DESIGN OF A 4X4 WALLACE TREE MULTIPLIER USING VERILOG HDL

A PROJECT REPORT

Submitted by

Abhinav R 2019504502

Charaan S 2019504511

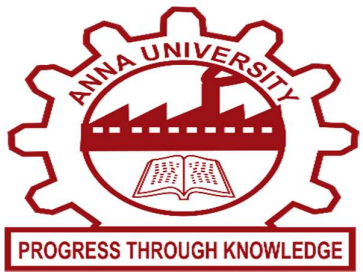
Sarath Vignesh A 2019504581

In partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING



DEPARTMENT OF ELECTRONICS ENGINEERING

MADRAS INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY: CHENNAI

600044

Introduction:-

Binary multipliers are digital circuits capable of multiplying two numbers. Many techniques involve computing the set of partial products, which are then summed together using binary adders in a process similar to long multiplication in binary format. Multipliers usually produce long delay so reducing it is critical. Wallace multipliers are fast multipliers compared to the easily accessible multipliers as they save addition bits in the algorithm for the final product addition. This project involves implementation of 4*4 Wallace Multiplier using Verilog HDL.

Software required:-

Altera Quartus II

Theory:-

Multipliers have gained significant importance with the introduction of the digital computers. Multipliers are most often used in digital signal processing applications and microprocessors designs. In contrast to the process of addition and subtraction, multipliers consume more time and more hardware resources. With the recent advances in technology, a number of multiplication techniques have been implemented for fulfilling the requirement of producing high speed, low power consumption, less area or a combination of them in one multiplier. Speed and area are the two major constraints which conflict each other. Therefore, it is the designer's task to decide proper balance in selecting an appropriate multiplication technique as per requirements.

Parallel multipliers are the high speed multipliers. Therefore, the enhanced speed of the multiplication operation is achieved using various schemes and Wallace tree is one of them. There are three phases in the multiplier architecture:

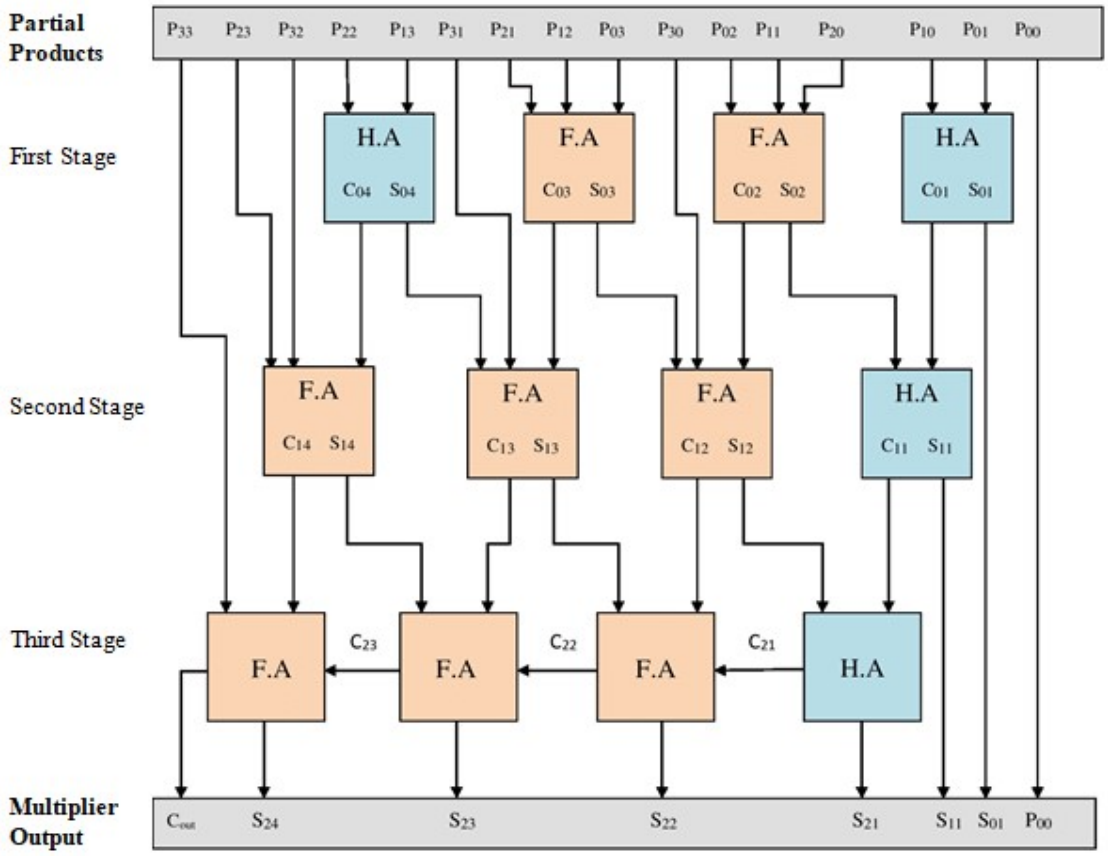
1. The first phase is the generation of partial products.
2. Accumulation of partial product in second phase.
3. The third phase is the final addition phase.

The Wallace tree is a variant of long multiplication. The first step is to multiply each digit (each bit) of one factor by each digit of the other. Each of these partial products has weight equal to the product of its factors. The final product is calculated by the weighted sum of all these partial products. The first step, as said above, is to multiply each bit of one number by each bit of the other, which is accomplished as a simple AND gate, resulting in n^2 bits, the partial product of bits a_m by b_m has weight $2^{(m+n).}$ In the second step, the resulting bits are reduced to two numbers; this is accomplished as follows: As long as there are three or more wires with the same weight, add a following layer:-

- Take any three wires with the same weights and input them into a full adder. The result will be an output wire of the same weight and an output wire with a higher weight for each three input wires.
- If there are two wires of the same weight left, input them into a half adder.
- If there is just one wire left, connect it to the next layer.

In the third and final step, the two resulting numbers are fed to an adder, obtaining the final product.

Circuit Diagram:-



Code:-

i) half_adder.v

```
module half_adder (a,b,s,c);
input a,b;
output s,c;
assign s = a^b;
assign c = a&b;
endmodule
```

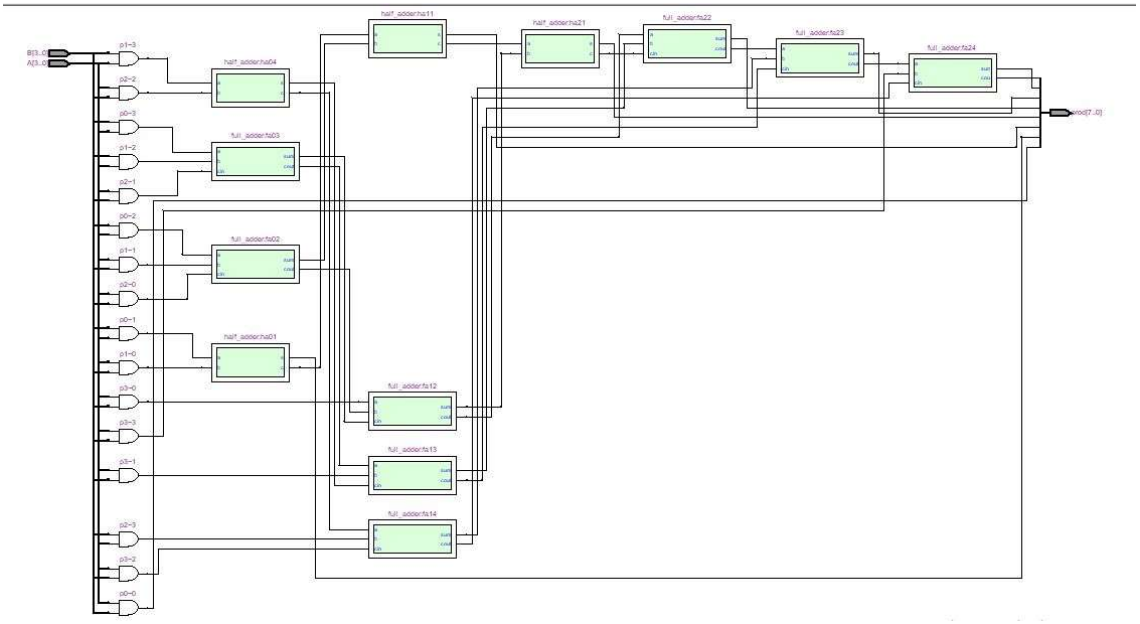
ii) full_adder.v

```
module full_adder(a,b,cin,sum,cout);
input a,b,cin;
output sum,cout;
wire s0,c1,c2;
half_adder ha1 (a,b,s0,c1);
half_adder ha2 (s0,cin,sum,c2);
or(cout,c1,c2);
endmodule
```

iv) wallace.v

```
module wallace_tree(A,B,prod);

    //inputs and outputs
    input [3:0] A,B;
    output [7:0] prod;
    //internal variables.
    wire s01,s02,s03,s04,s11,s12,s13,s14,s21,s22,s23,s24;
    wire c01,c02,c03,c04,c11,c12,c13,c14,c21,c22,c23,c24;
    wire [6:0] p0,p1,p2,p3;

//initialize the p's.
    assign  p0 = A & {4{B[0]}};
    assign  p1 = A & {4{B[1]}};
    assign  p2 = A & {4{B[2]}};
    assign  p3 = A & {4{B[3]}};

//final product assignments
    assign prod[0] = p0[0];
    assign prod[1] = s01;
    assign prod[2] = s11;
    assign prod[3] = s21;
    assign prod[4] = s22;
    assign prod[5] = s23;
    assign prod[6] = s24;
    assign prod[7] = c24;

//first stage
    half_adder ha01 (p0[1],p1[0],s01,c01);
    full_adder fa02(p0[2],p1[1],p2[0],s02,c02);
    full_adder fa03(p0[3],p1[2],p2[1],s03,c03);
    half_adder ha04(p1[3],p2[2],s04,c04);

//second stage
    half_adder ha11 (c01,s02,s11,c11);
    full_adder fa12 (p3[0],c02,s03,s12,c12);
    full_adder fa13 (c03,p3[1],s04,s13,c13);
    full_adder fa14 (c04,p2[3],p3[2],s14,c14);

//third stage
    half_adder ha21(c11,s12,s21,c21);
    full_adder fa22(c12,s13,c21,s22,c22);
    full_adder fa23(c22,s14,c13,s23,c23);
    full_adder fa24(c23,p3[3],c14,s24,c24);

endmodule
```

RTL Viewer:-



Flow Summary:-

Flow Status	Successful - Sun Jun 26 01:15:14 2022
Quartus II Version	9.0 Build 132 02/25/2009 SJ Web Edition
Revision Name	wallace_tree
Top-level Entity Name	wallace_tree
Family	Cyclone II
Device	EP2C5AF256A7
Timing Models	Final
Met timing requirements	Yes
Total logic elements	33 / 4,608 (< 1 %)
Total combinational functions	33 / 4,608 (< 1 %)
Dedicated logic registers	0 / 4,608 (0 %)
Total registers	0
Total pins	16 / 158 (10 %)
Total virtual pins	0
Total memory bits	0 / 119,808 (0 %)
Embedded Multiplier 9-bit elements	0 / 26 (0 %)
Total PLLs	0 / 2 (0 %)

Power Summary:-

PowerPlay Power Analyzer Status	Successful - Sun Jun 26 01:11:55 2022
Quartus II Version	9.0 Build 132 02/25/2009 SJ Web Edition
Revision Name	wallace_tree
Top-level Entity Name	wallace_tree
Family	Cyclone II
Device	EP2C5AF256A7
Power Models	Final
Total Thermal Power Dissipation	30.87 mW
Core Dynamic Thermal Power Dissipation	0.00 mW
Core Static Thermal Power Dissipation	18.00 mW
I/O Thermal Power Dissipation	12.87 mW
Power Estimation Confidence	Low: user provided insufficient toggle rate data

Timing Summary:-

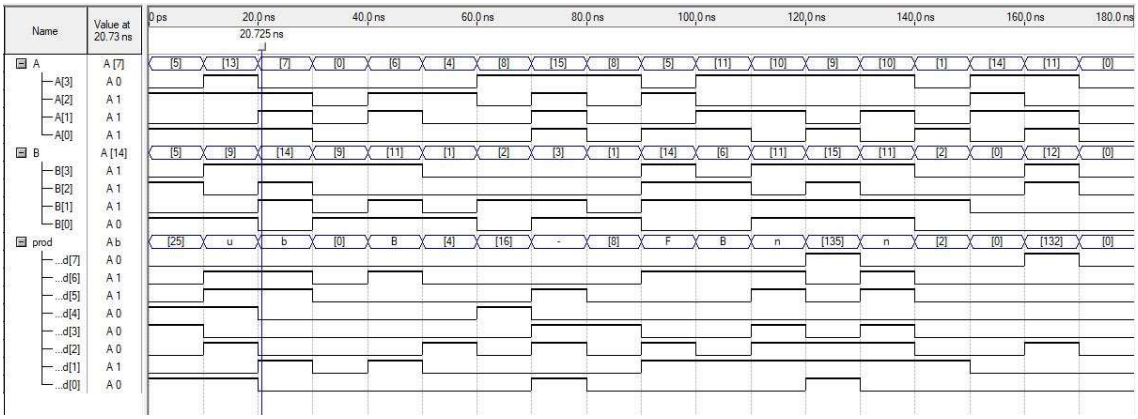
Timing Analyzer Summary

	Type	Slack	Required Time	Actual Time	From	To	From Clock	To Clock	Failed Paths
1	Worst-case tpd	N/A	None	16.696 ns	B[1]	prod[7]	--	--	0
2	Total number of failed paths								0

Registered Performance | tpd | tsu | tco | th | Custom Delays

	Slack	Required P2P Time	Actual P2P Time	From	To
1	N/A	None	16.696 ns	B[1]	prod[7]
2	N/A	None	16.659 ns	B[2]	prod[7]
3	N/A	None	16.552 ns	B[1]	prod[6]
4	N/A	None	16.541 ns	B[0]	prod[7]
5	N/A	None	16.515 ns	B[2]	prod[6]
6	N/A	None	16.436 ns	A[1]	prod[7]
7	N/A	None	16.397 ns	B[0]	prod[6]
8	N/A	None	16.292 ns	A[1]	prod[6]
9	N/A	None	16.064 ns	A[0]	prod[7]
10	N/A	None	15.986 ns	B[1]	prod[5]
11	N/A	None	15.949 ns	B[2]	prod[5]
12	N/A	None	15.920 ns	A[0]	prod[6]
13	N/A	None	15.831 ns	B[0]	prod[5]
14	N/A	None	15.726 ns	A[1]	prod[5]
15	N/A	None	15.423 ns	A[1]	prod[4]
16	N/A	None	15.354 ns	A[0]	prod[5]
17	N/A	None	15.225 ns	A[0]	prod[4]
18	N/A	None	15.156 ns	A[1]	prod[3]
19	N/A	None	15.042 ns	B[1]	prod[4]
20	N/A	None	14.981 ns	B[0]	prod[4]
21	N/A	None	14.775 ns	B[1]	prod[3]
22	N/A	None	14.716 ns	B[0]	prod[3]
23	N/A	None	14.638 ns	A[0]	prod[3]
24	N/A	None	14.470 ns	B[2]	prod[4]
25	N/A	None	13.891 ns	B[2]	prod[3]
26	N/A	None	13.593 ns	A[1]	prod[2]
27	N/A	None	13.212 ns	B[1]	prod[2]
28	N/A	None	13.153 ns	B[0]	prod[2]
29	N/A	None	13.020 ns	A[0]	prod[2]
30	N/A	None	12.764 ns	A[0]	prod[1]
31	N/A	None	12.715 ns	A[1]	prod[1]
32	N/A	None	12.536 ns	A[3]	prod[7]
33	N/A	None	12.536 ns	B[1]	prod[1]
34	N/A	None	12.522 ns	A[2]	prod[7]
35	N/A	None	12.483 ns	B[0]	prod[1]
36	N/A	None	12.392 ns	A[3]	prod[6]
37	N/A	None	12.378 ns	A[2]	prod[6]
38	N/A	None	12.250 ns	B[2]	prod[2]
39	N/A	None	11.918 ns	B[3]	prod[7]
40	N/A	None	11.826 ns	A[3]	prod[5]
41	N/A	None	11.812 ns	A[2]	prod[5]
42	N/A	None	11.774 ns	B[3]	prod[6]
43	N/A	None	11.656 ns	A[0]	prod[0]
44	N/A	None	11.379 ns	B[0]	prod[0]
45	N/A	None	11.208 ns	B[3]	prod[5]
46	N/A	None	10.406 ns	A[2]	prod[4]
47	N/A	None	10.141 ns	A[2]	prod[3]
48	N/A	None	10.039 ns	B[3]	prod[4]
49	N/A	None	10.029 ns	A[3]	prod[4]
50	N/A	None	9.452 ns	B[3]	prod[3]
51	N/A	None	9.451 ns	A[3]	prod[3]
52	N/A	None	8.578 ns	A[2]	prod[2]

Simulation Waveform:-



Result:-

Thus, Wallace tree has been designed using Verilog HDL in Altera Quartus II and its working has been checked along with flow analysis, timing analysis and power analysis. Simulation waveform has been attached.