Electronics 2 Coursework

by Robbie Herbert

Submission date: 17-Apr-2025 01:06PM (UTC+0100)
Submission ID: 255953367
File name: 57713_Robbie_Herbert_Electronics_2_Coursework_654553_678400638.docx (5.07M)
Word count: 5333
Character count: 28616



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

Electronics II ACDA6001

8x8 Brauns Multiplier & 8x8 Wallace tree Multiplier

By

Robbie Herbert

2405914

Submission Date: 17th April 2025

Lecturer: Jenifer Prince Devaraj

Word Count: 2473

1. Contents

1. Contents 2
2. List of Figures 3
2.1. List of Tables 5
3. Acronyms, Abbreviations and Units 6
4. Abstract 7
5. Introduction 7
5.1. Background research 7
5.2. Applications and Advantages 8
5.2.1. Brauns Multiplier 8
5.2.2. Wallace Multiplier 9
5.3. Flow Charts/Algorithms 10
5.4. Methodologies 12
6. Main Body 13
6.1. Multiplier block diagram and Algorithms 13
6.1.1. Brauns 13
6.1.2. Wallace Tree 15
6.2. VHDL Codes 17
6.2.1. Full Adder 17
6.2.2. Half Adder 17
6.2.3. Brauns Multiplier code 18
6.2.4. Wallace Multiplier code 22
6.3. Brauns Multiplier Device 25
6.4. Wallace Multiplier Device 26
6.5. Timing Diagrams and Schematics 26
6.5.1. Brauns Multiplier Timing, Longest and Shortest Delay Diagrams & Combinational Delays 27
6.5.2. Wallace Multiplier Timing Summary, Longest and Shortest delays & Combinational Delays 33
6.6. Design Screenshots and Pin Allocation 37
6.6.1. Brauns Multiplier Designs & Pin allocation 37
6.6.2. Wallace Multiplier Designs & Pin Allocation 41
7. Conclusion 44
7.1. Simulation Results 45

7.2. Comparison, Analysis/testing details, and interpretation 58
7.2.1. Interpretation of Power utilisation and LUTs 58
7.2.2. Details of testing 62
7.3. Recommendations 68
8. References 69
9. Appendix 71

2. List of Figures

Figure 1 flow chart of Brauns multiplier/Algorithm 10
Figure 2 Wallace Tree Flowchart/Algorithm 11
Figure 3 Brauns multiplier block diagram. 13
Figure 4 Brauns Multiplier Architecture. 14
Figure 5 Wallace Tree Multiplier Diagram 15
Figure 6 Wallace Tree Architecture. 16
Figure 7 Full Adder Code 17
Figure 8 Half Adder Code 17
Figure 9 Brauns Multiplier Start 18
Figure 10 Signal Generation 1 18
Figure 11 Signal Generation 4 & sum & carry signals p1-p4. 19
Figure 12 Sum & Carry signals P12, P13 & P14 19
Figure 13 Full & Half adder components. 20
Figure 14 Brauns Multiplier Partial Products 1 20
Figure 15 Brauns Multiplier Port Maps 1 21
Figure 16 Wallace Multiplier Code start. 22
Figure 17 Wallace Multiplier Code 2 22
Figure 18 Wallace Multiplier Code 3 & Sum & Carry Signals 22
Figure 19 Wallace Multiplier Signal Generation 1 23
Figure 20 Wallace Multiplier Partial Products 1 23
Figure 21 Wallace Multiplier Stage 1 & 2 Reduction 24
Figure 22 Wallace Multiplier Final Stage Reduction 24
Figure 23 Brauns Multiplier Device 25
Figure 24 Wallace Multiplier Device 26
Figure 25 Brauns Multiplier Timing Constraint 28
Figure 26 Brauns Multiplier Shortest Delay Diagram 29
Figure 27 Brauns Multiplier Longest Delay Diagram 30
Figure 28 Brauns Multiplier Synthesis Timing Summary 31
Figure 29 Brauns Multiplier Implementation Timing Summary 31
Figure 30 Brauns Multiplier Combinational Delay 1 32
Figure 31 Wallace Multiplier Timing Constraint 33
Figure 32 Wallace Multiplier Shortest Delay Diagram 33
Figure 33 Wallace Multiplier Longest Delay Diagram 34
Figure 34 Wallace Multiplier Synthesis Timing Summary 34
Figure 35 Wallace Multiplier Implementation Timing Summary 35

Figure 36 Wallace Multiplier Combinational Delays 1 36
Figure 37 RTL Design Brauns 37
Figure 38 Implementation Design Brauns 38
Figure 39 Pin Allocation Brauns 39
Figure 40 Pin Allocation Brauns 2 39
Figure 41 Brauns Multiplier I/O Planning 40
Figure 42 Wallace Multiplier RTL Design 41
Figure 43 Wallace Multiplier Implemented Design 42
Figure 44 Wallace Multiplier Pin Allocation 43
Figure 45 Wallace Multiplier Pin Allocation 2 43
Figure 46 Wallace Multiplier I/O Planning 44
Figure 47 Hex value 1 46
Figure 48 Brauns Simulation of Hex Value 1 47
Figure 49 Wallace Simulation of Hex Value 1 48
Figure 50 Hex value 2 49
Figure 51 Brauns Multiplier Hex Value 2 50
Figure 52 Wallace Simulation of Hex Value 2 51
Figure 53 Hex value 3 52
Figure 54 Brauns Simulation of Hex Value 3 53
Figure 55 Wallace Simulation of Hex Value 3 54
Figure 56 Hex value 4 55
Figure 57 Brauns Simulation of Hex Value 4 56
Figure 58 Wallace Simulation of Hex Value 4 57
Figure 59 Brauns Multiplier Power Summary 59
Figure 60 Wallace Multiplier Power Report 59
Figure 61 Brauns Multiplier Utilization Hierarchy 60
Figure 62 Wallace Multiplier Utilization Hierarchy 60
Figure 63 Brauns Multiplier Utilization Summary 60
Figure 64 Wallace Multiplier Utilization Summary 61
Figure 65 Brauns Multiplier Project Summary 61
Figure 66 Wallace Multiplier Project Summary 62
Figure 67 Basys 3 FF x FF 64
Figure 68 Basys 3 13 x 34 65
Figure 69 Basys 3 EF x EA 66
Figure 70 Basys 3 AA x AB 67
Figure 71 Signal Generation 2 71
Figure 72 Signal Generation 3 71
Figure 73 Sum & Carry signals P5 & P6 72
Figure 74 Sum & Carry signals P7. 72
Figure 75 Sum & Carry signals P8. 73
Figure 76 Sum & Carry signals P9. 73
Figure 77 Sum & Carry Signals P10 & P11 74
Figure 78 Brauns Multiplier Partial Products 2 74
Figure 79 Brauns Multiplier Partial Products 3 75
Figure 80 Brauns Multiplier Partial Products 4 75
Figure 81 Brauns Multiplier Port Maps 2 76
Figure 82 Brauns Multiplier Port Maps 3 76

Figure 83 Wallace Multiplier Signal Generation 2 77
Figure 84 Wallace Multiplier Signal Generation 4 77
Figure 85 Wallace Multiplier Signal Generation 5 78
Figure 86 Wallace Multiplier Partial Products 2 78
Figure 87 Wallace Multiplier Partial Products 3 79
Figure 88 Wallace Multiplier Partial Products 4 79
Figure 89 Wallace Multiplier Stage 3 & 4 Reduction 80
Figure 90 Wallace Multiplier Stage 5 & 6 Reduction 80
Figure 91 Brauns Multiplier Combinational Delay 2 81
Figure 92 Brauns Multiplier Combinational Delay 3 82
Figure 93 Brauns Multiplier Combinational Delay 4 83
Figure 94 Brauns Multiplier Combinational Delay 5 84
Figure 95 Brauns Multiplier Combinational Delay 6 85
Figure 96 Brauns Multiplier Combinational Delay 7 86
Figure 97 Brauns Multiplier Combinational Delay 8 87
Figure 98 Brauns Multiplier Combinational Delay 9 88
Figure 99 Brauns Multiplier Combinational Delay 10 89
Figure 100 Brauns Multiplier Combinational Max Delay 11 90
Figure 101 Wallace Multiplier Combinational Delays 2 91
Figure 102 Wallace Multiplier Combinational Delays 3 92
Figure 103 Wallace Multiplier Combinational Delays 4 93
Figure 104 Wallace Multiplier Combinational Delays 5 94
Figure 105 Wallace Multiplier Combinational Delays 6 95
Figure 106 Wallace Multiplier Combinational Delays 7 96
Figure 107 Wallace Multiplier Combinational Delays 8 97
Figure 108 Wallace Multiplier Combinational Delays 9 98
Figure 109 Wallace Multiplier Combinational Delays 10 99
Figure 110 Wallace Multiplier Combinational Delays 11 100
Figure 111 Basys 3 Diagram [14] 101

2.1. List of Tables

Table 1 Abbreviations, Acronyms and Units table 6

3. Acronyms, Abbreviations and Units

Ω	Ohms
%	Percent
TC	Time Constant
AC	Alternating Current
DC	Direct Current
ms	Milli-seconds
Hz	Hertz
I/O	Input/Output
DSP	Digital Signal Processing
ALU	Arithmetic Logic Unit
WNS	Worst Negative Slack
LUT	Lookup Table
IOB	Input/Output Buffer

Table 1 Abbreviations, Acronyms and Units table

4. Abstract

This Paper shows an analysis of the Brauns and Wallace Multipliers used for eight-by-eight hexadecimal multiplication. Shown within the paper are algorithms, architecture, and comparisons between the two multipliers regarding areas of efficiency and speed. Using VHDL and Vivado software, tests were conducted to also see accuracy and power consumption, hardware tests were also performed. The Paper shows within its findings that the Wallace multiplier has advantages with speed, whereas the Brauns multiplier will allow for lower power applications. Simulation results showed that the accuracy of both multipliers is high, the power consumption of the Brauns multiplier is 33.33% less than the power consumption of the Wallace multiplier. The simulations also show that the Wallace multiplier is ≈ 8ns quicker than the Brauns multiplier.

5. Introduction

Multiplication is found in all areas of life. Multiplication is fundamental within modern systems; the core operation is used within applications such as image processing and machine learning for example[1]. As a result of multiplication being crucial within digital systems, efficient ways to multiply are crucial for optimisation of applications [1]. With use of efficient multipliers consumption of power and speed of multiplication becomes quicker [1]. Two ways of completing fast multiplication are a Brauns multiplier and a Wallace multiplier [1]. This paper aims to compare the two multipliers together and find performance differences, whilst also providing improvements for the multipliers to increase the potential. The results which have been found will help designers in their fields make a choice suitable for their applications for which multiplier to choose between.

5.1. Background research

Brauns and Wallace multipliers are both unique in their own ways. They also both have varied advantages and disadvantages with them [1]. The Brauns and Wallace multipliers are used within applications and the applications/systems that they are within will vary in complexity. Current developments within multiplier techniques can be seen as follows [1]:

Multipliers, over many years have become more complex and quicker [1]. The most essential element of a multiplier is to be accurate and now, they have become faster, lower powered, and the accuracy is exceedingly high [1]. Within a practical application of a multiplier, hardware has improved making the multipliers faster [1].

Within the development of the VHDL code the Brauns and Wallace multipliers will use the product family Artix-7 which is an FPGA. The two multipliers will use the same project parts, the name of this project part is xc7a35tcg236-1. This will be used to be able to generate bitstream and apply the software to the Basys 3 hardware this will also be used to allocate pins so that the multipliers will work.

Both the Brauns multiplier and Wallace multiplier can achieve the same thing. However, the speed, power consumption and components used vary quite a large amount. The codes which will be compiled will also vary, especially when the code comes to the port mapping and structure. The code will be created from the diagrams which have been created using Microsoft Visio. The Brauns and the Wallace multipliers are both different and are used within different applications. This is due to the fact of speed of multiplication of the two multipliers.

5.2. Applications and Advantages

5.2.1. Brauns Multiplier

The Brauns multiplier is in many applications within daily life and can also be found in many niche applications for the workplace. The Brauns multiplier can be found in image processing [2]. The way that a multiplier works within image processing is that the multiplier will be used to multiply the pixels within the image to make the image much clearer [2]. Depending on the pixel value the image will be clear, within two scenarios the multiplier may create a much clearer image if the correct factor of multiplication has taken place [2]. However, if the factor of multiplication is high and exceeds maximum values then the image will become distorted [2].

The Brauns multiplier can also be found in multimedia technology, as multimedia is graphics, sound, text, video and animation, the multiplier is used to multiply bits within these areas to make the sound clearer for example [2]. As stated previously with image processing, video and animation will be similar as multiplication of pixels will be used to make the quality of the image clearer on the screen. [2]

The Brauns multiplier can also be found in digital signal processing [3]. By using a Brauns multiplier a signal is boosted as its efficiency will be increased [3]. The effect of using a multiplier will also help with the power consumption of the component that is completing the task. [3]

Two of the most critical advantages of the Brauns multiplier are as follows:

Linear multiplication, the Brauns multiplier uses a linear multiplication structure to be able to complete its multiplication tasks [4]. This is named “carry-save.” This improves the accuracy of the multiplier [4]. Critical path, the critical path of the Brauns adder is stated to be its main advantaged

for the multiplication task [4]. As a result of the single critical path that the Brauns multiplier has the multiplier can be found in digital signal processing because of the low power consumption [4].

5.2.2. Wallace Multiplier

The Wallace multiplier is used in many applications. This multiplier is an efficient way of multiplying two integers [7]. The Wallace tree multiplier is what is called a parallel multiplier [7]. This means that the multiplier will not contain any method or structure of feedback. Within the Wallace multiplier, the first step is to is to use an AND gate to multiply each bit to another bit [7]. The products of these AND gates are called partial products [7]. Each of these products will go through an array of half adders and full adders to be able to multiply the A and B values to get the full product, or in the case of this multiplier the Hexadecimal value [7]. The Wallace multiplier uses a specific multiplication method, this is because the logic of the multiplication will help power consumption improving the Wallace multiplier and having greater use than other multipliers which do the same thing [7].

There are many applications of the Wallace multiplier. These applications are used in everyday life and within very niche markets [8]. This is because of the multiplier adaptability. The multiplier can be used in digital signal processing (DSP), the multiplier is used because of its fast multiplication [8]. Another application for the Wallace multiplier is ALU's, the ALU is found in Microprocessors, due to consisting of logic gates. The Wallace multiplier by this logic is found in most electronics [8]. Due to the method of adder logic that the Wallace multiplier uses the Wallace multiplier is much faster than other multipliers [9]. In comparison to other multipliers the Wallace multiplier is much less Complex as it has less Half Adders [9]. Another advantage is the Latency and the power consumption which is less than other multipliers [9].

5.3. Flow Charts/Algorithms



Figure 1 flow chart of Brauns Multiplier/Algorithm.

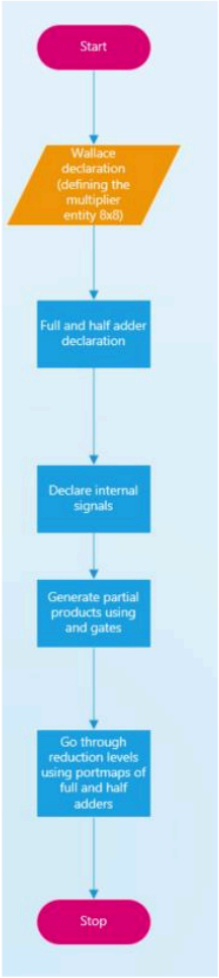


Figure 2 Wallace Tree Flowchart/Algorithm

5.4. Methodologies

The methodologies used to create both the Brauns multiplier, and the Wallace Tree multiplier are remarkably similar [6]. Throughout the research carried out many journals have been used and read thoroughly to ensure that they are dependable, and they are relating to the task [6]. CRAAP testing has been conducted on all the sources that are within this paper to create security and reliability of what is being stated within this task [6].

Similarly, within the code methodologies have been used to make sure the code works correctly and is accurate and dependable [6]. The agile methodology has been used to develop both codes. First, within this methodology is planning [6]. The plan is crucial within the rest of the code development as it will help keep the development on track. The plan helps with designing and developing the flowcharts within this paper [6].

The design would come next. The design of the codes helps to visualise what the code will look like [6]. The designs of both the Brauns and Wallace multipliers can be seen below with their respective algorithms [6].

After the design, the development of the code will come next. With use of the designs created the development of the code is much simpler than doing it blind [6]. As the help of the design will drive decisions for the portmaps created for the code [6].

Finally, the testing will come last as deployment of this product created will not occur. Within the test four sets of hexadecimal numbers were chosen [6]. These numbers were used to simulate the multipliers to guarantee the accuracy of the VHDL codes [6]. The hexadecimal values will be compared to the values found on an online calculator. Within the testing there will also be synthesis and then timing reports within the synthesis [6]. The RTL design will also be made with the implementation schematic to show how the code will look using l[illegible]. The device and I/O planning will also be completed and shown [6].

2

6. Main Body

6.1. Multiplier block diagram and Algorithms

6.1.1. Brauns

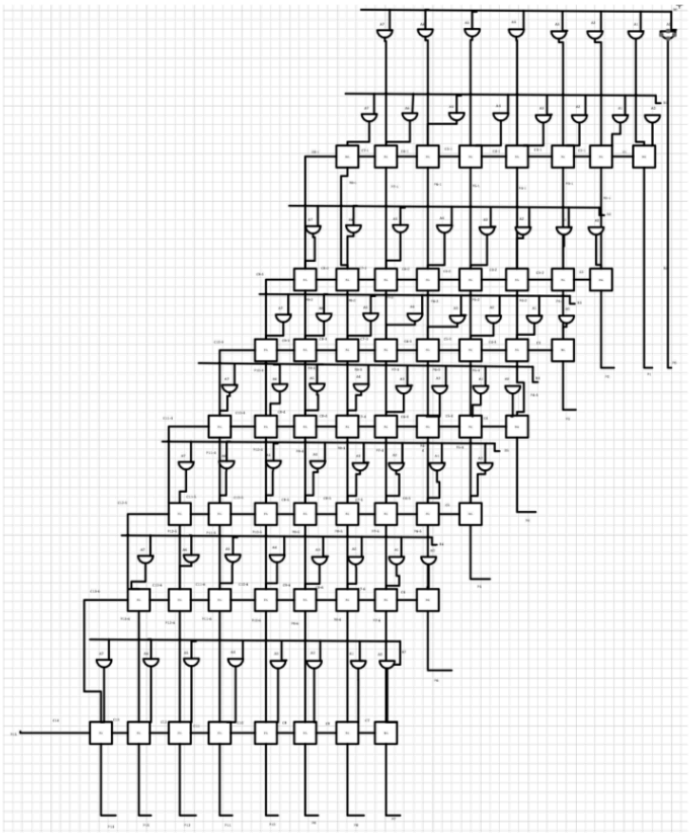


Figure 3 Brauns Multiplier block diagram

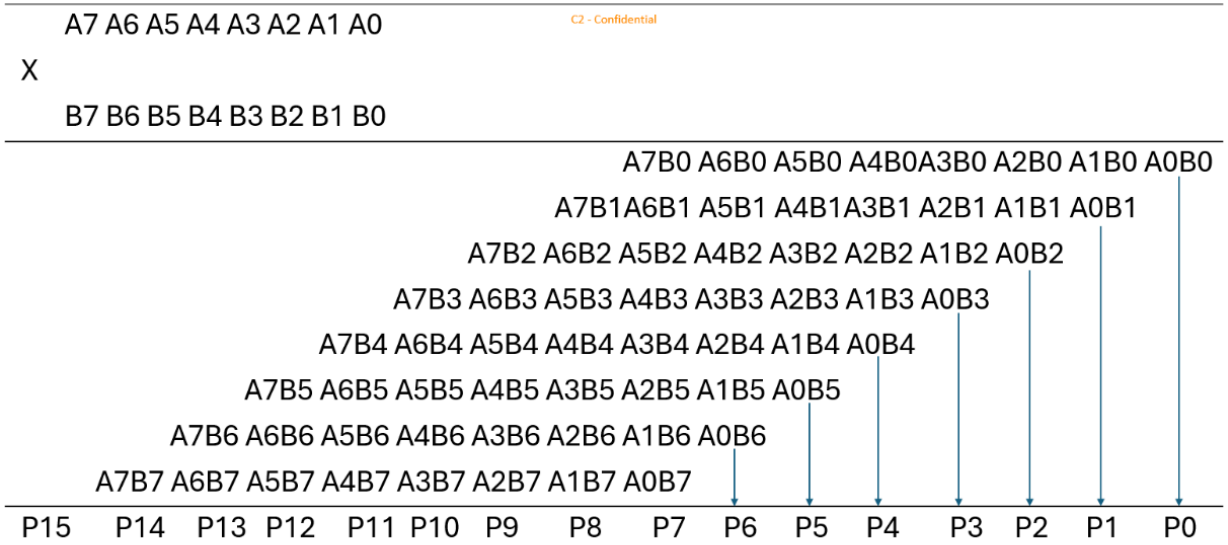


Figure 4 Brauns Multiplier Architecture.

6.1.2. Wallace Tree

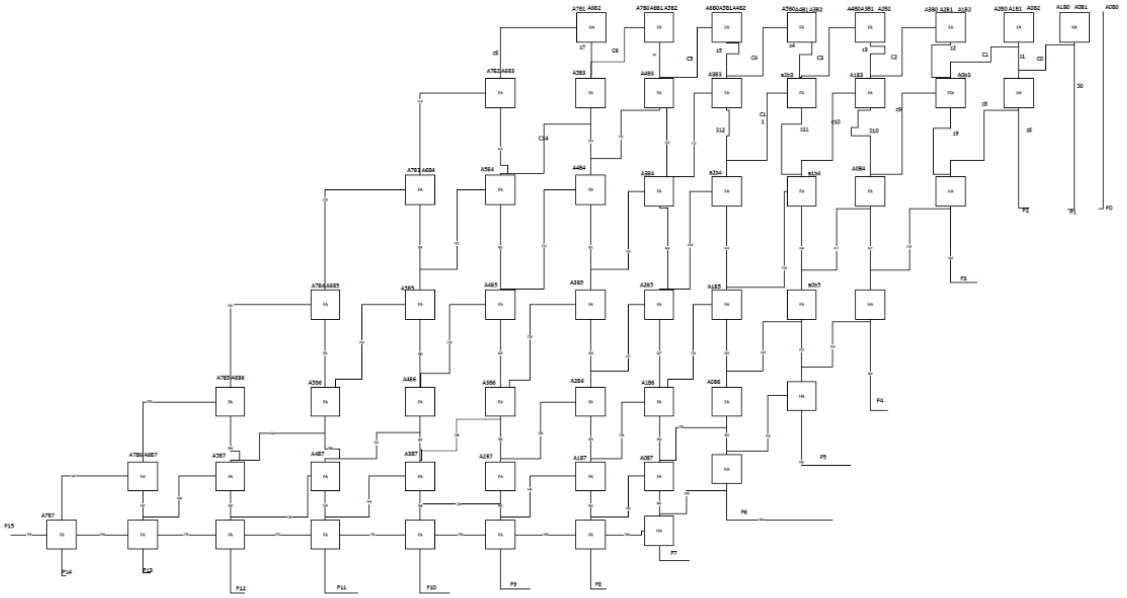


Figure 5 Wallace Tree Multiplier Diagram

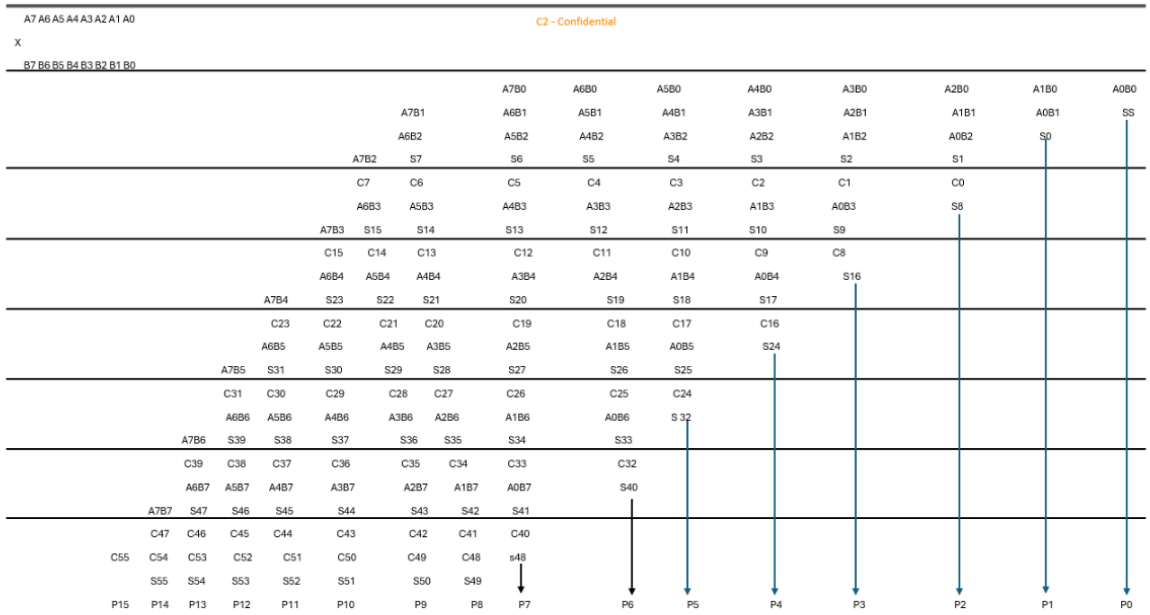


Figure 6 Wallace Tree Architecture.

6.2. VHDL Codes

6.2.1. Full Adder

```
entity Full_Adder is
    Port ( A : in STD_LOGIC;
           B : in STD_LOGIC;
           Cin : in STD_LOGIC;
           Cout : out STD_LOGIC;
           S : out STD_LOGIC);
end Full_Adder;

architecture Gate_level of Full_Adder is

begin

 S <= A XOR B XOR Cin ;
 Cout <= (A AND B) OR (Cin AND A) OR (Cin AND B);

end Gate_level ;
```

Figure 7 Full Adder Code

6.2.2. Half Adder

```
33
34  entity Half_adder is
35      Port ( A : in STD_LOGIC;
36             B : in STD_LOGIC;
37             S : out STD_LOGIC;
38             Cout : out STD_LOGIC);
39  end Half_adder;
40
41  architecture Gate_level of Half_adder is
42
43  begin
44
45   S <= A XOR B  ;
46   Cout <= (A AND B) ;
47
48  end Gate_level;
49
```

Figure 8 Half Adder Code

6.2.3. Brauns Multiplier code

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  entity brauns_multiplier is
7      Port (
8          A : in  STD_LOGIC_VECTOR (7 downto 0);
9          B : in  STD_LOGIC_VECTOR (7 downto 0);
10         P : out STD_LOGIC_VECTOR (15 downto 0)
11     );
12 end brauns_multiplier;
13
14 architecture Behavioural of brauns_multiplier is
15
16 signal A0B0 : std_logic;
17 signal A1B0 : std_logic;
18 signal A2B0 : std_logic;
19 signal A3B0 : std_logic;
20 signal A4B0 : std_logic;
21 signal A5B0 : std_logic;
22 signal A6B0 : std_logic;
23 signal A7B0 : std_logic;
```

Figure 9 Brauns Multiplier Start

```
24
25 signal A0B1 : std_logic;
26 signal A1B1 : std_logic;
27 signal A2B1 : std_logic;
28 signal A3B1 : std_logic;
29 signal A4B1 : std_logic;
30 signal A5B1 : std_logic;
31 signal A6B1 : std_logic;
32 signal A7B1 : std_logic;
33
34 signal A0B2 : std_logic;
35 signal A1B2 : std_logic;
36 signal A2B2 : std_logic;
37 signal A3B2 : std_logic;
38 signal A4B2 : std_logic;
39 signal A5B2 : std_logic;
40 signal A6B2 : std_logic;
41 signal A7B2 : std_logic;
```

Figure 10 Signal Generation 1

```
78
79  signal A0B7 : std_logic;
80  signal A1B7 : std_logic;
81  signal A2B7 : std_logic;
82  signal A3B7 : std_logic;
83  signal A4B7 : std_logic;
84  signal A5B7 : std_logic;
85  signal A6B7 : std_logic;
86  signal A7B7 : std_logic;
87
88  signal C1   : std_logic; -- P2
89  signal P2_1 : std_logic;
90
91  signal C2_1 : std_logic; -- P3
92  signal P3_1 : std_logic;
93  signal C2   : std_logic;
94  signal P3_2 : std_logic;
95
96  signal C3_1 : std_logic; -- P4
97  signal P4_1 : std_logic;
98  signal C3_2 : std_logic;
99  signal P4_2 : std_logic;
100 signal C3   : std_logic;
101 signal P4_3   : std_logic;
102 signal C4_3   : std_logic;
```

Figure 11 Signal Generation 4 & sum & carry signals p1-p4.

```
194  -- P12
195
196 signal C11   : std_logic;
197
198 signal P12_5 : std_logic;
199 signal P12_6 : std_logic;
200 signal C12_5 : std_logic;
201 signal C12_6 : std_logic;
202
203  -- P13
204
205 signal C12   : std_logic;
206
207
208 signal P13_6 : std_logic;
209 signal C13_6 : std_logic;
210
211  signal C13   : std_logic; -- P14
212 signal C14   : std_logic;
213
```

Figure 12 Sum & Carry signals P12, P13 & P14

```
214
215
216  component full_adder
217  port (A : in STD_LOGIC;
218   B : in STD_LOGIC;
219   Cin : in STD_LOGIC;
220   S : out STD_LOGIC;
221   Cout : out STD_LOGIC);
222  end component;
223
224  component half_adder
225  port (A : in STD_LOGIC;
226   B : in STD_LOGIC;
227    S : out STD_LOGIC;
228   Cout : out STD_LOGIC);
229  end component;
```

Figure 13 Full & Half adder components.

```
231  begin
232  A0B0 <= A(0) and B(0);
233  A1B0 <= A(1) and B(0);
234  A2B0 <= A(2) and B(0);
235  A3B0 <= A(3) and B(0);
236  A4B0 <= A(4) and B(0);
237  A5B0 <= A(5) and B(0);
238  A6B0 <= A(6) and B(0);
239  A7B0 <= A(7) and B(0);
240
241  A0B1 <= A(0) and B(1);
242  A1B1 <= A(1) and B(1);
243  A2B1 <= A(2) and B(1);
244  A3B1 <= A(3) and B(1);
245  A4B1 <= A(4) and B(1);
246  A5B1 <= A(5) and B(1);
247  A6B1 <= A(6) and B(1);
248  A7B1 <= A(7) and B(1);
```

Figure 14 Brauns Multiplier Partial Products 1

```
303
304     P0 : P(0) <= A(0) and B(0) ;
305   P1_0 : half_adder port map (A=>A1B0 , B=>A0B1 ,S=> P(1), Cout=>C1);
306
307   P21 : full_adder port map (A=>A2B0 , B=>A1B1 , Cin=>C1, S=>P2_1, Cout=>C2_1);
308   P22 : half_adder port map (A=>A0B2 , B=>P2_1 , S=>P(2), Cout=>C2);
309
310   P31 : full_adder port map (A=>A3B0 , B=>A2B1, Cin=>C2_1, S=>P3_1,Cout=>C3_1);
311   P32 : full_adder port map (A=>A1B2, B=>P3_1 ,Cin=>C2, S=>P3_2,Cout=>C3_2);
312   P3_0 : half_adder port map (A=>A0B3, B=>P3_2, S=>P(3),Cout=>C3);
313
314   P41 : full_adder port map (A=>A3B1 , B=>A4B0,Cin=> C3_1, S=>P4_1,Cout=>C4_1);
315   P42 : full_adder port map (A=>A2B2 , B=>P4_1, Cin=>C3_2, S=>P4_2,Cout=>C4_2);
316   P4_0 : full_adder port map (A=>A1B3 , B=>P4_2, Cin=>C3, S=>P4_3,Cout=>C4_3);
317   P4 : half_adder port map (A=>A0B4 , B=>P4_3, S=>P(4),Cout=>C4);
318
319   P51 : full_adder port map (A=>A5B0 , B=>A4B1, Cin=>C4_1, S=>P5_1,Cout=>C5_1);
320   P52   : full_adder port map (A=>A3B2 , B=>P5_1, Cin=>C4_2, S=>P5_2,Cout=>C5_2);
321   P5_0 : full_adder port map (A=>A2B3 , B=>P5_2, Cin=>C4_3, S=>P5_3, Cout=> C5_3);
322   P53 :full_adder port map (A=>A1B4 , B=>P5_3, Cin=>C4, S=> P5_4, Cout=> C5_4);
323   P5:  half_adder port map (A=> A0B5 , B=> P5_4, S=>P(5), Cout=> C5);
```

Figure 15 Brauns Multiplier Port Maps 1

6.2.4. Wallace Multiplier code

```
21
22      library IEEE;
23      use IEEE.STD_LOGIC_1164.ALL;
24      use IEEE.STD_LOGIC_ARITH.ALL;
25      use IEEE.STD_LOGIC_UNSIGNED.ALL;
26
27
28      entity Wallace is
29      Port ( A : in  STD_LOGIC_VECTOR (7 downto 0);
30      B : in  STD_LOGIC_VECTOR (7 downto 0);
31             P : out  STD_LOGIC_VECTOR (15 downto 0));
32      end Wallace;
33
34      architecture Behavioral of Wallace is
35      component Full_Adder is
36          Port ( A : in  STD_LOGIC;
37                 B : in  STD_LOGIC;
38               Cin : in  STD_LOGIC;
39                 S : out  STD_LOGIC;
40              Cout : out  STD_LOGIC);
41      end component;
```

Figure 16 Wallace Multiplier Code start.

```
component Half_adder is
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           S : out  STD_LOGIC;
        Cout : out  STD_LOGIC);
end component;

Signal S1,S2,S3,S4,S5,S6,S7,S9,S10,S11,S12,S13,S14,S15,S17,S18,S19,S20,S21,S22,S23,S25,S26,S27,S28,S29,S30,S31,S33,S34,S35,S36,S37,S38,S39
Signal C0,C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11,C12,C13,C14,C15,C16,C17,C18,C19,C20,C21,C22,C23,C24,C25,C26,C27,C28,C29,C30,C31,C32,C33,C34,C
```

Figure 17 Wallace Multiplier Code 2

```
,S39,S41,S42,S43,S44,S45,S46,S47 : std_logic;
34,C35,C36,C37,C38,C39,C40,C41,C42,C43,C44,C45,C46,C47,C48,C49,C50,C51,C52,C53,C54: std_logic;
```

Figure 18 Wallace Multiplier Code 3 & Sum & Carry Signals

```
52
53      signal A0B0 : std_logic;
54      signal A1B0 : std_logic;
55      signal A2B0 : std_logic;
56      signal A3B0 : std_logic;
57      signal A4B0 : std_logic;
58      signal A5B0 : std_logic;
59      signal A6B0 : std_logic;
60      signal A7B0 : std_logic;
61
62      signal A0B1 : std_logic;
63      signal A1B1 : std_logic;
64      signal A2B1 : std_logic;
65      signal A3B1 : std_logic;
66      signal A4B1 : std_logic;
67      signal A5B1 : std_logic;
68      signal A6B1 : std_logic;
69      signal A7B1 : std_logic;
```

Figure 19 Wallace Multiplier Signal Generation 1

```
126     begin
127
128     A0B0 <= A(0) and B(0);
129     A1B0 <= A(1) and B(0);
130     A2B0 <= A(2) and B(0);
131     A3B0 <= A(3) and B(0);
132     A4B0 <= A(4) and B(0);
133     A5B0 <= A(5) and B(0);
134     A6B0 <= A(6) and B(0);
135     A7B0 <= A(7) and B(0);
136
137     A0B1 <= A(0) and B(1);
138     A1B1 <= A(1) and B(1);
139     A2B1 <= A(2) and B(1);
140     A3B1 <= A(3) and B(1);
141     A4B1 <= A(4) and B(1);
142     A5B1 <= A(5) and B(1);
143     A6B1 <= A(6) and B(1);
144     A7B1 <= A(7) and B(1);
```

Figure 20 Wallace Multiplier Partial Products 1

```
200     --STAGE 1
201     P0: P(0)<=A(0) and B(0) ;
202
203     HA1: Half_adder port map(A=>A1B0,B=>A0B1,S=>P(1),Cout=>C0);
204     FA1: Full_Adder port map(A=>A2B0,B=>A1B1,Cin=>A0B2,S=>S1,Cout=>C1);
205     FA2: Full_Adder port map(A=>A3B0,B=>A2B1,Cin=>A1B2,S=>S2,Cout=>C2);
206     FA3: Full_Adder port map(A=>A4B0,B=>A3B1,Cin=>A2B2,S=>S3,Cout=>C3);
207     FA4: Full_Adder port map(A=>A5B0,B=>A4B1,Cin=>A3B2,S=>S4,Cout=>C4);
208     FA5: Full_Adder port map(A=>A6B0,B=>A5B1,Cin=>A4B2,S=>S5,Cout=>C5);
209     FA6: Full_Adder port map(A=>A7B0,B=>A6B1,Cin=>A5B2,S=>S6,Cout=>C6);
210     HA2: Half_adder port map(A=>A7B1,B=>A6B2,S=>S7,Cout=>C7);
211
212     --STAGE 2
213     HA3: Half_adder port map(A=>S1,B=>C0,S=>P(2),Cout=>C8);
214     FA7: Full_Adder port map(A=>C1,B=>A0B3,Cin=>S2,S=>S9,Cout=>C9);
215     FA8: Full_Adder port map(A=>A1B3,B=>S3,Cin=>C2,S=>S10,Cout=>C10);
216     FA9: Full_Adder port map(A=>A2B3,B=>S4,Cin=>C3,S=>S11,Cout=>C11);
217     FA10: Full_Adder port map(A=>A3B3,B=>S5,Cin=>C4,S=>S12,Cout=>C12);
218     FA11: Full_Adder port map(A=>A4B3,B=>S6,Cin=>C5,S=>S13,Cout=>C13);
219     FA12: Full_Adder port map(A=>A5B3,B=>S7,Cin=>C6,S=>S14,Cout=>C14);
220     FA13: Full_Adder port map(A=>A7B2,B=>A6B3,Cin=>C7,S=>S15,Cout=>C15);
```

Figure 21 Wallace Multiplier Stage 1 & 2 Reduction

```
262     --FINAL STAGE
263     HA8: Half_adder port map(A=>S41,B=>C40,S=>P(7),Cout=>C48);
264     FA42: Full_Adder port map(A=>S42,B=>C41,Cin=>C48,S=>P(8),Cout=>C49);
265     FA43: Full_Adder port map(A=>S43,B=>C42,Cin=>C49,S=>P(9),Cout=>C50);
266     FA44: Full_Adder port map(A=>S44,B=>C43,Cin=>C50,S=>P(10),Cout=>C51);
267     FA45: Full_Adder port map(A=>S45,B=>C44,Cin=>C51,S=>P(11),Cout=>C52);
268     FA46: Full_Adder port map(A=>S46,B=>C45,Cin=>C52,S=>P(12),Cout=>C53);
269     FA47: Full_Adder port map(A=>S47,B=>C46,Cin=>C53,S=>P(13),Cout=>C54);
270     FA48: Full_Adder port map(A=>A7B7,B=>C47,Cin=>C54,S=>P(14),Cout=>P(15));
271
272
273     end Behavioral;
```

Figure 22 Wallace Multiplier Final Stage Reduction

Shown in the pictures above is the code for both the Brauns and Wallace multiplier, the figures show the declaration of the multiplier, the full and half adder declaration, partial product storage/internal signals, partial products using AND gates, and finally the Port Maps. Additional figures seen in Appendix. Brauns figures 71-82. Wallace figures 83- 90.

6.3. Brauns Multiplier Device

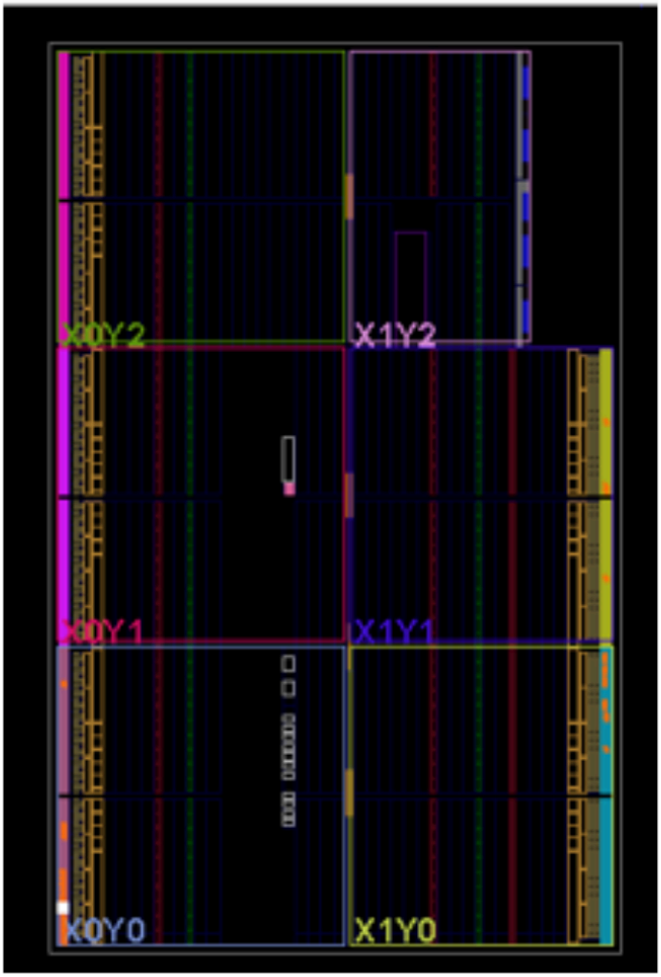


Figure 23 Brauns Multiplier Device

This Figure is to show the Brauns multiplier Device from the Vivado software.

6.4. Wallace Multiplier Device

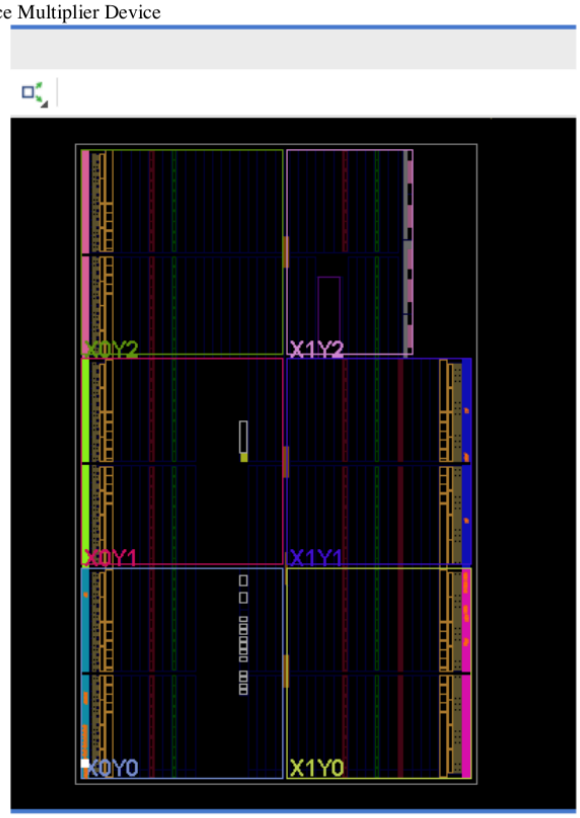


Figure 24 Wallace Multiplier Device

This Figure is to show the Wallace multiplier Device from Vivado Software.

6.5. Timing Diagrams and Schematics

The Brauns and Wallace multipliers both have the shortest delay to P0, P15 and the longest delay to P15. This is measured by adding up the delays possible. The delay will amount to the number of full and half adders within the two multipliers. The delays will only account for the very first AND gate that is used to gain the P value, all adders will then be accounted for regardless of being full or half adders. The shortest delay for P0 within both multipliers is 1ns.

Shortest Brauns multiplier delay for P15.

$$1ns + 7 \times (2ns) = 15ns$$

Longest Brauns multiplier delay for P15.

$$1ns + 20 \times (2ns) = 41ns$$

Shortest Wallace multiplier delay for P15.

$$1ns + 7 \times (2ns) = 15ns$$

Longest Wallace multiplier delay for P15

$$1ns + 14 \times (2ns) = 29ns$$

5

6.5.1. Brauns Multiplier Timing, Longest and Shortest Delay Diagrams & Combinational Delays

Package | Device | Schematic | **Timing.xdc**

C:/Users/robbi/Brauns_Multiplier/Brauns_Multiplier.srcs/constrs_1/new/Timing.xdc

```
1  create_clock -period 45.000 -name virtual_clk
2
3  # Set delays based on analytical calculations
4  set_input_delay -clock virtual_clk -max 1.000 [get_ports A*]
5  set_input_delay -clock virtual_clk -max 1.000 [get_ports B*]
6  set_output_delay -clock virtual_clk -max 2.000 [get_ports P*]
7
8  # Set max delay for the critical path (longest delay)
9  set_max_delay -from [get_ports A*] -to [get_ports P*] 41.000
10
```

Figure 25 Brauns Multiplier Timing Constraint

Figure to show code used for Brauns multiplier Timing constraint. 6

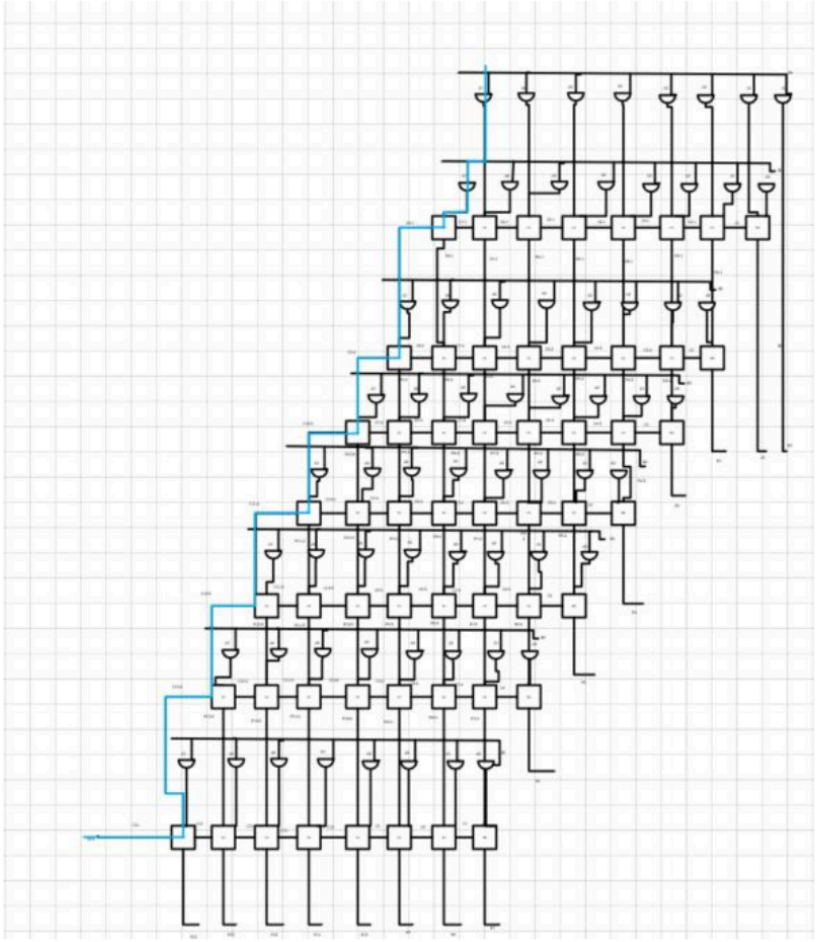


Figure 26 Brauns Multiplier Shortest Delay Diagram

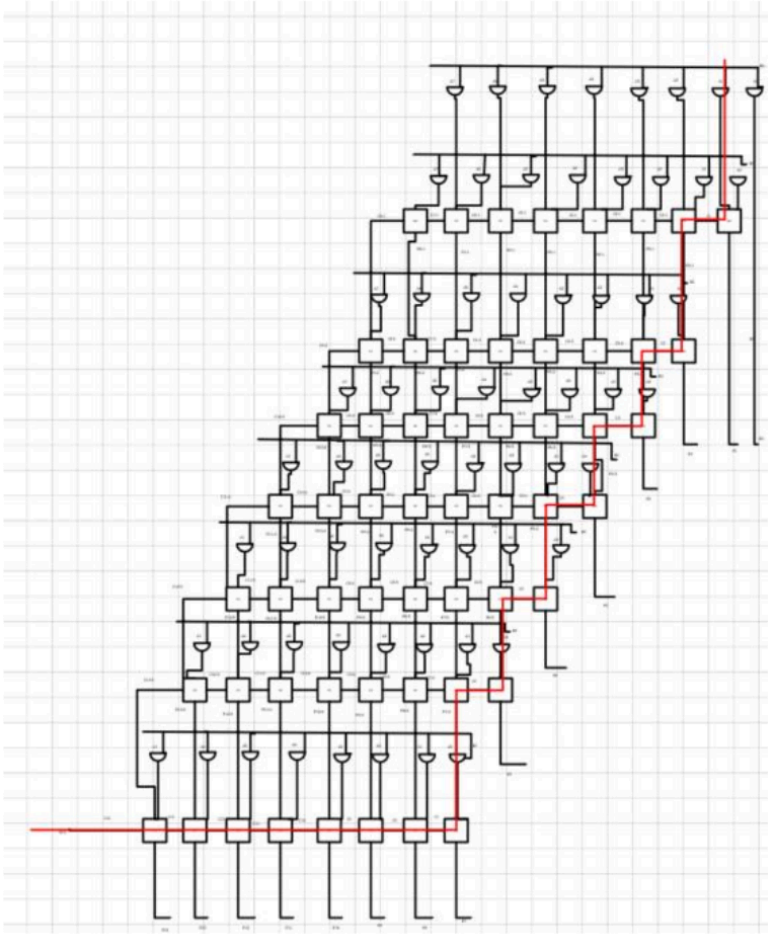


Figure 27 Brauns Multiplier Longest Delay Diagram

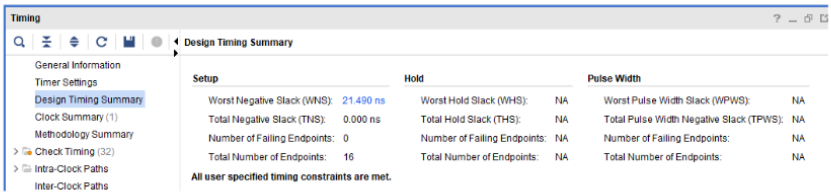


Figure 28 Brauns Multiplier Synthesis Timing Summary

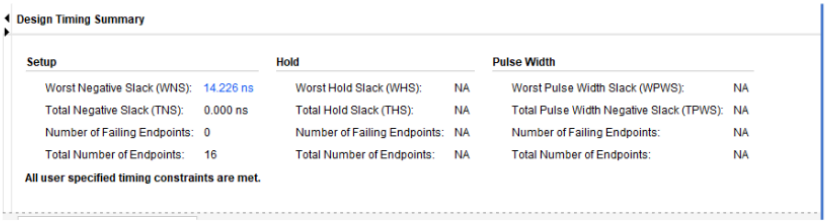


Figure 29 Brauns Multiplier Implementation Timing Summary

Figures showing the Timing summary of the Synthesis and implementation of the Brauns multiplier.

Combinational Delays

From Port	To Port	Max Delay	Max Process Corner
A[0]	P[0]	11.487	SLOW
A[0]	P[1]	11.021	SLOW
A[0]	P[2]	10.506	SLOW
A[0]	P[3]	11.057	SLOW
A[0]	P[4]	13.323	SLOW
A[0]	P[5]	14.492	SLOW
A[0]	P[6]	15.401	SLOW
A[0]	P[7]	16.110	SLOW
A[0]	P[8]	17.200	SLOW
A[0]	P[9]	19.225	SLOW
A[0]	P[10]	19.746	SLOW
A[0]	P[11]	20.718	SLOW
A[0]	P[12]	21.496	SLOW
A[0]	P[13]	22.530	SLOW
A[0]	P[14]	22.131	SLOW
A[0]	P[15]	22.739	SLOW
A[1]	P[1]	11.764	SLOW
A[1]	P[2]	11.324	SLOW
A[1]	P[3]	11.841	SLOW

Figure 30 Brauns Multiplier Combinational Delay 1

Figures to show combinational delays of the Brauns multiplier. Additional combinational delays within appendix figures 91-100.

6.5.2. Wallace Multiplier Timing Summary, Longest and Shortest delays & Combinational Delays

Untitled 1 | Timing.xdc (2)

C:/Users/robbi/Wallace Multiplier/Wallace Multiplier.srcs/constrs_1/new/Timing.xdc

```
1 create_clock -period 32.000 -name virtual_clk
2
3 # Set delays based on analytical calculations
4 set_input_delay -clock virtual_clk -max 1.000 [get_ports A*]
5 set_input_delay -clock virtual_clk -max 1.000 [get_ports B*]
6 set_output_delay -clock virtual_clk -max 2.000 [get_ports P*]
7
8 # Set max delay for the critical path (longest delay)
9 set_max_delay -from [get_ports A*] -to [get_ports P*] 29.000
```

Figure 31 Wallace Multiplier Timing Constraint

Figure to show constraint code for Wallace multiplier.

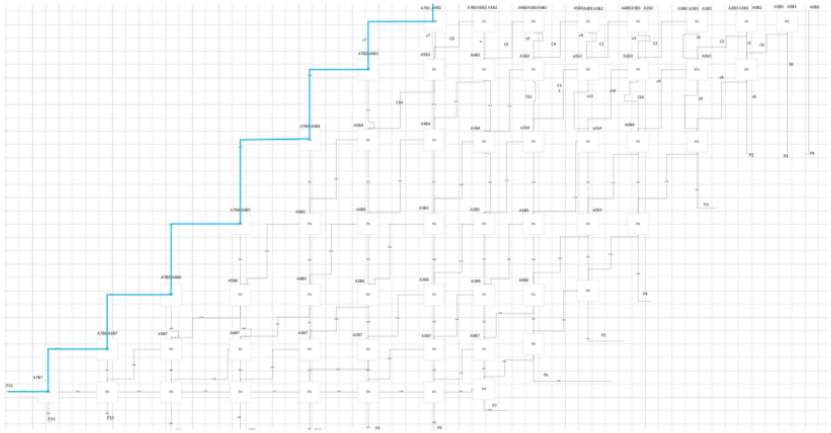


Figure 32 Wallace Multiplier Shortest Delay Diagram

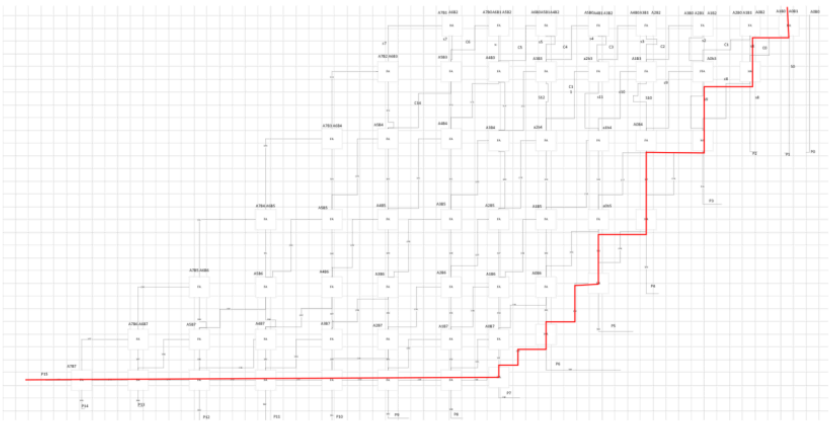


Figure 33 Wallace Multiplier Longest Delay Diagram

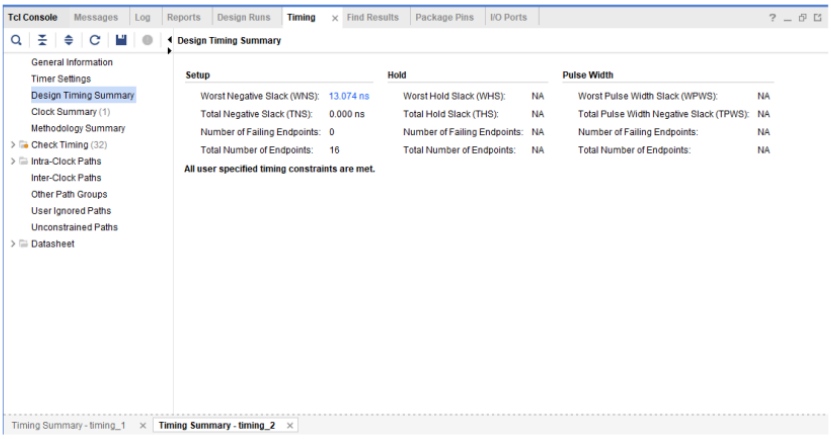


Figure 34 Wallace Multiplier Synthesis Timing Summary

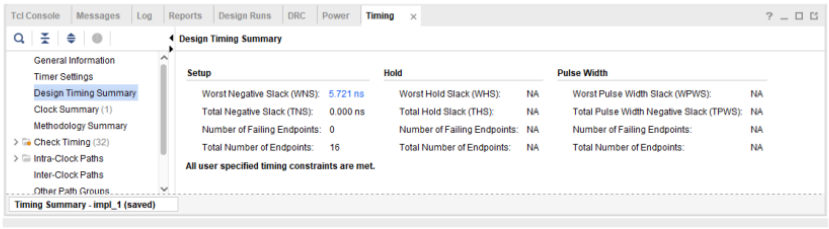


Figure 35 Wallace Multiplier Implementation Timing Summary

Figures show Wallace multiplier synthesis and implementation timing summary.

Combinational Delays

From Port	To Port	Max Delay	Max Process Corner
A[0]	P[0]	11.453	SLOW
A[0]	P[1]	10.991	SLOW
A[0]	P[2]	10.587	SLOW
A[0]	P[3]	11.292	SLOW
A[0]	P[4]	12.068	SLOW
A[0]	P[5]	13.629	SLOW
A[0]	P[6]	14.003	SLOW
A[0]	P[7]	14.578	SLOW
A[0]	P[8]	14.525	SLOW
A[0]	P[9]	16.017	SLOW
A[0]	P[10]	16.711	SLOW
A[0]	P[11]	16.654	SLOW
A[0]	P[12]	18.185	SLOW
A[0]	P[13]	18.123	SLOW
A[0]	P[14]	18.450	SLOW
A[0]	P[15]	18.898	SLOW
A[1]	P[1]	11.007	SLOW
A[1]	P[2]	10.623	SLOW
A[1]	P[3]	11.564	SLOW

Figure 36 Wallace Multiplier Combinational Delays 1

Figures showing additional combinational delays in Appendix Figures 101-110.

8

6.6. Design Screenshots and Pin Allocation

6.6.1. Brauns Multiplier Designs & Pin allocation

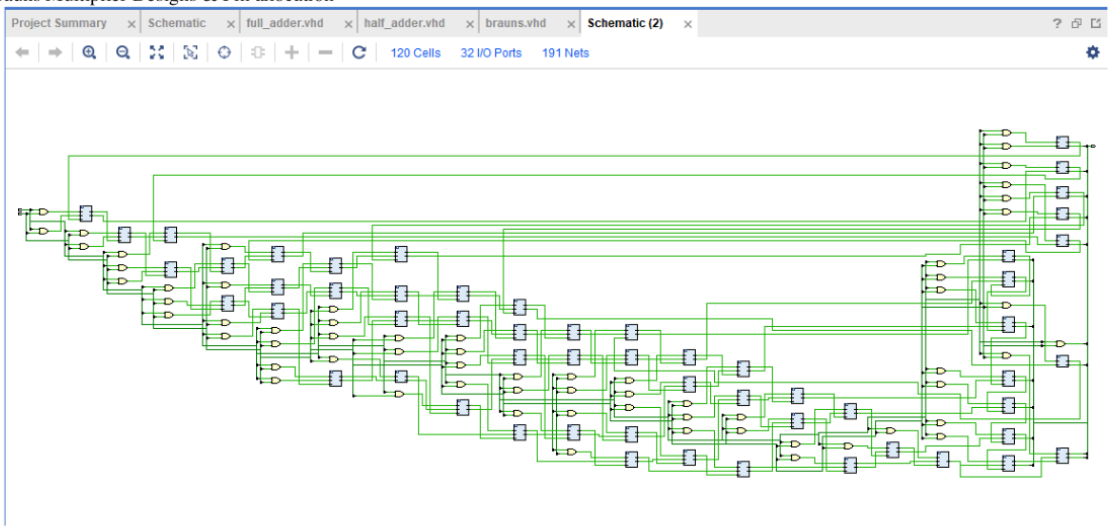


Figure 37 RTL Design Brauns

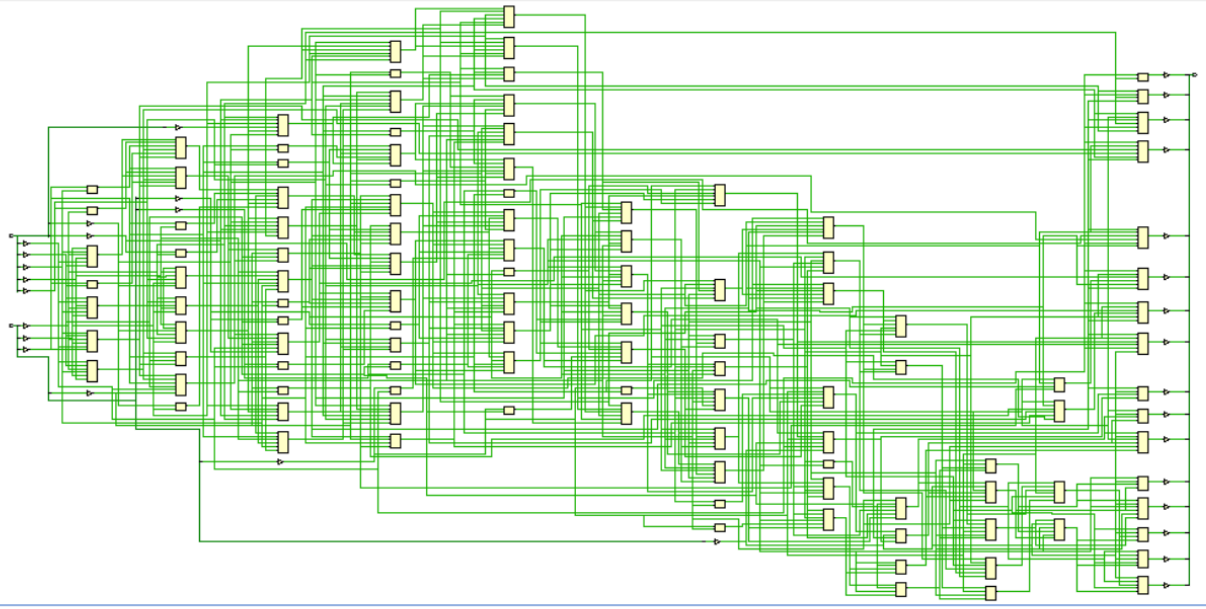
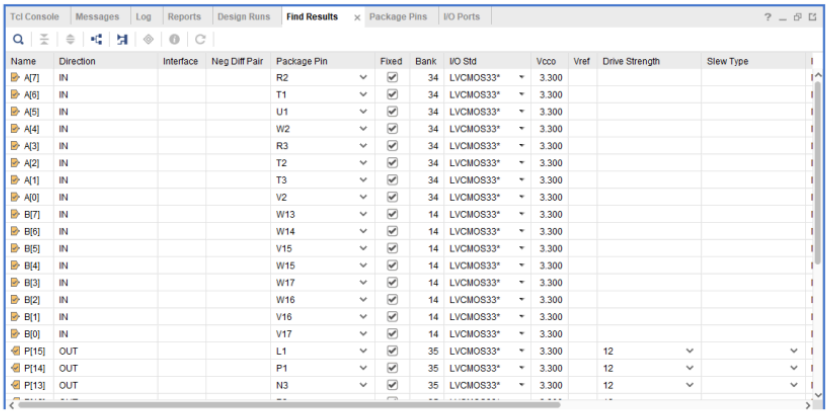


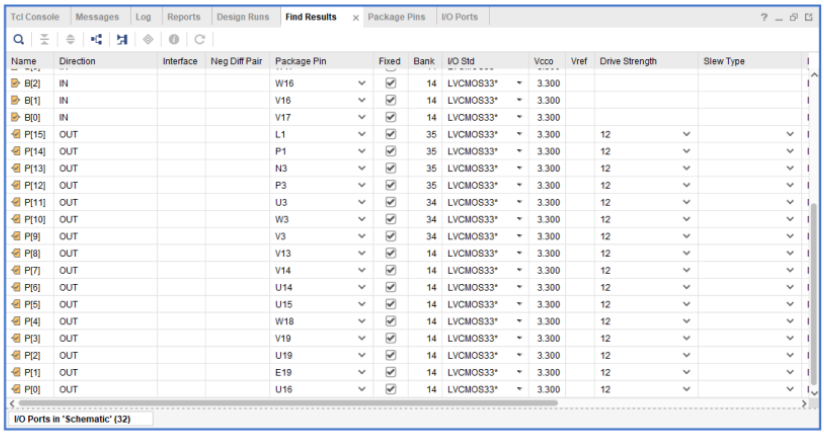
Figure 38 Implementation Design Brauns

9



Name	Direction	Interface	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type
A[7]	IN			R2	☑	34	LVCMOS33*	3.300			
A[6]	IN			T1	☑	34	LVCMOS33*	3.300			
A[5]	IN			U1	☑	34	LVCMOS33*	3.300			
A[4]	IN			W2	☑	34	LVCMOS33*	3.300			
A[3]	IN			R3	☑	34	LVCMOS33*	3.300			
A[2]	IN			T2	☑	34	LVCMOS33*	3.300			
A[1]	IN			T3	☑	34	LVCMOS33*	3.300			
A[0]	IN			V2	☑	34	LVCMOS33*	3.300			
B[7]	IN			W13	☑	14	LVCMOS33*	3.300			
B[6]	IN			W14	☑	14	LVCMOS33*	3.300			
B[5]	IN			V15	☑	14	LVCMOS33*	3.300			
B[4]	IN			W15	☑	14	LVCMOS33*	3.300			
B[3]	IN			W17	☑	14	LVCMOS33*	3.300			
B[2]	IN			W16	☑	14	LVCMOS33*	3.300			
B[1]	IN			V16	☑	14	LVCMOS33*	3.300			
B[0]	IN			V17	☑	14	LVCMOS33*	3.300			
P[15]	OUT			L1	☑	35	LVCMOS33*	3.300		12	
P[14]	OUT			P1	☑	35	LVCMOS33*	3.300		12	
P[13]	OUT			N3	☑	35	LVCMOS33*	3.300		12	

Figure 39 Pin Allocation Brauns



Name	Direction	Interface	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type
B[2]	IN			W16	☑	14	LVCMOS33*	3.300			
B[1]	IN			V16	☑	14	LVCMOS33*	3.300			
B[0]	IN			V17	☑	14	LVCMOS33*	3.300			
P[15]	OUT			L1	☑	35	LVCMOS33*	3.300		12	
P[14]	OUT			P1	☑	35	LVCMOS33*	3.300		12	
P[13]	OUT			N3	☑	35	LVCMOS33*	3.300		12	
P[12]	OUT			P3	☑	35	LVCMOS33*	3.300		12	
P[11]	OUT			U3	☑	34	LVCMOS33*	3.300		12	
P[10]	OUT			W3	☑	34	LVCMOS33*	3.300		12	
P[9]	OUT			V3	☑	34	LVCMOS33*	3.300		12	
P[8]	OUT			V13	☑	14	LVCMOS33*	3.300		12	
P[7]	OUT			V14	☑	14	LVCMOS33*	3.300		12	
P[6]	OUT			U14	☑	14	LVCMOS33*	3.300		12	
P[5]	OUT			U15	☑	14	LVCMOS33*	3.300		12	
P[4]	OUT			W18	☑	14	LVCMOS33*	3.300		12	
P[3]	OUT			V19	☑	14	LVCMOS33*	3.300		12	
P[2]	OUT			U19	☑	14	LVCMOS33*	3.300		12	
P[1]	OUT			E19	☑	14	LVCMOS33*	3.300		12	
P[0]	OUT			U16	☑	14	LVCMOS33*	3.300		12	

Figure 40 Pin Allocation Brauns 2



Figure 41 Brauns Multiplier I/O Planning

Figures show the Brauns multiplier designs and Pin Allocation with the I/O planning.

6.6.2. Wallace Multiplier Designs & Pin Allocation

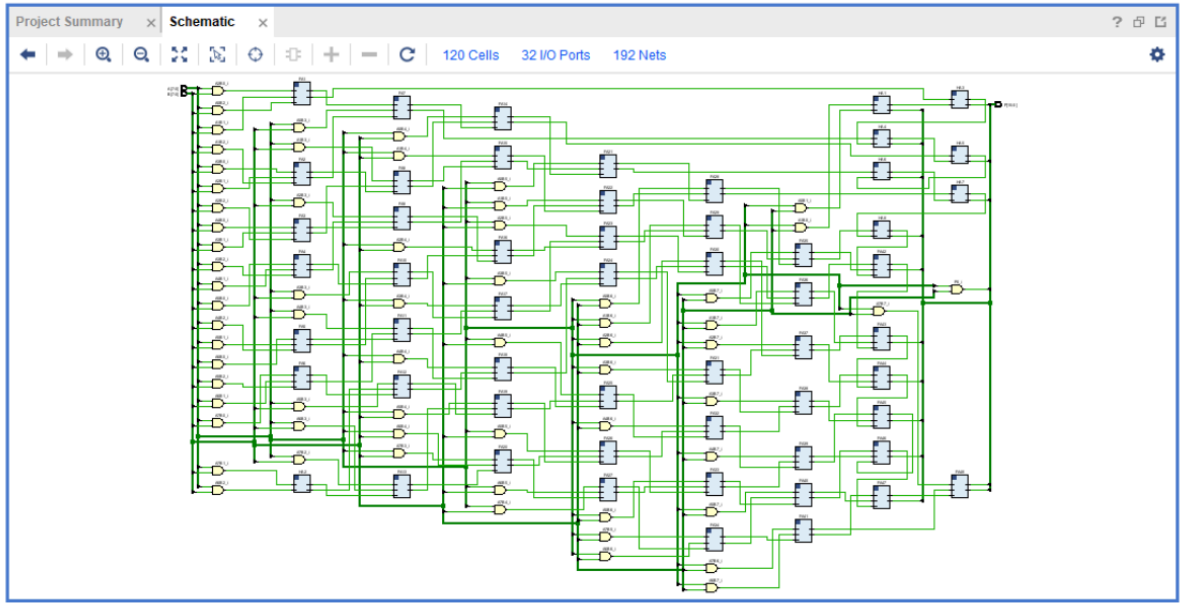


Figure 42 Wallace Multiplier RTL Design

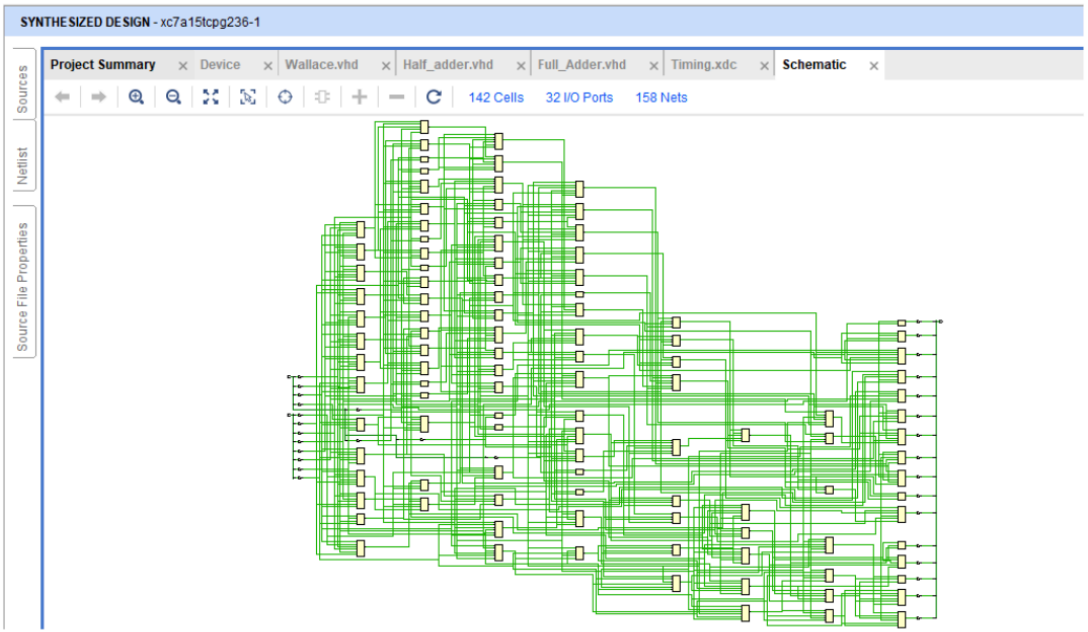
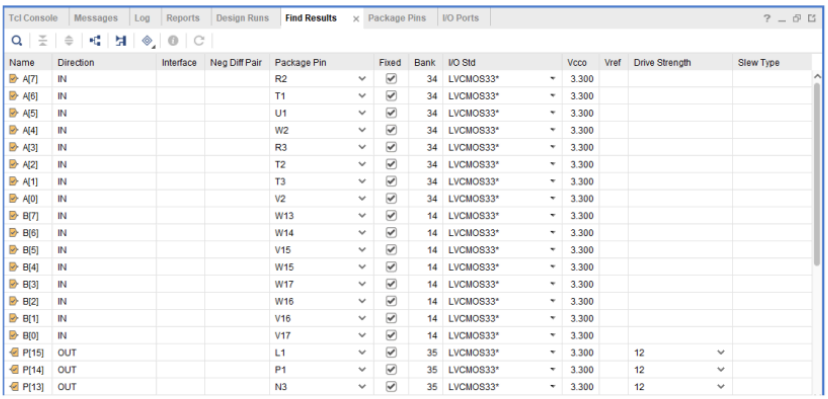
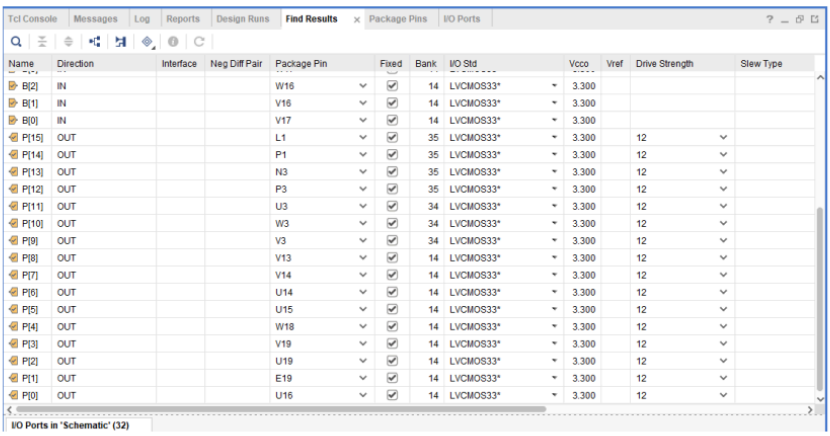


Figure 43 Wallace Multiplier Implemented Design



Name	Direction	Interface	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type
A[7]	IN			R2	☑	34	LVCMOS33*	3.300			
A[6]	IN			T1	☑	34	LVCMOS33*	3.300			
A[5]	IN			U1	☑	34	LVCMOS33*	3.300			
A[4]	IN			W2	☑	34	LVCMOS33*	3.300			
A[3]	IN			R3	☑	34	LVCMOS33*	3.300			
A[2]	IN			T2	☑	34	LVCMOS33*	3.300			
A[1]	IN			T3	☑	34	LVCMOS33*	3.300			
A[0]	IN			V2	☑	34	LVCMOS33*	3.300			
B[7]	IN			W13	☑	14	LVCMOS33*	3.300			
B[6]	IN			W14	☑	14	LVCMOS33*	3.300			
B[5]	IN			V15	☑	14	LVCMOS33*	3.300			
B[4]	IN			W15	☑	14	LVCMOS33*	3.300			
B[3]	IN			W17	☑	14	LVCMOS33*	3.300			
B[2]	IN			W16	☑	14	LVCMOS33*	3.300			
B[1]	IN			V16	☑	14	LVCMOS33*	3.300			
B[0]	IN			V17	☑	14	LVCMOS33*	3.300			
P[15]	OUT			L1	☑	35	LVCMOS33*	3.300		12	
P[14]	OUT			P1	☑	35	LVCMOS33*	3.300		12	
P[13]	OUT			N3	☑	35	LVCMOS33*	3.300		12	

Figure 44 Wallace Multiplier Pin Allocation



Name	Direction	Interface	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type
B[2]	IN			W16	☑	14	LVCMOS33*	3.300			
B[1]	IN			V16	☑	14	LVCMOS33*	3.300			
B[0]	IN			V17	☑	14	LVCMOS33*	3.300			
P[15]	OUT			L1	☑	35	LVCMOS33*	3.300		12	
P[14]	OUT			P1	☑	35	LVCMOS33*	3.300		12	
P[13]	OUT			N3	☑	35	LVCMOS33*	3.300		12	
P[12]	OUT			P3	☑	35	LVCMOS33*	3.300		12	
P[11]	OUT			U3	☑	34	LVCMOS33*	3.300		12	
P[10]	OUT			W3	☑	34	LVCMOS33*	3.300		12	
P[9]	OUT			V3	☑	34	LVCMOS33*	3.300		12	
P[8]	OUT			V13	☑	14	LVCMOS33*	3.300		12	
P[7]	OUT			V14	☑	14	LVCMOS33*	3.300		12	
P[6]	OUT			U14	☑	14	LVCMOS33*	3.300		12	
P[5]	OUT			U15	☑	14	LVCMOS33*	3.300		12	
P[4]	OUT			W18	☑	14	LVCMOS33*	3.300		12	
P[3]	OUT			V19	☑	14	LVCMOS33*	3.300		12	
P[2]	OUT			U19	☑	14	LVCMOS33*	3.300		12	
P[1]	OUT			E19	☑	14	LVCMOS33*	3.300		12	
P[0]	OUT			U16	☑	14	LVCMOS33*	3.300		12	

Figure 45 Wallace Multiplier Pin Allocation 2

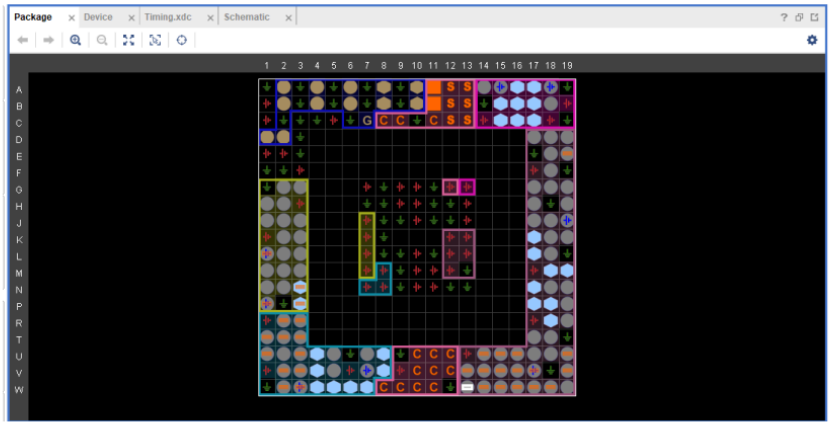


Figure 46 Wallace Multiplier I/O Planning

Figures to show the Wallace multiplier Designs with Pin Allocation and I/O planning.

7. Conclusion

7.1. Simulation Results

Within the Brauns and Wallace multiplier, both were used with the same hexadecimal values to multiply. These values were AA x AB, FF x FF, EF x EA and finally 18 x 34.

The results were all the same with 100% accuracy with the results chosen, comparing the values of the multiplication to an online hexadecimal calculator, As seen in the figures below.

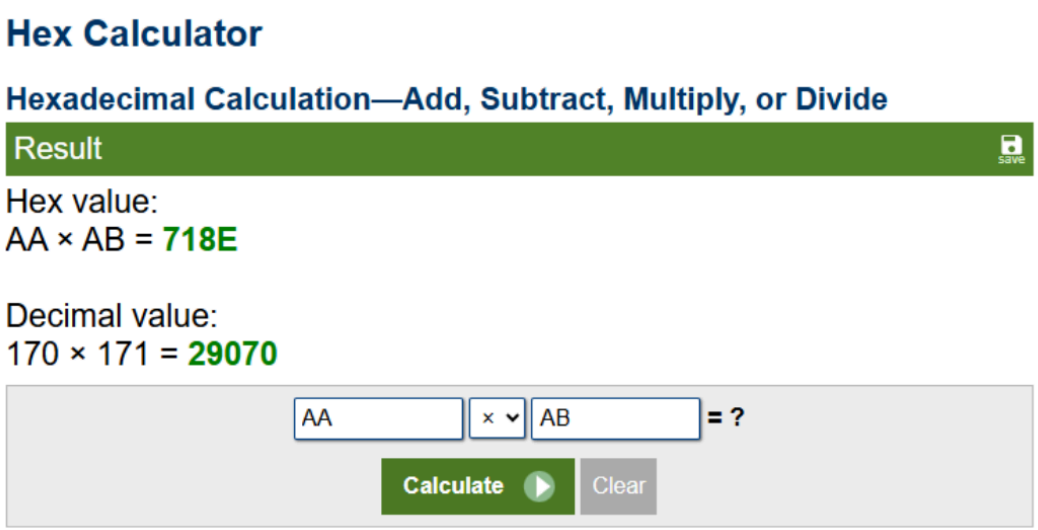


Figure 47 Hex value 1.

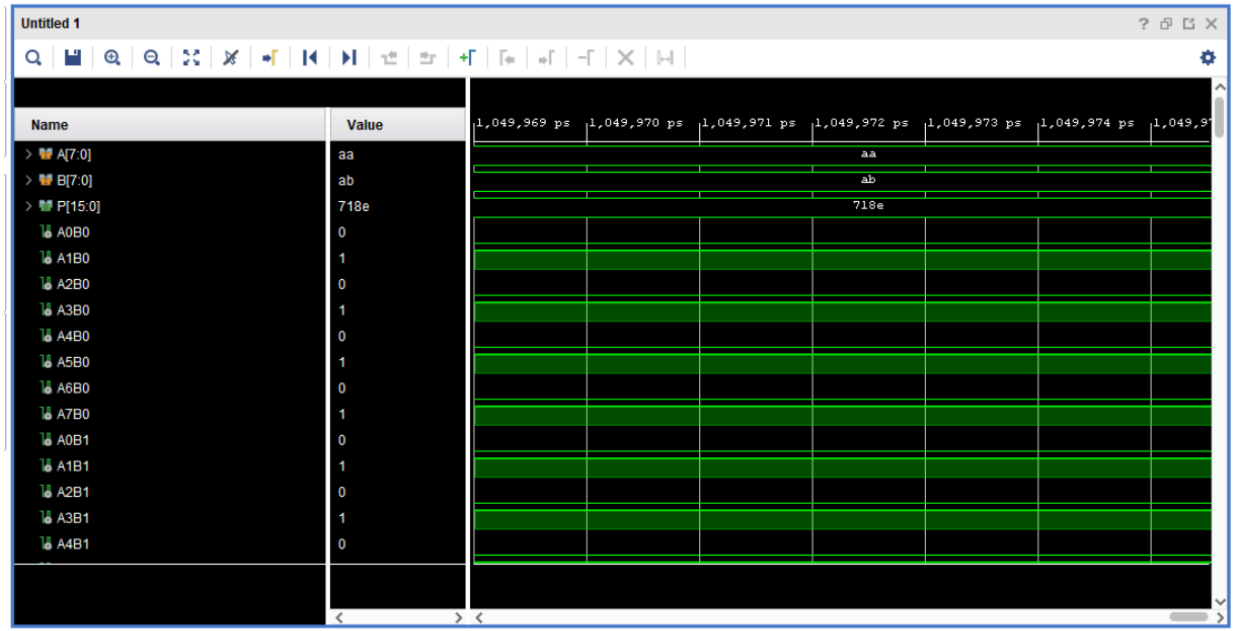


Figure 48 Brauns Simulation of Hex Value 1

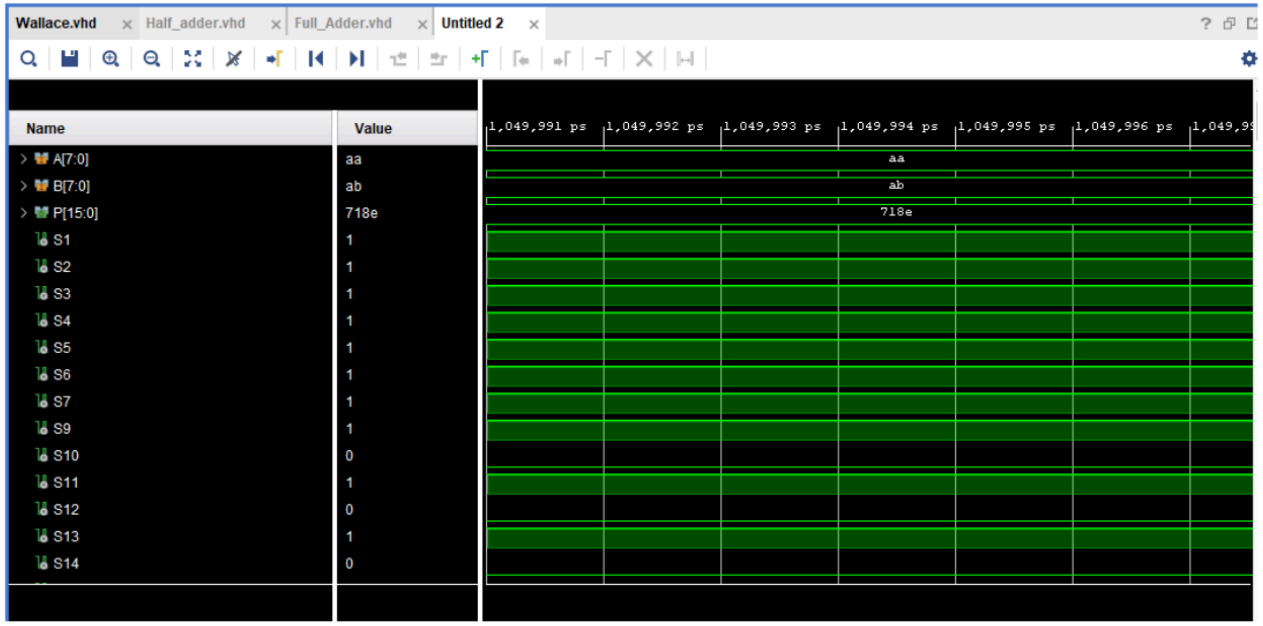


Figure 49 Wallace Simulation of Hex Value 1

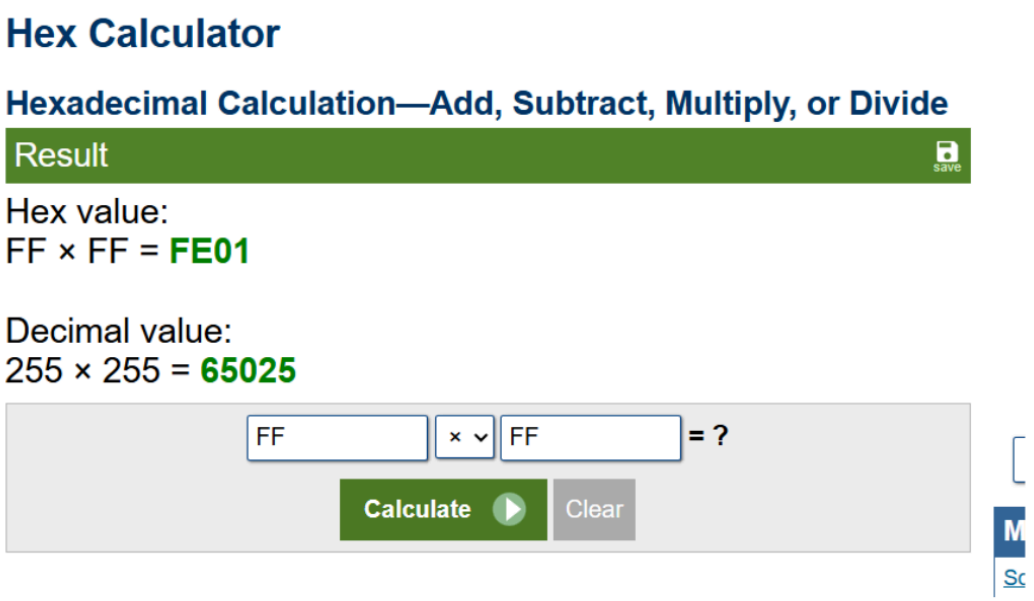


Figure 50 Hex value 2.

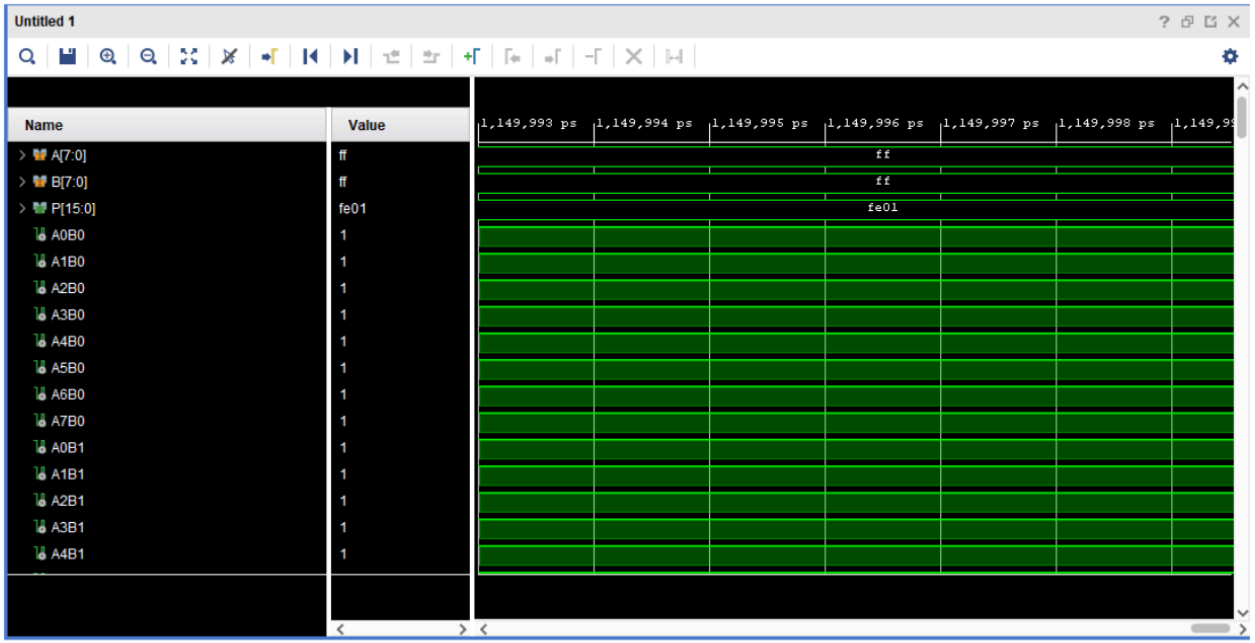


Figure 51 Brauns Multiplier Hex Value 2

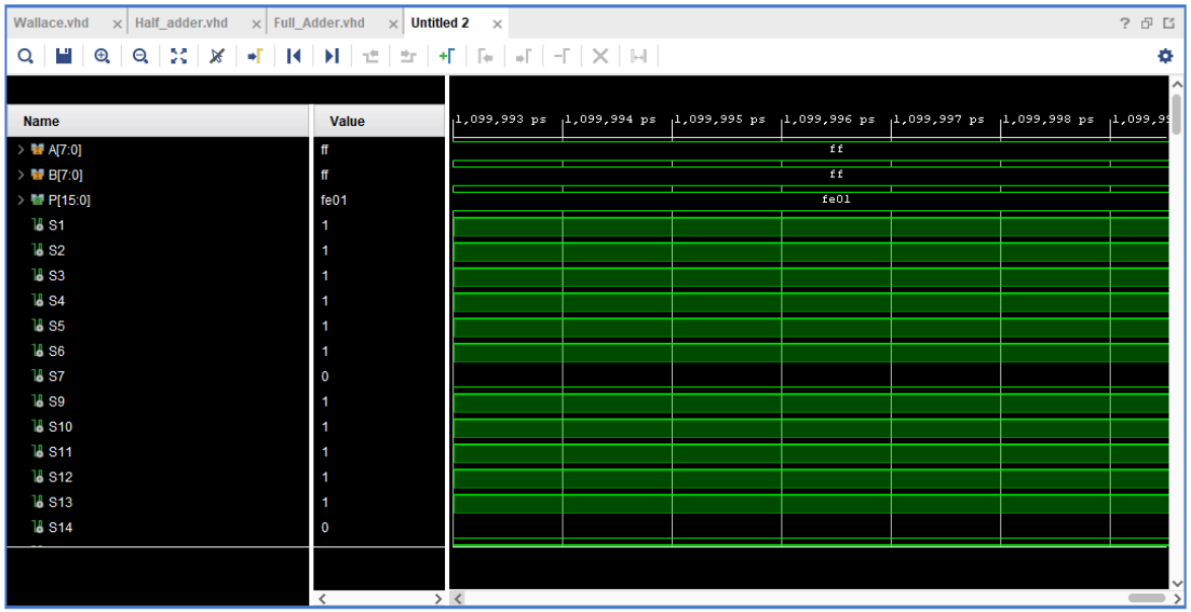


Figure 52 Wallace Simulation of Hex Value 2

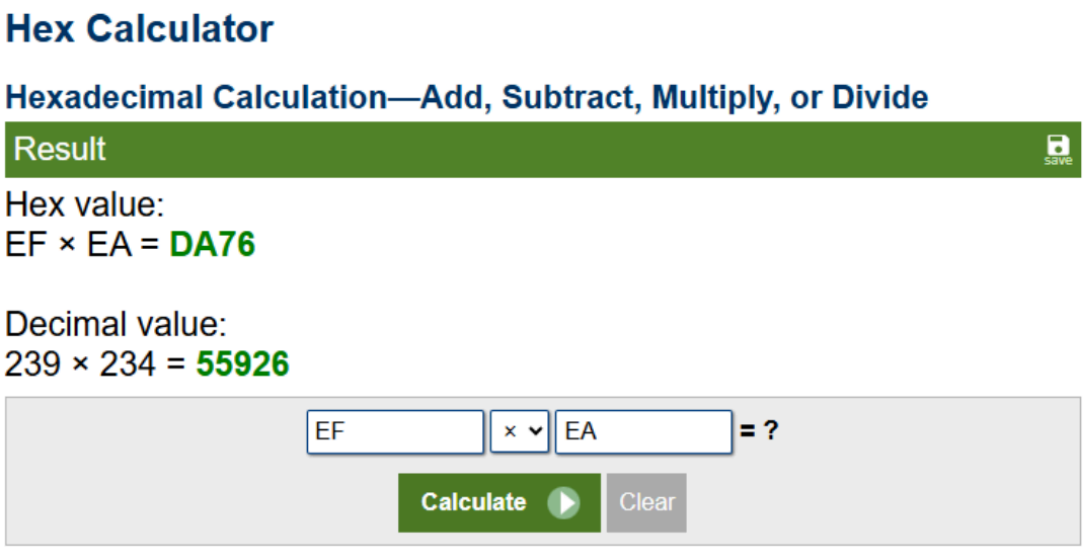


Figure 53 Hex value 3.

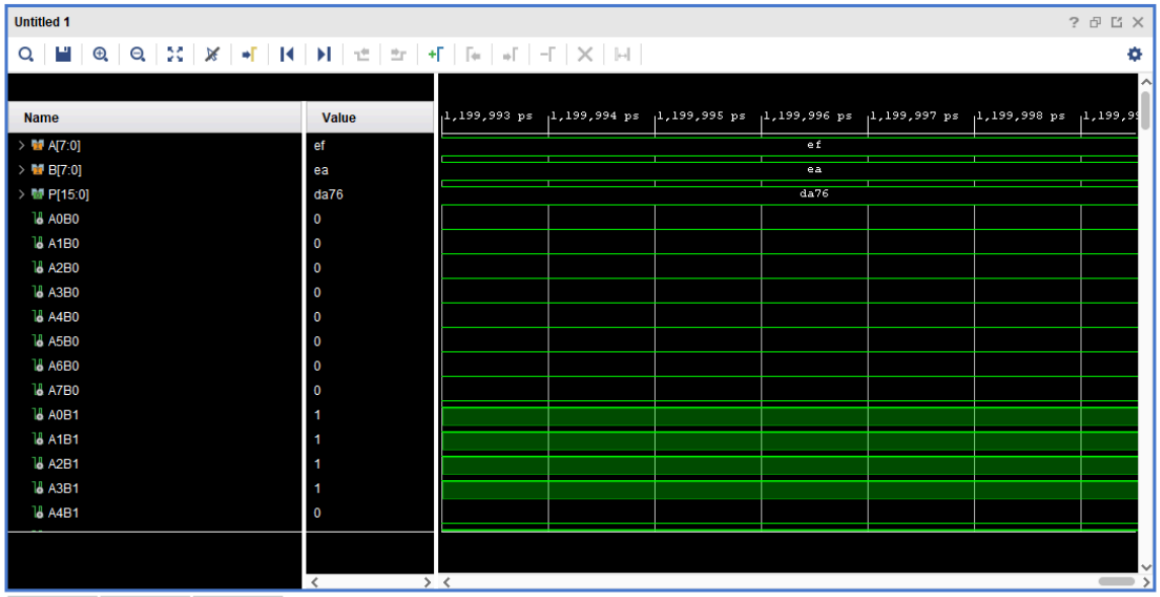


Figure 54 Brauns Simulation of Hex Value 3

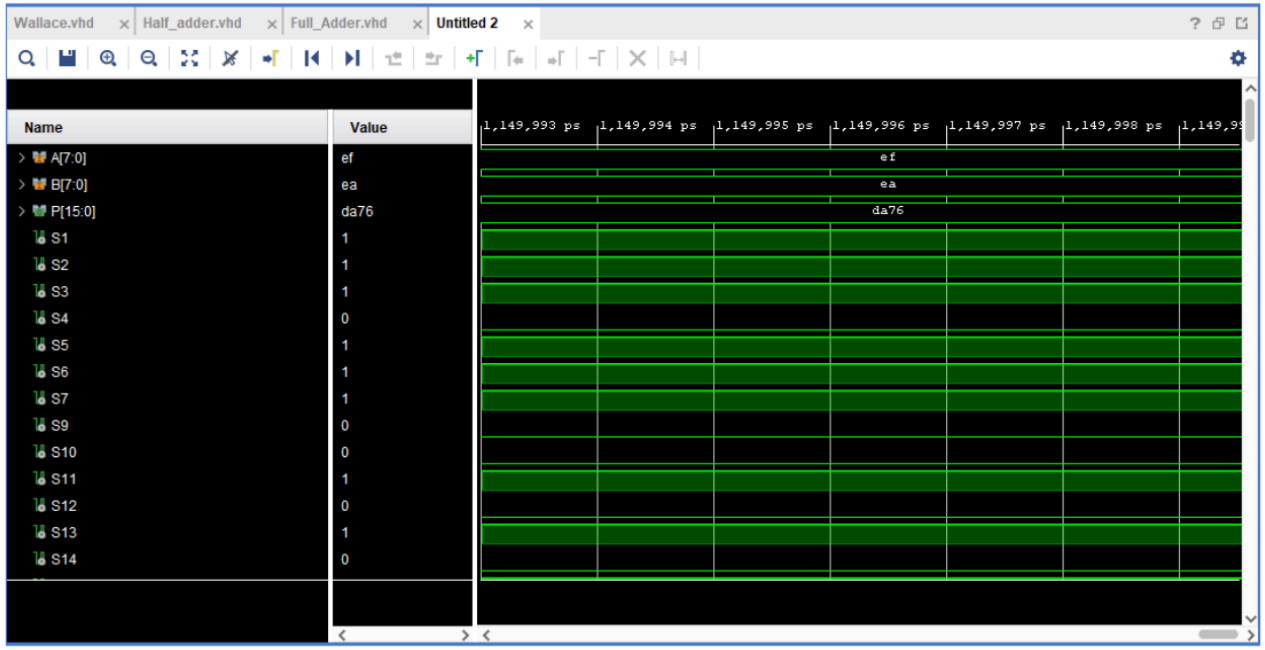


Figure 55 Wallace Simulation of Hex Value 3

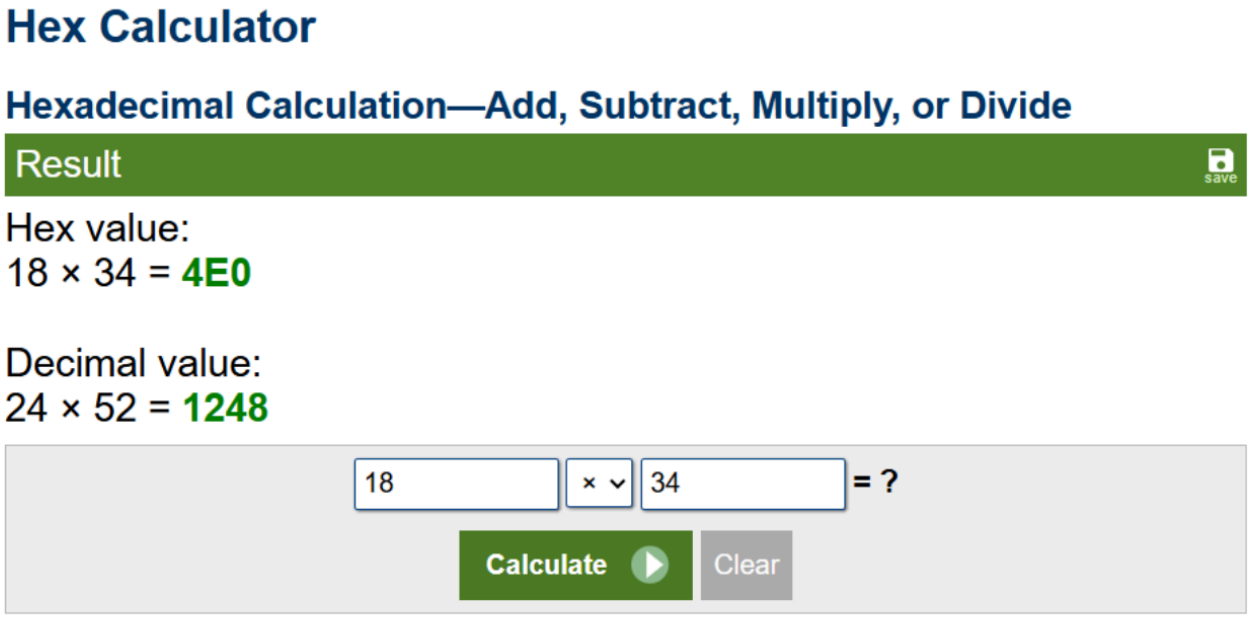


Figure 56 Hex value 4.

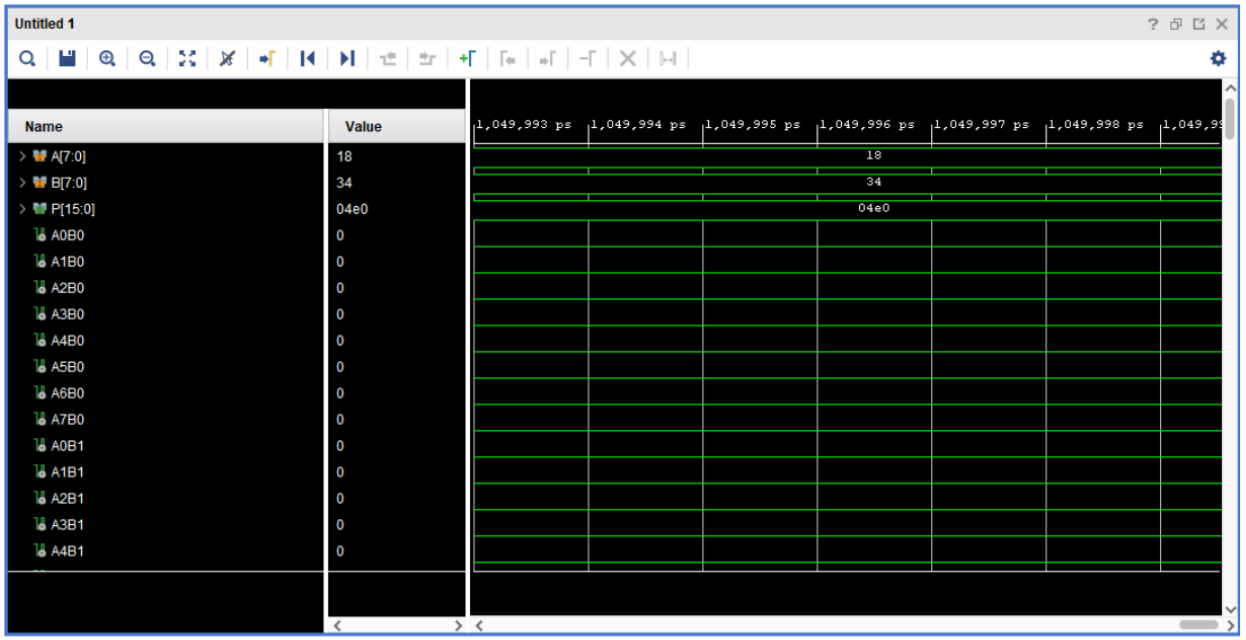


Figure 57 Brauns Simulation of Hex Value 4

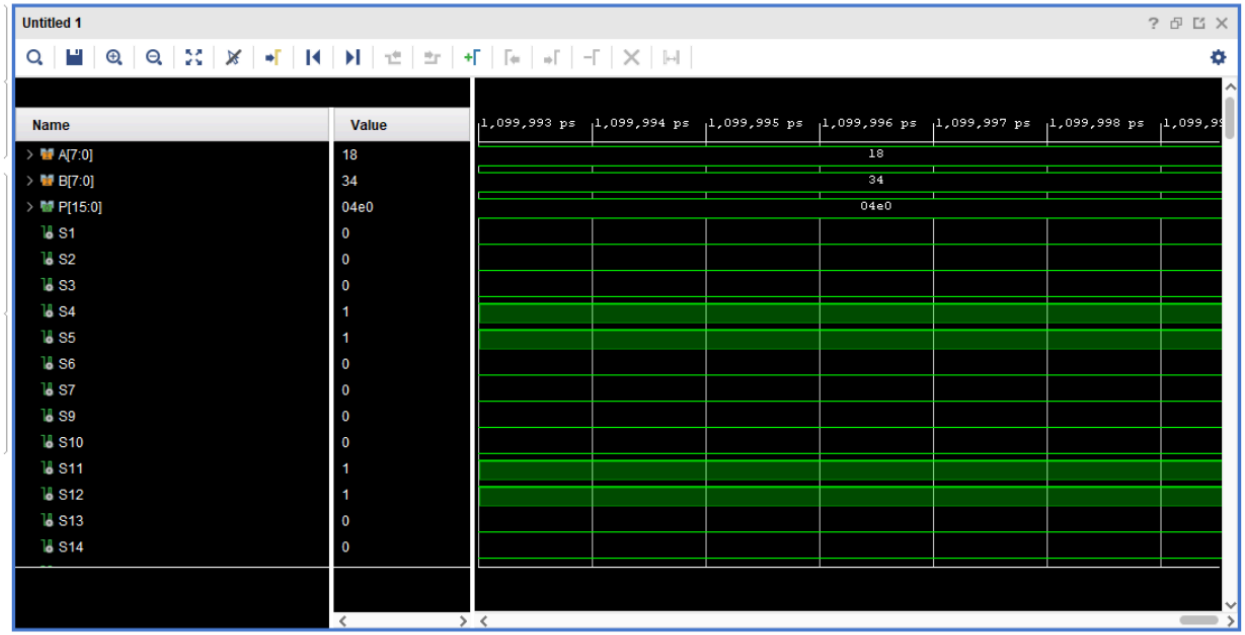


Figure 58 Wallace Simulation of Hex Value 4

7.2. Comparison, Analysis/testing details, and interpretation

Whilst the Brauns and Wallace tree multipliers are both quick and efficient. They both have some differences between them both. In comparison to each other regarding the accuracy, the hexadecimal values that have been chosen, the accuracy is the same as they have both produced the same correct hexadecimal values. The Two multipliers are both fast however the Wallace multiplier has a lower input-output time as the highest within the full list of combinational delays is 20.006ns. However, looking at the Brauns multiplier the highest combinational delay is 23.749ns. A second difference between the two multipliers is the Worst Negative Slack [10]. Worst negative slack means the time taken to get through the critical path [10]. Within the Brauns and Wallace multiplier both WNS are positive. As the WNS needs to be positive or equal to 0 [10]. However, in the case of the two multipliers the Wallace multiplier has a better WNS, this is because it is closer to zero in both synthesis (5.721ns) and implementation (13.074ns). Compared to the Brauns multipliers synthesis (14.226ns) and implementation (21.490ns) WNS the Brauns multiplier is slower by ≈8.416ns within synthesis and ≈ 8.505ns within implementation.

7.2.1. Interpretation of Power utilisation and LUTs

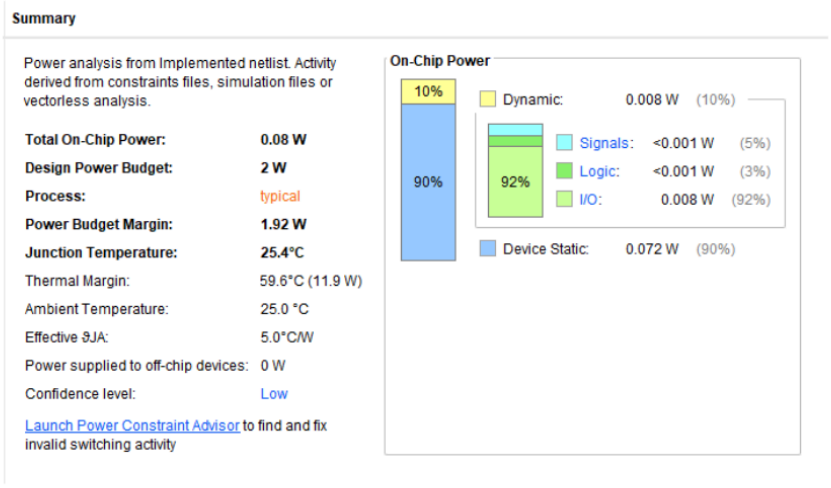


Figure 59 Brauns Multiplier Power Summary

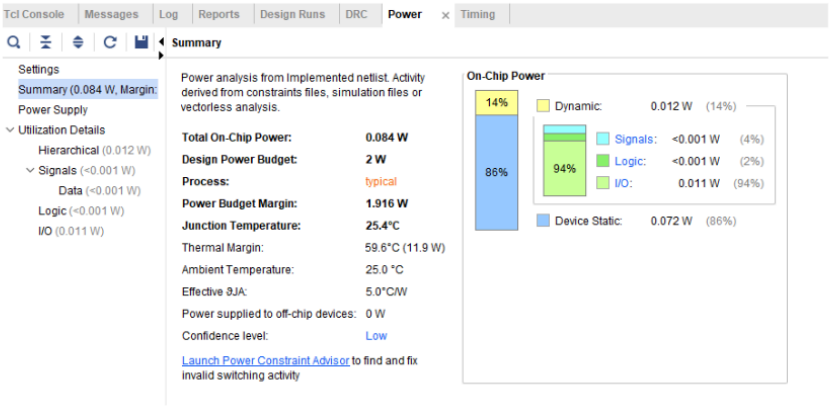


Figure 60 Wallace Multiplier Power Report

Hierarchy

Name	Slice LUTs (20800)	Slice (8150)	LUT as Logic (20800)	Bonded IOB (106)
brauns_multiplier	97	30	97	32

Figure 61 Brauns Multiplier Utilization Hierarchy

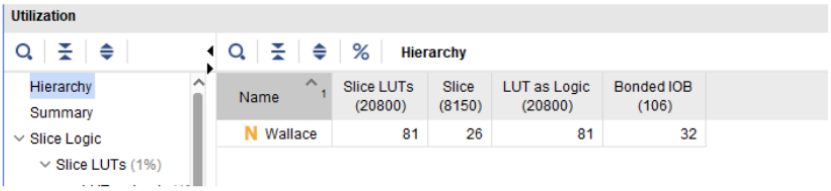


Figure 62 Wallace Multiplier Utilization Hierarchy

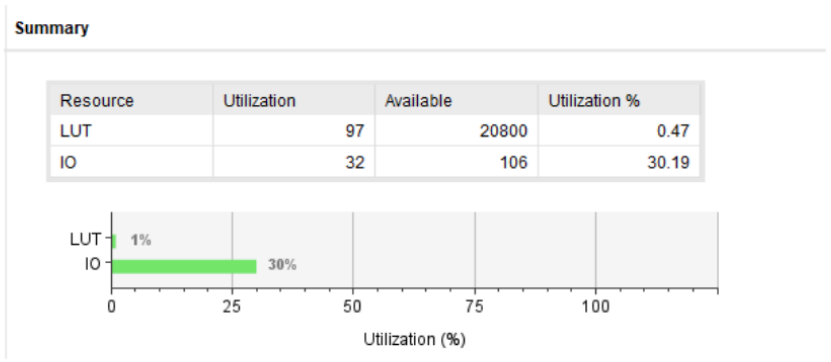


Figure 63 Brauns Multiplier Utilization Summary

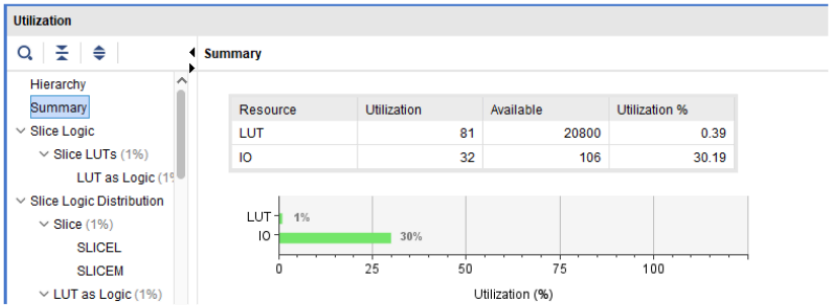


Figure 64 Wallace Multiplier Utilization Summary

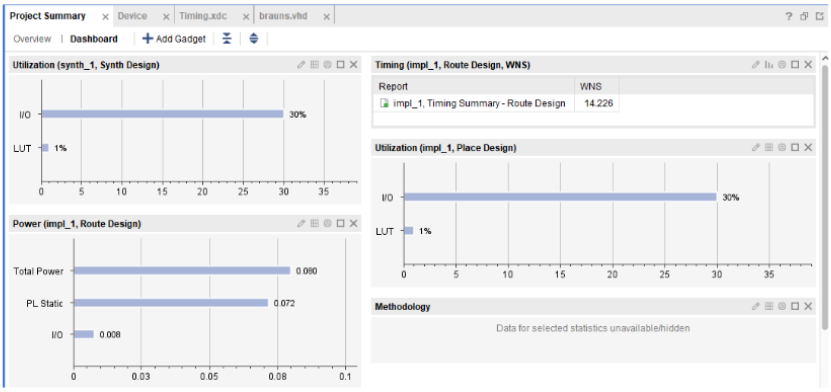


Figure 65 Brauns Multiplier Project Summary

11

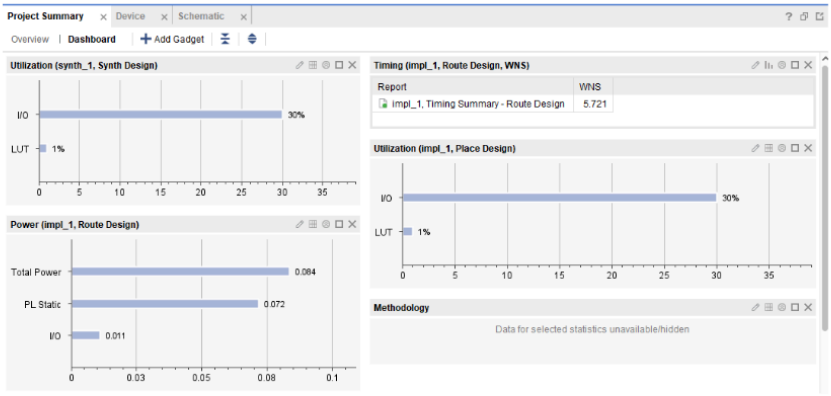


Figure 66 Wallace Multiplier Project Summary

Within the power report with regards to figure 59 and 60 the Brauns multiplier uses slightly less on chip power than the Wallace multiplier, the Brauns multiplier has 10% dynamic on chip power whereas the Wallace multiplier uses 14% of the dynamic on chip power. Other than the power figures the rest of the values within the power report are the same.

The utilization hierarchy also shows the Wallace multiplier utilises less LUTs and Slices than the Brauns multiplier. The Brauns multiplier uses ≈0.47% of the Slice LUTs available. The Wallace multiplier uses ≈0.4% of the available Slice LUTs. The LUT as logic is also the same values as previously stated for the two multipliers. The Brauns multiplier uses more Slice, with ≈0.37% used and the Wallace multiplier with ≈0.32% of the Slice utilised. The final figure within the hierarchy is the Bonded IOB which is the same value for both with ≈30.2% of the IOB utilised. Thirdly the utilisation summary shows the percentage of LUT and percentage of I/O which is both the same for the two multipliers. Finally, the project summary gives an overall of the things previously spoken of.

7.2.2. Details of testing

Once the code had been written to evaluate the VHDL code the first step in the testing process is to run a simulation. This will help to show if there are any problems within the code. For example, if certain lines are producing a value of U, then the code will have some er[12]hin them which need to be addressed. Once the simulation has been completed and the value of the calculator matches the value of the multiplication online then the next step may be conducted.

The second step within testing was to run synthesis. This will compile the code and map it all within FPGA for example. Synthesizing the code will also allow for timing data when the timing constraint has been applied to the code. Third step is to show schematic of the synthesis, this will show the RTL Design. Within synthesis the Pin Allocation can be done for the I/O planning, with use of the Basys 3 diagram the two multipliers were allocated I/O pins.

The fourth step is to start the implementation. The implementation will complete and then you can see the implementation schematic. The timing summary can also be seen within the implementation. These steps will be done for both the Brauns and Wallace multipliers.

Within Testing there will be the timing constraints. With addition to the timing constraint source the timing summary can be found, this will be found in the implementation and the synthesis portions of the software. All the ports will be connected so that the timing of each port is visible, all the Port combinational delays should be a lower value than the one calculated using the diagrams. Additionally, Power Reports can be made within the implementation stage of the software, this can be used to find the usage of ports, wattage and working temperature. Testing on hardware was also completed using the Basys 3. The hardware was able to use the multipliers software to answer the requested multiplier values. Figures seen below.

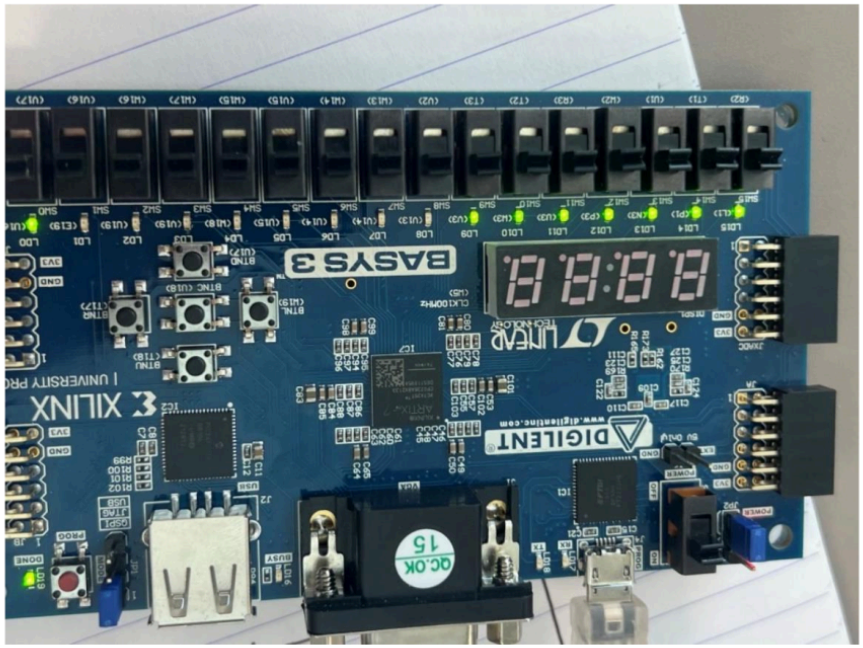


Figure 67 Basys 3 FF x F

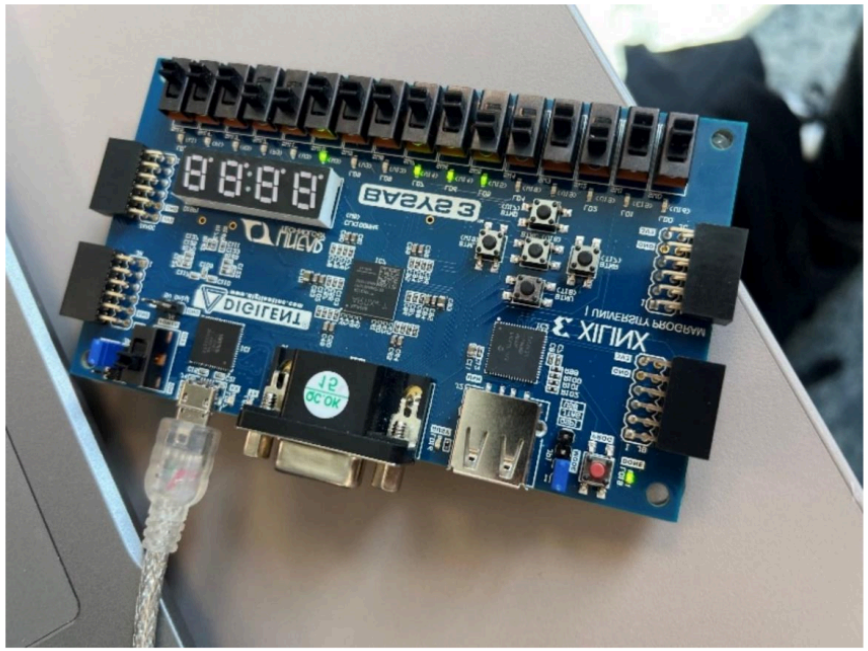


Figure 68 Basys 3 13 x 34.

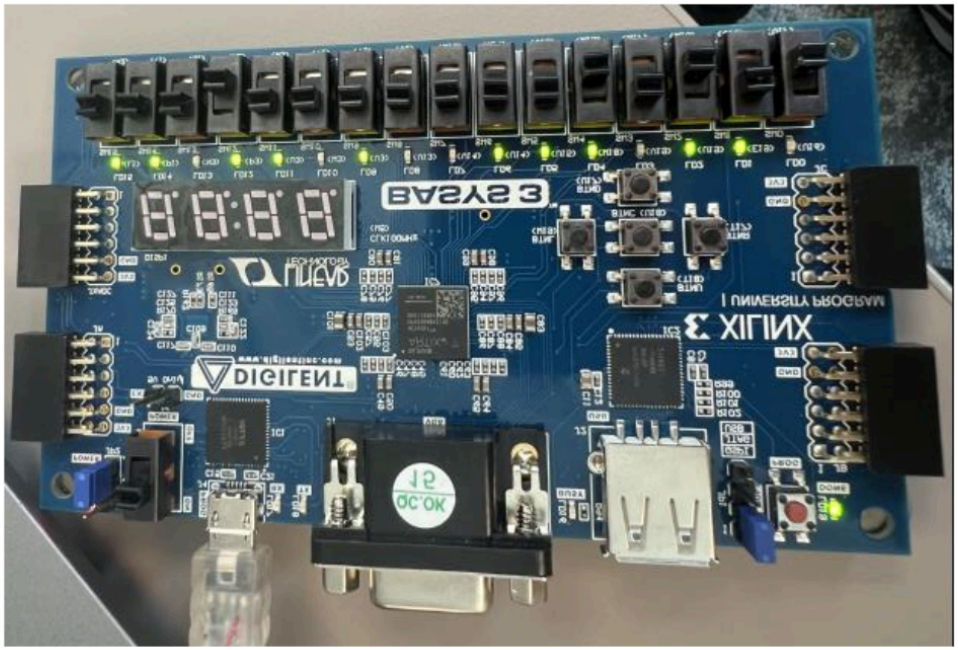


Figure 69 Basys 3 EF x EA.

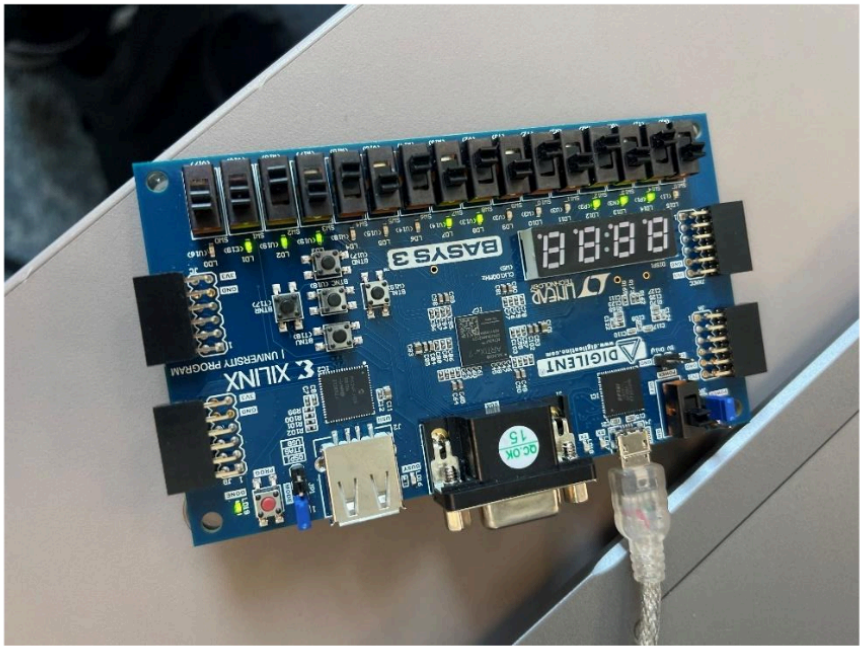


Figure 70 Basys 3 AA x AB

7.3. Recommendations

Within the code that has been developed, there are some simplifications that can be done to make the VHDL code much clearer to read, for the Brauns multiplier, tidying up the code within the VHDL where the partial product signals are held, for example C2_1 and p2_1, improved would be C3, C4 etc. similarly with P2_1 this could be changed to S3, S4 etc... This would make it so that the two codes are mostly identical, this will help with readability and the debugging aspects if there were any changes made to the code. From research, to improve the speed of the Brauns multiplier, an adder which is known as Brent kung or B-K adder can be used to improve the speed [11].

Another type of adder can be used to improve the system which is known as a Kogge-Stone adder [11]. This type of adder can reduce the delay of the multiplier and increase the speed of the multiplier, the adders will reduce delay, however when designing the power constraints must be looked at due to the increase in complexity and the amount of LUTs or logic slices used will increase making the multiplier more intricate [11]. With the addition of these types of adders the multiplier can be utilised in low-mid performance applications.

The Wallace multiplier also has a few recommendations for improvements. The Wallace multiplier can be improved from research [13]. The multiplier can use prefix adders along the final stage of the multiplication [13]. The prefix adders allow for less delay and faster multiplication, however there is a trade off when using prefix adders as they will increase the power consumption of the multiplier [13]. Certain adders which can be used are also able to reduce the complexity of the multiplier [13].

With the inclusion of the Prefix adders the Wallace multiplier is more competitive with the other multipliers that are out there for example the Booth multiplier [13]. The improvements with the Prefix adders will also help with real time applications, as when high speed operations are needed the multiplier will be able to complete them on time [13]. Similarly 14 Brauns multiplier the introduction to prefix adders will also increase the power consumption, LUTs and slice logic used which should be looked at when designing to make sure that the multiplier is appropriate for the application [13].

8. References

[1] D. V. N. Bharathi, V. Nikhil, T. Priya Raghavi, U. A. Manas, and O. Vijay Kumar, *Review on Performance and Hardware Complexity of Multipliers,* vol. 15, no. 5, pp. 1–3, 2020. doi:10.9790/2834-1505011416

[2] A. Sharma, "Design and implementation of Braun multiplier using Verilog: Research," Ijraset, https://www.ijraset.com/research-paper/design-and-implementation-of-braun-multiplier-using-verilog-research#:~:text=Braun%20multiplier%20is%20mainly%20used,technology%20and%20digital%20signal%20processing. (accessed Mar. 13, 2025).

[3] S. Ananthakumaran, M. Abdullah, C. Kamalanathan, K. Nehru, and S. Karthick, "Design and analysis of Braun multiplier using CMOS/GDI technique for signal processing," *International Journal of Emerging Trends in Engineering Research*, vol. 8, no. 10, pp. 6917–6924, Oct. 2020. doi:10.30534/ijeter/2020/488102020

[4] B. Kasturi, *Effective GDI Based Array Multiplier and Braun Multiplier with TSMC 45nm*, vol. 7, no. 5, pp. 46–51, 2020.

[5] "Hex Calculator," Calculator.net, https://www.calculator.net/hex-calculator.html?number1=FA&c2op=x&number2=AC&calctype=op&x=Calculate (accessed Mar. 26, 2025).

[6] A. Oladele, "Top 15 software development methodologies," Velvetech, https://www.velvetech.com/blog/software-development-methodologies/ (accessed Mar. 28, 2025).

[7] S. N. 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*, vol. 391, p. 01025, 2023. doi:10.1051/e3sconf/202339101025

[8] K. Rambabu, H. Venkatesh Kumar, and N. Jayalakshmi, "High speed and efficient Wallace tree multiplier for DSP applications," *Journal of VLSI circuits and systems*, vol. 2, no. 1, Jan. 2020. doi:10.31838/jvcs/02.01.01

[9] H. Bansal, K. G. Sharma, and T. Sharma , "Wallace Tree Multiplier Designs: A Performance Comparison Review," *Wallace Tree Multiplier Designs: A Performance Comparison Review* , vol. 5, no. 5, pp. 60–67, 2014.

[10] "What is the Worst Negative Slack?," AMD Customer Community, https://adaptivesupport.amd.com/s/question/0D54U00006qTJxwSAG/what-is-the-worst-negative-slack?language=en_US (accessed Apr. 7, 2025).

[11] S. C. Bikkina and Nekkanti. Mouni, "Design and implementation of Braun multiplier using parallel prefix adders," *International Journal of Engineering and Advanced Technology*, vol. 9, no. 4, pp. 1251–1254, Apr. 2020. doi:10.35940/ijeat.d8075.049420

[12] M. Bansal, V. Bharti, and V. Chander, "Comparison between conventional fast multipliers and improved fast multipliers using PTL Logic," *IOP Conference Series: Materials Science and Engineering*, vol. 1126, no. 1, p. 012041, Mar. 2021. doi:10.1088/1757-899x/1126/1/012041

[13] I. Journal, "Reduced complexity Wallace multiplier using parallel prefix adders.," International Journal of Engineering Sciences & Research T[15]ogy, https://www.academia.edu/5673653/Reduced_Complexity_Wallace_Multiplier_using_Parallel_Prefix_Adders (accessed Apr. 8, 2025).

[14] Rajagiritech, "FAMILIARISATION OF BASYS 3 ARTIX-7 FPGA BOARD," rajagiritech, https://www.rajagiritech.ac.in/Stud/KTU/RIMS/Brochure/11099.pdf (accessed Apr. 11, 2025).

9. Appendix

```
43  signal A0B3 : std_logic;
44  signal A1B3 : std_logic;
45  signal A2B3 : std_logic;
46  signal A3B3 : std_logic;
47  signal A4B3 : std_logic;
48  signal A5B3 : std_logic;
49  signal A6B3 : std_logic;
50  signal A7B3 : std_logic;
51
52  signal A0B4 : std_logic;
53  signal A1B4 : std_logic;
54  signal A2B4 : std_logic;
55  signal A3B4 : std_logic;
56  signal A4B4 : std_logic;
57  signal A5B4 : std_logic;
58  signal A6B4 : std_logic;
59  signal A7B4 : std_logic;
```

Figure 71 Signal Generation 2

```
60
61  signal A0B5 : std_logic;
62  signal A1B5 : std_logic;
63  signal A2B5 : std_logic;
64  signal A3B5 : std_logic;
65  signal A4B5 : std_logic;
66  signal A5B5 : std_logic;
67  signal A6B5 : std_logic;
68  signal A7B5 : std_logic;
69
70  signal A0B6 : std_logic;
71  signal A1B6 : std_logic;
72  signal A2B6 : std_logic;
73  signal A3B6 : std_logic;
74  signal A4B6 : std_logic;
75  signal A5B6 : std_logic;
76  signal A6B6 : std_logic;
77  signal A7B6 : std_logic;
```

Figure 72 Signal Generation 3

```
104 ¦ signal C4_1 : std_logic; -- P5
105 ¦ signal P5_1 : std_logic;
106 ¦ signal C4_2 : std_logic;
107 ¦ signal C4   : std_logic;
108 ¦ signal P5_2 : std_logic;
109 ¦ signal P5_3 : std_logic;
110 ¦ signal C5_3 : std_logic;
111 ¦ signal P5_4 : std_logic;
112 ¦ signal C5_4 : std_logic;
113 ¦
114 ¦ signal C5_1 : std_logic; -- P6
115 ¦ signal C5_2 : std_logic;
116 ¦ signal P6_1 : std_logic;
117 ¦ signal C5   : std_logic;
118 ¦ signal P6_2 : std_logic;
119 ¦ signal P6_3 : std_logic;
120 ¦ signal P6_4 : std_logic;
121 ¦ signal C6_3 : std_logic;
122 ¦ signal C6_4 : std_logic;
123 ¦ signal C6_5 : std_logic;
124 ¦ signal P6_5 : std_logic;
```

Figure 73 Sum & Carry signals P5 & P6

```
126 ¦ signal C6_1 : std_logic; -- P7
127 ¦ signal C6_2 : std_logic;
128 ¦ signal P7_1 : std_logic;
129 ¦ signal C6   : std_logic;
130 ¦ signal P7_2 : std_logic;
131 ¦ signal C7_3   : std_logic;
132 ¦ signal P7_3 : std_logic;
133 ¦ signal C7_4   : std_logic;
134 ¦ signal P7_4 : std_logic;
135 ¦ signal C7_5   : std_logic;
136 ¦ signal P7_5 : std_logic;
137 ¦ signal C7_6   : std_logic;
138 ¦ signal P7_6 : std logic;
```

Figure 74 Sum & Carry signals P7.

```
signal C7_1 : std_logic; -- P8
signal C7_2 : std_logic;
signal P8_1 : std_logic;
signal C7   : std_logic;
signal P8_2 : std_logic;
signal P8_3 : std_logic;
signal P8_4 : std_logic;
signal P8_5 : std_logic;
signal P8_6 : std_logic;
signal C8_3 : std_logic;
signal C8_4 : std_logic;
signal C8_5 : std_logic;
signal C8_6 : std_logic;
```

Figure 75 Sum & Carry signals P8.

```
154  signal C8_1 : std_logic; -- P9
155  signal C8_2 : std_logic;
156  signal C8   : std_logic;
157  signal P9_2 : std_logic;
158  signal P9_3 : std_logic;
159  signal P9_4 : std_logic;
160  signal P9_5 : std_logic;
161  signal P9_6 : std_logic;
162  signal C9_3 : std_logic;
163  signal C9_4 : std_logic;
164  signal C9_5 : std_logic;
165  signal C9_6 : std_logic;
166  signal C9_2 : std_logic;
```

Figure 76 Sum & Carry signals P9.

```
168   -- P10
169
170
171   signal C9   : std_logic;
172
173   signal P10_3 : std_logic;
174   signal P10_4 : std_logic;
175   signal P10_5 : std_logic;
176   signal P10_6 : std_logic;
177   signal C10_3 : std_logic;
178   signal C10_4 : std_logic;
179   signal C10_5 : std_logic;
180   signal C10_6 : std_logic;
181
182
183   -- P11
184
185   signal C10   : std_logic;
186   signal P11_4 : std_logic;
187   signal P11_5 : std_logic;
188   signal P11_6 : std_logic;
189   signal C11_4 : std_logic;
190   signal C11_5 : std_logic;
191   signal C11_6 : std_logic;
```

Figure 77 Sum & Carry Signals P10 & P11

```
250   A0B2 <= A(0) and B(2);
251   A1B2 <= A(1) and B(2);
252   A2B2 <= A(2) and B(2);
253   A3B2 <= A(3) and B(2);
254   A4B2 <= A(4) and B(2);
255   A5B2 <= A(5) and B(2);
256   A6B2 <= A(6) and B(2);
257   A7B2 <= A(7) and B(2);
258
259   A0B3 <= A(0) and B(3);
260   A1B3 <= A(1) and B(3);
261   A2B3 <= A(2) and B(3);
262   A3B3 <= A(3) and B(3);
263   A4B3 <= A(4) and B(3);
264   A5B3 <= A(5) and B(3);
265   A6B3 <= A(6) and B(3);
266   A7B3 <= A(7) and B(3);
```

Figure 78 Brauns Multiplier Partial Products 2

```
268   A0B4 <= A(0) and B(4);
269   A1B4 <= A(1) and B(4);
270   A2B4 <= A(2) and B(4);
271   A3B4 <= A(3) and B(4);
272   A4B4 <= A(4) and B(4);
273   A5B4 <= A(5) and B(4);
274   A6B4 <= A(6) and B(4);
275   A7B4 <= A(7) and B(4);
276
277   A0B5 <= A(0) and B(5);
278   A1B5 <= A(1) and B(5);
279   A2B5 <= A(2) and B(5);
280   A3B5 <= A(3) and B(5);
281   A4B5 <= A(4) and B(5);
282   A5B5 <= A(5) and B(5);
283   A6B5 <= A(6) and B(5);
284   A7B5 <= A(7) and B(5);
```

Figure 79 Brauns Multiplier Partial Products 3

```
285
286   A0B6 <= A(0) and B(6);
287   A1B6 <= A(1) and B(6);
288   A2B6 <= A(2) and B(6);
289   A3B6 <= A(3) and B(6);
290   A4B6 <= A(4) and B(6);
291   A5B6 <= A(5) and B(6);
292   A6B6 <= A(6) and B(6);
293   A7B6 <= A(7) and B(6);
294
295   A0B7 <= A(0) and B(7);
296   A1B7 <= A(1) and B(7);
297   A2B7 <= A(2) and B(7);
298   A3B7 <= A(3) and B(7);
299   A4B7 <= A(4) and B(7);
300   A5B7 <= A(5) and B(7);
301   A6B7 <= A(6) and B(7);
302   A7B7 <= A(7) and B(7);
```

Figure 80 Brauns Multiplier Partial Products 4

```
P61 : full_adder port map (A => A6B0, B => A5B1, Cin => C5_1, S => P6_1, Cout => C6_1);
P62 : full_adder port map (A => A4B2, B => P6_1, Cin => C5_2, S => P6_2, Cout => C6_2);
P63 :full_adder port map (A => A3B3, B => P6_2, Cin => C5_3, S => P6_3, Cout => C6_3);
P64 :full_adder port map (A => A2B4, B => P6_3, Cin => C5_4, S => P6_4, Cout => C6_4);
p65 : full_adder port map (A => A1B5, B => P6_4, Cin => C5, S => P6_5, Cout => C6_5);
P6_0 : half_adder port map (A => A0B6, B => P6_5, S => P(6), Cout => C6);

P71 : full_adder port map (A => A7B0, B => A6B1, Cin => C6_1, S => P7_1, Cout => C7_1);
P72 : full_adder port map (A => A5B2, B => P7_1, Cin => C6_2, S => P7_2, Cout => C7_2);
P73 : full_adder port map (A => A4B3, B => P7_2, Cin => C6_3, S => P7_3, Cout => C7_3);
P74 : full_adder port map (A => A3B4, B => P7_3, Cin => C6_4, S => P7_4, Cout => C7_4);
P75 : full_adder port map (A => A2B5, B => P7_4, Cin => C6_5, S => P7_5, Cout => C7_5);
P76 : full_adder port map (A => A1B6, B => P7_5, Cin => C6, S => P7_6, Cout => C7_6);
P7_0 : half_adder port map (A => A0B7, B => P7_6, S => P(7), Cout => C7);

P81 : half_adder port map (A => A7B1, B => C7_1, S => P8_1, Cout => C8_1);
P82 : full_adder port map (A => A6B2, B => P8_1, Cin => C7_2, S => P8_2, Cout => C8_2);
P83 : full_adder port map (A => A5B3, B => P8_2, Cin => C7_3, S => P8_3, Cout => C8_3);
P84 : full_adder port map (A => A4B4, B => P8_3, Cin => C7_4, S => P8_4, Cout => C8_4);
P85 : full_adder port map (A => A3B5, B => P8_4, Cin => C7_5, S => P8_5, Cout => C8_5);
p86 : full_adder port map (A => A2B6, B => P8_5, Cin => C7_6, S => P8_6, Cout => C8_6);
P8_0 : full_adder port map (A => A1B7, B => P8_6,Cin=> C7, S => P(8), Cout => C8);
```

Figure 81 Brauns Multiplier Port Maps 2

```
P91 : full_adder port map (A => A7B2, B => C8_1, Cin => C8_2, S => P9_2, Cout => C9_2);
P92 : full_adder port map (A => A6B3, B => P9_2, Cin => C8_3, S => P9_3, Cout => C9_3);
P93 : full_adder port map (A => A5B4, B => P9_3, Cin => C8_4, S => P9_4, Cout => C9_4);
P94 : full_adder port map (A => A4B5, B => P9_4, Cin => C8_5, S => P9_5, Cout => C9_5);
P95 : full_adder port map (A => A3B6, B => P9_5, Cin => C8_6, S => P9_6, Cout => C9_6);
P9_0 : full_adder port map (A => A2B7, B => P9_6,Cin=>C8, S => P(9), Cout => C9);

P101 : full_adder port map (A => A7B3, B => C9_2, Cin => C9_3, S => P10_3, Cout => C10_3);
P102 : full_adder port map (A => A6B4, B => P10_3, Cin => C9_4, S => P10_4, Cout => C10_4);
P103 : full_adder port map (A => A5B5, B => P10_4, Cin => C9_5, S => P10_5, Cout => C10_5);
P104 : full_adder port map (A => A4B6, B => P10_5, Cin => C9_6, S => P10_6, Cout => C10_6);
P10_0 : full_adder port map (A => A3B7, B => P10_6, Cin => C9, S => P(10), Cout => C10);

P111 : full_adder port map (A => A7B4, B => C10_3, Cin => C10_4, S => P11_4, Cout => C11_4);
P112 : full_adder port map (A => A6B5, B => P11_4, Cin => C10_5, S => P11_5, Cout => C11_5);
P113 : full_adder port map (A => A5B6, B => P11_5, Cin => C10_6, S => P11_6, Cout => C11_6);
P11_0 : full_adder port map (A => A4B7, B => P11_6, Cin => C10, S => P(11), Cout => C11);

P121 : full_adder port map (A => A7B5, B => C11_4, Cin => C11_5, S => P12_5, Cout => C12_5);
P122 : full_adder port map (A => A6B6, B => P12_5, Cin => C11_6, S => P12_6, Cout => C12_6);
P12_0 : full_adder port map (A => A5B7, B => P12_6, Cin => C11, S => P(12), Cout => C12);
```

Figure 82 Brauns Multiplier Port Maps 3

```
70
71      signal A0B2 : std_logic;
72      signal A1B2 : std_logic;
73      signal A2B2 : std_logic;
74      signal A3B2 : std_logic;
75      signal A4B2 : std_logic;
76      signal A5B2 : std_logic;
77      signal A6B2 : std_logic;
78      signal A7B2 : std_logic;
79
80      signal A0B3 : std_logic;
81      signal A1B3 : std_logic;
82      signal A2B3 : std_logic;
83      signal A3B3 : std_logic;
84      signal A4B3 : std_logic;
85      signal A5B3 : std_logic;
86      signal A6B3 : std_logic;
87      signal A7B3 : std_logic;
```

Figure 83 Wallace Multiplier Signal Generation 2

```
88
89      signal A0B4 : std_logic;
90      signal A1B4 : std_logic;
91      signal A2B4 : std_logic;
92      signal A3B4 : std_logic;
93      signal A4B4 : std_logic;
94      signal A5B4 : std_logic;
95      signal A6B4 : std_logic;
96      signal A7B4 : std_logic;
97
98      signal A0B5 : std_logic;
99      signal A1B5 : std_logic;
00      signal A2B5 : std_logic;
01      signal A3B5 : std_logic;
02      signal A4B5 : std_logic;
03      signal A5B5 : std_logic;
04      signal A6B5 : std_logic;
05      signal A7B5 : std_logic;
06
```

Figure 84 Wallace Multiplier Signal Generation 4

```
107     signal A0B6 : std_logic;
108     signal A1B6 : std_logic;
109     signal A2B6 : std_logic;
110     signal A3B6 : std_logic;
111     signal A4B6 : std_logic;
112     signal A5B6 : std_logic;
113     signal A6B6 : std_logic;
114     signal A7B6 : std_logic;
115
116     signal A0B7 : std_logic;
117     signal A1B7 : std_logic;
118     signal A2B7 : std_logic;
119     signal A3B7 : std_logic;
120     signal A4B7 : std_logic;
121     signal A5B7 : std_logic;
122     signal A6B7 : std_logic;
123     signal A7B7 : std_logic;
```

Figure 85 Wallace Multiplier Signal Generation 5

```
146     A0B2 <= A(0) and B(2);
147     A1B2 <= A(1) and B(2);
148     A2B2 <= A(2) and B(2);
149     A3B2 <= A(3) and B(2);
150     A4B2 <= A(4) and B(2);
151     A5B2 <= A(5) and B(2);
152     A6B2 <= A(6) and B(2);
153     A7B2 <= A(7) and B(2);
154
155     A0B3 <= A(0) and B(3);
156     A1B3 <= A(1) and B(3);
157     A2B3 <= A(2) and B(3);
158     A3B3 <= A(3) and B(3);
159     A4B3 <= A(4) and B(3);
160     A5B3 <= A(5) and B(3);
161     A6B3 <= A(6) and B(3);
162     A7B3 <= A(7) and B(3);
```

Figure 86 Wallace Multiplier Partial Products 2

```
164     A0B4 <= A(0) and B(4);
165     A1B4 <= A(1) and B(4);
166     A2B4 <= A(2) and B(4);
167     A3B4 <= A(3) and B(4);
168     A4B4 <= A(4) and B(4);
169     A5B4 <= A(5) and B(4);
170     A6B4 <= A(6) and B(4);
171     A7B4 <= A(7) and B(4);
172
173     A0B5 <= A(0) and B(5);
174     A1B5 <= A(1) and B(5);
175     A2B5 <= A(2) and B(5);
176     A3B5 <= A(3) and B(5);
177     A4B5 <= A(4) and B(5);
178     A5B5 <= A(5) and B(5);
179     A6B5 <= A(6) and B(5);
180     A7B5 <= A(7) and B(5);
```

Figure 87 Wallace Multiplier Partial Products 3

```
182     A0B6 <= A(0) and B(6);
183     A1B6 <= A(1) and B(6);
184     A2B6 <= A(2) and B(6);
185     A3B6 <= A(3) and B(6);
186     A4B6 <= A(4) and B(6);
187     A5B6 <= A(5) and B(6);
188     A6B6 <= A(6) and B(6);
189     A7B6 <= A(7) and B(6);
190
191     A0B7 <= A(0) and B(7);
192     A1B7 <= A(1) and B(7);
193     A2B7 <= A(2) and B(7);
194     A3B7 <= A(3) and B(7);
195     A4B7 <= A(4) and B(7);
196     A5B7 <= A(5) and B(7);
197     A6B7 <= A(6) and B(7);
198     A7B7 <= A(7) and B(7);
```

Figure 88 Wallace Multiplier Partial Products 4

```
222     --STAGE 3
223     HA4: Half_adder port map(A=>S9,B=>C8,S=>P(3),Cout=>C16);
224     FA14: Full_Adder port map(A=>A0B4,B=>S10,Cin=>C9,S=>S17,Cout=>C17);
225     FA15: Full_Adder port map(A=>A1B4,B=>S11,Cin=>C10,S=>S18,Cout=>C18);
226     FA16: Full_Adder port map(A=>A2B4,B=>S12,Cin=>C11,S=>S19,Cout=>C19);
227     FA17: Full_Adder port map(A=>A3B4,B=>S13,Cin=>C12,S=>S20,Cout=>C20);
228     FA18: Full_Adder port map(A=>A4B4,B=>S14,Cin=>C13,S=>S21,Cout=>C21);
229     FA19: Full_Adder port map(A=>A5B4,B=>S15,Cin=>C14,S=>S22,Cout=>C22);
230     FA20: Full_Adder port map(A=>A7B3,B=>A6B4,Cin=>C15,S=>S23,Cout=>C23);
231
232     --STAGE 4
233     HA5: Half_adder port map(A=>S17,B=>C16,S=>P(4),Cout=>C24);
234     FA21: Full_Adder port map(A=>A0B5,B=>S18,Cin=>C17,S=>S25,Cout=>C25);
235     FA22: Full_Adder port map(A=>A1B5,B=>S19,Cin=>C18,S=>S26,Cout=>C26);
236     FA23: Full_Adder port map(A=>A2B5,B=>S20,Cin=>C19,S=>S27,Cout=>C27);
237     FA24: Full_Adder port map(A=>A3B5,B=>S21,Cin=>C20,S=>S28,Cout=>C28);
238     FA25: Full_Adder port map(A=>A4B5,B=>S22,Cin=>C21,S=>S29,Cout=>C29);
239     FA26: Full_Adder port map(A=>A5B5,B=>S23,Cin=>C22,S=>S30,Cout=>C30);
240     FA27: Full_Adder port map(A=>A7B4,B=>A6B5,Cin=>C23,S=>S31,Cout=>C31);
```

Figure 89 Wallace Multiplier Stage 3 & 4 Reduction

```
242     --STAGE 5
243     HA6: Half_adder port map(A=>S25,B=>C24,S=>P(5),Cout=>C32);
244     FA28: Full_Adder port map(A=>A0B6,B=>S26,Cin=>C25,S=>S33,Cout=>C33);
245     FA29: Full_Adder port map(A=>A1B6,B=>S27,Cin=>C26,S=>S34,Cout=>C34);
246     FA30: Full_Adder port map(A=>A2B6,B=>S28,Cin=>C27,S=>S35,Cout=>C35);
247     FA31: Full_Adder port map(A=>A3B6,B=>S29,Cin=>C28,S=>S36,Cout=>C36);
248     FA32: Full_Adder port map(A=>A4B6,B=>S30,Cin=>C29,S=>S37,Cout=>C37);
249     FA33: Full_Adder port map(A=>A5B6,B=>S31,Cin=>C30,S=>S38,Cout=>C38);
250     FA34: Full_Adder port map(A=>A7B5,B=>A6B6,Cin=>C31,S=>S39,Cout=>C39);
251
252     --STAGE 6
253     HA7: Half_adder port map(A=>S33,B=>C32,S=>P(6),Cout=>C40);
254     FA35: Full_Adder port map(A=>A0B7,B=>S34,Cin=>C33,S=>S41,Cout=>C41);
255     FA36: Full_Adder port map(A=>A1B7,B=>S35,Cin=>C34,S=>S42,Cout=>C42);
256     FA37: Full_Adder port map(A=>A2B7,B=>S36,Cin=>C35,S=>S43,Cout=>C43);
257     FA38: Full_Adder port map(A=>A3B7,B=>S37,Cin=>C36,S=>S44,Cout=>C44);
258     FA39: Full_Adder port map(A=>A4B7,B=>S38,Cin=>C37,S=>S45,Cout=>C45);
259     FA40: Full_Adder port map(A=>A5B7,B=>S39,Cin=>C38,S=>S46,Cout=>C46);
260     FA41: Full_Adder port map(A=>A7B6,B=>A6B7,Cin=>C39,S=>S47,Cout=>C47);
```

Figure 90 Wallace Multiplier Stage 5 & 6 Reduction

From Port	To Port	Max Delay	Max Process Corner
A[1]	P[4]	14.133	SLOW
A[1]	P[5]	15.354	SLOW
A[1]	P[6]	16.325	SLOW
A[1]	P[7]	17.034	SLOW
A[1]	P[8]	18.124	SLOW
A[1]	P[9]	20.149	SLOW
A[1]	P[10]	20.670	SLOW
A[1]	P[11]	21.643	SLOW
A[1]	P[12]	22.420	SLOW
A[1]	P[13]	23.454	SLOW
A[1]	P[14]	23.055	SLOW
A[1]	P[15]	23.663	SLOW
A[2]	P[2]	10.617	SLOW
A[2]	P[3]	11.820	SLOW
A[2]	P[4]	14.201	SLOW
A[2]	P[5]	15.380	SLOW
A[2]	P[6]	16.273	SLOW
A[2]	P[7]	16.982	SLOW
A[2]	P[8]	18.072	SLOW

Figure 91 Brauns Multiplier Combinational Delay 2

Combinational Delays

From Port	To Port	Max Delay	Max Process Corner
A[2]	P[9]	20.097	SLOW
A[2]	P[10]	20.618	SLOW
A[2]	P[11]	21.591	SLOW
A[2]	P[12]	22.368	SLOW
A[2]	P[13]	23.402	SLOW
A[2]	P[14]	23.003	SLOW
A[2]	P[15]	23.612	SLOW
A[3]	P[3]	11.957	SLOW
A[3]	P[4]	14.339	SLOW
A[3]	P[5]	15.518	SLOW
A[3]	P[6]	16.411	SLOW
A[3]	P[7]	17.120	SLOW
A[3]	P[8]	18.209	SLOW
A[3]	P[9]	20.235	SLOW
A[3]	P[10]	20.755	SLOW
A[3]	P[11]	21.728	SLOW
A[3]	P[12]	22.505	SLOW
A[3]	P[13]	23.539	SLOW
A[3]	P[14]	23.141	SLOW

Figure 92 Brauns Multiplier Combinational Delay 3

Combinational Delays

From Port	To Port	Max Delay	Max Process Corner
A[3]	P[15]	23.749	SLOW
A[4]	P[4]	12.851	SLOW
A[4]	P[5]	14.073	SLOW
A[4]	P[6]	15.044	SLOW
A[4]	P[7]	15.753	SLOW
A[4]	P[8]	16.843	SLOW
A[4]	P[9]	18.868	SLOW
A[4]	P[10]	19.389	SLOW
A[4]	P[11]	20.361	SLOW
A[4]	P[12]	21.139	SLOW
A[4]	P[13]	22.173	SLOW
A[4]	P[14]	21.774	SLOW
A[4]	P[15]	22.382	SLOW
A[5]	P[5]	13.806	SLOW
A[5]	P[6]	15.316	SLOW
A[5]	P[7]	15.893	SLOW
A[5]	P[8]	16.907	SLOW
A[5]	P[9]	18.932	SLOW
A[5]	P[10]	19.627	SLOW

Figure 93 Brauns Multiplier Combinational Delay 4

Combinational Delays

From Port	To Port	Max Delay	Max Process Corner
A[5]	P[11]	20.657	SLOW
A[5]	P[12]	21.475	SLOW
A[5]	P[13]	22.455	SLOW
A[5]	P[14]	22.067	SLOW
A[5]	P[15]	22.676	SLOW
A[6]	P[6]	13.850	SLOW
A[6]	P[7]	14.427	SLOW
A[6]	P[8]	15.441	SLOW
A[6]	P[9]	17.466	SLOW
A[6]	P[10]	17.987	SLOW
A[6]	P[11]	18.960	SLOW
A[6]	P[12]	19.754	SLOW
A[6]	P[13]	20.771	SLOW
A[6]	P[14]	20.372	SLOW
A[6]	P[15]	20.980	SLOW
A[7]	P[7]	13.791	SLOW
A[7]	P[8]	14.853	SLOW
A[7]	P[9]	16.878	SLOW
A[7]	P[10]	17.620	SLOW

Figure 94 Brauns Multiplier Combinational Delay 5

Combinational Delays

From Port	To Port	Max Delay	Max Process Corner
A[7]	P[11]	18.650	SLOW
A[7]	P[12]	19.468	SLOW
A[7]	P[13]	20.448	SLOW
A[7]	P[14]	20.061	SLOW
A[7]	P[15]	20.669	SLOW
B[0]	P[0]	10.058	SLOW
B[0]	P[1]	9.595	SLOW
B[0]	P[2]	9.519	SLOW
B[0]	P[3]	10.853	SLOW
B[0]	P[4]	13.120	SLOW
B[0]	P[5]	14.227	SLOW
B[0]	P[6]	15.198	SLOW
B[0]	P[7]	15.907	SLOW
B[0]	P[8]	16.996	SLOW
B[0]	P[9]	19.021	SLOW
B[0]	P[10]	19.542	SLOW
B[0]	P[11]	20.515	SLOW
B[0]	P[12]	21.292	SLOW
B[0]	P[13]	22.326	SLOW

Figure 95 Brauns Multiplier Combinational Delay 6

Combinational Delays

From Port	To Port	Max Delay	Max Process Corner
B[0]	P[14]	21.928	SLOW
B[0]	P[15]	22.536	SLOW
B[1]	P[1]	9.502	SLOW
B[1]	P[2]	9.286	SLOW
B[1]	P[3]	10.943	SLOW
B[1]	P[4]	13.210	SLOW
B[1]	P[5]	14.317	SLOW
B[1]	P[6]	15.288	SLOW
B[1]	P[7]	15.997	SLOW
B[1]	P[8]	17.086	SLOW
B[1]	P[9]	19.111	SLOW
B[1]	P[10]	19.632	SLOW
B[1]	P[11]	20.605	SLOW
B[1]	P[12]	21.382	SLOW
B[1]	P[13]	22.416	SLOW
B[1]	P[14]	22.018	SLOW
B[1]	P[15]	22.626	SLOW
B[2]	P[2]	8.960	SLOW
B[2]	P[3]	10.763	SLOW

Figure 96 Brauns Multiplier Combinational Delay 7

Combinational Delays

From Port	To Port	Max Delay	Max Process Corner
B[2]	P[4]	13.029	SLOW
B[2]	P[5]	14.136	SLOW
B[2]	P[6]	15.107	SLOW
B[2]	P[7]	15.816	SLOW
B[2]	P[8]	16.905	SLOW
B[2]	P[9]	18.931	SLOW
B[2]	P[10]	19.452	SLOW
B[2]	P[11]	20.424	SLOW
B[2]	P[12]	21.201	SLOW
B[2]	P[13]	22.235	SLOW
B[2]	P[14]	21.837	SLOW
B[2]	P[15]	22.445	SLOW
B[3]	P[3]	9.029	SLOW
B[3]	P[4]	12.050	SLOW
B[3]	P[5]	13.273	SLOW
B[3]	P[6]	14.244	SLOW
B[3]	P[7]	14.953	SLOW
B[3]	P[8]	16.043	SLOW
B[3]	P[9]	18.068	SLOW

Figure 97 Brauns Multiplier Combinational Delay 8

Combinational Delays

From Port	To Port	Max Delay	Max Process Corner
B[3]	P[10]	18.589	SLOW
B[3]	P[11]	19.561	SLOW
B[3]	P[12]	20.339	SLOW
B[3]	P[13]	21.373	SLOW
B[3]	P[14]	20.974	SLOW
B[3]	P[15]	21.582	SLOW
B[4]	P[4]	9.872	SLOW
B[4]	P[5]	12.117	SLOW
B[4]	P[6]	12.977	SLOW
B[4]	P[7]	13.686	SLOW
B[4]	P[8]	14.776	SLOW
B[4]	P[9]	16.801	SLOW
B[4]	P[10]	17.322	SLOW
B[4]	P[11]	18.295	SLOW
B[4]	P[12]	19.074	SLOW
B[4]	P[13]	20.106	SLOW
B[4]	P[14]	19.707	SLOW
B[4]	P[15]	20.316	SLOW
B[5]	P[5]	9.825	SLOW

Figure 98 Brauns Multiplier Combinational Delay 9

Combinational Delays

From Port	To Port	Max Delay	Max Process Corner
B[5]	P[6]	12.242	SLOW
B[5]	P[7]	12.951	SLOW
B[5]	P[8]	14.040	SLOW
B[5]	P[9]	16.065	SLOW
B[5]	P[10]	16.586	SLOW
B[5]	P[11]	17.559	SLOW
B[5]	P[12]	18.336	SLOW
B[5]	P[13]	19.370	SLOW
B[5]	P[14]	18.972	SLOW
B[5]	P[15]	19.580	SLOW
B[6]	P[6]	10.143	SLOW
B[6]	P[7]	10.135	SLOW
B[6]	P[8]	14.061	SLOW
B[6]	P[9]	16.087	SLOW
B[6]	P[10]	16.608	SLOW
B[6]	P[11]	17.580	SLOW
B[6]	P[12]	18.357	SLOW
B[6]	P[13]	19.391	SLOW
B[6]	P[14]	18.993	SLOW

Figure 99 Brauns Multiplier Combinational Delay 10

Combinational Delays

From Port	To Port	Max Delay	Max Process Corner
B[6]	P[6]	10.143	SLOW
B[6]	P[7]	10.135	SLOW
B[6]	P[8]	14.061	SLOW
B[6]	P[9]	16.087	SLOW
B[6]	P[10]	16.608	SLOW
B[6]	P[11]	17.580	SLOW
B[6]	P[12]	18.357	SLOW
B[6]	P[13]	19.391	SLOW
B[6]	P[14]	18.993	SLOW
B[6]	P[15]	19.601	SLOW
B[7]	P[7]	10.202	SLOW
B[7]	P[8]	10.907	SLOW
B[7]	P[9]	12.964	SLOW
B[7]	P[10]	12.681	SLOW
B[7]	P[11]	13.544	SLOW
B[7]	P[12]	14.280	SLOW
B[7]	P[13]	15.700	SLOW
B[7]	P[14]	15.200	SLOW
B[7]	P[15]	15.840	SLOW

Figure 100 Brauns Multiplier Combinational Max Delay 11

Combinational Delays

From Port	To Port	Max Delay	Max Process Corner
A[1]	P[4]	12.558	SLOW
A[1]	P[5]	14.130	SLOW
A[1]	P[6]	14.504	SLOW
A[1]	P[7]	15.079	SLOW
A[1]	P[8]	15.026	SLOW
A[1]	P[9]	16.518	SLOW
A[1]	P[10]	17.212	SLOW
A[1]	P[11]	17.155	SLOW
A[1]	P[12]	18.686	SLOW
A[1]	P[13]	18.624	SLOW
A[1]	P[14]	18.952	SLOW
A[1]	P[15]	19.400	SLOW
A[2]	P[2]	11.185	SLOW
A[2]	P[3]	12.098	SLOW
A[2]	P[4]	12.905	SLOW
A[2]	P[5]	14.445	SLOW
A[2]	P[6]	14.299	SLOW
A[2]	P[7]	15.393	SLOW
A[2]	P[8]	15.506	SLOW

Figure 101 Wallace Multiplier Combinational Delays 2

Combinational Delays

From Port	To Port	Max Delay	Max Process Corner
A[2]	P[9]	17.038	SLOW
A[2]	P[10]	17.527	SLOW
A[2]	P[11]	17.470	SLOW
A[2]	P[12]	19.000	SLOW
A[2]	P[13]	18.939	SLOW
A[2]	P[14]	19.266	SLOW
A[2]	P[15]	19.714	SLOW
A[3]	P[3]	11.498	SLOW
A[3]	P[4]	12.565	SLOW
A[3]	P[5]	14.331	SLOW
A[3]	P[6]	14.439	SLOW
A[3]	P[7]	15.086	SLOW
A[3]	P[8]	15.388	SLOW
A[3]	P[9]	16.920	SLOW
A[3]	P[10]	17.394	SLOW
A[3]	P[11]	17.337	SLOW
A[3]	P[12]	19.007	SLOW
A[3]	P[13]	18.912	SLOW
A[3]	P[14]	19.239	SLOW

Figure 102 Wallace Multiplier Combinational Delays 3

Combinational Delays

From Port	To Port	Max Delay	Max Process Corner
A[3]	P[15]	19.687	SLOW
A[4]	P[4]	12.501	SLOW
A[4]	P[5]	14.266	SLOW
A[4]	P[6]	14.454	SLOW
A[4]	P[7]	15.022	SLOW
A[4]	P[8]	15.400	SLOW
A[4]	P[9]	16.932	SLOW
A[4]	P[10]	17.405	SLOW
A[4]	P[11]	17.596	SLOW
A[4]	P[12]	19.574	SLOW
A[4]	P[13]	19.479	SLOW
A[4]	P[14]	19.806	SLOW
A[4]	P[15]	20.254	SLOW
A[5]	P[5]	13.672	SLOW
A[5]	P[6]	14.198	SLOW
A[5]	P[7]	14.766	SLOW
A[5]	P[8]	15.260	SLOW
A[5]	P[9]	16.792	SLOW
A[5]	P[10]	17.265	SLOW

Figure 103 Wallace Multiplier Combinational Delays 4

Combinational Delays

From Port	To Port	Max Delay	Max Process Corner
A[5]	P[11]	17.209	SLOW
A[5]	P[12]	19.046	SLOW
A[5]	P[13]	18.951	SLOW
A[5]	P[14]	19.278	SLOW
A[5]	P[15]	19.726	SLOW
A[6]	P[6]	14.562	SLOW
A[6]	P[7]	15.130	SLOW
A[6]	P[8]	15.313	SLOW
A[6]	P[9]	16.845	SLOW
A[6]	P[10]	17.318	SLOW
A[6]	P[11]	17.348	SLOW
A[6]	P[12]	19.326	SLOW
A[6]	P[13]	19.231	SLOW
A[6]	P[14]	19.558	SLOW
A[6]	P[15]	20.006	SLOW
A[7]	P[7]	13.912	SLOW
A[7]	P[8]	15.210	SLOW
A[7]	P[9]	16.707	SLOW
A[7]	P[10]	17.180	SLOW

Figure 104 Wallace Multiplier Combinational Delays 5

Combinational Delays

From Port	To Port	Max Delay	Max Process Corner
A[7]	P[11]	17.274	SLOW
A[7]	P[12]	19.252	SLOW
A[7]	P[13]	19.157	SLOW
A[7]	P[14]	19.484	SLOW
A[7]	P[15]	19.932	SLOW
B[0]	P[0]	10.075	SLOW
B[0]	P[1]	9.612	SLOW
B[0]	P[2]	9.214	SLOW
B[0]	P[3]	10.552	SLOW
B[0]	P[4]	11.546	SLOW
B[0]	P[5]	13.119	SLOW
B[0]	P[6]	13.556	SLOW
B[0]	P[7]	14.125	SLOW
B[0]	P[8]	14.308	SLOW
B[0]	P[9]	15.840	SLOW
B[0]	P[10]	16.313	SLOW
B[0]	P[11]	16.343	SLOW
B[0]	P[12]	18.321	SLOW
B[0]	P[13]	18.226	SLOW

Figure 105 Wallace Multiplier Combinational Delays 6

Combinational Delays

From Port	To Port	Max Delay	Max Process Corner
B[0]	P[14]	18.553	SLOW
B[0]	P[15]	19.001	SLOW
B[1]	P[1]	9.693	SLOW
B[1]	P[2]	9.007	SLOW
B[1]	P[3]	10.745	SLOW
B[1]	P[4]	11.739	SLOW
B[1]	P[5]	13.311	SLOW
B[1]	P[6]	14.061	SLOW
B[1]	P[7]	14.630	SLOW
B[1]	P[8]	14.813	SLOW
B[1]	P[9]	16.344	SLOW
B[1]	P[10]	16.818	SLOW
B[1]	P[11]	16.848	SLOW
B[1]	P[12]	18.826	SLOW
B[1]	P[13]	18.731	SLOW
B[1]	P[14]	19.058	SLOW
B[1]	P[15]	19.506	SLOW
B[2]	P[2]	9.633	SLOW
B[2]	P[3]	10.131	SLOW

Figure 106 Wallace Multiplier Combinational Delays 7

Combinational Delays

From Port	To Port	Max Delay	Max Process Corner
B[2]	P[4]	11.421	SLOW
B[2]	P[5]	13.187	SLOW
B[2]	P[6]	13.551	SLOW
B[2]	P[7]	14.120	SLOW
B[2]	P[8]	14.419	SLOW
B[2]	P[9]	15.951	SLOW
B[2]	P[10]	16.425	SLOW
B[2]	P[11]	16.693	SLOW
B[2]	P[12]	18.671	SLOW
B[2]	P[13]	18.576	SLOW
B[2]	P[14]	18.903	SLOW
B[2]	P[15]	19.351	SLOW
B[3]	P[3]	9.122	SLOW
B[3]	P[4]	10.722	SLOW
B[3]	P[5]	12.485	SLOW
B[3]	P[6]	12.312	SLOW
B[3]	P[7]	13.243	SLOW
B[3]	P[8]	13.685	SLOW
B[3]	P[9]	16.136	SLOW

Figure 107 Wallace Multiplier Combinational Delays 8

Combinational Delays

From Port	To Port	Max Delay	Max Process Corner
B[3]	P[10]	16.540	SLOW
B[3]	P[11]	16.483	SLOW
B[3]	P[12]	18.014	SLOW
B[3]	P[13]	17.952	SLOW
B[3]	P[14]	18.279	SLOW
B[3]	P[15]	18.728	SLOW
B[4]	P[4]	9.428	SLOW
B[4]	P[5]	11.138	SLOW
B[4]	P[6]	12.510	SLOW
B[4]	P[7]	13.079	SLOW
B[4]	P[8]	13.490	SLOW
B[4]	P[9]	15.184	SLOW
B[4]	P[10]	15.588	SLOW
B[4]	P[11]	15.531	SLOW
B[4]	P[12]	17.116	SLOW
B[4]	P[13]	17.021	SLOW
B[4]	P[14]	17.348	SLOW
B[4]	P[15]	17.796	SLOW
B[5]	P[5]	9.944	SLOW

Figure 108 Wallace Multiplier Combinational Delays 9

Combinational Delays

From Port	To Port	Max Delay	Max Process Corner
B[5]	P[6]	12.502	SLOW
B[5]	P[7]	13.071	SLOW
B[5]	P[8]	13.794	SLOW
B[5]	P[9]	15.291	SLOW
B[5]	P[10]	15.765	SLOW
B[5]	P[11]	15.708	SLOW
B[5]	P[12]	17.238	SLOW
B[5]	P[13]	17.177	SLOW
B[5]	P[14]	17.504	SLOW
B[5]	P[15]	17.952	SLOW
B[6]	P[6]	10.319	SLOW
B[6]	P[7]	12.125	SLOW
B[6]	P[8]	11.951	SLOW
B[6]	P[9]	13.563	SLOW
B[6]	P[10]	14.258	SLOW
B[6]	P[11]	14.201	SLOW
B[6]	P[12]	15.731	SLOW
B[6]	P[13]	15.670	SLOW
B[6]	P[14]	15.997	SLOW

Figure 109 Wallace Multiplier Combinational Delays 10

Combinational Delays

From Port	To Port	Max Delay	Max Process Corner
B[6]	P[6]	10.319	SLOW
B[6]	P[7]	12.125	SLOW
B[6]	P[8]	11.951	SLOW
B[6]	P[9]	13.563	SLOW
B[6]	P[10]	14.258	SLOW
B[6]	P[11]	14.201	SLOW
B[6]	P[12]	15.731	SLOW
B[6]	P[13]	15.670	SLOW
B[6]	P[14]	15.997	SLOW
B[6]	P[15]	16.445	SLOW
B[7]	P[7]	10.200	SLOW
B[7]	P[8]	10.773	SLOW
B[7]	P[9]	12.940	SLOW
B[7]	P[10]	13.635	SLOW
B[7]	P[11]	13.622	SLOW
B[7]	P[12]	15.153	SLOW
B[7]	P[13]	15.091	SLOW
B[7]	P[14]	15.419	SLOW
B[7]	P[15]	15.867	SLOW

Figure 110 Wallace Multiplier Combinational Delays 11

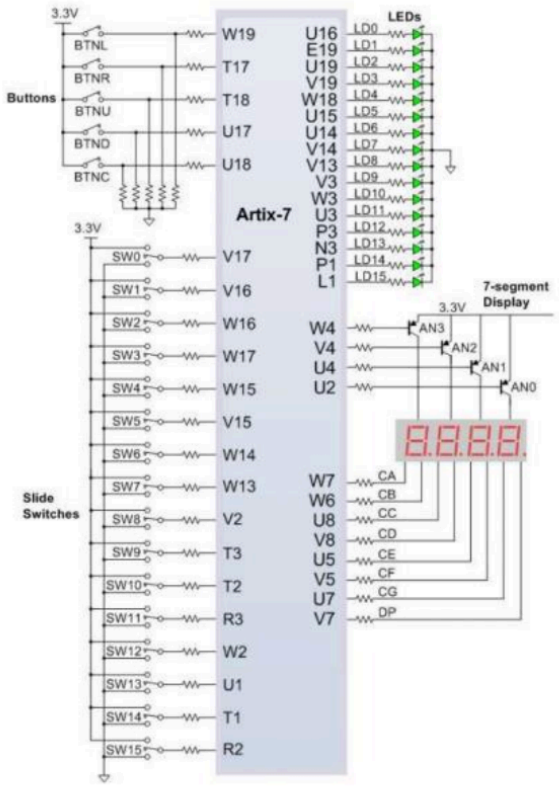


Figure 111 Basys 3 Diagram [14]

Electronics 2 Coursework

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

70/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

Comment 1 | Engagement with literature(20)

The
inclusion of their applications and advantages provides useful context. It will be helpful to start with the basic idea of binary multiplication and explain logic circuits like full adder and half adder clearly.

PAGE 9

PAGE 10

PAGE 11

PAGE 12

Comment 2 | Knowledge & Understanding(20)

More
in-depth explanation of the working process, signal flow, and technical comparisons between the two multipliers would strengthen this section.

PAGE 13

Comment 3 | Knowledge & Understanding(20)

Architecture
diagrams for both Braun and Wallace are shown.

However, labels for inputs, outputs or partial products are not completely visible.

PAGE 14

PAGE 15

PAGE 16

PAGE 17

Comment 4 | Knowledge & Understanding(20)

The coding's of the Full and the half adders are seen. It could be better if the logic design and the concepts of adders is also explained.

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

Comment 5 | Cognitive skills (20)

critical
path calculations for both designs are shown clearly.

PAGE 28

Comment 6 | Practical Application (25)

XDC file showing the Timing Constraints is clearly shown.

PAGE 29

Comment 7 | Knowledge & Understanding(20)

Delay
diagrams both designs are shown clearly.

PAGE 30

PAGE 31

PAGE 32

PAGE 33

PAGE 34

PAGE 35

PAGE 36

Comment 8 | Cognitive skills (20)

The
report includes critical path diagrams, power analysis screenshots, LUT usage data, and area details, which shows effort towards evaluating the design performance and resource utilisation.

PAGE 37

PAGE 38

Comment 9 | Practical Application (25)

Screenshots
like schematics, RTL diagrams, and I/O planning are provided is appreciable.

PAGE 39

Comment 10 | Practical Application (25)

The
report mentions pin mapping in the design flow, indicating awareness of hardware setup.

PAGE 40

PAGE 41

PAGE 42

PAGE 43

PAGE 44

PAGE 45

PAGE 46

PAGE 47

PAGE 48

PAGE 49

PAGE 50

PAGE 51

PAGE 52

PAGE 53

PAGE 54

PAGE 55

PAGE 56

PAGE 57

PAGE 58

PAGE 59

PAGE 60

PAGE 61

Comment 11 | Cognitive skills (20)

The
report includes critical path diagrams, power analysis screenshots, LUT usage data, and area details, which shows effort towards evaluating the design performance and resource utilisation.

PAGE 62

Comment 12 | Cognitive skills (20)

The
report presents output performance analysis using parameters such as power, LUT usage, speed, and area, which indicates that the designs were evaluated thoroughly.

PAGE 63

PAGE 64

Comment 13 | Practical Application (25)

Simulation
and hardware results are shown properly, which proves the design works.

PAGE 65

PAGE 66

PAGE 67

PAGE 68

Comment 14 | Cognitive skills (20)

The
future recommendations are well noted and show a good understanding of how the work can be taken forward.

PAGE 69

PAGE 70

Comment 15 | Engagement with literature(20)

The
report makes an effort to refer to relevant sources, which shows awareness of existing work in the field.

PAGE 71

PAGE 72

PAGE 73

PAGE 74

PAGE 75

PAGE 76

PAGE 77

PAGE 78

PAGE 79

PAGE 80

Comment 16 | Practical Application (25)

It is good that the coding's are done clearly, however, explain how port mapping logic is assumed.

PAGE 81

PAGE 82

PAGE 83

PAGE 84

PAGE 85

PAGE 86

PAGE 87

PAGE 88

PAGE 89

PAGE 90

PAGE 91

PAGE 92

PAGE 93

PAGE 94
PAGE 95
PAGE 96
PAGE 97
PAGE 98
PAGE 99
PAGE 100
PAGE 101

GRADING FORM: UNTITLED

ROBBIE HERBERT **69.90**

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.

Literature engagement is good, though some references may lack full relevance. Analysis is present but lacks depth in parts. Research gaps are acknowledged but not thoroughly explored. Citations are generally correct, with a few inconsistencies. Source variety is adequate. **12.7**

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.

Adequate but limited understanding. Key topics are mentioned (e.g., Braun, Wallace, logic design), but with shallow explanation. Neccessity for the aspects like delay paths, power basics, and slack needs more explaination.

11.7

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

Good level of analysis shown. Partial product logic is missing. Architectural comparisons are mostly correct but needs more clarity. Conclusions are reasonable, though the conclusion should include numerical values on the performance of both multipliers.

12.7

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.

Demonstrates excellent practical skills with thorough VHDL implementation, accurate port mapping, correct FPGA pin assignments, and complete simulation with screenshots. Hardware proof is clear and well-documented. Synthesis and constraint application are precise, and tool reports (LUTs, delay, power, timing) are correctly interpreted and presented.

21.5

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.

11.3