

DESIGN OF WALLACE TREE MULTIPLIER USING 45nm TECHNOLOGY

Nithin Kumar Y K¹

¹ M.Tech student, VLSI Design and Embedded Systems, CMRIT, Karnataka, INDIA

Abstract - Many electronic circuits and digital signal processing systems comprises an ALU which contains a multiplier to perform various computations. Performance of the multiplier directly affects the performance of the electronic circuit. Hence there is a need of designing a multiplier with optimized performance parameters assuring high performance of the electronic circuit. VLSI trends like Area, Power and Delay are major parameters which constitutes the performance of the system. This paper presents a Wallace Tree Multiplier to multiply two 5-bit unsigned numbers. The multiplier structure is outlined using Carry Save Adder and MUX implementation of Full Adder. Both the designs of Wallace Tree Multiplier are compared for performance parameters. The designs of Wallace Tree Multiplier using Carry Save Adder and MUX implementation of Full Adder is mapped into Cadence Encounter (R) RTL Compiler RC13.0-v13.10-s006_01 and the results are viewed using RTL synthesis tool in Cadence at 45nm Technology. Comparative analysis of results shows that the design of Wallace Tree Multiplier using Carry Save Adder achieved 33.3% reduction in Area, 34.9% reduction in Power and 26.2% reduction in Delay over the design of Wallace Tree Multiplier using MUX implementation of Full Adder.

Key Words: Area, Power, Delay, VLSI, Carry Save Adder, MUX implementation, Wallace Tree Multiplier, Cadence, 45nm Technology.

1.INTRODUCTION

A computer spends more of its time in executing multiplication. It is suggested that the economics of present large scale scientific computers could benefit from a greater investment in hardware to mechanize multiplication and division than is now common. [1]. Multiplier plays a very important role in present DSP and other digital applications. As technology progresses there is a great need of designing a multiplier unit with high performance which assures good performance of the overall digital system. Multiplication is a mathematical operation that is done by adding a number to itself a predefined number of times. Multiplicand is added to itself number of times as indicated by the multiplier. MSB of the result shows the sign of the product. The multiplication performed is binary multiplication i.e. in the form of 0s and 1s.

1.1 Binary Multiplication

The multiplication operation is performed in the form of 0s and 1s. The output obtained can be represented in decimal as well as in the form of 0s and 1s.

There are two types of binary multiplication, they are

- Unsigned multiplication.
- Signed multiplication.

In unsigned multiplication only positive numbers are considered for multiplication. In this case the MSB of both the operands is 0. The MSB of the final product obtained is also 0.

In signed multiplication both positive and negative numbers are considered for multiplication. The MSB of negative number is 1. The MSB of final product obtained might be 0 in the case of both operands are negative and 1 if any one of the operand is positive and other is negative.

1.2 Multiplication Process

The diagram showing the multiplication process is shown in fig 1. [5]

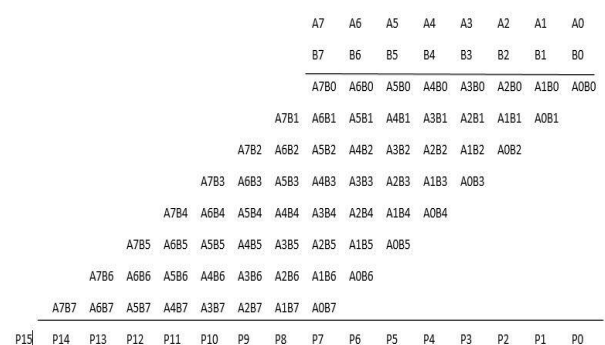


Fig -1: Multiplication

Multiplication is done using two operands one called as multiplicand and the other as multiplier. Each bit of multiplicand is multiplied with single bit of multiplier and is placed row wise as shown in Fig -1.

In Fig -1 each row represents partial product with each partial product being shifted by one bit and the final row

represents the final product. To arrive at the final product, partial product addition is performed. The bits of partial products are fed to adders column wise in order to obtain final product.

1.2 Types of Multipliers

There are various types of multipliers

- Wallace Tree Multiplier.
- Braun Multiplier.
- Baugh Wooley Multiplier.
- Dadda Multiplier.
- Booth Multiplier.

The above mentioned multipliers can be used for Unsigned and Signed Multiplication.

Wallace Tree Multiplier is recommended only for Unsigned Multiplication.

The multiplication process in Wallace Tree Multiplier consists of three stages to arrive at the final product they are,

- Generation of partial products.
- Reduction of partial products.
- Final addition.

2. DESIGN AND IMPLEMENTATION

The Wallace Tree Multiplier is designed using Carry Save Adder and MUX implementation of Full Adder.

2.1 Carry Save Adder implementation of Full Adder

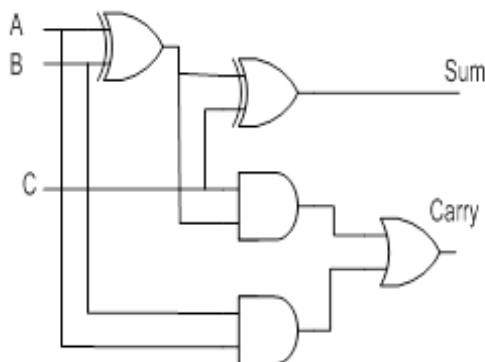


Fig -2: Carry Save Adder implementation of Full Adder

Fig -2 represents conventional Full Adder design [3] which is used as a Carry Save Adder in the proposed design since the Carry of one Full Adder is given as input to Full Adder in the next stage.

The conventional full adder uses two AND gates, two XOR gates and an OR gate taking A, B, C as inputs and producing Sum and Carry as outputs.

The expression for sum and carry:

$$SUM = a \oplus b \oplus c. [3]$$

$$CARRY = ab + bc + ac. [3]$$

The above mentioned conventional full adder forms a Carry Save Adder block as shown in Fig -3.

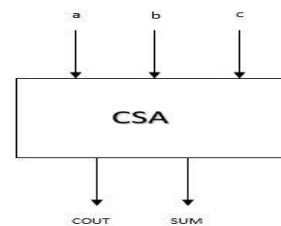


Fig -3: Block of Carry Save Adder

2.2 MUX implementation of Full Adder

The diagram showing the MUX implementation of Full Adder is shown in Fig -4 [3].

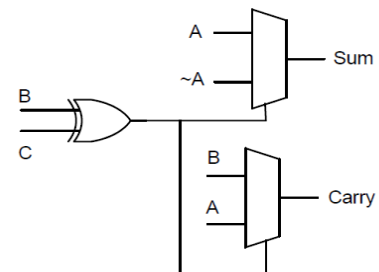


Fig -4: MUX implementation of Full Adder

The MUX implementation of Full Adder consists of two 2:1 MUX and a XOR gate. The proper functioning of MUX implementation of Full Adder as a Full Adder is verified by using the Truth Table of Full Adder.

In this implementation when

- Select line=0, Sum=A and Carry=B.
- Select line=1, Sum=!A and Carry=A.

The expression for Sum is

$$Sum = !(b \wedge c)a + (b \wedge c) !a. [3]$$

The expression for Carry is

$$Carry = b \cdot !(b \wedge c) + a(b \wedge c). [3]$$

The Carry Save Adder and MUX implementation of Full Adder operates as a proper Full Adder and both the implementations of Full Adder are utilized to design a

Wallace Tree Multiplier. The paper proposes a 5*5 Wallace Tree Multiplier to multiply unsigned numbers.

2.3 5*5 Multiplication

The diagram showing 5*5 multiplication process is shown below. [4]

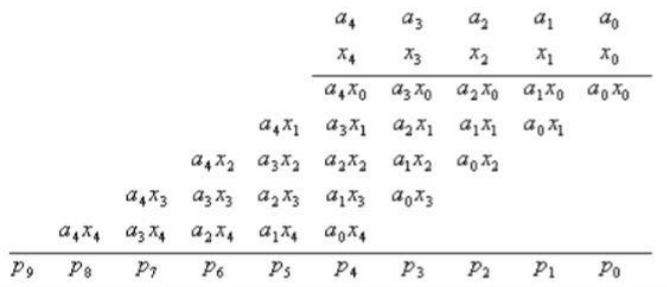


Fig -5: 5*5 multiplication

Considering two 5-bit numbers for multiplication, the process obtains five partial products which are to be added to obtain the final product which is 10-bit in size. The required number of zeros are concatenated in each column to design a Wallace Tree Multiplier. In the first, second, third and fourth column four, three, two and one zeros are concatenated respectively below the partial product bits. In the fifth column no zero is concatenated since five partial product bits are present. In the sixth, seventh, eighth and ninth columns one, two, three and four zeros are concatenated respectively above the partial product bits.

2.4 Wallace Tree to add Five bits

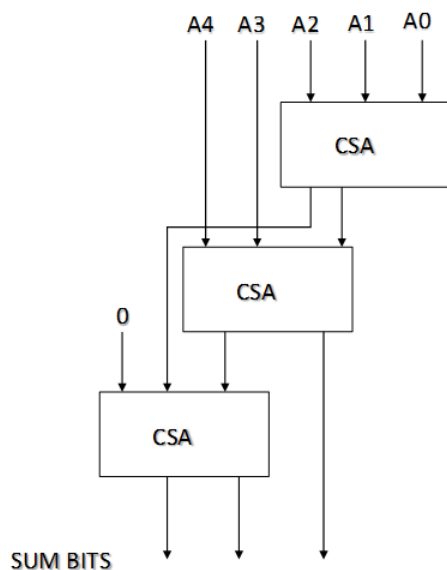


Fig -6: WT to add five bits using Carry Save Adder

The Carry Save Adder is instantiated in three stages with a Carry Save Adder in each stage to form a Wallace Tree acting as a compressor to perform partial product reduction. The Wallace Tree shown in Fig -6 takes 5 inputs and produces a 3-bit output which is given as input to next stages in the proposed multiplier structure. The above shown WT acts as a 5:3 compressors which is actually used in the partial product reduction stage in the original Wallace Tree Multiplier [1].

The same Wallace Tree is designed using MUX implementation of Full Adder which is shown in Fig -7.

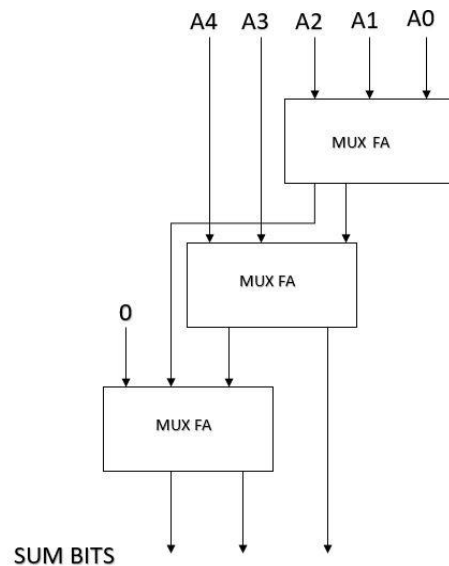


Fig -7: WT to add five bits using MUX implementation of Full Adder

MUX FA is instantiated in three stages with a MUX FA in each stage which takes 5-bit input and produces 3-bit output which is given as input to next stages in the proposed multiplier structure.

The WT shown in Fig -6 and Fig-7 takes partial product bits in vertical manner as input and produces a compressed output. The structure shown in Fig -6 and Fig -7 is used in the proposed Wallace Tree Multiplier using Carry Save Adder and MUX implementation of Full Adder respectively by instantiating the WT nine times in the first stage. The partial product bits from first column to ninth column are fed as input to these nine WT's in the design of Wallace Tree Multiplier.

2.5 Wallace Tree Multiplier

The diagram showing the proposed design of Wallace Tree Multiplier using Carry Save Adder is shown in Fig -8.

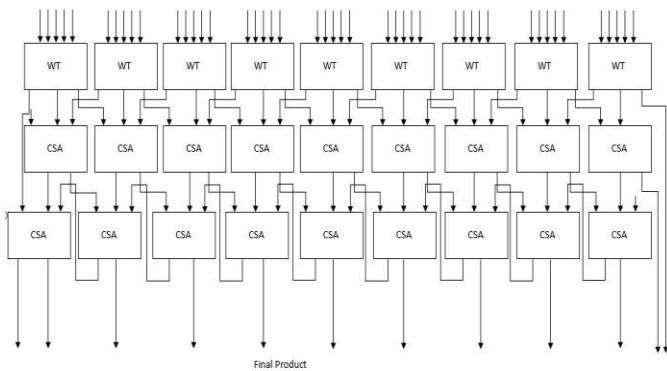


Fig -8: Wallace Tree Multiplier using Carry Save Adder

In Fig -8 WT represents Wallace Tree to add five bits shown in Fig -6.

The multiplier is designed is three stages with WT being instantiated nine times in the first stage and Carry Save Adder being instantiated nine times each in second and third stages thus totally 18 times.

The diagram showing the proposed design of Wallace Tree Multiplier using MUX implementation of Full Adder is shown in Fig -9.

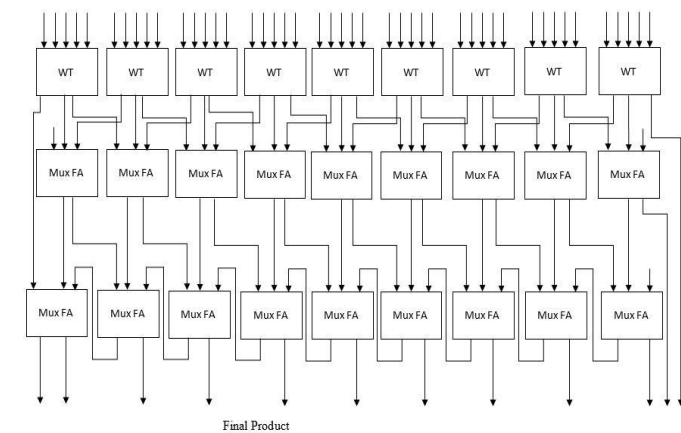


Fig -9: Wallace Tree Multiplier using MUX implementation of Full Adder

In Fig -9 WT represents Wallace Tree to add five bits shown in Fig -7.

The partial products obtained from Fig -5 is given as input column wise to Wallace Tree Multiplier design shown in Fig -8 and Fig -9.

The zeroth, first, second, third, fourth, fifth, sixth, seventh, eighth bits of partial products are fed as input to first, second, third, fourth, fifth, sixth, seventh, eighth and ninth WT respectively which is shown in Fig -8 and Fig -9.

In the second stage the output of nine WTs are fed as input to nine Carry Save Adders in Fig -8 and Fig -9.

In the third stage final addition is performed to obtain the final product by feeding output of second stage as input to third stage. Here final addition is carried out by instantiating nine Carry Save Adders in Fig -8 and nine MUX implementation of Full Adder in Fig -9.

The proposed design of Wallace Tree Multiplier using Carry Save Adder and MUX implementation of Full Adder takes 45-bits as input and produces a 12-bit output which is nothing but the final product obtained by multiplying two 5-bit numbers. Since the result of 5*5 multiplication should be 10-bit in size the last two MSB bits of the result obtained from the proposed Wallace Tree Multiplier design have to be neglected which yields a 10-bit output.

3 RESULTS

The diagram showing the output waveform of proposed Wallace Tree Multiplier using Carry Save Adder and MUX implementation of Full Adder is shown in Fig -10.

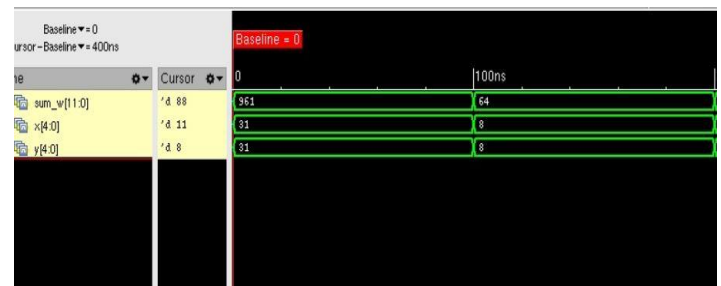


Fig -10: Output waveform of proposed Wallace Tree Multiplier using Carry Save Adder and MUX implementation of Full Adder

The diagram showing output of proposed Wallace Tree Multiplier in binary format is shown in Fig -11.



Fig -11: Output waveform of proposed Wallace Tree Multiplier using Carry Save Adder and MUX implementation of Full Adder in binary format

Fig -10 depicts the product obtained by multiplying 31 by 31 is 961 which is 111100001 in binary format shown in Fig -11 and the product obtained by multiplying 8 by 8 is 64 which is 000100000 in binary format shown in Fig -11.

The diagram showing the area occupied by Wallace Tree Multiplier using MUX implementation of Full Adder is shown in Fig -12.

Instance	Cells	Cell Area	Net Area	Total Area	Wireload	WL Flag
wallace_tree_5_opnds	116	366.52	0.00	366.52	<none>	(D)
wallace_tree_5_opnds/inst	2	7.04	0.00	7.04	<none>	(D)
wallace_tree_5_opnds/inst	2	7.04	0.00	7.04	<none>	(D)
wallace_tree_5_opnds/inst	1	4.69	0.00	4.69	<none>	(D)
wallace_tree_5_opnds/inst	1	2.35	0.00	2.35	<none>	(D)
wallace_tree_5_opnds/inst	4	14.55	0.00	14.55	<none>	(D)
wallace_tree_5_opnds/inst	4	14.55	0.00	14.55	<none>	(D)
wallace_tree_5_opnds/inst	1	4.69	0.00	4.69	<none>	(D)
wallace_tree_5_opnds/inst	2	5.16	0.00	5.16	<none>	(D)
wallace_tree_5_opnds/inst	9	30.04	0.00	30.04	<none>	(D)
wallace_tree_5_opnds/inst	2	7.04	0.00	7.04	<none>	(D)
wallace_tree_5_opnds/inst	1	4.69	0.00	4.69	<none>	(L)

Fig -5.12: Area occupied by Wallace Tree Multiplier using MUX implementation of Full Adder

The Diagram showing the power consumed by Wallace Tree Multiplier using MUX implementation of Full Adder is shown in Fig -5.13.

Instance	Cells	Leakage (nW)	Internal (nW)	Net (nW)	Switching (nW)
wallace_tree_5_opnds	116	2743.90	24692.46	16333.11	41025.57
wallace_tree_5_opnds/ins	2	60.76	335.87	46.64	382.51
wallace_tree_5_opnds/ins	2	60.76	335.87	46.64	382.51
wallace_tree_5_opnds/ins	1	45.15	300.08	0.00	300.08
wallace_tree_5_opnds/ins	1	15.61	35.80	46.64	82.43
wallace_tree_5_opnds/ins	4	110.59	1074.32	598.70	1673.02
wallace_tree_5_opnds/ins	4	110.59	1074.32	598.70	1673.02
wallace_tree_5_opnds/ins	1	45.15	518.28	196.62	714.90
wallace_tree_5_opnds/ins	2	20.29	216.47	170.16	386.63
wallace_tree_5_opnds/ins	9	233.84	2160.04	1241.83	3401.86
wallace_tree_5_opnds/ins	2	60.76	460.63	240.74	701.37
wallace_tree_5_opnds/ins	1	45.15	388.35	147.47	535.82

Fig -5.13: Power consumed by Wallace Tree Multiplier using MUX implementation of Full Adder

The diagram showing the timing report of Wallace Tree Multiplier using MUX implementation of Full Adder is shown in Fig -5.14.

```

MUX1/out
inst20/cout
inst21/c
g11/A
g11/Y
MUX1/sel
g19/S
g19/Y
g18/A
g18/Y
MUX1/out
inst21/cout
inst22/c
MUX1/sel
g9/A
g9/Y
MUX1/out
inst22/cout
sum_w[10]
-----
X0R2X1      2  9.5  78  +0  932
                989 R
MUX2X1      1  1.6  38  +48 1037 R
                1037
INVX1       2  6.9  24  +33 1070 F
                1070
AND2X1      1  0.0  10  +30 1099 F
                1099 F
out port                +0 1099 F
    
```

Fig -5.14: Timing report of Wallace Tree Multiplier using MUX implementation of Full Adder.

The diagram showing the RTL of Wallace Tree Multiplier using Carry Save Adder and MUX implementation of Full Adder is shown in Fig -5.15

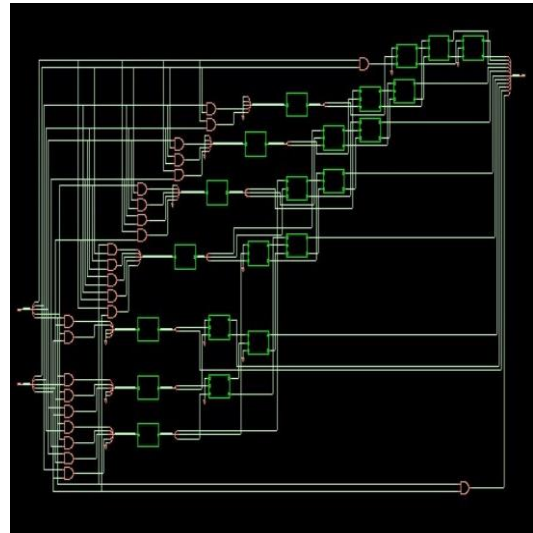


Fig -15: RTL of proposed Wallace Tree Multiplier using

Carry Save Adder and MUX implementation of Full Adder The design of Wallace Tree Multiplier using MUX implementation of Full Adder occupies a total cell area of 367, consumes a total power of 43769.473nW and gives a delay of 1099F.

The diagram showing the area occupied by Wallace Tree Multiplier using Carry Save Adder is shown in Fig -5.16.

Instance	Cells	Cell Area	Net Area	Total Area	Wireload	WL Flag
wallace_tree_5_opnds	52	244.51	0.00	244.51	<none>	(D)
wallace_tree_5_opnds/inst	1	4.69	0.00	4.69	<none>	(D)
wallace_tree_5_opnds/inst	1	4.69	0.00	4.69	<none>	(D)
wallace_tree_5_opnds/inst	1	8.92	0.00	8.92	<none>	(D)
wallace_tree_5_opnds/inst	1	8.92	0.00	8.92	<none>	(D)
wallace_tree_5_opnds/inst	3	18.30	0.00	18.30	<none>	(D)
wallace_tree_5_opnds/inst	1	4.69	0.00	4.69	<none>	(D)
wallace_tree_5_opnds/inst	1	8.92	0.00	8.92	<none>	(D)
wallace_tree_5_opnds/inst	1	4.69	0.00	4.69	<none>	(D)
wallace_tree_5_opnds/inst	3	22.53	0.00	22.53	<none>	(D)
wallace_tree_5_opnds/inst	1	8.92	0.00	8.92	<none>	(D)
wallace_tree_5_opnds/inst	1	8.92	0.00	8.92	<none>	(L)

Fig -5.16: Area occupied by Wallace Tree Multiplier using Carry Save Adder

The diagram showing the power consumed by Wallace Tree Multiplier using Carry Save Adder is shown in Fig -5.17.

Instance	Cells	Leakage (nW)	Internal (nW)	Net (nW)	Switching (nW)
wallace_tree_5_opnds	52	1557.44	15923.95	11293.65	27217.60
wallace_tree_5_opnds/ins	1	47.56	228.45	27.10	255.55
wallace_tree_5_opnds/ins	1	47.56	228.45	27.10	255.55
wallace_tree_5_opnds/ins	1	39.21	732.09	217.42	949.52
wallace_tree_5_opnds/ins	1	39.21	732.09	217.42	949.52
wallace_tree_5_opnds/ins	3	134.34	1421.19	674.64	2095.83
wallace_tree_5_opnds/ins	1	47.56	332.12	269.73	601.84
wallace_tree_5_opnds/ins	1	39.21	865.83	249.56	1115.39
wallace_tree_5_opnds/ins	1	47.56	223.25	155.35	378.59
wallace_tree_5_opnds/ins	3	125.99	1806.67	841.96	2648.63
wallace_tree_5_opnds/ins	1	39.21	609.37	317.63	927.00
wallace_tree_5_opnds/ins	1	39.21	950.35	272.25	

Fig -5.17: Power consumed by Wallace Tree Multiplier using Carry Save Adder

The diagram showing the timing report of Wallace Tree Multiplier using Carry Save Adder is shown in Fig -5.18.

```

-----
g21/Y5      HAX1      1  3.4  17  +58  268  F
inst12/sum
inst20/b
g21/B
g21/YC     HAX1      1 10.9  41  +65  333  F
inst20/cout
inst21/c
g2/A
g2/YC     FAX1      1 10.9  40  +99  432  F
inst21/cout
inst22/c
g2/A
g2/YC     FAX1      1  7.1  29  +92  524  F
inst22/cout
inst23/c
g2/C
g2/YC     FAX1      1  7.1  29  +82  606  F
inst23/cout
inst24/c
g2/C
g2/YC     FAX1      1  7.1  29  +82  687  F
inst24/cout
inst25/c
g2/C
g2/YC     FAX1      1  3.4  19  +74  761  F
inst25/cout
inst26/c
g21/B
g21/Y5     HAX1      1  0.0   7  +51  812  F
inst26/sum
sum_w[9]   out port          +0  812  F
-----

```

Fig -5.18: Timing report of Wallace Tree Multiplier using Carry Save Adder.

The design of Wallace Tree Multiplier using Carry Save Adder occupies a total cell area of 245, consumes power of 28775nW and the delay is 812F

The table showing the comparison of Area, Power and Delay of Wallace Tree Multiplier using Carry Save Adder and MUX implementation of Full Adder for Area, Power and Delay is shown in Table -1.

Wallace Tree Multiplier using	Area	Power(nW)	Delay
Carry Save Adder	245	28775	812F
MUX implementation of Full Adder	367	43769	1099F

Table -1: Comparison of Wallace Tree Multiplier using Carry Save Adder and MUX implementation of Full Adder for Area, Power and Delay

3. CONCLUSIONS

The paper presents a 5*5 Wallace Tree Multiplier using Carry Save Adder and MUX implementation of Full Adder to multiply two unsigned numbers. Both the designs of Wallace Tree Multiplier are compared for Area, Power and Delay. The Design of Wallace Tree Multiplier using Carry Save Adder achieved 33.3% reduction in area, 34.9% reduction in power consumption and 26.2% reduction in critical path delay over the design of Wallace Tree Multiplier using MUX implementation of Full Adder at 45nm technology. Hence the Design of Wallace Tree Multiplier using Carry Save Adder is more efficient than the design of Wallace Tree Multiplier using MUX implementation of Full Adder.

In future the same design can be extended to Booth Multiplier to multiply two unsigned and signed numbers.

REFERENCES

- [1] C. S. WALLACE, "A Suggestion for a Fast Multiplier" IEEE Transactions on Electronic Computers, Year: 1964, Pages: 14 – 17.
- [2] Sudhakar Alluri, B.RajendraNaik, N.S.S.Reddy, "Mapping of Five input Wallace tree using Cadence Tool for Low Power, Low area and High Speed" 2016 International Conference on Communication and Signal Processing (ICCSP), Year: 2016, Pages: 0304 – 0310.
- [3] Kokila Bharti Jaiswal, Nithish Kumar V, Pavithra Seshadri and Lakshminarayanan G, "Low Power Wallace Tree Multiplier Using Modified Full Adder" 2015 3rd International Conference on Signal Processing, Communication and Networking (ICSCN), Year: 2015, Pages: 1 – 4.
- [4] Pramodini Mohanty, "An Efficient Baugh-Wooley Architecture for Signed & Unsigned Fast Multiplication" NIET Journal of Engineering & Technology, Vol. 1, Issue 2, 2013.
- [5] Charles R Baugh, Bruce A Wooley, "A Two's Complement Parallel Array Multiplication Algorithm" IEEE TRANSACTIONS ON COMPUTERS, VOL. C-22, NO. 12, DECEMBER 1973.

BIOGRAPHIES



Nithin Kumar Y K received his Bachelor of Engineering in Electronics and Communication Engineering from MVJCE, Bengaluru in 2015. He is currently pursuing M.Tech in VLSI Design and Embedded Systems at CMRIT, Bengaluru.