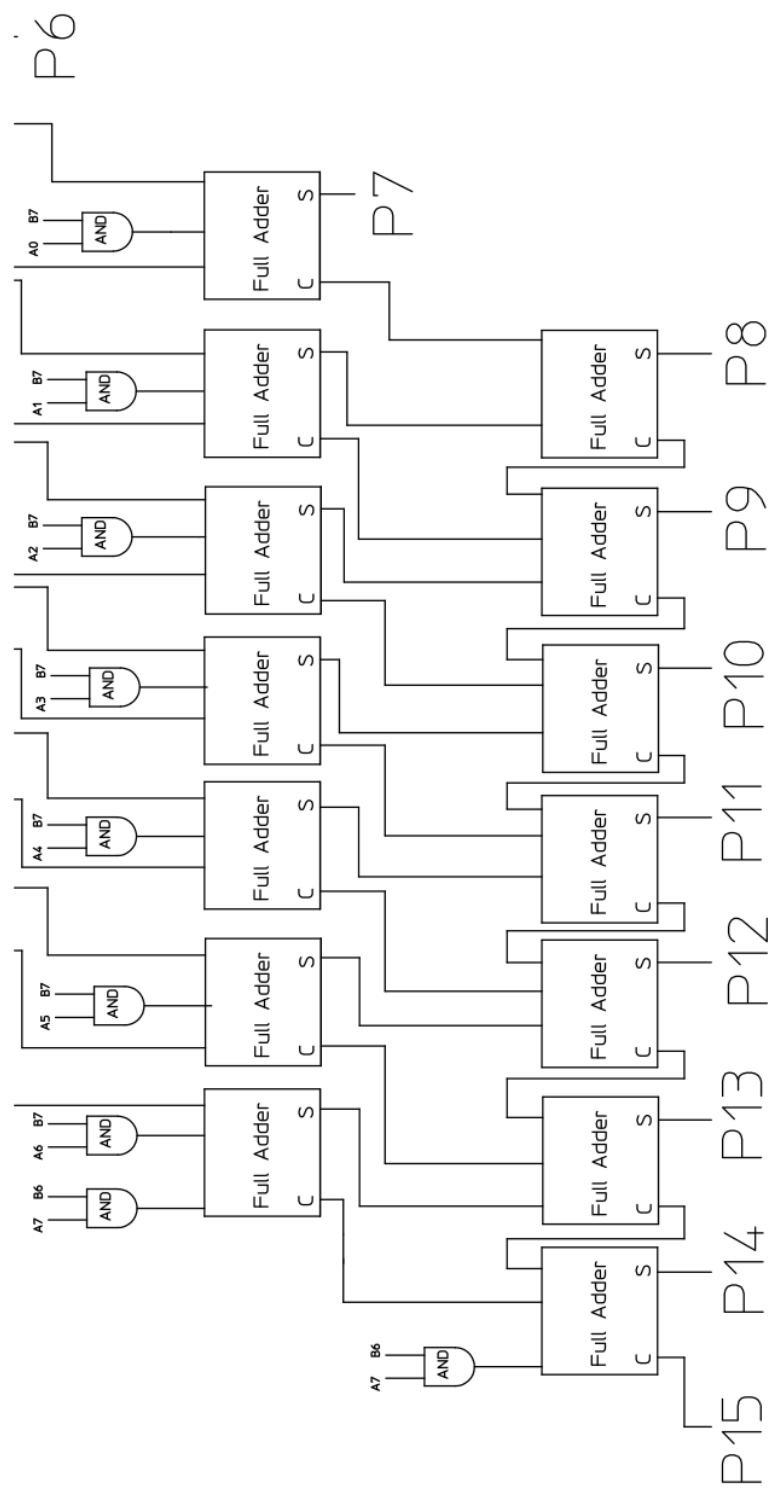


Figure 17 – Braun's logic diagram bottom section

Once these design stages were complete, the code for the Braun's multiplier was written. Structural VHDL coding was used which utilizes port mapping. This can be found in Appendix A. Appendix B shows the testbench used.

Simulating the code in Xilinx Vivado produced results to prove the design worked correctly. The testbench simulated 16 random input combinations. All 16 input combinations provided the expected results. This is shown in figure 18. Table 1 shows the inputs, expected outputs and the actual output from these 16 combinations and confirms the multiplier works.

Input A	Input B	Expected output	Actual output
11111111 (255)	11111111 (255)	1111111000000001 (65,025)	1111111000000001 (65,025)
00000000 (0)	00000000 (0)	0000000000000000 (0)	0000000000000000 (0)
00001111 (15)	11110000 (240)	0000111000010000 (3,600)	0000111000010000 (3,600)
00000000 (0)	11111111 (255)	0000000000000000 (0)	0000000000000000 (0)
01010101 (85)	10101010 (170)	0011100001110010 (14,450)	0011100001110010 (14,450)
11001100 (204)	00110011 (51)	0010100010100100 (10,404)	0010100010100100 (10,404)
00011000 (24)	11100111 (231)	0001010110101000 (5,544)	0001010110101000 (5,544)
00111111 (63)	11111111 (255)	0011111011000001 (16,065)	0011111011000001 (16,065)
00000011 (3)	00000001 (1)	0000000000000011 (3)	0000000000000011 (3)
11000111 (199)	11110011 (243)	1011110011100101 (48,357)	1011110011100101 (48,357)
01001111 (79)	10110000 (176)	0011011001010000 (13,904)	0011011001010000 (13,904)
00011001 (25)	11110111 (247)	0001100000011111 (6,175)	0001100000011111 (6,175)
01000101 (69)	10101010 (170)	0010110111010010 (11,730)	0010110111010010 (11,730)
01111100 (124)	00111011 (59)	0001110010010100 (7,316)	0001110010010100 (7,316)
00011011 (27)	11100111 (231)	0001100001011101 (6,237)	0001100001011101 (6,237)
00100111 (39)	11111111 (255)	0010011011011001 (9,945)	0010011011011001 (9,945)

Table 1 – Braun's multiplier inputs and outputs



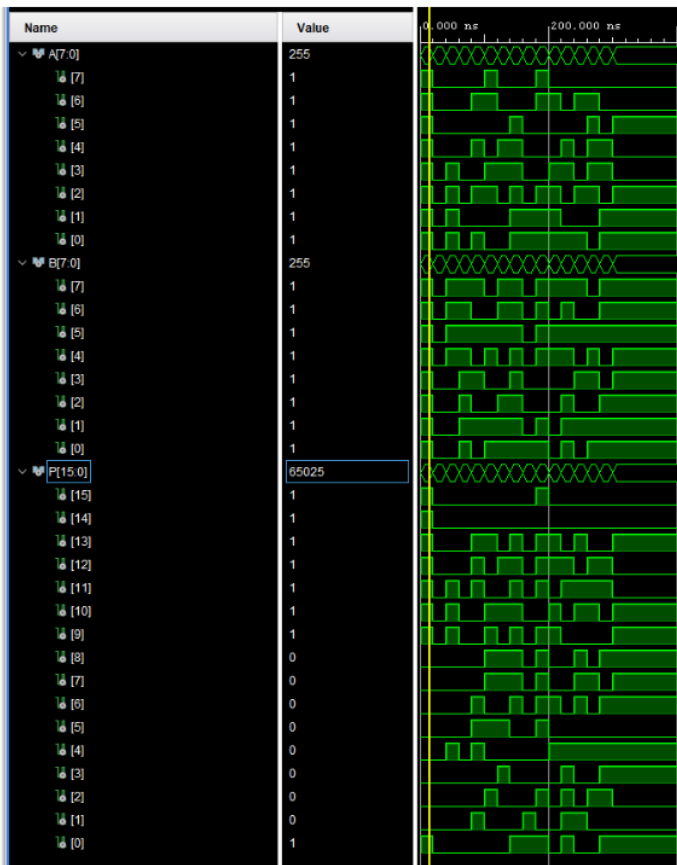


Figure 18 - Braun's multiplier results

The design was then implemented to produce a circuit schematic utilizing LUTs. Figure 19 shows the circuit schematic. This is a very complicated circuit making it difficult to properly inspect it from this expanded view. Figures 20-22 demonstrate zoomed in, detailed views of the inputs and outputs.

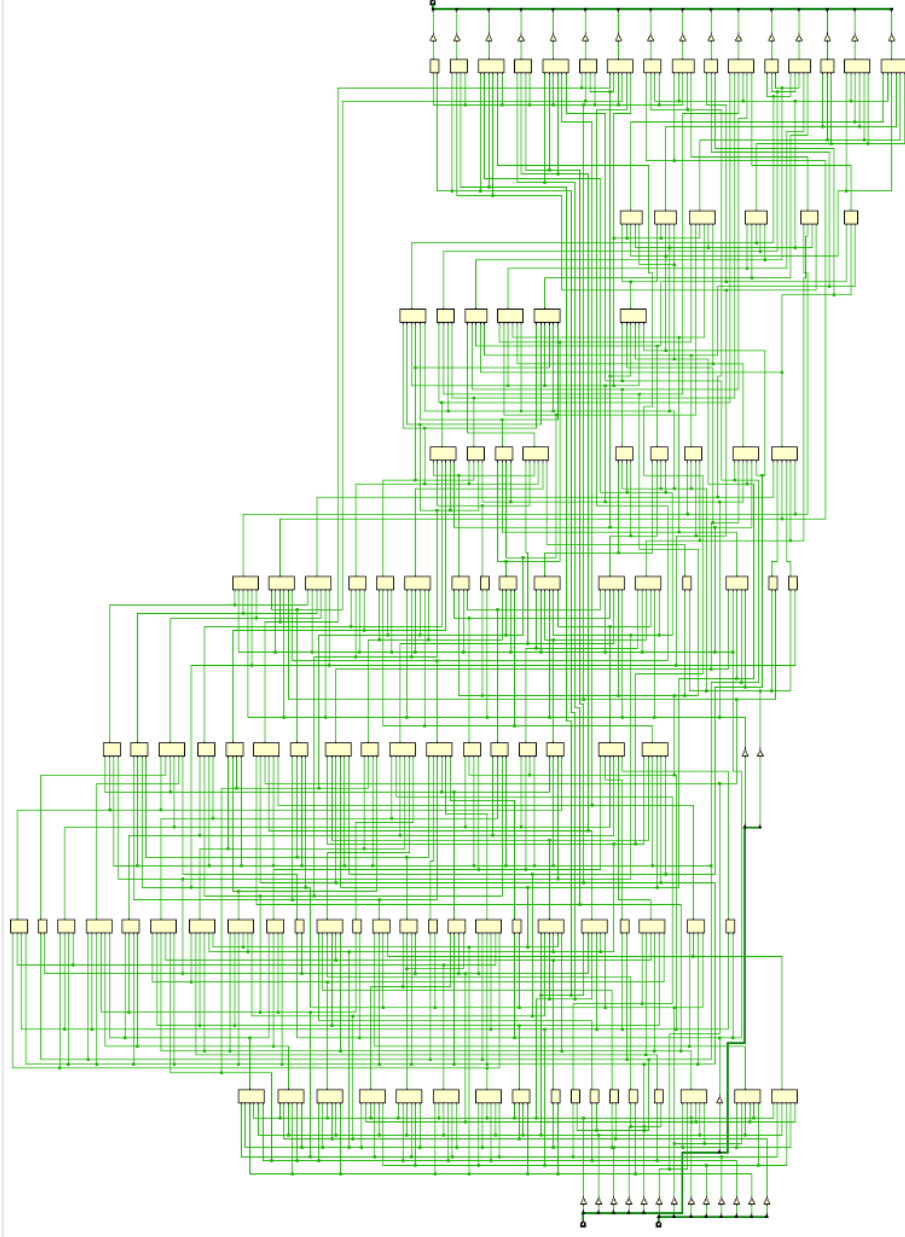


Figure 19 – expanded view of the Braun's multiplier schematic diagram



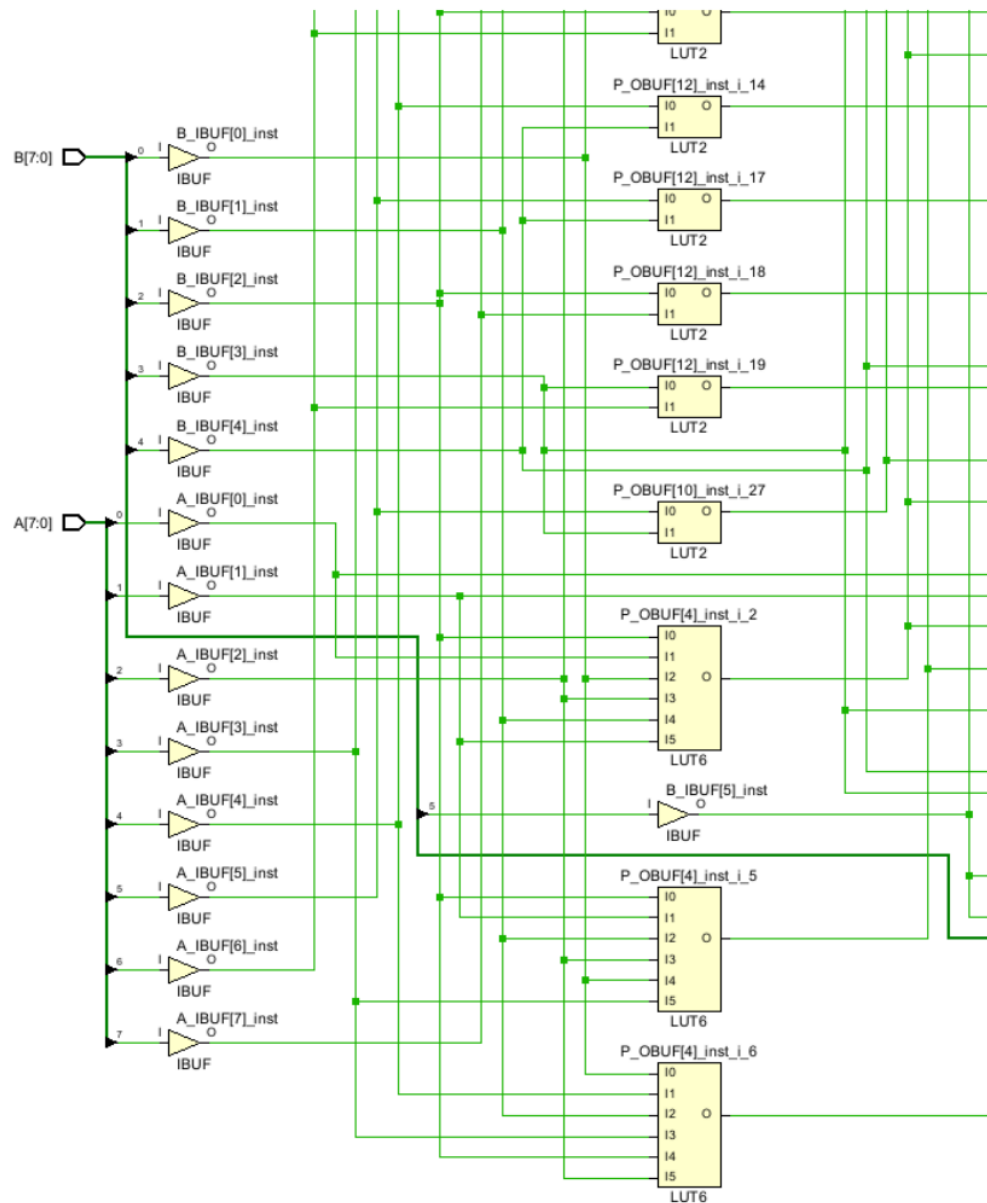


Figure 20 – Detailed view of the input section of the Braun's multiplier schematic diagram

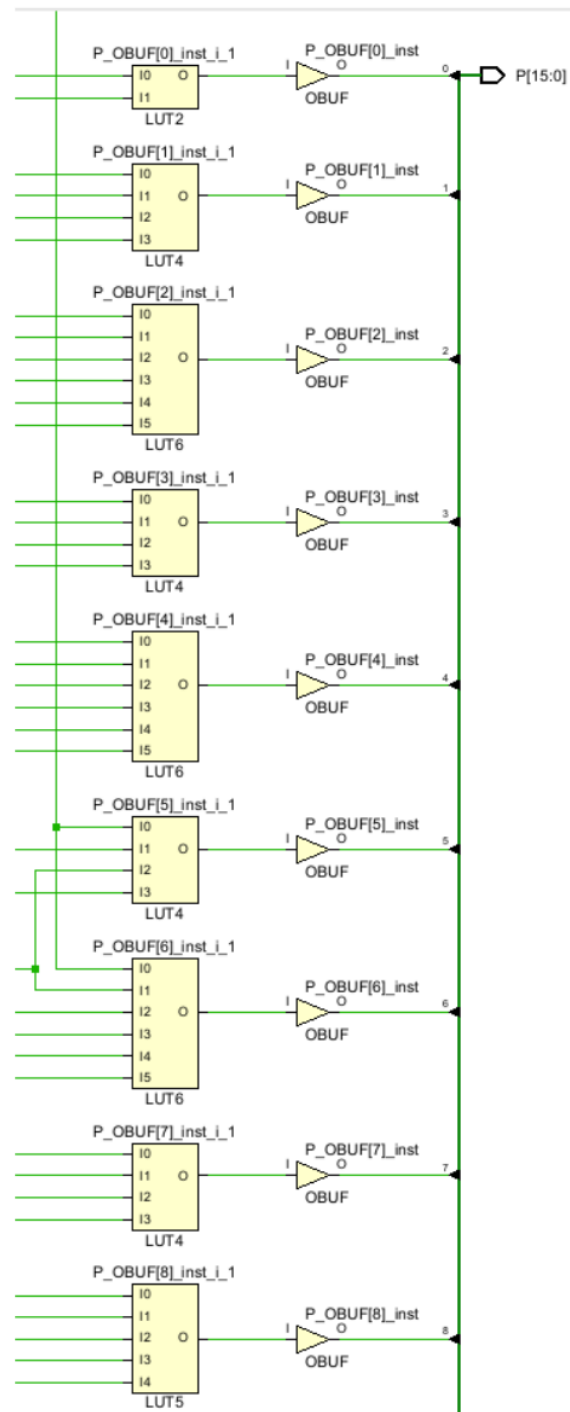


Figure 21 – Detailed view of the Braun's multiplier schematic diagram output

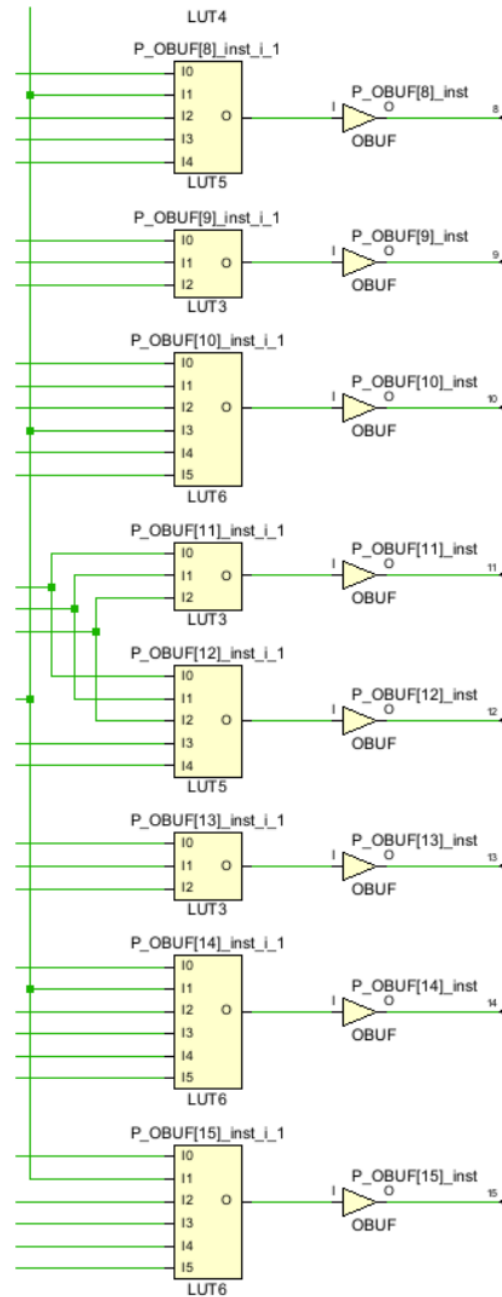


Figure 22 – Detailed view of the Braun's multiplier schematic diagram output

Once the schematic was generated, the input and output ports were specified according to the datasheet for the FPGA used [15]. The FPGA used was the XC7a100T csg324-L. Figure 23 shows the pinouts being selected in Xilinx Vivado. This allowed the FPGA diagram to be generated, this is shown in figure 24.

Figure 25 shows a detailed view of some of the active I/O pins within the FPGA. This shows that the input B5 is allocated to the pin R16 and the output P4 is connected to the input T16. This shows that the package pins in the FPGA match up to figure 23. Figure 26 shows one of the active LUTs within the FPGA. This same LUT is also shown in figure 21 connected to P2. This demonstrates how the FPGA diagram, and the schematic diagram are linked. Figure 27 shows a view of the LUTs within the FPGA diagram.

Name	Direction	Interface	Neg Diff Pair	Package Pin	
☒ A[7]	IN			J15	▼
☒ A[6]	IN			L16	▼
☒ A[5]	IN			M13	▼
☒ A[4]	IN			R15	▼
☒ A[3]	IN			R17	▼
☒ A[2]	IN			T18	▼
☒ A[1]	IN			U18	▼
☒ A[0]	IN			R13	▼
☒ B[7]	IN			T8	▼
☒ B[6]	IN			U8	▼
☒ B[5]	IN			R16	▼
☒ B[4]	IN			T13	▼
☒ B[3]	IN			H6	▼
☒ B[2]	IN			U12	▼
☒ B[1]	IN			U11	▼
☒ B[0]	IN			V10	▼
☒ P[15]	OUT			H17	▼
☒ P[14]	OUT			K15	▼
☒ P[13]	OUT			J13	▼
☒ P[12]	OUT			N14	▼
☒ P[11]	OUT			R18	▼
☒ P[10]	OUT			V17	▼
☒ P[9]	OUT			U17	▼
☒ P[8]	OUT			U16	▼
☒ P[7]	OUT			V16	▼
☒ P[6]	OUT			T15	▼
☒ P[5]	OUT			U14	▼
☒ P[4]	OUT			T16	▼
☒ P[3]	OUT			V15	▼
☒ P[2]	OUT			V14	▼
☒ P[1]	OUT			V12	▼
☒ P[0]	OUT			V11	▼

Figure 23 – I/O input and output pin specification

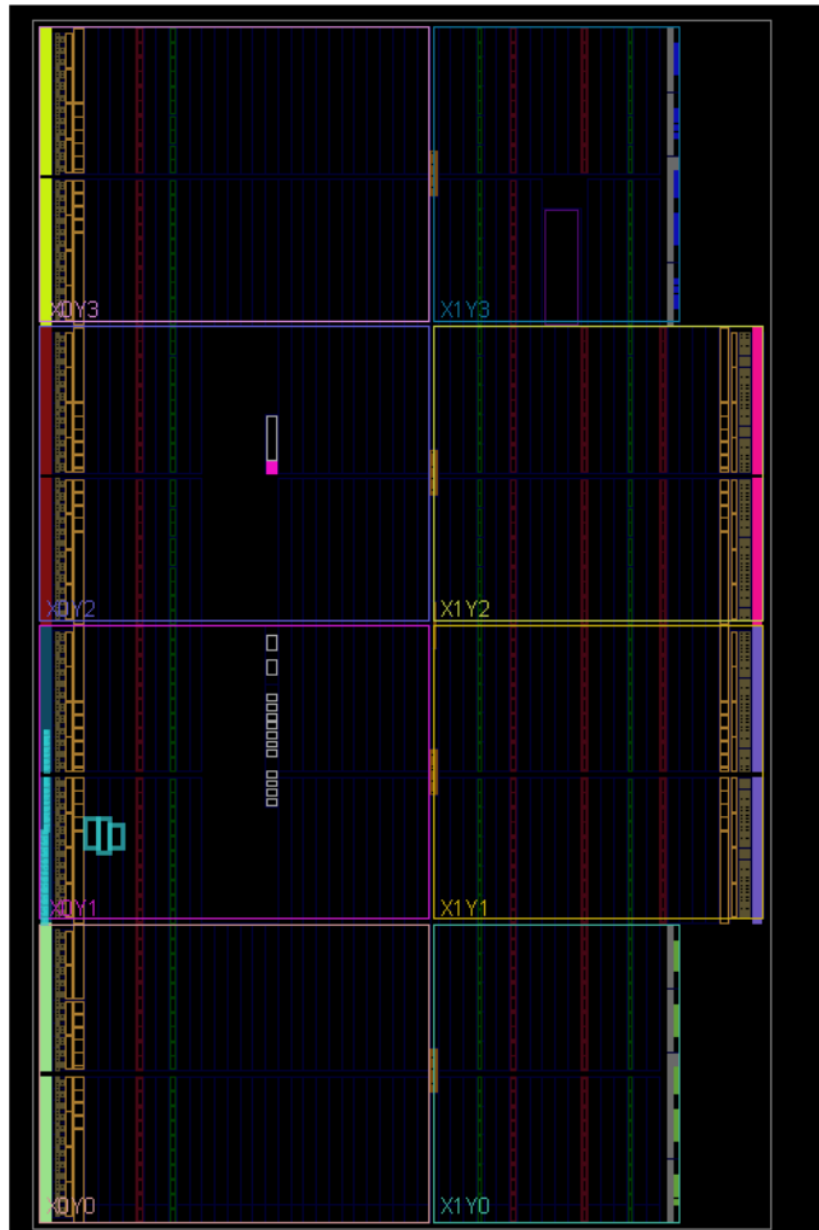


Figure 24 – Expanded view of the Braun's multiplier FPGA diagram

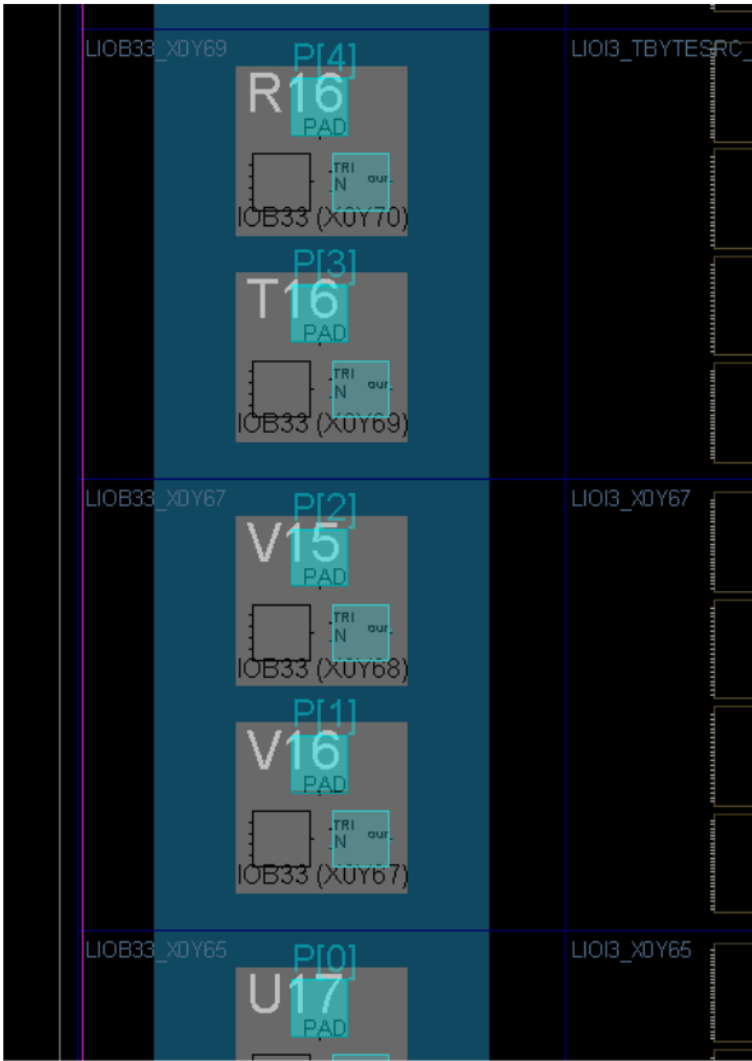


Figure 25 – FPGA I/O ports for Braun's multiplier design

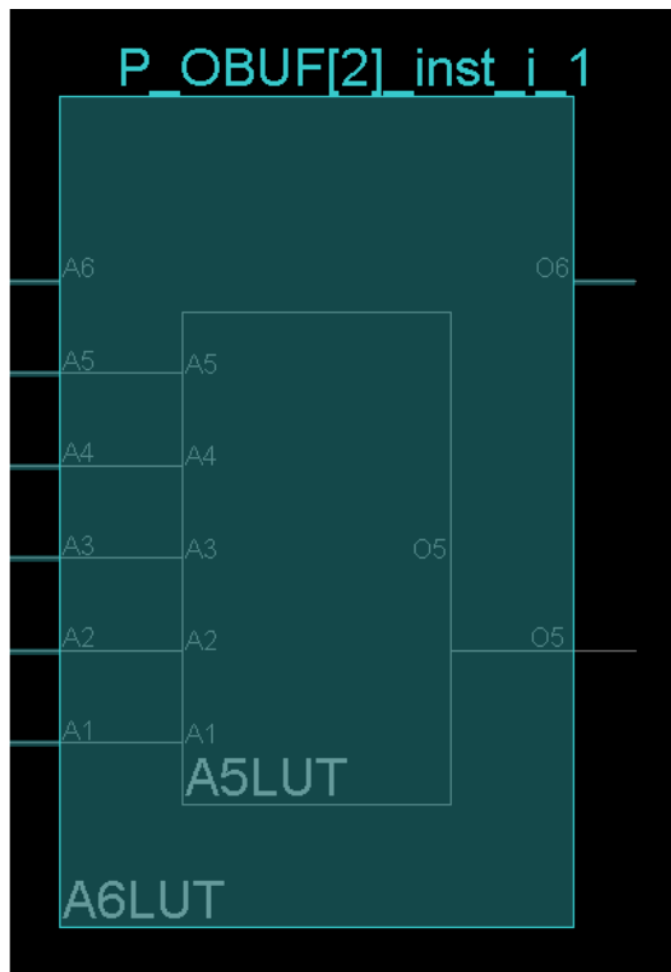


Figure 26 – Output buffer for P2 shown in the FPGA diagram



Figure 27 – Overview of LUTs within Braun’s multiplier



4.2 Wallace multiplier

The same design aspects from the Braun's multiplier can be applied to the Wallace tree multiplier. The only major difference is that the Wallace Tree multiplier refers to a dot diagram in place of the partial products. This is shown in figure 28. Figure 29 shows the flowchart and figure 30 shows an expanded view of the logic diagram. Figures 31-33 show a more detailed view of the logic diagram.

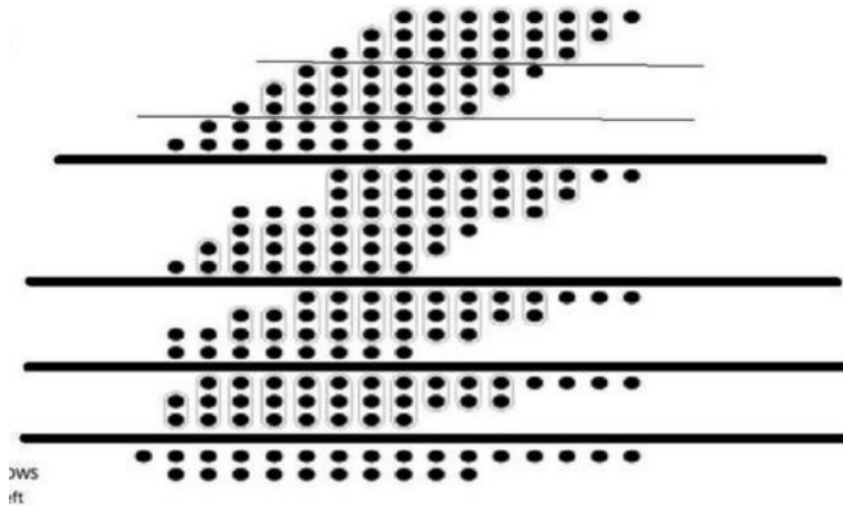


Figure 28 – Dot diagram for Wallace tree multiplier [16]

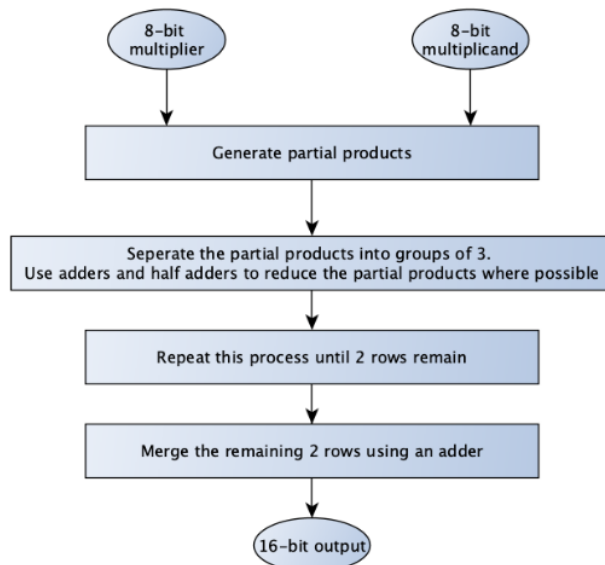


Figure 29 – Flowchart for Wallace tree multiplier

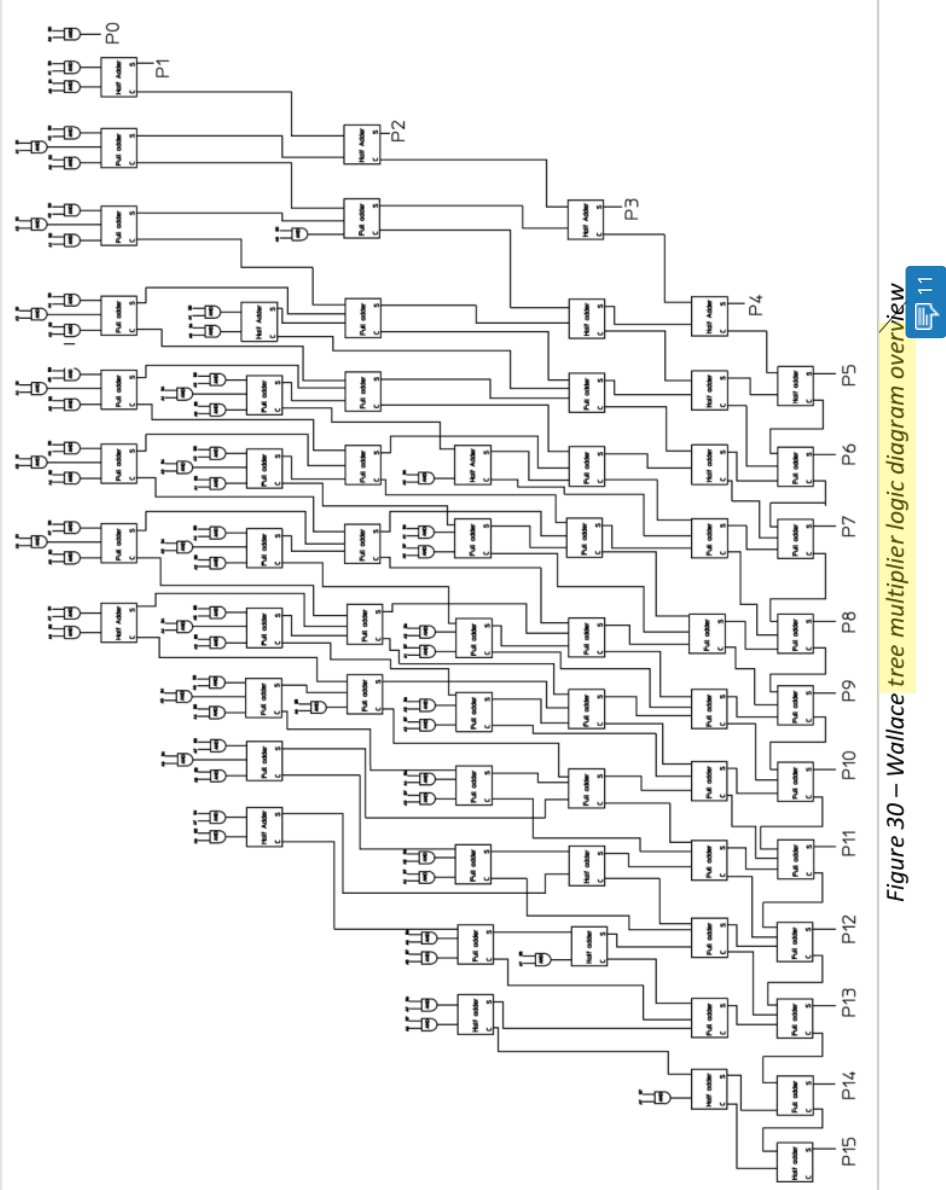


Figure 30 – Wallace tree multiplier logic diagram overview

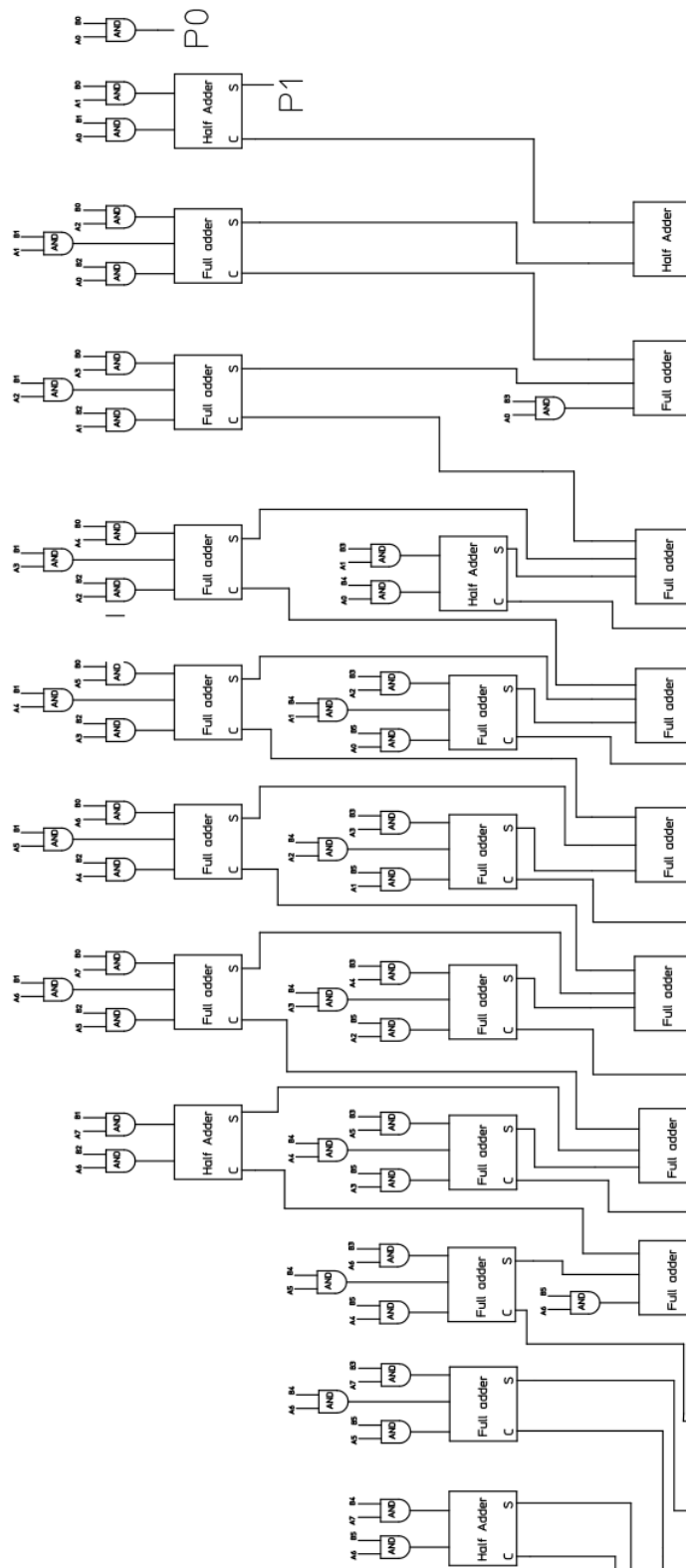


Figure 31 – Wallace tree multiplier logic diagram top section

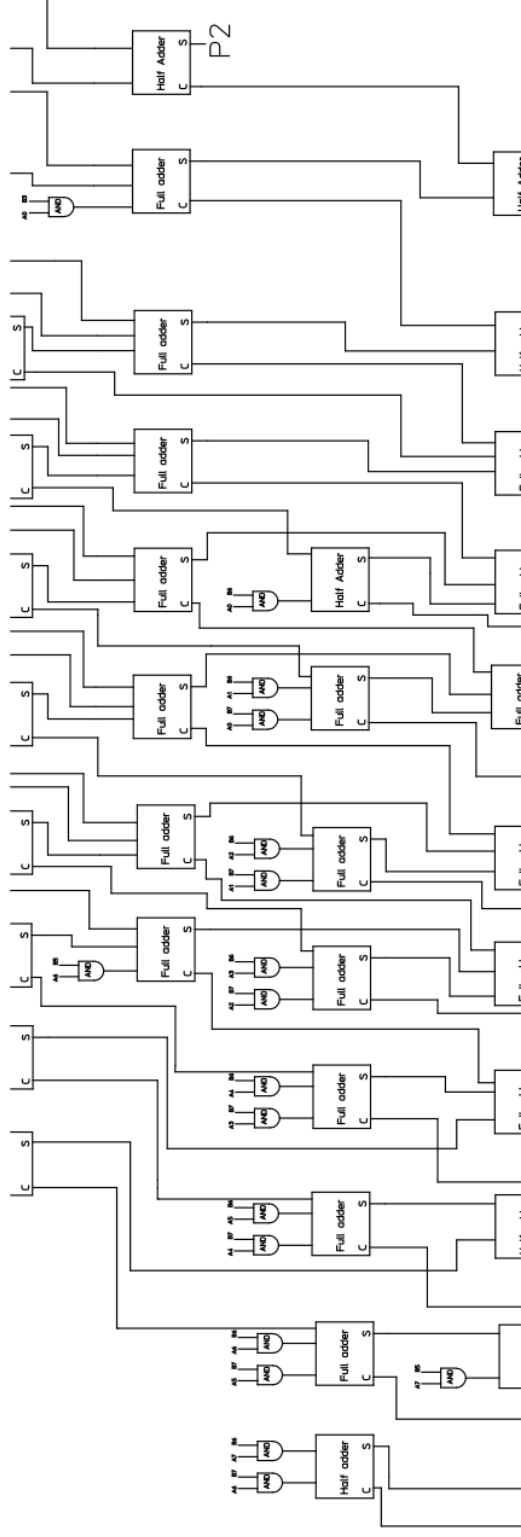


Figure 32 – Wallace tree multiplier logic diagram middle section

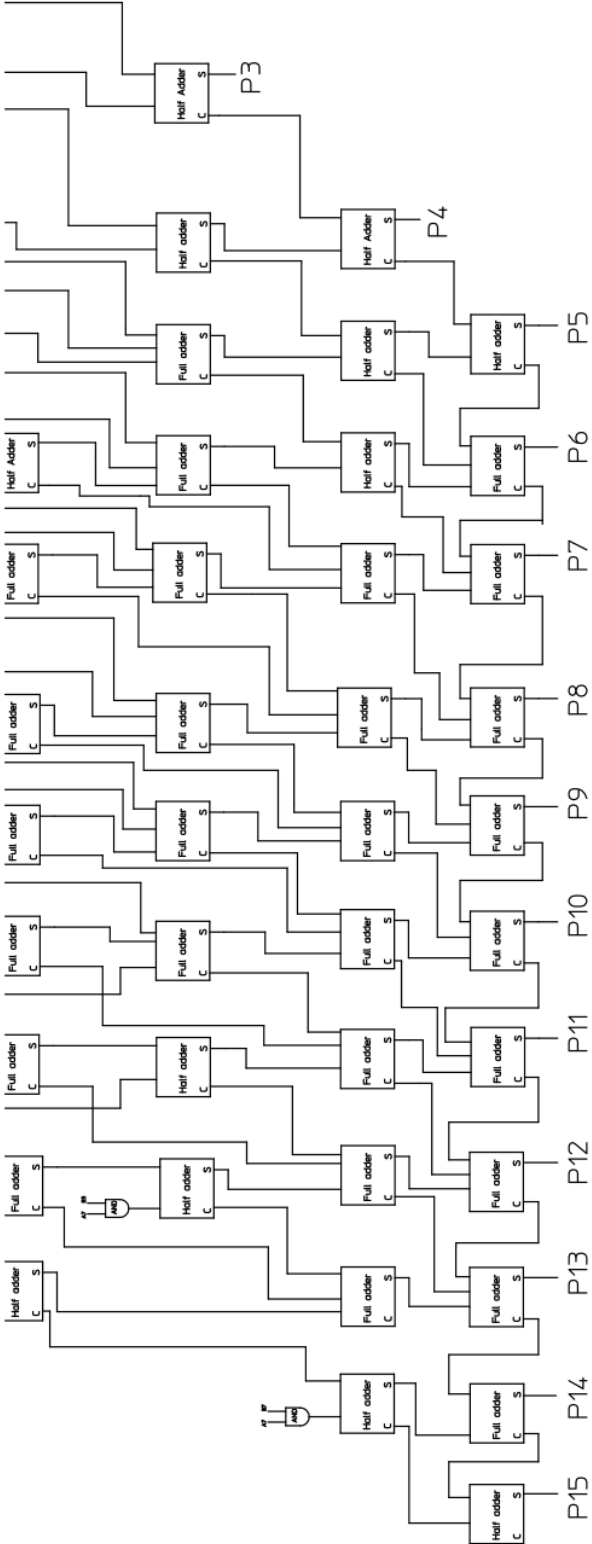


Figure 33 – Wallace tree multiplier logic diagram top section



Once these design stages were complete, the code for the Wallace tree multiplier was written. The same coding as the Braun’s multiplier was used. This can be found in Appendix C.

Appendix B shows the testbench. The testbench used for the Braun’s multiplier was also used for the Wallace tree multiplier. The actual outputs all matched the expected outputs, as table 1 showed for the Braun’s multiplier. Figure 34 shows the output of the Wallace tree multiplier in Xilinx Vivado.

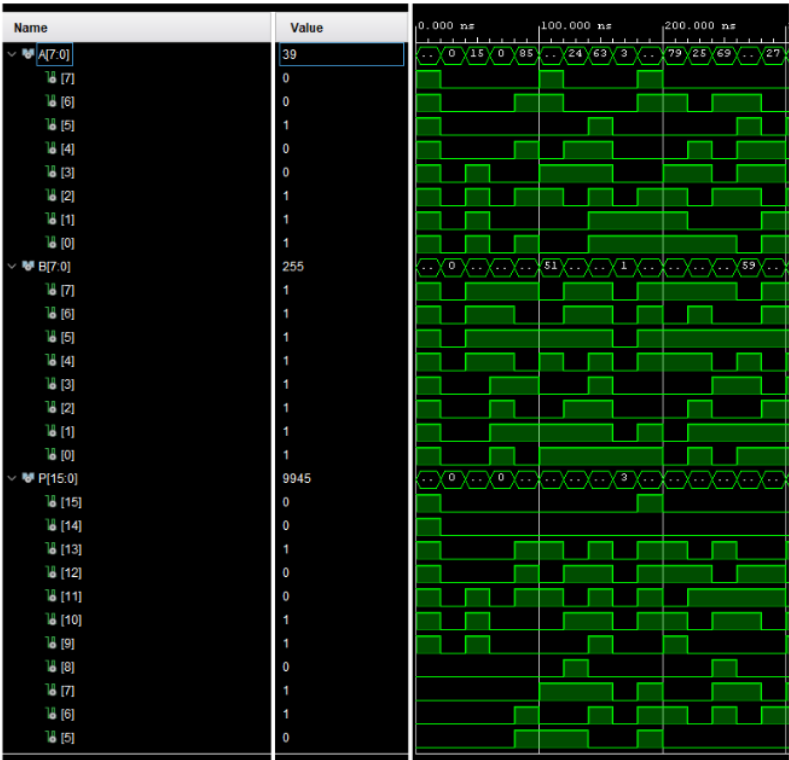


Figure 34 – Wallace tree multiplier results

Next, as with the Braun’s multiplier, a schematic was produced. Figure 35 shows this. Figures 36-38 show more detailed view.

Figures 39 & 40 show the FPGA diagram. This is similar to the Braun’s multiplier, however slight differences can be seen when comparing figure 40 to figure 27. This is because the Wallace tree multiplier utilizes more LUTs.

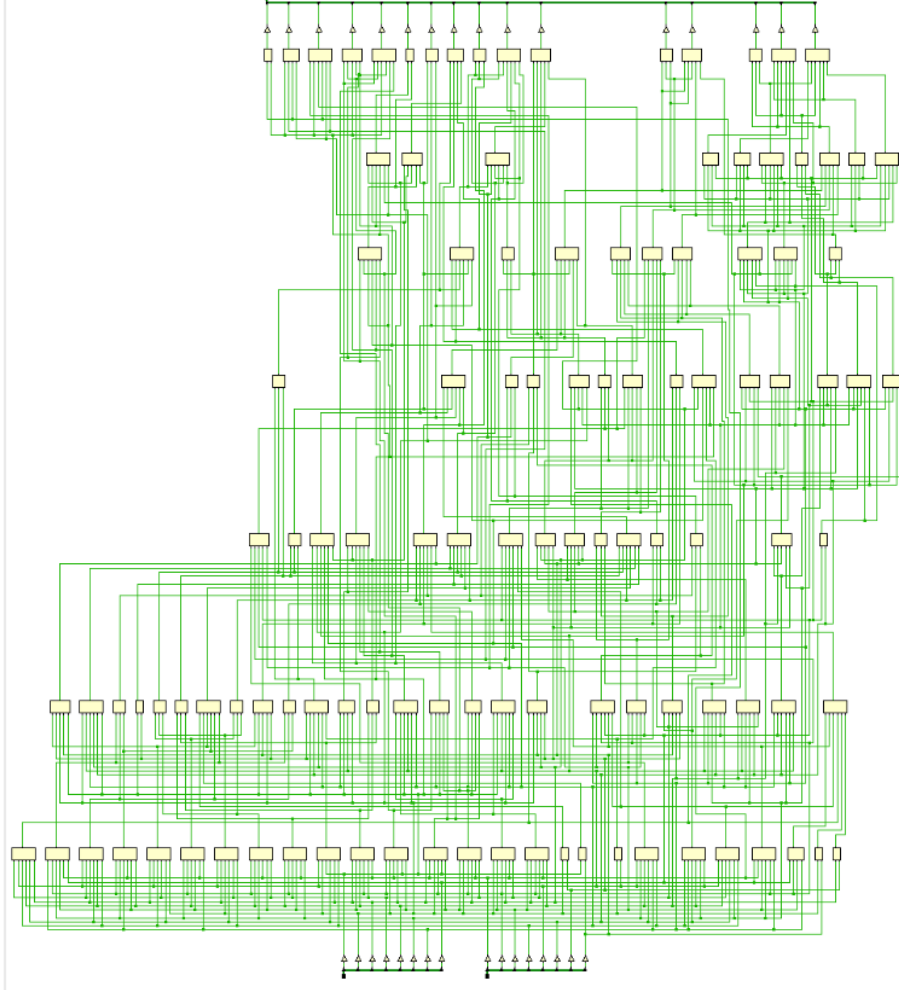


Figure 35 – expanded view of the Wallace-Tree multiplier schematic diagram

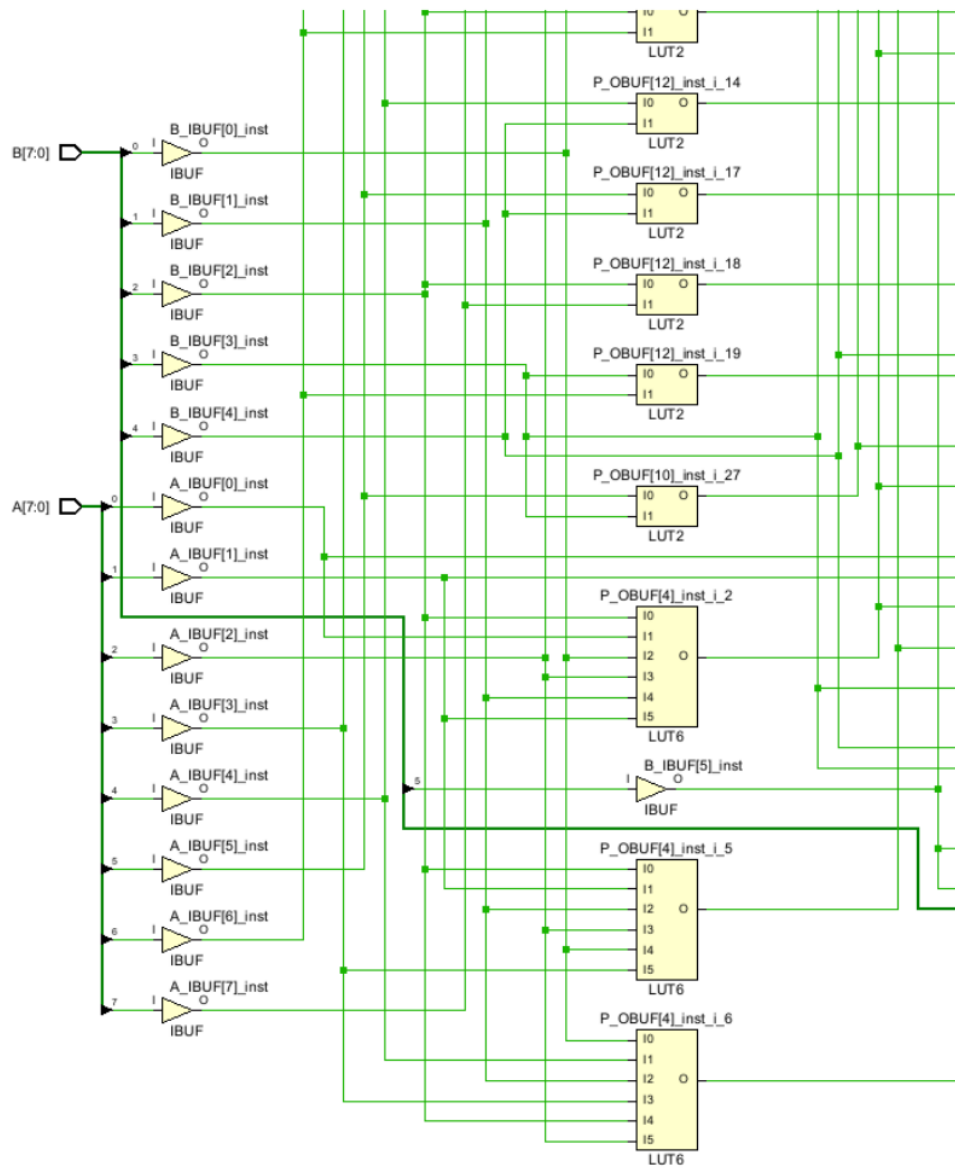


Figure 36 – Detailed view of the input of the Wallace tree multiplier schematic diagram

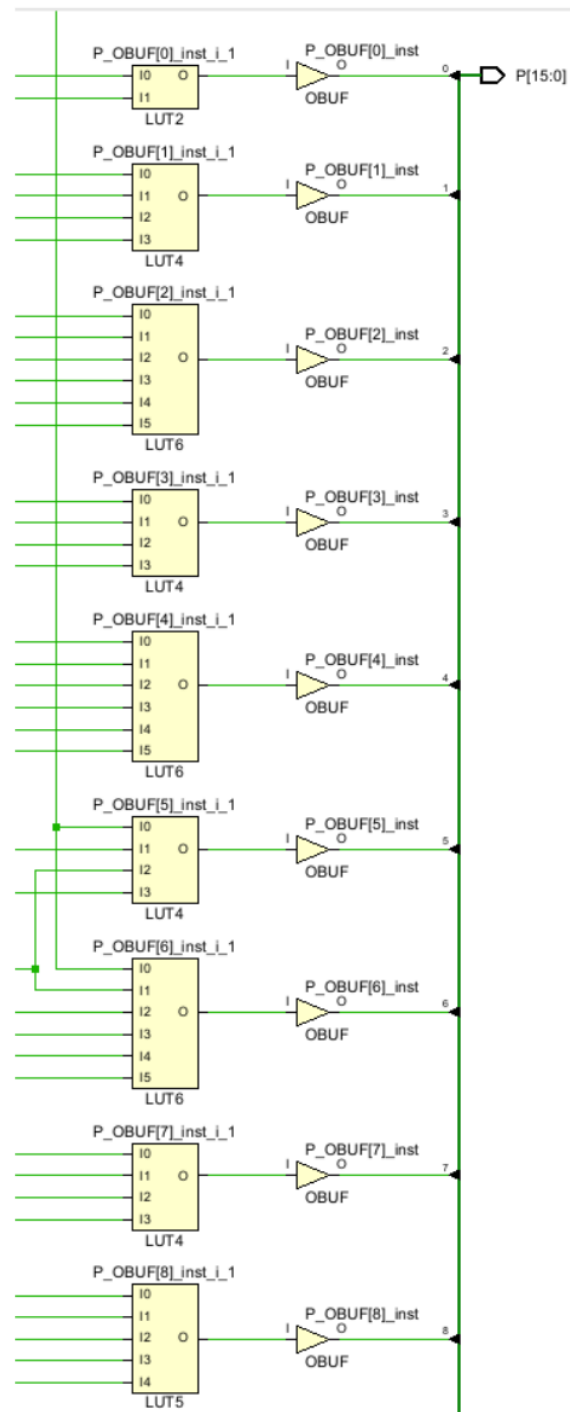


Figure 37 – Detailed view of the Wallace tree multiplier schematic diagram output

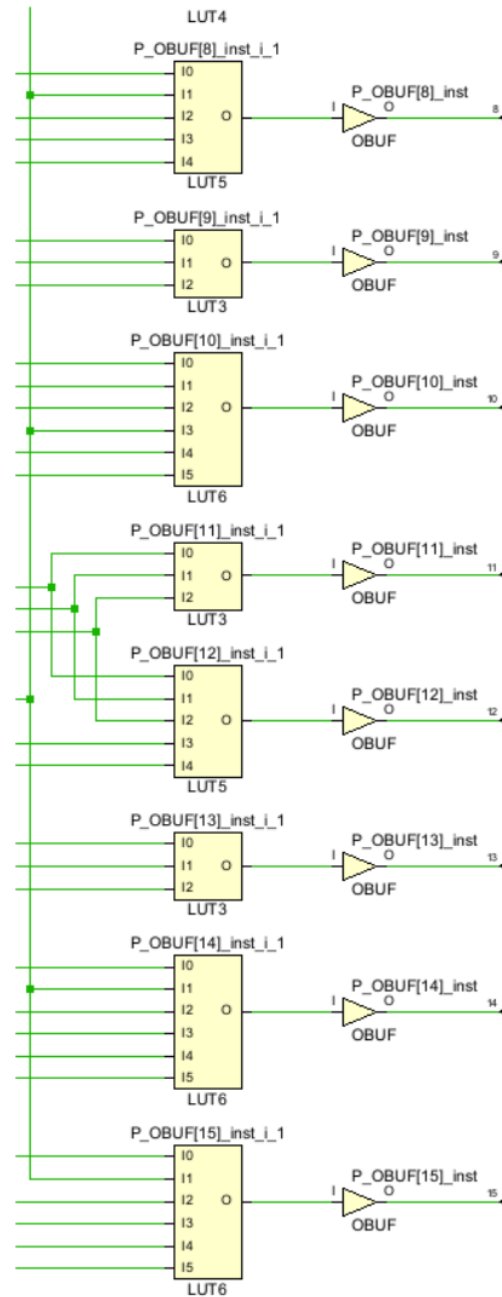


Figure 38 – Detailed view of the Wallace tree multiplier schematic diagram output

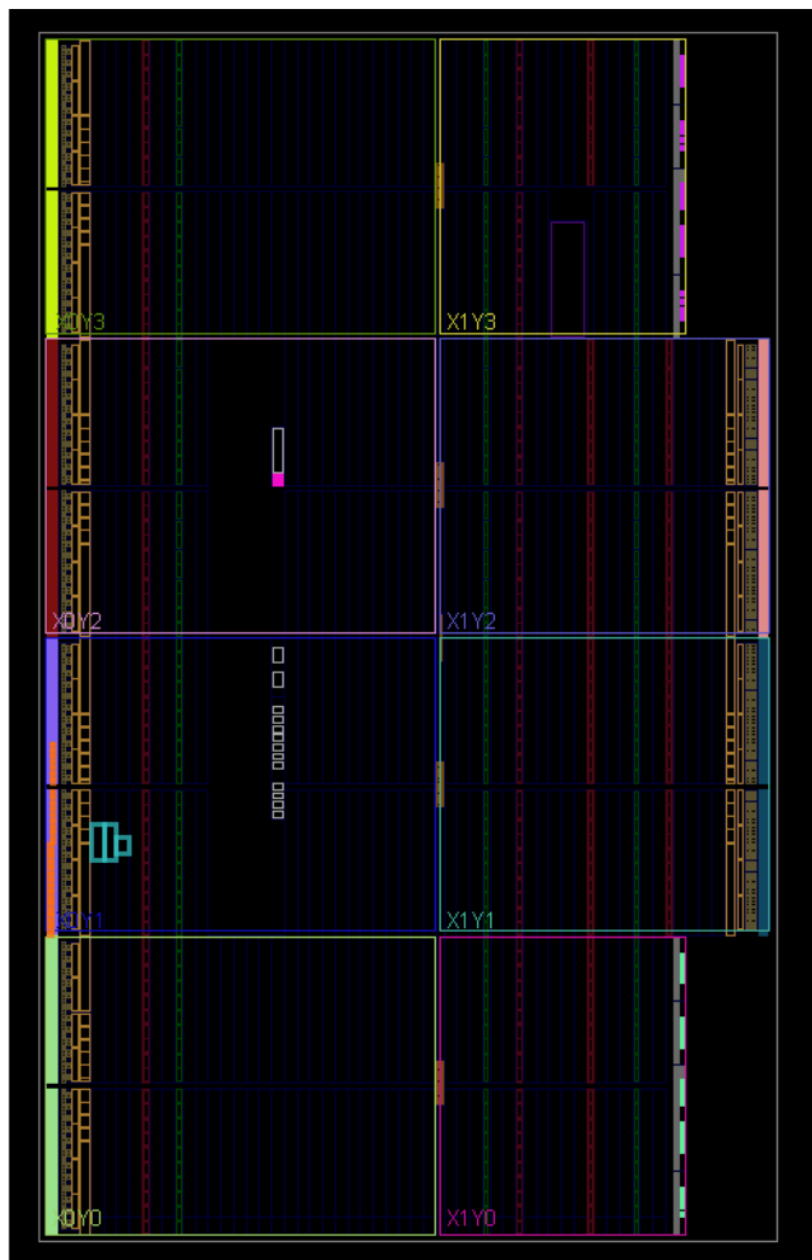


Figure 39 – Expanded view of the Wallace tree multiplier FPGA diagram

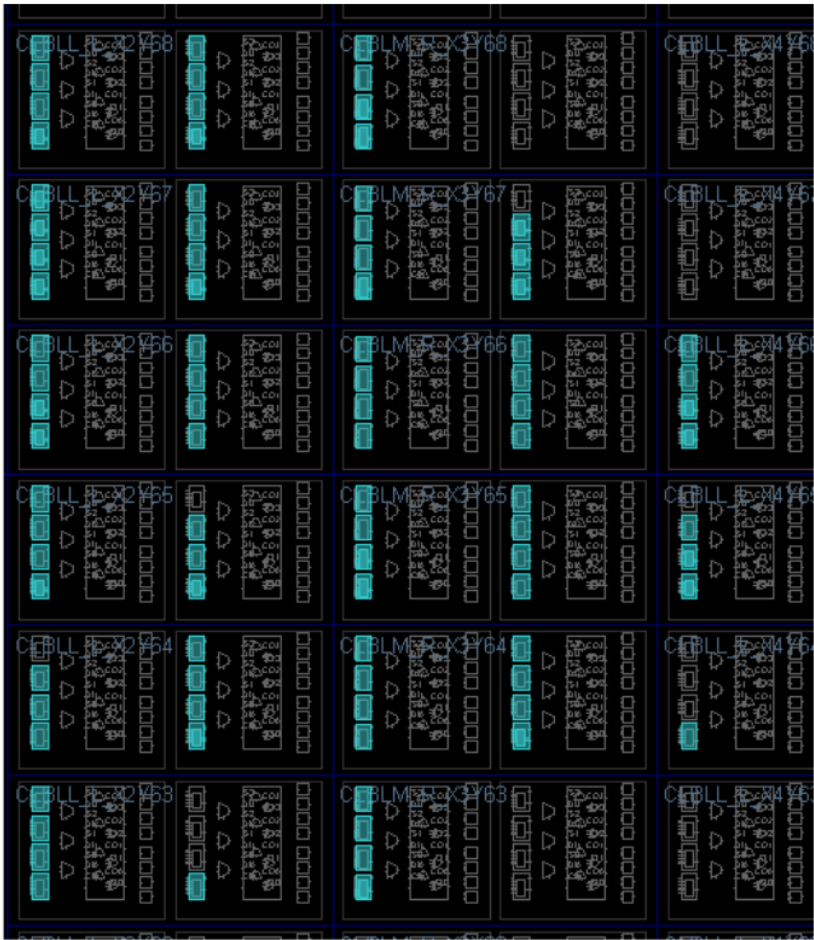


Figure 40 – Overview of LUT utilization in the Wallace tree multiplier

5 Comparison of the results

This section will compare the simulation reports from the Braun’s multiplier and the Wallace tree multiplier.

Power consumption & temperature

The research suggested that the Wallace tree multiplier would consume more power due to the increased LUT utilization. This has not been proven in the simulations. Figure 41 shows the power consumption, which is identical for both multipliers.

It is possible this power consumption is inaccurate, since the confidence level is stated to be low.

The breakdown of on-chip power consumption displays the power drawn by different sections of the design. Dynamic power is drawn by logic components, IO ports and signals. The static power is the power consumption when the device is not completing any tasks. It is also known as standby power [17].

Summary

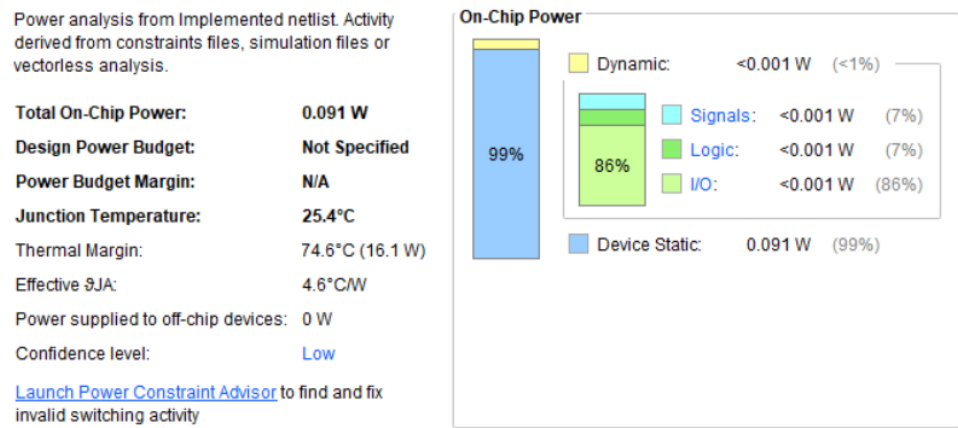


Figure 41 – Power consumption of both multipliers

14

Hardware utilization

Figure 42 shows the hardware utilization of both multipliers as a percentage. Figure 43 & 44 shows a more detailed representation of the hardware utilization. It shows that the Wallace-Tree design uses more LUTs, however due to the large number of LUTs available it does not affect the percentage in figure 42.

The Wallace tree multiplier utilizes 90 LUTs compared to the Braun’s multiplier which utilizes 83 LUTs. This means that the Wallace tree multiplier utilizes 8.4% more LUTs than

the Braun’s multiplier. This can be critical in larger circuits where hardware utilization is vital. This difference is the cause of the additional power consumption.

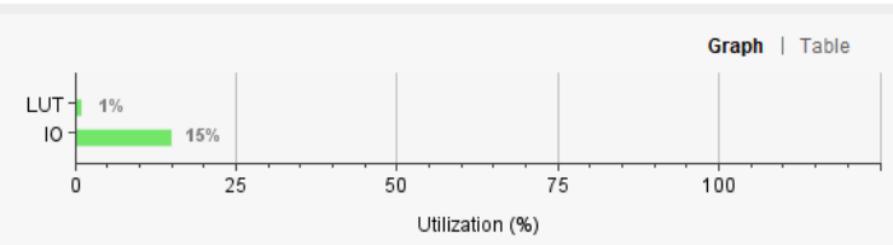


Figure 42 – Basic display of the I/O port and LUT usage for both multipliers

Utilization			
		Post-Synthesis	Post-Implementation
		Graph Table	
Resource	Utilization	Available	Utilization %
LUT	83	63400	0.13
IO	32	210	15.24

Figure 43 – hardware utilization of the Braun’s multiplier

Utilization			
		Post-Synthesis	Post-Implementation
		Graph Table	
Resource	Utilization	Available	Utilization %
LUT	90	63400	0.14
IO	32	210	15.24

Figure 44 – Hardware utilization of the Wallace design

Circuit delays

The timing of each design varies. Since the multiplication is carried out in parallel, by the time the longest delayed path is complete all other outputs will have been calculated. This means that the longest possible delay is equal to the longest delayed path, which is known as the critical path.

The specification states that an AND gate should be considered to have a 1ns delay. Both half adders and full adders should be considered to have a 2ns delay for both sum and carry outputs. By analysing the logic schematics critical path and these delays, the longest possible delay can be found. Figures 45 and 46 show the critical path for the Braun’s and Wallace tree multipliers respectively. The longest delay for the Braun’s multiplier is 35ns. This includes 14 adders and 7 AND gates. The longest delay for the Wallace tree multiplier is

This includes 15 adders and 2 AND gates. This backs up the research suggesting the Wallace tree multiplier has improved speeds. 11 of these adders are part of the RCA. This accounts for 22 seconds of the delay. This highlights how the delay can be drastically reduced by incorporating a CLA.

The minimum delay for both multipliers is 1ns.

The Xilinx Vivado software can also display the critical path on the circuit schematic which utilizes LUTs. This is shown in figures 47 & 48. Figures 49 & 50 show the minimum delayed paths in the circuit schematic. Figures 51 & 52 show the critical path highlighted on the FPGA diagram.

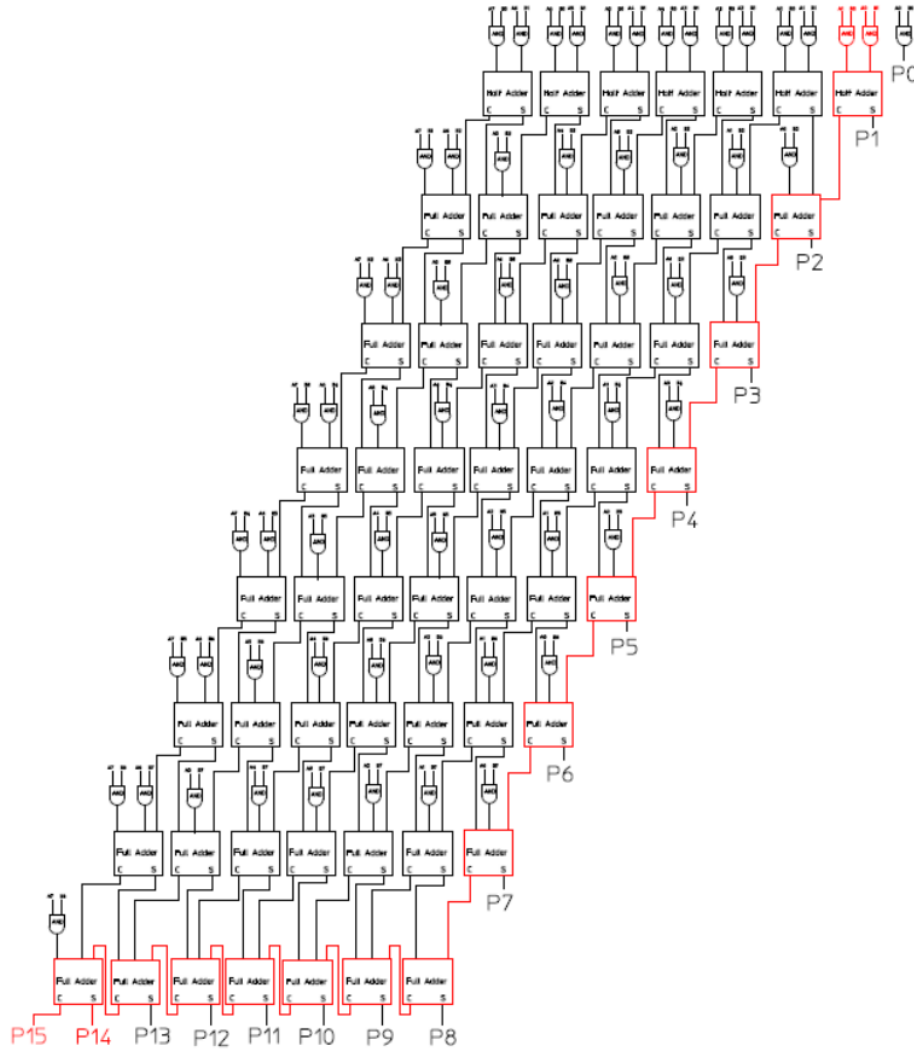


Figure 45 – Critical path of Braun's multiplier

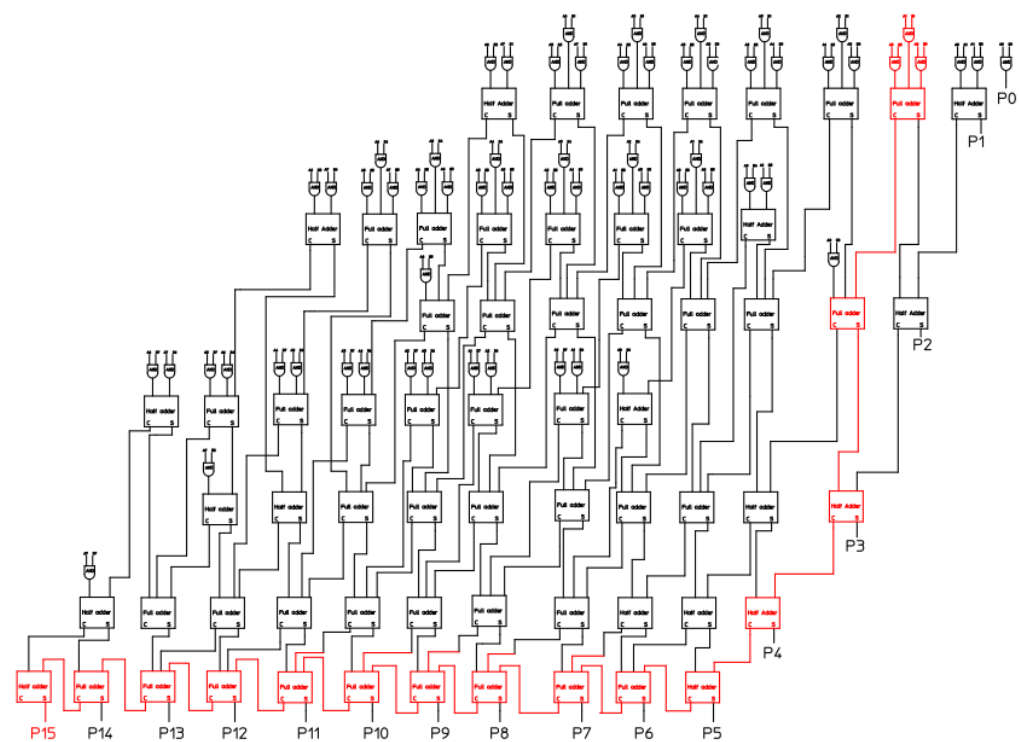


Figure 46 – Critical path of Wallace tree multiplier

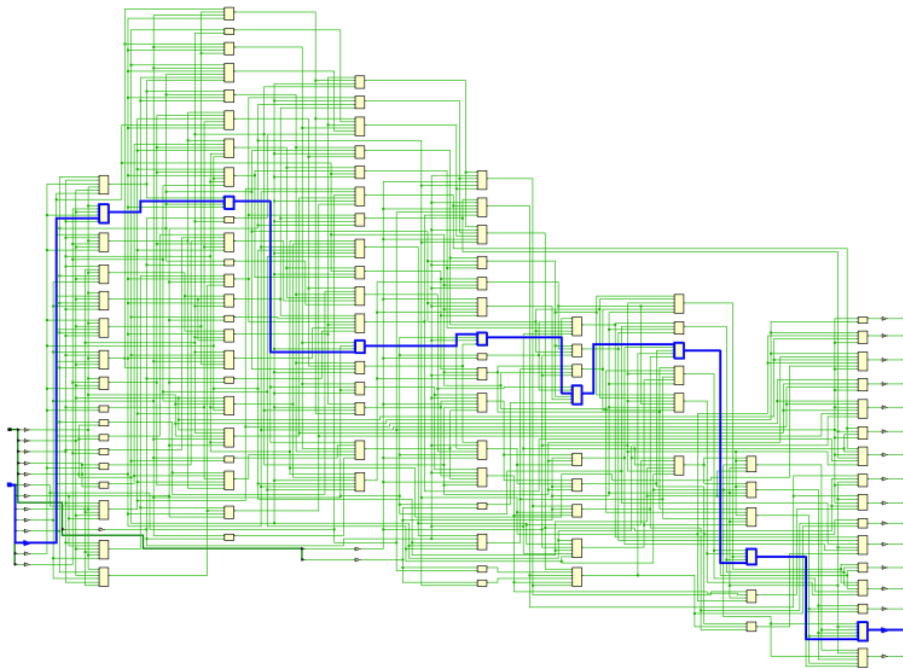


Figure 47 - Braun's multiplier critical path in the circuit schematic

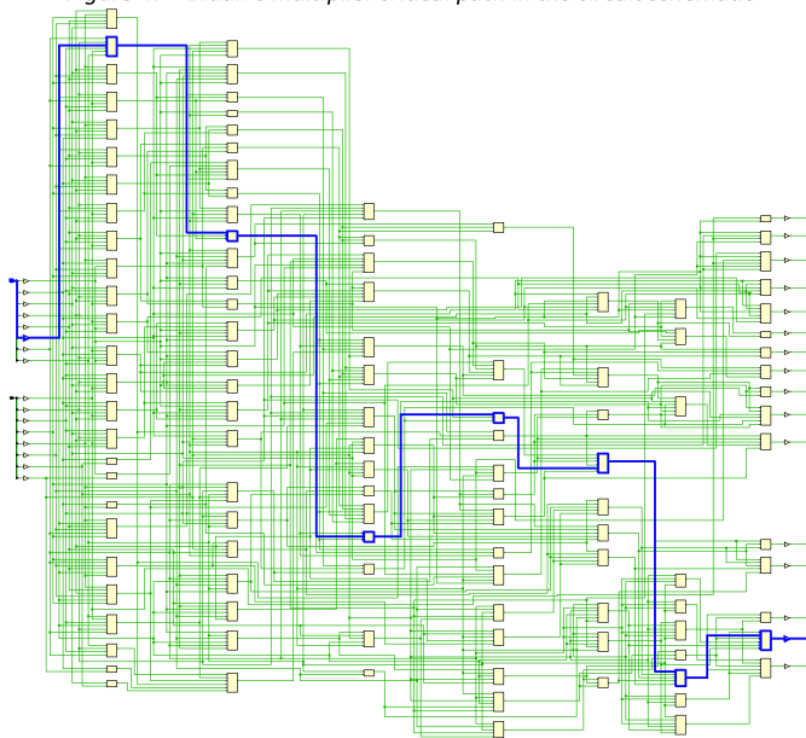


Figure 48 – Wallace tree multiplier critical path in the circuit schematic

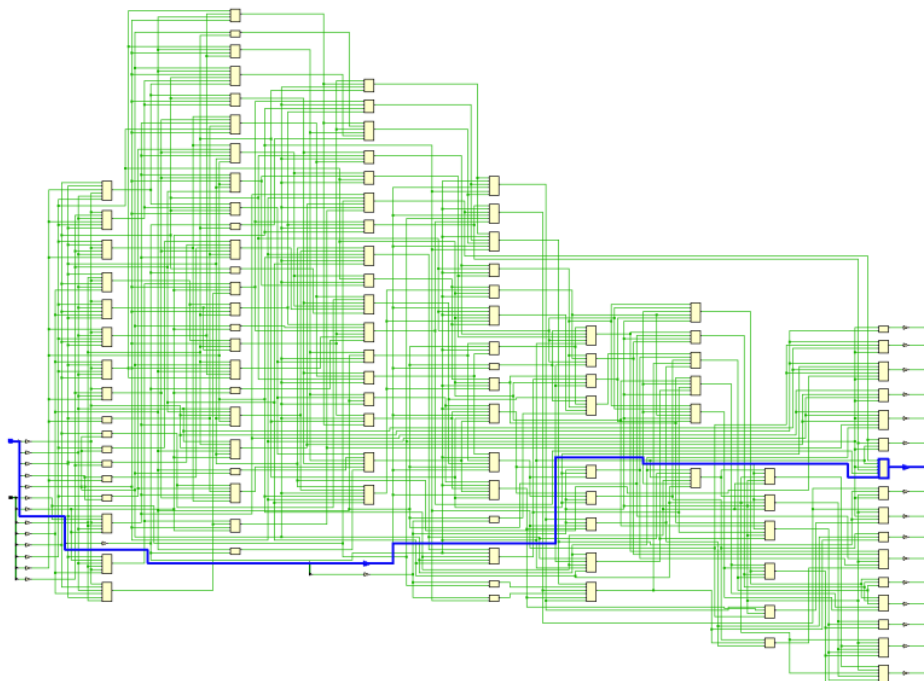


Figure 49 –Braun's multiplier minimum delayed path in the circuit schematic

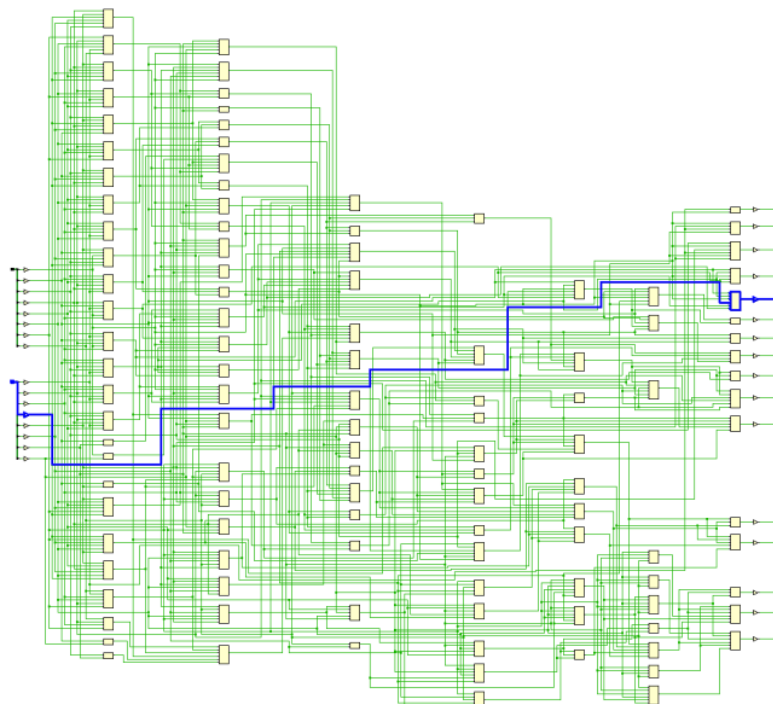


Figure 50 – Wallace tree multiplier minimum delayed path in the circuit schematic

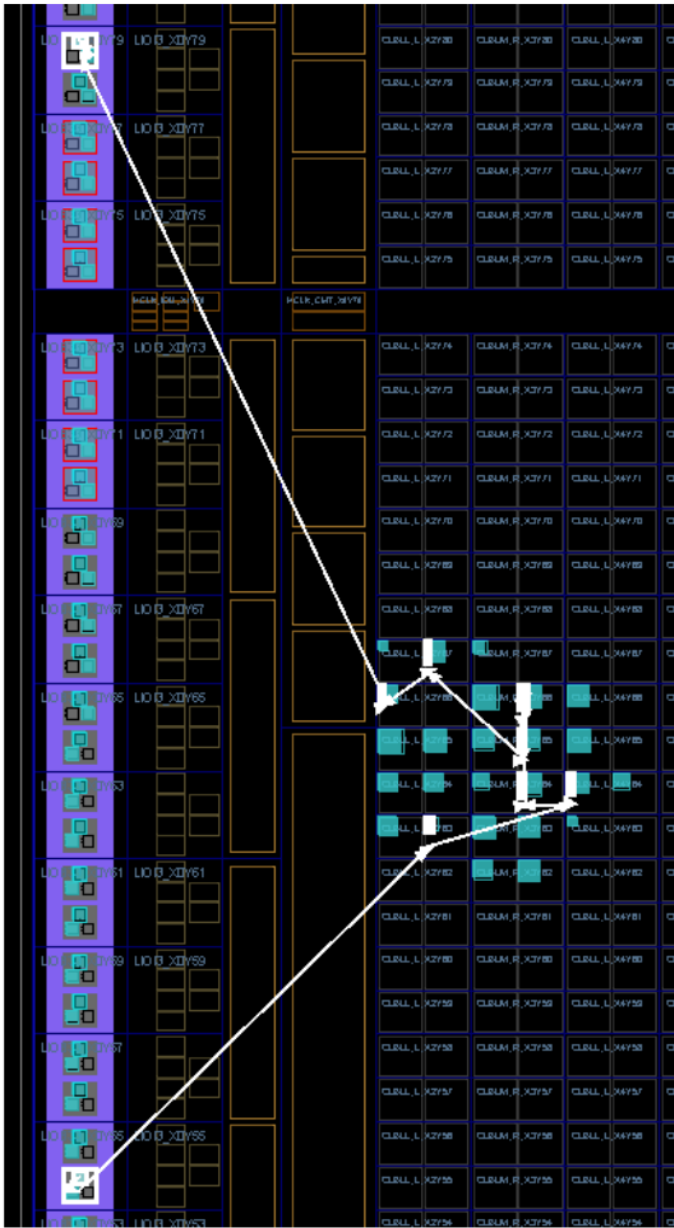


Figure 51 –Braun’s multiplier critical path in FPGA diagram



Figure 52 – Wallace tree multiplier critical path in FPGA diagram

The minimum and maximum delays can be used to implement the timing constraints and simulate the timings of each multiplier. Figures 53 & 54 show the paths with the longest delays from the Braun's multiplier and the Wallace-Tree multiplier simulations respectively. The longest delay in the Braun's multiplier is 12.916ns. The longest delay in the Wallace tree multiplier is 11.693ns. This shows the Braun's multiplier delay is 10.46% longer than the Wallace tree multiplier.

Figures 55 & 56 show the minimum delays of both multipliers. These are very similar although the minimum delays for the Braun's multiplier are slightly lower. This will have no impact on the on the circuit performance since the total output is not complete until the longest delayed path has completed.

Combinational Delays

From Port	To Port	Max Delay	Max Process Corner	Min Delay	Min Process Corner
A[5]	P[14]	12.916	SLOW	3.182	FAST
A[5]	P[13]	12.853	SLOW	3.100	FAST
B[2]	P[14]	12.835	SLOW	3.305	FAST
B[2]	P[13]	12.772	SLOW	3.260	FAST
A[5]	P[15]	12.761	SLOW	3.087	FAST
A[6]	P[14]	12.761	SLOW	2.842	FAST
B[1]	P[14]	12.733	SLOW	3.439	FAST
A[6]	P[13]	12.698	SLOW	2.932	FAST
B[2]	P[15]	12.680	SLOW	3.247	FAST
B[1]	P[13]	12.670	SLOW	3.394	FAST

Figure 53 – Longest delayed paths of the Braun's multiplier

Combinational Delays

From Port	To Port	Max Delay	Max Process Corner	Min Delay	Min Process Corner
A[5]	P[14]	11.693	SLOW	2.871	FAST
A[6]	P[14]	11.652	SLOW	2.717	FAST
A[3]	P[14]	11.645	SLOW	3.623	FAST
B[3]	P[14]	11.572	SLOW	3.566	FAST
A[1]	P[14]	11.492	SLOW	3.669	FAST
A[5]	P[13]	11.468	SLOW	3.329	FAST
B[1]	P[14]	11.437	SLOW	3.748	FAST
B[0]	P[14]	11.430	SLOW	3.666	FAST
A[6]	P[13]	11.427	SLOW	2.895	FAST
A[3]	P[13]	11.420	SLOW	3.536	FAST

Figure 54 – Longest delayed paths of the Wallace tree multiplier

 Combinational Delays

From Port	To Port	Max Delay	Max Process Corner	Min Delay 	Min Process Corner
 A[1]	 P[14]	12.633	SLOW	3.751	FAST
 A[1]	 P[13]	12.570	SLOW	3.707	FAST
 A[0]	 P[14]	12.646	SLOW	3.699	FAST
 A[1]	 P[15]	12.478	SLOW	3.693	FAST
 A[0]	 P[13]	12.583	SLOW	3.659	FAST
 A[0]	 P[15]	12.491	SLOW	3.647	FAST
 A[2]	 P[14]	12.122	SLOW	3.532	FAST
 A[1]	 P[12]	12.105	SLOW	3.505	FAST
 B[0]	 P[10]	10.958	SLOW	3.503	FAST
 A[3]	 P[14]	12.056	SLOW	3.500	FAST

Figure 55 – Braun's multiplier minimum delayed path

 Combinational Delays

From Port	To Port	Max Delay	Max Process Corner	M 	Min Process Corner
 A[2]	 P[14]	11.370	SLOW	3.799	FAST
 B[1]	 P[14]	11.437	SLOW	3.748	FAST
 A[2]	 P[13]	11.145	SLOW	3.712	FAST
 A[1]	 P[14]	11.492	SLOW	3.669	FAST
 B[0]	 P[14]	11.430	SLOW	3.666	FAST
 B[1]	 P[13]	11.211	SLOW	3.661	FAST
 A[0]	 P[14]	11.417	SLOW	3.649	FAST
 A[2]	 P[15]	10.943	SLOW	3.646	FAST
 A[3]	 P[14]	11.645	SLOW	3.623	FAST
 B[1]	 P[15]	11.010	SLOW	3.595	FAST

Figure 56 – Wallace tree multiplier minimum delayed path

Xilinx Vivado allows each path to be analysed in more detail. The paths with the longest delays are shown in figure 57 & 58. It shows that the Wallace tree multipliers longest path only has 9 logic levels. The Braun's multipliers longest path has 10. This is the reason for the improved speed in the Wallace tree multiplier.

Figures 59 & 60 show the exact components each path contains. This is known as the data path.

Summary	
Name	Path 1
Slack	22.084ns
Source	A[5] (input port)
Destination	P[14]
Path Group	**default**
Path Type	Max at Slow Process Corner
Requirement	35.000ns (MaxDelay Path 35.000ns)
Data Path Delay	12.916ns (logic 4.646ns (35.968%) route 8.271ns (64.032%))
Logic Levels	10 (IBUF=1 LUT4=3 LUT5=2 LUT6=3 OBUF=1)
Output Delay	0.000ns
Timing ...ception	MaxDelay Path 35.000ns

Figure 57 – Longest delayed path of the Braun's multiplier

Summary	
Name	Path 1
Slack	20.307ns
Source	A[5] (input port)
Destination	P[14]
Path Group	**default**
Path Type	Max at Slow Process Corner
Requirement	32.000ns (MaxDelay Path 32.000ns)
Data Path Delay	11.693ns (logic 4.554ns (38.945%) route 7.139ns (61.055%))
Logic Levels	9 (IBUF=1 LUT3=3 LUT5=1 LUT6=3 OBUF=1)
Output Delay	0.000ns
Timing ...ception	MaxDelay Path 32.000ns

Figure 58 – Longest delayed path of the Wallace-Tree multiplier

Data Path				
Delay Type	Incr (ns)	Path (...)	Location	Netlist Resource(s)
	(r) 0.000	0.000	Site: V14	A[5]
net (fo=0)	0.000	0.000		A[5]
			Site: V14	A_IBUF[5]_inst/I
IBUF (Prop...ibuf I O)	(r) 0.916	0.916	Site: V14	A_IBUF[5]_inst/O
net (fo=17, routed)	1.869	2.785		A_IBUF[5]
			Site: S..._X1Y63	P_OBUF[10]_inst_i_23/I4
LUT6 (Pro...t6 I4 O)	(r) 0.105	2.890	Site: S..._X1Y63	P_OBUF[10]_inst_i_23/O
net (fo=3, routed)	0.641	3.531		S_85
			Site: S..._X4Y64	P_OBUF[10]_inst_i_21/I1
LUT4 (Pro...t4 I1 O)	(r) 0.124	3.655	Site: S..._X4Y64	P_OBUF[10]_inst_i_21/O
net (fo=3, routed)	0.687	4.342		S_96
			Site: S..._X3Y64	P_OBUF[10]_inst_i_12/I3
LUT4 (Pro...t4 I3 O)	(r) 0.281	4.623	Site: S..._X3Y64	P_OBUF[10]_inst_i_12/O
net (fo=2, routed)	0.906	5.529		S_108
			Site: S..._X3Y66	P_OBUF[10]_inst_i_6/I0
LUT4 (Pro...t4 I0 O)	(r) 0.267	5.796	Site: S..._X3Y66	P_OBUF[10]_inst_i_6/O
net (fo=2, routed)	0.557	6.353		S_121
			Site: S..._X3Y66	P_OBUF[12]_inst_i_10/I2
LUT6 (Pro...t6 I2 O)	(r) 0.105	6.458	Site: S..._X3Y66	P_OBUF[12]_inst_i_10/O
net (fo=1, routed)	0.819	7.276		S_146
			Site: S..._X3Y65	P_OBUF[12]_inst_i_4/I0
LUT5 (Pro...t5 I0 O)	(r) 0.115	7.391	Site: S..._X3Y65	P_OBUF[12]_inst_i_4/O
net (fo=3, routed)	0.585	7.976		S_156
			Site: S..._X1Y67	P_OBUF[15]_inst_i_5/I4
LUT5 (Pro...t5 I4 O)	(r) 0.267	8.243	Site: S..._X1Y67	P_OBUF[15]_inst_i_5/O
net (fo=3, routed)	0.474	8.718		S_158
			Site: S..._X0Y66	P_OBUF[14]_inst_i_1/I5
LUT6 (Pro...t6 I5 O)	(r) 0.105	8.823	Site: S..._X0Y66	P_OBUF[14]_inst_i_1/O
net (fo=1, routed)	1.733	10.556		P_OBUF[14]
			Site: M16	P_OBUF[14]_inst/I
OBUF (Pr...buf I O)	(r) 2.361	12.916	Site: M16	P_OBUF[14]_inst/O
net (fo=0)	0.000	12.916		P[14]
			Site: M16	P[14]
Arrival Time		12.916		

Figure 59 – Braun's multipliers longest delayed path detailed view

Delay Type	Incr (ns)	Path (...)	Location	Netlist Resource(s)
	(r) 0.000	0.000	Site: V14	A[5]
net (fo=0)	0.000	0.000		A[5]
			Site: V14	A_IBUF[5]_inst/I
IBUF (Prop ibuf I O)	(r) 0.916	0.916	Site: V14	A_IBUF[5]_inst/O
net (fo=17, routed)	2.172	3.088		A_IBUF[5]
			Site: S...X2Y66	P_OBUF[10]_inst_i_11/I2
LUT6 (Pro...t6 I2 O)	(r) 0.105	3.193	Site: S...X2Y66	P_OBUF[10]_inst_i_11/O
net (fo=3, routed)	0.492	3.685		P_OBUF[10]_inst_i_11_n_0
			Site: S...X4Y66	P_OBUF[9]_inst_i_15/I0
LUT3 (Pro...t3 I0 O)	(f) 0.126	3.811	Site: S...X4Y66	P_OBUF[9]_inst_i_15/O
net (fo=3, routed)	0.686	4.497		P_OBUF[9]_inst_i_15_n_0
			Site: S...X4Y65	P_OBUF[9]_inst_i_4/I1
LUT3 (Pro...t3 I1 O)	(r) 0.281	4.778	Site: S...X4Y65	P_OBUF[9]_inst_i_4/O
net (fo=3, routed)	0.510	5.288		P_OBUF[9]_inst_i_4_n_0
			Site: S...X3Y66	P_OBUF[12]_inst_i_8/I0
LUT3 (Pro...t3 I0 O)	(r) 0.267	5.555	Site: S...X3Y66	P_OBUF[12]_inst_i_8/O
net (fo=1, routed)	0.478	6.033		P_OBUF[12]_inst_i_8_n_0
			Site: S...X2Y68	P_OBUF[12]_inst_i_2/I4
LUT6 (Pro...t6 I4 O)	(r) 0.105	6.138	Site: S...X2Y68	P_OBUF[12]_inst_i_2/O
net (fo=3, routed)	0.471	6.609		P_OBUF[12]_inst_i_2_n_0
			Site: S...X0Y68	P_OBUF[15]_inst_i_5/I0
LUT5 (Pro...t5 I0 O)	(r) 0.126	6.735	Site: S...X0Y68	P_OBUF[15]_inst_i_5/O
net (fo=3, routed)	0.717	7.452		P_OBUF[15]_inst_i_5_n_0
			Site: S...X1Y68	P_OBUF[14]_inst_i_1/I1
LUT6 (Pro...t6 I1 O)	(r) 0.267	7.719	Site: S...X1Y68	P_OBUF[14]_inst_i_1/O
net (fo=1, routed)	1.613	9.332		P_OBUF[14]
			Site: M16	P_OBUF[14]_inst/I
OBUF (Pr...buf I O)	(r) 2.361	11.693	Site: M16	P_OBUF[14]_inst/O
net (fo=0)	0.000	11.693		P[14]
			Site: M16	P[14]
Arrival Time		11.693		

Figure 60 – Wallace tree multipliers longest delayed path detailed view

The Worst Negative Slack (WNS) is also generated by Xilinx Vivado. This is the difference between the maximum delay set in the timing constraints and the longest delay found in the simulation. Both simulations have a positive value, meaning all data paths met the timing constraints. In the event a data paths delay exceeded the limit set in the timing constraints, this value would be negative. The slack of each individual path can be found by analysing the path. This can be utilized to highlight paths which are causing delays to the system and make the necessary adjustments to minimize them.

The timing summaries for the Braun’s and Wallace tree multipliers are shown in figures 61 and 62.

Design Timing Summary			
Setup	Hold	Pulse Width	
Worst Negative Slack (WNS): 22.084 ns	Worst Hold Slack (WHS): 1.203 ns	Worst Pulse Width Slack (WPWS):	NA
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS):	NA
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints:	NA
Total Number of Endpoints: 16	Total Number of Endpoints: 16	Total Number of Endpoints:	NA
All user specified timing constraints are met.			

Figure 61 – Braun’s multiplier timing summary

Design Timing Summary			
Setup	Hold	Pulse Width	
Worst Negative Slack (WNS): 20.307 ns	Worst Hold Slack (WHS): 1.410 ns	Worst Pulse Width Slack (WPWS):	NA
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS):	NA
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints:	NA
Total Number of Endpoints: 16	Total Number of Endpoints: 16	Total Number of Endpoints:	NA
All user specified timing constraints are met.			

Figure 62 – Wallace tree multiplier timing summary

6 Conclusion

Throughout this report different types of parallel multipliers for use in FPGAs have been investigated. Particular attention has been given to the Braun's and Wallace-Tree multipliers.

The background research conducted on these multipliers concluded that the Wallace-Tree multiplier is more complex, requires more hardware and consumes more power than the Braun's multiplier. Despite this, the research suggested that the Wallace-Tree had improved speed performance, making it a superior choice. Further research also found that implementing a carry lookahead adder in place of a ripple carry adder could further reduce delays.

To verify that the Wallace tree multiplier was faster, an 8x8 bit multiplier was designed using both design theories. Analysing the critical path found that the longest possible delay on the Braun's multiplier contained more logic gates, suggesting the delays would be higher. Both multipliers were then simulated in Xilinx Vivado. A comparison of the results found that the Wallace-Tree multiplier used 8.4% more LUTs and found the power consumption to be equal. It is possible this is incorrect, since it contradicts the background research. Xilinx Vivado states the confidence level is low, meaning the power consumption may be inaccurate. The Braun's multipliers maximum delay was 10.46% longer than the Wallace tree multiplier.

Besides the power consumption, these statistics confirm that the research conducted in section 3 was correct. Based on these differences in characteristics between the two multipliers, the Wallace-Tree multiplier is the best choice for most applications. The slightly increased LUT utilization is made up for by the significant speed increase. If there was an application where speed was not important, but hardware utilization was critical, the Braun's multiplier would be the better choice.

7 References

- [1] S. Brown, Z. Vranesic, Fundamentals of Digital Logic with VHDL Design, 3rd ed. New York: McGraw-Hill, 2008.
- [2] Arm Technology, "What is an FPGA," Arm Technology. [Online]. Available: <https://www.arm.com/glossary/fpga#:~:text=What%20is%20an%20FPGA%3F,requirements%20after%20the%20manufacturing%20process> [Accessed: Apr. 03, 2023].
- [3] Millennium Circuits Limited, "Field Programmable Gate Array (FPGA) VS. Microcontroller – What's the Difference?," Millennium Circuits Limited. [Online]. Available: <https://www.mclpcb.com/blog/fpga-vs-microcontroller/#:~:text=The%20main%20difference%20is%20in,of%20code%20at%20a%20time>. [Accessed: Apr. 03, 2023].
- [4] All about circuits, "Multiple-input gates," All about circuits. [Online]. Available: <https://www.allaboutcircuits.com/textbook/digital/chpt-3/multiple-input-gates/> [Accessed: Apr. 04, 2023].
- [5] Subham, "Half adder and Full adder circuit," Electronics Study Centre, 2014. [Online]. Available: <https://www.electronicengineering.nbcafe.in/half-adder-and-full-adder-circuit/> [Accessed: Apr. 04, 2023].
- [6] P. J. Ashenden, Digital Design – An Embedded Systems Approach using VHDL. Amsterdam: Elsevier, 2007.
- [7] P. Balasubramanian, C. Dang, D. L. Maskell and K. Prasad, "Approximate ripple carry and carry lookahead adders — A comparative analysis," 2017 IEEE 30th International Conference on Microelectronics (MIEL), Nis, Serbia, 2017, pp. 299-304, doi: 10.1109/MIEL.2017.8190125.
- [8] S. P. and A. A. Khan, "Comparison of Braun Multiplier and Wallace Multiplier Techniques in VLSI," 2018 4th International Conference on Devices, Circuits and Systems (ICDCS), Coimbatore, India, 2018, pp. 48-53, doi: 10.1109/ICDCSyst.2018.8605173.
- [9] M. Sadeghi, M. Zahedi and M. Ali, "The Cascade Carry Array Multiplier – A Novel Structure of Digital Unsigned Multipliers for Low-Power Consumption and Ultra-Fast applications," Research Gate, 2019. [Online]. Available: https://www.researchgate.net/figure/A-44-bit-basic-Braun-multiplier-12-16_fig3_333968081 [Accessed: Apr. 07, 2023].
- [10] P. Mishra and S. Nayak, "A study on Wallace tree multiplier," in International Journal of Advance Research in Science and Engineering, 2018. [Online]. Available:

http://www.ijarse.com/images/fullpdf/1523964041_IIMT19.pdf [Accessed: Apr. 07, 2023].

- [11] V. Manu, A. M. Vijaya Prakash and M. U. Chandra, "Design and Implementation of Sixteen-bit Low Power and Area Efficient Dadda Multiplier," 2019 4th International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT), Bangalore, India, 2019, pp. 631-636, doi: 10.1109/RTEICT46194.2019.9016834.
- [12] A.L. Siridhara, M. Vucha and T. Ravinder, "Performance Evaluation of Parallel Multipliers," in Journal of Engineering and Applied Sciences, 2017.
- [13] R. Fatemeh, "High Performance 8-bit Approximate Multiplier using Novel 4:2 approximate Compressors for Fast Image Processing," Research Gate, 2018. [Online]. Available: https://www.researchgate.net/figure/An-8-bit-Dadda-multiplier-constructed-by-only-some-half-and-full-adders-each-rectangle_fig2_325740406 [Accessed: Apr. 15, 2023].
- [14] M. Sjalander, "Multiplication Acceleration Through Twin Precision," Research Gate, 2009. [Online]. Available: https://www.researchgate.net/figure/Illustration-of-an-unsigned-8-bit-multiplication_fig2_224440119 [Accessed: Apr. 16, 2023].
- [15] Digilent, "Nexys A7™ FPGA Board Reference Manual," Nexys A7 datasheet. [Online]. Available: https://digilent.com/reference/_media/reference/programmable-logic/nexys-a7/nexys-a7_rm.pdf [Accessed: Apr. 16, 2023].
- [16] Ansari, S.J., Verma, P., Choudhary, S.D. (2022). High-Performance Wallace Tree Multiplier Design Using Novel 8-4 Compressor Implementation for Image Processing. In: Rawat, S., Kumar, A., Kumar, P., Anguera, J. (eds) Proceedings of First International Conference on Computational Electronics for Wireless Communications. Lecture Notes in Networks and Systems, vol 329. Springer, Singapore. https://doi.org/10.1007/978-981-16-6246-1_41
- [17] AMD, "UltraFast Design Methodology Guide for FPGAs and SoCs (UG949)," AMD. [Online]. Available: <https://docs.xilinx.com/r/en-US/ug949-vivado-design-methodology/Startup-Power> [Accessed: Apr. 20, 2023].

Appendix A

Braun's multiplier code

```

2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4
5  -- Uncomment the following library declaration if using
6  -- arithmetic functions with Signed or Unsigned values
7  --use IEEE.NUMERIC_STD.ALL;
8
9  -- Uncomment the following library declaration if instantiating
10 -- any Xilinx leaf cells in this code.
11 --library UNISIM;
12 --use UNISIM.VComponents.all;
13
14 entity AND2 is -- Define AND I/O
15     port(
16         A,B: in BIT;
17         X: buffer BIT); -- set as buffer so it can be input or output
18 end AND2;
19
20 architecture behavioral of AND2 is -- Define AND behavior
21 begin
22     X <= A and B;
23 end behavioral;
24
25 entity HA is -- Define Half adder I/O
26     port(
27         a,b: in BIT;
28         sum, carry: buffer BIT);
29 end HA;
30
31 architecture behavioral of HA is -- Define Half adder Behavior
32 begin
33     sum <= a xor b; -- Calculate sum
34     carry <= a and b; -- Calculate Carry
35
36 end behavioral;
37
38 entity FA is -- Define full adder I/O
39     port(
40         a,b,c: in BIT;
41         sum, carry: buffer BIT);
42 end FA;
43
44 architecture behavioral of FA is -- Define full adder Behavior
45 begin
46     sum <= a xor b XOR c; -- Calculate Sum
47     carry <= ((a and b) or (b and c) or (a and c)); --Calculate Carry
48
49 end behavioral;
50
51 entity Multi is -- set inputs and outputs
52     port(A,B: in bit_vector(7 downto 0));

```

```

50
51 entity Multi is -- set inputs and outputs
52     port(A,B: in bit_vector(7 downto 0);
53         P: buffer bit_vector(15 downto 0));
54 end Multi;
55
56 architecture structural of Multi is
57     component AND2 -- Defines AND gate I/Os for port mapping
58     Port(
59         A,B: in BIT;
60         X: buffer BIT);
61 end component;
62
63     component HA -- Defines Half adder gate I/Os for port mapping
64     port(
65         a,b: in BIT;
66         sum, carry: buffer BIT);
67 end component;
68
69     component FA -- Defines full adder gate I/Os for port mapping
70     port(
71         a,b,c: in BIT;
72         sum, carry: buffer BIT);
73 end component;
74
75     signal S: BIT_vector(160 downto 1); -- Sets signals
76
77 begin
78
79     A1: AND2 port map (A(0),B(0),P(0)); -- AND gates for calculating partial products
80     A2: AND2 port map (A(0),B(1),S(1));
81     A3: AND2 port map (A(1),B(0),S(2));
82     A4: AND2 port map (A(1),B(1),S(3));
83     A5: AND2 port map (A(2),B(0),S(4));
84     A6: AND2 port map (A(2),B(1),S(5));
85     A7: AND2 port map (A(3),B(0),S(6));
86     A8: AND2 port map (A(3),B(1),S(7));
87     A9: AND2 port map (A(4),B(0),S(8));
88     A10: AND2 port map (A(4),B(1),S(9));
89     A11: AND2 port map (A(5),B(0),S(10));
90     A12: AND2 port map (A(5),B(1),S(11));
91     A13: AND2 port map (A(6),B(0),S(12));
92     A14: AND2 port map (A(6),B(1),S(13));
93     A15: AND2 port map (A(7),B(0),S(14));
94
95     A16: AND2 port map (A(0),B(2),S(15)); -- and gates 2nd row
96     A17: AND2 port map (A(1),B(2),S(16));
97     A18: AND2 port map (A(2),B(2),S(17));
98     A19: AND2 port map (A(3),B(2),S(18));
99     A20: AND2 port map (A(4),B(2),S(19));
100    A21: AND2 port map (A(5),B(2),S(20));
101    A22: AND2 port map (A(6),B(2),S(21));

```

```

100  A21: AND2 port map (A(5),B(2),S(20));
101  A22: AND2 port map (A(6),B(2),S(21));
102  A23: AND2 port map (A(7),B(1),S(22));
103
104  A24: AND2 port map (A(0),B(3),S(23)); -- and gates 3rd row
105  A25: AND2 port map (A(1),B(3),S(24));
106  A26: AND2 port map (A(2),B(3),S(25));
107  A27: AND2 port map (A(3),B(3),S(26));
108  A28: AND2 port map (A(4),B(3),S(27));
109  A29: AND2 port map (A(5),B(3),S(28));
110  A30: AND2 port map (A(6),B(3),S(29));
111  A31: AND2 port map (A(7),B(2),S(30));
112
113  A32: AND2 port map (A(0),B(4),S(31)); -- and gates 4th row
114  A33: AND2 port map (A(1),B(4),S(32));
115  A34: AND2 port map (A(2),B(4),S(33));
116  A35: AND2 port map (A(3),B(4),S(34));
117  A36: AND2 port map (A(4),B(4),S(35));
118  A37: AND2 port map (A(5),B(4),S(36));
119  A38: AND2 port map (A(6),B(4),S(37));
120  A39: AND2 port map (A(7),B(3),S(38));
121
122  A40: AND2 port map (A(0),B(5),S(39)); -- and gates 5th row
123  A41: AND2 port map (A(1),B(5),S(40));
124  A42: AND2 port map (A(2),B(5),S(41));
125  A43: AND2 port map (A(3),B(5),S(42));
126  A44: AND2 port map (A(4),B(5),S(43));
127  A45: AND2 port map (A(5),B(5),S(44));
128  A46: AND2 port map (A(6),B(5),S(45));
129  A47: AND2 port map (A(7),B(4),S(46));
130
131  A48: AND2 port map (A(0),B(6),S(47)); -- and gates 6th row
132  A49: AND2 port map (A(1),B(6),S(48));
133  A50: AND2 port map (A(2),B(6),S(49));
134  A51: AND2 port map (A(3),B(6),S(50));
135  A52: AND2 port map (A(4),B(6),S(51));
136  A53: AND2 port map (A(5),B(6),S(52));
137  A54: AND2 port map (A(6),B(6),S(53));
138  A55: AND2 port map (A(7),B(5),S(54));
139
140  A56: AND2 port map (A(0),B(7),S(55)); -- and gates bottom row
141  A57: AND2 port map (A(1),B(7),S(56));
142  A58: AND2 port map (A(2),B(7),S(57));
143  A59: AND2 port map (A(3),B(7),S(58));
144  A60: AND2 port map (A(4),B(7),S(59));
145  A61: AND2 port map (A(5),B(7),S(60));
146  A62: AND2 port map (A(6),B(7),S(61));
147  A63: AND2 port map (A(7),B(6),S(62));
148  A64: AND2 port map (A(7),B(7),S(63));
149
150
151  H1: HA port map (S(1),S(2),P(1),S(64)); -- Half adders top row
152  H2: HA port map (S(3),S(4),S(65),S(66));

```

```

150
151 H1: HA port map (S(1),S(2),P(1),S(64)); -- Half adders top row
152 H2: HA port map (S(3),S(4),S(65),S(66));
153 H3: HA port map (S(5),S(6),S(67),S(68));
154 H4: HA port map (S(7),S(8),S(69),S(70));
155 H5: HA port map (S(9),S(10),S(71),S(72));
156 H6: HA port map (S(11),S(12),S(73),S(74));
157 H7: HA port map (S(13),S(14),S(75),S(76));
158
159
160 F1: FA port map (S(15),S(64),S(65),P(2),S(77)); -- Full adders 2nd row b2
161 F2: FA port map (S(16),S(66),S(67),S(78),S(79));
162 F3: FA port map (S(17),S(68),S(69),S(80),S(81));
163 F4: FA port map (S(18),S(70),S(71),S(82),S(83));
164 F5: FA port map (S(19),S(72),S(73),S(84),S(85));
165 F6: FA port map (S(20),S(74),S(75),S(86),S(87));
166 F7: FA port map (S(21),S(76),S(77),S(88),S(89));
167
168 F8: FA port map (S(23),S(77),S(78),P(3),S(90)); -- Full adders 3rd row b3
169 F9: FA port map (S(24),S(79),S(80),S(91),S(92));
170 F10: FA port map (S(25),S(81),S(82),S(93),S(94));
171 F11: FA port map (S(26),S(83),S(84),S(95),S(96));
172 F12: FA port map (S(27),S(85),S(86),S(97),S(98));
173 F13: FA port map (S(28),S(87),S(88),S(99),S(100));
174 F14: FA port map (S(29),S(89),S(90),S(101),S(102));
175
176 F15: FA port map (S(31),S(90),S(91),P(4),S(103)); -- Full adders 4th row b4
177 F16: FA port map (S(32),S(92),S(93),S(104),S(105));
178 F17: FA port map (S(33),S(94),S(95),S(106),S(107));
179 F18: FA port map (S(34),S(96),S(97),S(108),S(109));
180 F19: FA port map (S(35),S(98),S(99),S(110),S(111));
181 F20: FA port map (S(36),S(100),S(101),S(112),S(113));
182 F21: FA port map (S(37),S(102),S(103),S(114),S(115));
183
184 F22: FA port map (S(39),S(103),S(104),P(5),S(116)); -- Full adders 5th row b5
185 F23: FA port map (S(40),S(105),S(106),S(117),S(118));
186 F24: FA port map (S(41),S(107),S(108),S(119),S(120));
187 F25: FA port map (S(42),S(109),S(110),S(121),S(122));
188 F26: FA port map (S(43),S(111),S(112),S(123),S(124));
189 F27: FA port map (S(44),S(113),S(114),S(125),S(126));
190 F28: FA port map (S(45),S(115),S(116),S(127),S(128));
191
192 F29: FA port map (S(47),S(116),S(117),P(6),S(129)); -- Full adders 6th row b6
193 F30: FA port map (S(48),S(118),S(119),S(130),S(131));
194 F31: FA port map (S(49),S(120),S(121),S(132),S(133));
195 F32: FA port map (S(50),S(122),S(123),S(134),S(135));
196 F33: FA port map (S(51),S(124),S(125),S(136),S(137));
197 F34: FA port map (S(52),S(126),S(127),S(138),S(139));
198 F35: FA port map (S(53),S(128),S(54),S(140),S(141));
199
200 F36: FA port map (S(55),S(129),S(130),P(7),S(142)); -- Full adders 7th row b7
201 F37: FA port map (S(56),S(131),S(132),S(143),S(144));

```

```

199 :
200 : F36: FA port map (S(55),S(129),S(130),P(7),S(142)); -- Full adders 7th row b7
201 : F37: FA port map (S(56),S(131),S(132),S(143),S(144));
202 : F38: FA port map (S(57),S(133),S(134),S(145),S(146));
203 : F39: FA port map (S(58),S(135),S(136),S(147),S(148));
204 : F40: FA port map (S(59),S(137),S(138),S(149),S(150));
205 : F41: FA port map (S(60),S(139),S(140),S(151),S(152));
206 : F42: FA port map (S(61),S(141),S(62),S(153),S(154));
207 :
208 : H8: HA port map (S(142),S(143),P(8),S(155)); -- Half adder bottom row
209 :
210 : F43: FA port map (S(144),S(145),S(155),P(9),S(156)); -- Full adders bottom row
211 : F44: FA port map (S(146),S(147),S(156),P(10),S(157));
212 : F45: FA port map (S(148),S(149),S(157),P(11),S(158));
213 : F46: FA port map (S(150),S(151),S(158),P(12),S(159));
214 : F47: FA port map (S(152),S(153),S(159),P(13),S(160));
215 : F48: FA port map (S(154),S(63),S(160),P(14),P(15));
216 :
217 : end architecture;
218 :

```

Appendix B

Testbench code

```

5  library IEEE;
6  use IEEE.Std_logic_1164.all;
7  use IEEE.Numeric_Std.all;
8  use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10 entity Multi_tb is
11 end;
12
13 architecture bench of Multi_tb is
14
15     component Multi
16         port(A,B: in bit_vector(7 downto 0);
17             P: buffer bit_vector(15 downto 0));
18     end component;
19
20     signal A,B: bit_vector(7 downto 0);
21     signal P: bit_vector(15 downto 0);
22
23 begin
24
25     uut: Multi port map ( A => A,
26                         B => B,
27                         P => P );
28
29     Force: process
30         constant period: time := 20 ns; -- Sets constant time period
31         begin
32
33
34         A <= "11111111"; B <= "11111111"; WAIT FOR period; -- Sets values for A and B manually
35
36         A <= "00000000"; B <= "00000000"; WAIT FOR period;
37
38         A <= "00001111"; B <= "11110000"; WAIT FOR period;
39
40         A <= "00000000"; B <= "11111111"; WAIT FOR period;
41
42         A <= "01010101"; B <= "10101010"; WAIT FOR period;
43
44         A <= "11001100"; B <= "00110011"; WAIT FOR period;
45
46         A <= "00011000"; B <= "11100111"; WAIT FOR period;
47
48         A <= "00111111"; B <= "11111111"; WAIT FOR period;
49
50         A <= "00000011"; B <= "00000001"; WAIT FOR period;
51
52         A <= "11000111"; B <= "11110011"; WAIT FOR period;
53
54         A <= "01001111"; B <= "10110000"; WAIT FOR period;
55
56         A <= "00011001"; B <= "11110111"; WAIT FOR period;

```

```
50 | A <= "00000011"; B <= "00000001"; WAIT FOR period;
51 |
52 | A <= "11000111"; B <= "11110011"; WAIT FOR period;
53 |
54 | A <= "01001111"; B <= "10110000"; WAIT FOR period;
55 |
56 | A <= "00011001"; B <= "11110111"; WAIT FOR period;
57 |
58 | A <= "01000101"; B <= "10101010"; WAIT FOR period;
59 |
60 | A <= "01111100"; B <= "00111011"; WAIT FOR period;
61 |
62 | A <= "00011011"; B <= "11100111"; WAIT FOR period;
63 |
64 | A <= "00100111"; B <= "11111111"; WAIT FOR period;
65 |
66 |
67 |     wait;
68 | end process;
69 |
70 |
71 | end;
72 |
73 |
```

Appendix C

Wallace tree VHDL code

17

```

1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 -- Uncomment the following library declaration if using
5 -- arithmetic functions with Signed or Unsigned values
6 --use IEEE.NUMERIC_STD.ALL;
7
8 -- Uncomment the following library declaration if instantiating
9 -- any Xilinx leaf cells in this code.
10 --library UNISIM;
11 --use UNISIM.VComponents.all;
12
13 entity AND2 is -- Define AND I/O
14     port(
15         A,B: in BIT;
16         X: buffer BIT); -- set as buffer so can be input or output
17 end AND2;
18
19 architecture behavioral of AND2 is -- Define AND behavior
20 begin
21     X <= A and B;
22 end behavioral;
23
24 entity HA is -- Define Half adder behavior
25     port(
26         a,b: in BIT;
27         sum, carry: buffer BIT);
28 end HA;
29
30 architecture behavioral of HA is -- Define Half adder Behavior
31 begin
32     sum <= a xor b;
33     carry <= a and b;
34
35 end behavioral;
36
37 entity FA is -- Define full adder behavior
38     port(
39         a,b,c: in BIT;
40         sum, carry: buffer BIT);
41 end FA;
42
43 architecture behavioral of FA is -- Define full adder Behavior
44 begin
45     sum <= a xor b XOR c;
46     carry <= ((a and b) or (b and c) or (a and c));
47
48 end behavioral;
49
50 entity WallaceMultiplier is -- set inputs and outputs
51     port(A,B: in bit_vector(7 downto 0);

```

```

50 entity WallaceMultiplier is -- set inputs and outputs
51     port(A,B: in bit_vector(7 downto 0);
52           P: buffer bit_vector(15 downto 0)); -- buffer output can be used as input
53 end WallaceMultiplier;
54
55 architecture structural of WallaceMultiplier is
56     component AND2
57     Port(
58         A,B: in BIT;
59         X: buffer BIT);
60     end component;
61
62     component HA
63     port(
64         a,b: in BIT;
65         sum, carry: buffer BIT);
66     end component;
67
68     component FA
69     port(
70         a,b,c: in BIT;
71         sum, carry: buffer BIT);
72     end component;
73
74     signal S: BIT_vector(175 downto 1); -- Sets signals?
75
76 begin
77
78     A1: AND2 port map (A(0),B(0),P(0)); -- and gates top row WORK FROM RIGHT TO LEFT FROM THE TOP
79     A2: AND2 port map (A(0),B(1),S(1));
80     A3: AND2 port map (A(1),B(0),S(2));
81     A4: AND2 port map (A(1),B(1),S(3));
82     A5: AND2 port map (A(2),B(0),S(4));
83     A6: AND2 port map (A(2),B(1),S(5));
84     A7: AND2 port map (A(3),B(0),S(6));
85     A8: AND2 port map (A(3),B(1),S(7));
86     A9: AND2 port map (A(4),B(0),S(8));
87     A10: AND2 port map (A(4),B(1),S(9));
88     A11: AND2 port map (A(5),B(0),S(10));
89     A12: AND2 port map (A(5),B(1),S(11));
90     A13: AND2 port map (A(6),B(0),S(12));
91     A14: AND2 port map (A(6),B(1),S(13));
92     A15: AND2 port map (A(7),B(0),S(14));
93
94     A16: AND2 port map (A(0),B(2),S(15)); -- and gates 2nd row
95     A17: AND2 port map (A(1),B(2),S(16));
96     A18: AND2 port map (A(2),B(2),S(17));
97     A19: AND2 port map (A(3),B(2),S(18));
98     A20: AND2 port map (A(4),B(2),S(19));
99     A21: AND2 port map (A(5),B(2),S(20));
100    A22: AND2 port map (A(6),B(2),S(21));
101    A23: AND2 port map (A(7),B(1),S(22));

```

```

100  A22: AND2 port map (A(6),B(2),S(21));
101  A23: AND2 port map (A(7),B(1),S(22));
102
103  A24: AND2 port map (A(0),B(3),S(23)); -- and gates 3rd row
104  A25: AND2 port map (A(1),B(3),S(24));
105  A26: AND2 port map (A(2),B(3),S(25));
106  A27: AND2 port map (A(3),B(3),S(26));
107  A28: AND2 port map (A(4),B(3),S(27));
108  A29: AND2 port map (A(5),B(3),S(28));
109  A30: AND2 port map (A(6),B(3),S(29));
110  A31: AND2 port map (A(7),B(2),S(30));
111
112  A32: AND2 port map (A(0),B(4),S(31)); -- and gates 4th row
113  A33: AND2 port map (A(1),B(4),S(32));
114  A34: AND2 port map (A(2),B(4),S(33));
115  A35: AND2 port map (A(3),B(4),S(34));
116  A36: AND2 port map (A(4),B(4),S(35));
117  A37: AND2 port map (A(5),B(4),S(36));
118  A38: AND2 port map (A(6),B(4),S(37));
119  A39: AND2 port map (A(7),B(3),S(38));
120
121  A40: AND2 port map (A(0),B(5),S(39)); -- and gates 5th row
122  A41: AND2 port map (A(1),B(5),S(40));
123  A42: AND2 port map (A(2),B(5),S(41));
124  A43: AND2 port map (A(3),B(5),S(42));
125  A44: AND2 port map (A(4),B(5),S(43));
126  A45: AND2 port map (A(5),B(5),S(44));
127  A46: AND2 port map (A(6),B(5),S(45));
128  A47: AND2 port map (A(7),B(4),S(46));
129
130  A48: AND2 port map (A(0),B(6),S(47)); -- and gates 6th row
131  A49: AND2 port map (A(1),B(6),S(48));
132  A50: AND2 port map (A(2),B(6),S(49));
133  A51: AND2 port map (A(3),B(6),S(50));
134  A52: AND2 port map (A(4),B(6),S(51));
135  A53: AND2 port map (A(5),B(6),S(52));
136  A54: AND2 port map (A(6),B(6),S(53));
137  A55: AND2 port map (A(7),B(5),S(54));
138
139  A56: AND2 port map (A(0),B(7),S(55)); -- and gates bottom row
140  A57: AND2 port map (A(1),B(7),S(56));
141  A58: AND2 port map (A(2),B(7),S(57));
142  A59: AND2 port map (A(3),B(7),S(58));
143  A60: AND2 port map (A(4),B(7),S(59));
144  A61: AND2 port map (A(5),B(7),S(60));
145  A62: AND2 port map (A(6),B(7),S(61));
146  A63: AND2 port map (A(7),B(6),S(62));
147  A64: AND2 port map (A(7),B(7),S(63));
148
149
150  H1: HA port map (S(1),S(2),P(1),S(64)); -- First half adder level 1 row 1
151  F1: FA port map (S(4),S(3),S(15),S(65),S(66)); -- Full adders top row

```

```

149 :
150 H1: HA port map (S(1),S(2),P(1),S(64)); -- First half adder level 1 row 1
151 F1: FA port map (S(4),S(3),S(15),S(65),S(66)); -- Full adders top row
152 F2: FA port map (S(6),S(5),S(16),S(67),S(68));
153 F3: FA port map (S(8),S(7),S(17),S(69),S(70));
154 F4: FA port map (S(10),S(9),S(18),S(71),S(72));
155 F5: FA port map (S(12),S(11),S(19),S(73),S(74));
156 F6: FA port map (S(14),S(13),S(20),S(75),S(76));
157 H2: HA port map (S(22),S(21),S(77),S(78)); -- 2nd half adder top row
158
159 H3: HA port map (S(24),S(31),S(79),S(80)); -- first half adder level 1 row 2
160 F7: FA port map (S(25),S(32),S(39),S(81),S(82)); -- Full adders top row
161 F8: FA port map (S(26),S(33),S(40),S(83),S(84));
162 F9: FA port map (S(27),S(34),S(41),S(85),S(86));
163 F10: FA port map (S(28),S(35),S(42),S(87),S(88));
164 F11: FA port map (S(29),S(36),S(43),S(89),S(90));
165 F12: FA port map (S(38),S(37),S(44),S(91),S(92));
166 H4: HA port map (S(46),S(45),S(93),S(94)); -- 2nd half adder top row
167
168
169 H5: HA port map (S(64),S(65),P(2),S(95)); -- first half adder level 2 row 1
170 F13: FA port map (S(23),S(67),S(66),S(96),S(97)); -- Full adders level 2 row 1
171 F14: FA port map (S(79),S(69),S(68),S(98),S(99));
172 F15: FA port map (S(81),S(71),S(70),S(100),S(101));
173 F16: FA port map (S(83),S(73),S(72),S(102),S(103));
174 F17: FA port map (S(85),S(75),S(74),S(104),S(105));
175 F18: FA port map (S(87),S(77),S(76),S(106),S(107));
176 F19: FA port map (S(89),S(30),S(78),S(108),S(109));
177
178 H6: HA port map (S(47),S(82),S(110),S(111)); -- level 2 row 2
179 F20: FA port map (S(48),S(84),S(55),S(112),S(113));
180 F21: FA port map (S(49),S(86),S(56),S(114),S(115));
181 F22: FA port map (S(50),S(88),S(57),S(116),S(117));
182 F23: FA port map (S(51),S(90),S(58),S(118),S(119));
183 F24: FA port map (S(52),S(92),S(59),S(120),S(121));
184 F25: FA port map (S(53),S(94),S(60),S(122),S(123));
185 H7: HA port map (S(62),S(61),S(124),S(125));
186
187
188 H8: HA port map (S(96),S(95),P(3),S(126)); -- level 3
189 H9: HA port map (S(98),S(97),S(127),S(128));
190 F26: FA port map (S(100),S(99),S(80),S(129),S(130)); -- S80??
191 F27: FA port map (S(102),S(101),S(110),S(131),S(132));
192 F28: FA port map (S(104),S(103),S(112),S(133),S(134));
193 F29: FA port map (S(106),S(105),S(114),S(135),S(136));
194 F30: FA port map (S(108),S(107),S(116),S(137),S(138));
195 F31: FA port map (S(91),S(109),S(118),S(139),S(140));
196 H10: HA port map (S(120),S(93),S(141),S(142));
197 H11: HA port map (S(122),S(54),S(143),S(144));
198
199
200 H12: HA port map (S(127),S(126),P(4),S(145)); -- level 4

```

```

200      H12: HA port map (S(127),S(126),P(4),S(145)); -- level 4
201      H13: HA port map (S(129),S(128),S(146),S(147));
202      H14: HA port map (S(131),S(130),S(148),S(149));
203      F32: FA port map (S(133),S(132),S(111),S(150),S(151)); -- S80??
204      F33: FA port map (S(135),S(134),S(113),S(152),S(153));
205      F34: FA port map (S(137),S(136),S(115),S(154),S(155));
206      F35: FA port map (S(139),S(138),S(117),S(156),S(157));
207      F36: FA port map (S(141),S(140),S(119),S(158),S(159));
208      F37: FA port map (S(143),S(142),S(121),S(160),S(161));
209      F38: FA port map (S(124),S(144),S(123),S(162),S(163)); --124??
210      H15: HA port map (S(125),S(63),S(164),S(165));
211
212      H16: HA port map (S(145),S(146),P(5),S(166)); -- level 5
213      F39: FA port map (S(147),S(148),S(166),P(6),S(167)); -- S80??
214      F40: FA port map (S(149),S(150),S(167),P(7),S(168));
215      F41: FA port map (S(151),S(152),S(168),P(8),S(169));
216      F42: FA port map (S(153),S(154),S(169),P(9),S(170));
217      F43: FA port map (S(155),S(156),S(170),P(10),S(171));
218      F44: FA port map (S(157),S(158),S(171),P(11),S(172));
219      F45: FA port map (S(159),S(160),S(172),P(12),S(173)); --124??
220      F46: FA port map (S(161),S(162),S(173),P(13),S(174));
221      F47: FA port map (S(163),S(164),S(174),P(14),S(175));
222      H17: HA port map (S(165),S(175),P(15));
223
224
225
226  end architecture;
--

```

FINAL GRADE

GENERAL COMMENTS

80/100

Instructor

a. Introduction to Multipliers/ Various Multipliers/ Advantages and Disadvantages/ Applications of Multipliers (8) :4 Marks

Very shallow writing on the introduction. Needs to address more on the types, benefits of Multipliers.

b. Background Research (8) :6 Marks

Excellent! Engagement with an extensive range of relevant and credible literature.

c. Diversity of sources and Referencing Skills(4) :3 Marks
Consistently accurate application of referencing.

a. Task 1: Identifying the correct Multiplier (4), Architecture of Braun's Multiplier(6) :9 Marks

Well Done! The correct Multiplier is identified. Its key concepts were explained. It is evident that you have a clear idea about multipliers. Architecture of 8X8 Braun's is well captured and the diagram is very clear and precise which is appreciable. Assumptions of partial signals made is very clear.

b. Task 2: Dot Diagram(4), Architecture of Wallace Multiplier(6) :9 Marks

Good Job- The Dot diagram is correct and well captured. Architecture of Wallace is also seen which is up to the Expectations. The consideration of the Partial Signals are also found. The usage of Full Adders and Half Adders is seen which is up to the satisfaction.

a. Finding

Critical Path, Longest Delay, Min and Max Delay, Power Consumption(12). :9 Marks

Good Job- Critical evaluation on finding the Critical path, longest min, max delay, Power consumption were made. The range slightly differs with varies respect to the no. of LUTs and I/O used. Based upon the Evidence Based judgements the solutions you have stated for delay are within the expected range. But the Power consumption for Wallace should be different. The longest delay shows that you have large no. of adders. But the delay between the two multipliers is almost similar which is not the actual case.

b. Innovative thinking in problem solving and feature additions(8) :6 Marks

Clearly distinguishes between fact, opinion, & acknowledges value judgments are seen.

a.

Simulation (7) :6 Marks

Good- The simulations were clearly matching with the expected outputs. Proper usage of test bench were seen.

b. VHDL

Coding's (10) :9 Marks

Good - Both the codes are relevant and produces the expected outputs and results without any errors. The products are matching with the Manual Calculations. The collection of samples taken is evidently seen in your report that is appreciable.

b. Circuit Schematic (4) :4 Marks

Good- The screen dump provided for the Circuit schematic is neat and precise in both the cases of multipliers. It covers most of the Blocks.

d. FPGA Synthesis (4) :4 Marks

Synthesis is provided. It clearly shows that the you have a very good experience in obtaining the synthesis and Implementation reports. The procedures to capture FPGA synthesis were done perfectly

a. Logical order of ideas, Flow of ideas in the report from the beginning to end with appropriate chapters, sub chapters and continuation(7) :6 Marks

Work is coherent, very fluent and is presented proficiently.

b. Definitions, Acronyms and Abbreviations (3) :2 Marks

Very Good- Most of the Definitions, Acronyms and Abbreviations are captured.

c. Professionalism, Spelling and Grammar, Complete Report, Proper Presentation, Quality of the report(5) :3 Marks

Good level of professionalism is maintained in the complete report. Quality of the report is good with minimal spelling and grammar mistakes.

TOTAL MARKS : 80%

Findings from the Codes and Simulations :

Codes have port and signal declarations . No errors or warnings found when compiling the code. Too many signals were taken which limits the efficiency of the Multipliers.

[Click here to view the feed back Report](#)

PAGE 1

PAGE 2

PAGE 3

PAGE 4



Comment 1

Very shallow writing on the introduction. Needs to address more on the purpose and benefits of Multipliers.

PAGE 5



Comment 2

It could be better if the Multipliers are addressed initially



Comment 3

Good

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11



Comment 4

Good- Various Multipliers are seen

PAGE 12

PAGE 13

PAGE 14



Comment 5

Well Done! The correct Multiplier is identified. Its key concepts were explained. It is evident that you have a clear idea about multipliers.

PAGE 15



Comment 6

Well Done- Architecture of 8X8 Braun's is well captured. Assumptions of partial signals made is very clear.

PAGE 16

PAGE 17

PAGE 18

PAGE 19



Comment 7

The collection of samples taken is evidently seen in your report that is appreciable.

PAGE 20

PAGE 21



Comment 8

Good- The screen dump provided for the Circuit schematic is neat and precise in both the cases of multipliers.

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29



Comment 9

You have a very good experience in obtaining the synthesis and Implementation reports.

PAGE 30



Comment 10

Good Job- The Dot diagram is correct and well captured.

PAGE 31



Comment 11

Architecture of Wallace is also seen which is up to the Expectations. The consideration of the Partial Signals are also found.

PAGE 32

PAGE 33

PAGE 34



Comment 12

The usage of Full Adders and Half Adders is seen which is up to the satisfaction

PAGE 35



Comment 13

The simulations were clearly matching with the expected outputs.

PAGE 36

PAGE 37

PAGE 38

PAGE 39

PAGE 40

PAGE 41

PAGE 42



Comment 14

Good Job- Critical evaluation on finding the Power consumption were made. But the Power consumption for Wallace should be different.

PAGE 43



Comment 15

The longest delay shows that you have large no. of adders. But the delay between the two multipliers is almost similar which is not the actual case.

PAGE 44

PAGE 45

PAGE 46

PAGE 47

PAGE 48

PAGE 49

PAGE 50



Comment 16

Good Job- Critical evaluation on finding the Critical path, longest min, max delay were made. The range slightly differs with varies respect to the no. of LUTs and I/O used. Based upon the Evidence Based judgements the solutions you have stated for delay are within the expected range.

PAGE 51

PAGE 52

PAGE 53

PAGE 54

PAGE 55

PAGE 56

PAGE 57

PAGE 58

PAGE 59

PAGE 60

PAGE 61

PAGE 62

PAGE 63

PAGE 64

PAGE 65

PAGE 66



Comment 17

Good - Both the codes are relevant and produces the expected outputs and results without any errors. The products are matching with the Manual Calculations.

PAGE 67

PAGE 68

PAGE 69

PAGE 70
