

# Design of Optimized Wallace Tree Multiplier in Cadence

Anindita Dash, Swetapadma Dash, S.K.Mandal

VLSI and Embedded Systems, KIIT University, India

## ABSTRACT

Shrinking device size has been a major area of interest for VLSI design engineers. In order to achieve multiple functions on the same substrate using the standard technology, designers have to deal with the issues of area and power dissipation. So there is a pressing demand for designers to work in the area of optimization. In this paper, we propose to optimize a Wallace Tree multiplier. The multiplier was implemented at the circuit level of design abstraction with Virtuoso® tool in Cadence.

**Keywords:** VLSI, power dissipation, Optimization, Wallace Tree Multiplier, FPM, Cadence

## 1. INTRODUCTION

The device size has been decreasing at a very fast pace which in the Very Large Scale Integration (VLSI) technology has enabled to fabricate huge and complex designs into a single chip and simultaneously this has increased the power density of the systems. The optimization of design for power becomes the crucial feature to maximize the runtime efficiency. There are different techniques followed at different abstraction level of VLSI design flow. We propose to optimize the Wallace Tree multiplier (WTM) at the circuit level of design abstraction.

The main bottle neck of the any arithmetic calculation performance lies in multiplication [1][2]. There are various optimization techniques implemented at various level of design abstraction. The multiplier increases the hardware requirement and thus is expensive [1][2]. The multiplier for low power is implemented using 1.2um CMOS (Complementary Metal Oxide Semiconductor) technology with booth recoding and reduction in power supply [3].

The WTM is faster than Array Multiplier but it comes with a disadvantage of layout complexity [4][5]. The layout feasibility is of prime importance in the VLSI field. To overcome this drawback we went onto implement the WTM with compressors [6]. Hence different topologies of compressors were studied and were used to implement the WTM.

## 2. COMPRESSORS BASED MULTIPLIER

The compressor based multiplication of 5x5 bits can be done as seen in the figure:

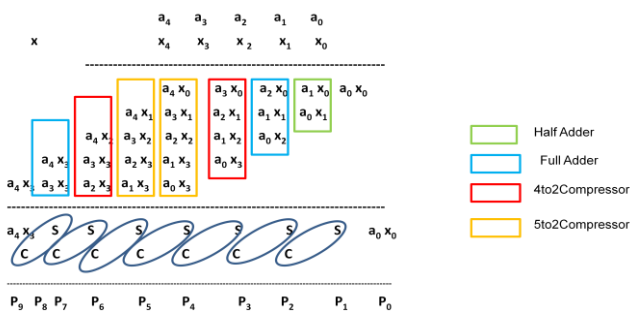


Fig1 Multiplication with compressors [5]

Unlike WTM process of step by step of row reduction; in the compressor technique the partial products in a single column are together reduced. There's a final 8bit carry propagate adder which helps to generate the product terms. From the compressor technique shown in figure 1, it is seen that there are two different compressors used apart from the full adder and the half adder, where one is the 4to2 compressor and the other is the 5to2 compressors. At first, different 4to2 Compressors and 5to2 compressors implementations were studied and compared. Then they were used in the multiplier. This optimized multiplier was then used in a floating point multiplier and compared for results.

### 2.1 4to2 Compressors

The 4to2 Compressor has five inputs and three different outputs, where four inputs and the output sum have of same weight. The output carry is of one bit higher weight. The fifth input is the carry from the preceding column.

The 4 to 2 Compression technique can be done using two different topologies. The basic implementation is to use two adders namely Topology I. The two full adders are in cascaded fashion, where the first adder takes in the three input bits to generate sum entering to the cascaded adder and carry-out to succeeding column. The next adder takes in the fourth bit, sum generated by previous adder and the carry from preceding column to give the final sum and carry. It can be better understood by looking into its block diagram in figure 2

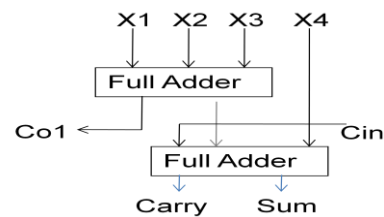


Fig.2 Block Diagram of Topology I 4to2 Compressor [6]

There is another topology of 4to2 Compressors, Topology II, and figure 3 where the logic of above adders is used and implemented in different logic style to give fastness. The topology I has 4XOR delay for the sum, topology II has 3XOR delay for the sum.

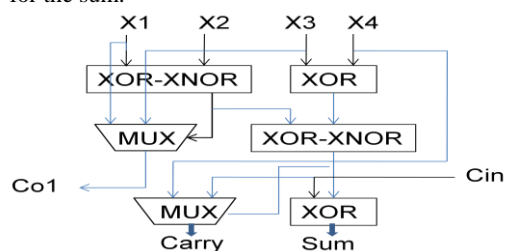


Fig.3 Block Diagram of Topology II 4to2 Compressors [6]

## 2.2 5to2 Compressors

The 5to 2 Compressors has seven inputs out of which four are primary input and the other two are the carry bit received from lower column compressor. It has total four outputs namely sum of same weight as the inputs and three carry outputs of one bit higher weight.

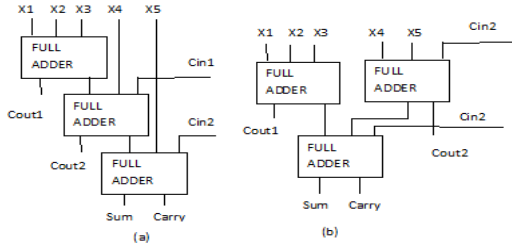


Fig 4 Different Architectures in Topology I of 5to2 Compressor [6]

The Topology-I of 5to2 compressors consists of three adders in cascaded fashion in which the sum has a delay of 6xor delay and topology II has 5xor delay. The Topology-I can be of two different architectures, where the architecture (b) as shown in figure 4 fares better than architecture (a).

The Topology-II of 5to2 compressors, just like Topology-II of 4to2 compressors uses different logic styles to implement the different blocks as seen in the figure 5. It consists of XOR-XNOR block to generate simultaneous XOR and XNOR signal with whose help the carry and sum are generated. The carry is generated by multiplexer unit and sum by a XOR unit.

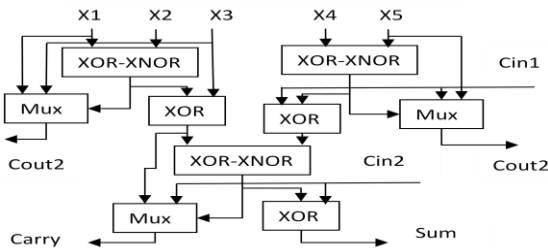


Fig.5 Topology II of 5to2 Compressor [6]

## 3. CIRCUIT LEVEL IMPLEMENTATION

We began our work by implementing WTM at circuit level in 45nm Technology in Cadence using Virtuoso tool.

### 3.1 Circuit Level Implementation of Wallace Tree Multiplier

In the WTM, the partial products were generated by AND gates and the partial product reduction were done with full adders in Static CMOS logic style. With Virtuoso tool in Cadence the circuits were implemented and their symbols were generated to implement the WTM. The circuit level implementation can be seen in the figure 6 and its simulation in figure 7. The worst case power and delay was found to be the worst case power of 2.058  $\mu$ W and 186.0ps respectively.

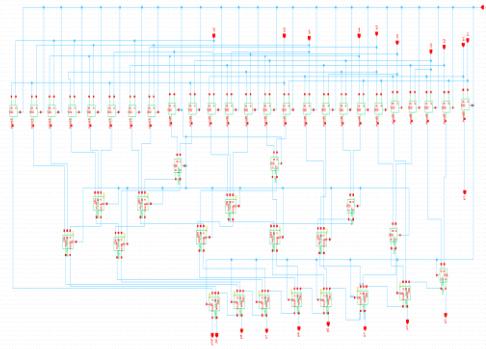


Fig. 6 Circuit Level Implementation of WTM

### 3.2 Circuit Level Implementation of Different 4to2 Compressor Topologies

As shown in figure 4, adders in different logic styles were chosen and implemented in it. The adders chosen for the implementation were Static CMOS logic, pass transistor logic, transmission gate (TG) logic style. The Static CMOS logic used was of 28 transistors. The pass logic gives a threshold loss. Hence a swing restorer pass logic adder of 38 transistors was chosen. The TG adder was of 18 transistors.

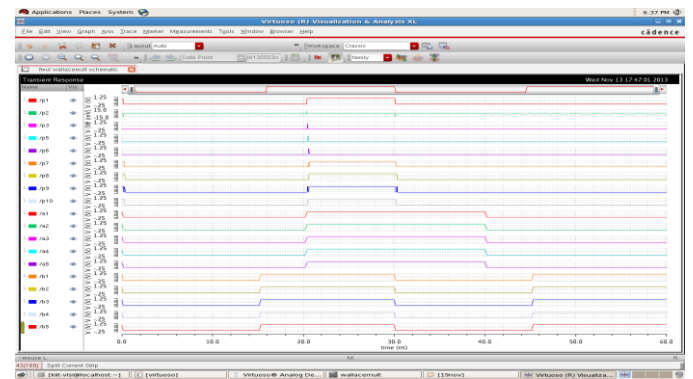


Fig. 7 Simulation Result of the WTM

#### 3.2.1 4to2 Compressors-Topology I using Static CMOS Adder:

The block diagram in figure 2 was implemented with Static CMOS Adder. The transistor implementation and the simulation can be seen in figure 8 and figure 9 respectively. The worst case power and delay is found to be 110nW and 131.7ps. The circuit consists of 56 transistors.

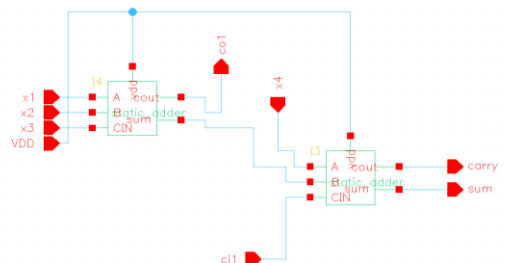


Fig.8 4to2 Compressors Topology I with Static CMOS Adder

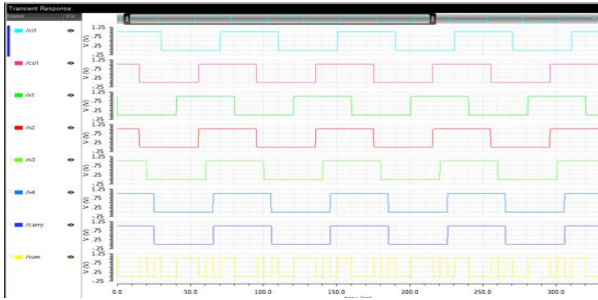


Fig.9 Simulation result of 4to2 compressor Topology I with static adder

### 3.2.2 4to2 Compressors-Topology I using PTL logic Adder:

The next implementation in the figure 2 was done using Complementary PTL adders. This implementation consists of 76 transistors. The simulation and transistor level can be seen in 10 and fig 11. The worst case power and delay was found to be 380nW and 152ps.

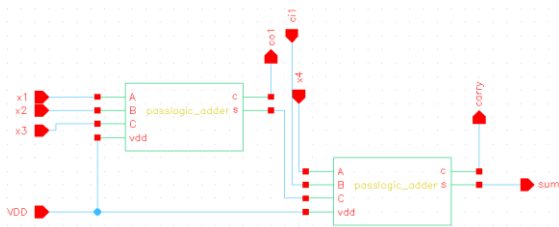


Fig.10 4to2 compressor topology I using Complementary Pass logic Adders

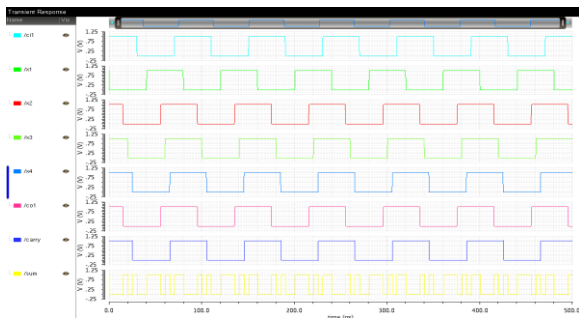


Fig.11 Simulation result of 4to2 Compressor Topology I using Pass logic Adder

### 3.2.3 4to2 Compressors-Topology I using TG Adder:

The TG based adder was implemented in Topology I as in figure 12 whose schematic and simulation can be seen in figure 13. This implementation consists of 36 transistors. The worst case power and delay was found to be 120.9nW and 69.29ps.

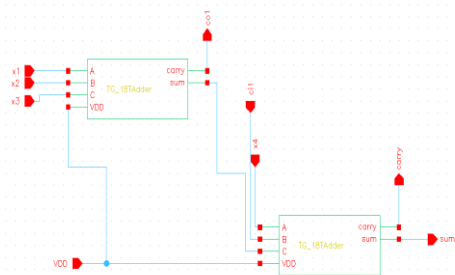


Fig.12 4to2 compressor topology I using TG Adder

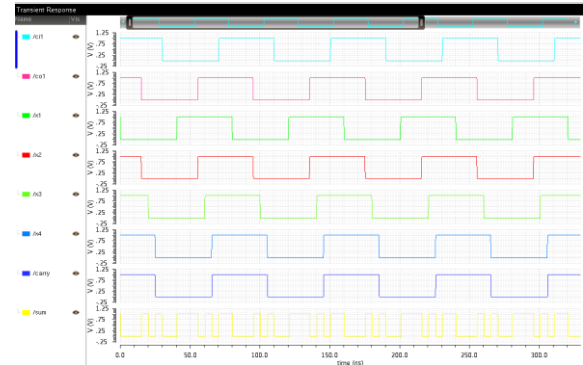


Fig.13 Simulation result of 4to2 compressor topology I using TG Adder

### 3.2.4 4to2 Compressors-Topology II:

In topology II, the logic modules for simultaneous generation of XOR signal and XNOR signal was implemented with 10 transistors. The signals XOR and XNOR from this module were used as select signal in the multiplexer implementation of the sum and carry unit. The carry unit is of multiplexer implementation using TG logic style to have output of full swing. The topology II of 4to2 Compressor consists of 42 transistors. The worst case power and delay of this topology is 42.31nW and 56.73ps respectively. The implementation and simulation can be seen in figure 14 and figure 15.

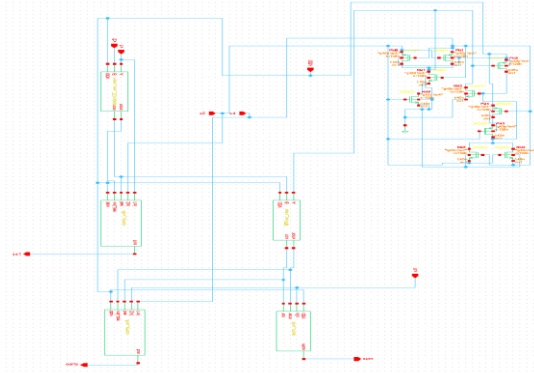


Fig.14 4to2 compressor Topology II with 10T XOR-XNOR circuit

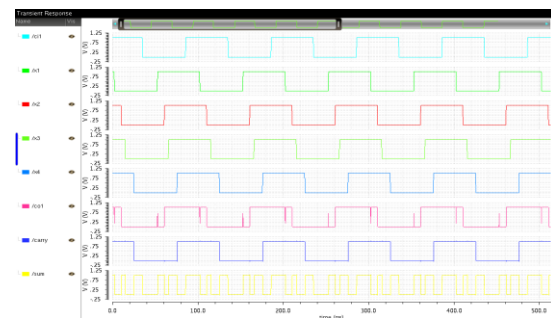


Fig.15 Simulation result of 4to2 compressor Topology II with 10T XOR-XNOR circuit

## 3.3 Circuit Level Implementation of Different 5to2 Compressor Topologies

The 5to2 Compressors were implemented at circuit level just like the 4to2 Compressors and simulated.

### 3.3.1 5to2 Compressor Topology I using Static CMOS Adders:

The topology I of 5to2 Compressor was first implemented using static CMOS adders in the circuit implementation as in figure 4b. The transistor level implementation and simulation can be seen in figure 16 and figure 17. The worst case power and delay of this configuration is found to be 324.1nW and 102 ps respectively.

### 3.3.2 5to2 Compressor Topology I using pass Transistor Logic Adders:

The Complementary pass transistor logic was next implemented in the figure 3.21(b) of topology I 5to2compressor. The transistor level implementation can be seen in figure 18 and figure 19. The worst case power and delay was found to be 833.9nW and 243.6 ps.

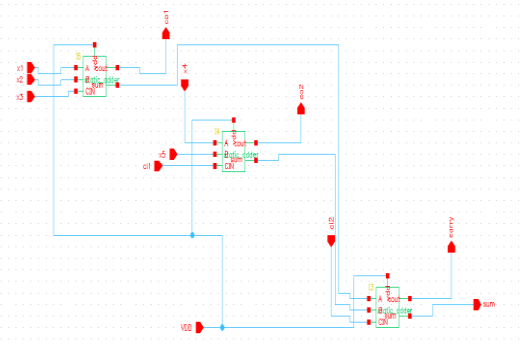


Fig.16 Circuit implementation of the 5to2 Compressor topology I with Static Adders

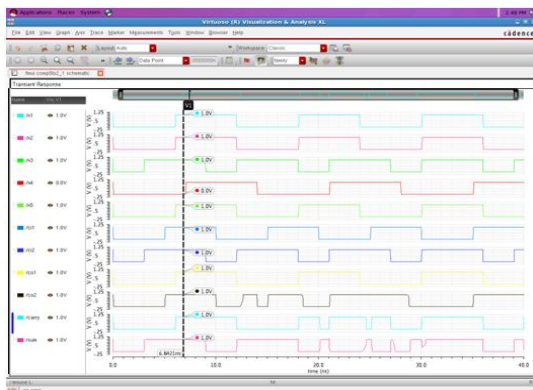


Fig17 Simulation of 5to2compressor topology II using Static adder

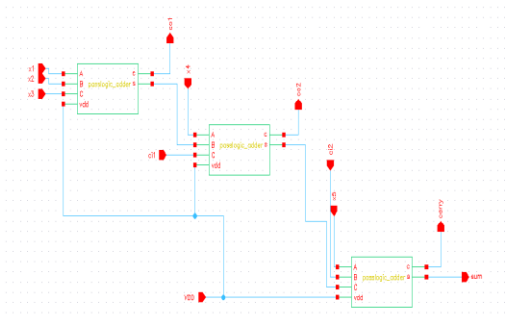


Fig.18 Circuit implementation of the 5to2 Compressor topology I with PTL Adders

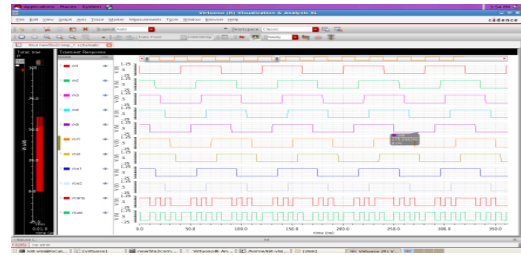


Fig.19 Simulation of 5to2compressor topology II using PTL adder

### 3.3.3 5to2 Compressor Topology I using TG logic Adders:

The TG adder transistor implementation in topology I of 5to 2 compressors can be seen in figure 19 and figure20. The worst case power and delay of this topology was found to be 267.6nW and 43.88ps.

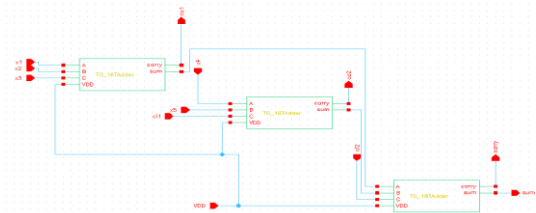


Fig.19 5to2 Compressor topology I circuit implementation with TG Adders

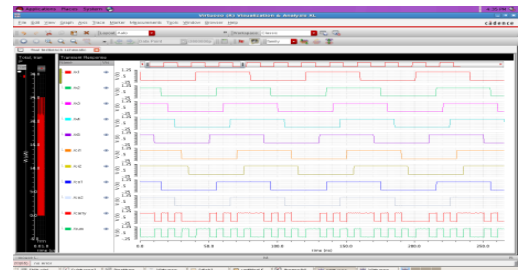


Fig.20 Simulation of 5to2compressor Topology I using TG Adder

### 3.3.4 5to2 Compressors-Topology II:

The XOR-XNOR signal is generated using 10 Transistors circuit. The sum unit and carry unit is implemented using TG logic style. The worst case power and delay of this topology is 320.1nW and 41.34 respectively. The circuit implementation can be seen in figure 21 and figure22.

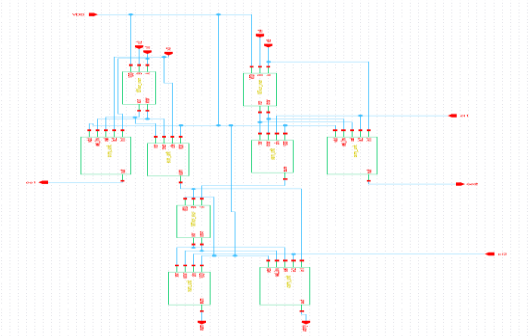


Fig.21 circuit implementation of the 5to2 Compressor topology II

### 3.4 Circuit Level Implementation of Compressed based Multiplier

As from the figure1 the multiplier with compressors was implemented at circuit level.

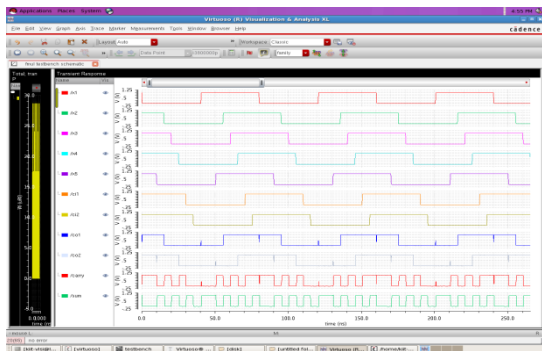


Fig.22 Simulation of 5to2compressor topology II

The circuit 5x5 bit multiplier in figure23was implemented using the different compressors topology and compared which can be seen in section IV.

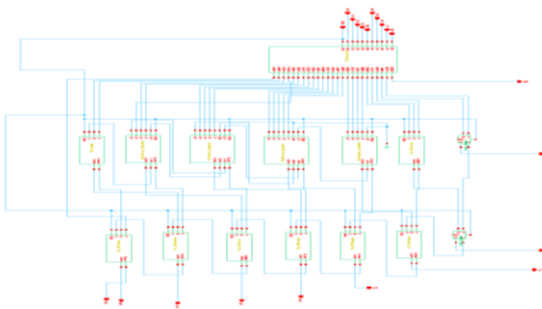


Fig.23 Circuit level implementation of 5x5Multiplier using Compressors

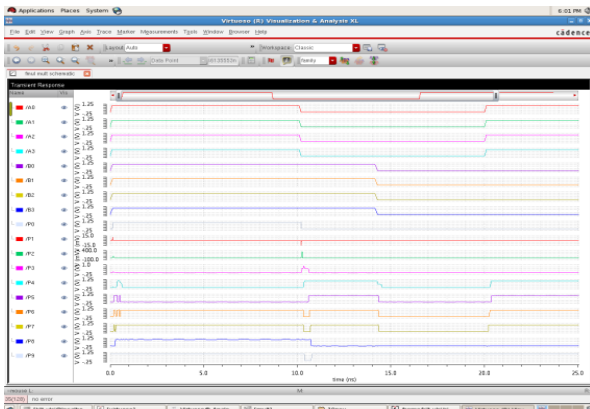


Fig.24 Simulation result of the Compressor based Multiplier

## 4. COMPARISON AND RESULTS

The comparative results of the various 4to2 Compressors can be in the table I. The comparison was done for power, delay and for power-delay product (PDP) and the transistor count.

The different topologies of 5to2 Compressors implemented were compared and can be seen in Table2.

Comparative study of compressors implementation was done. As the results were not that well defined. Hence all the topologies were implemented in the Multiplier unit and compared as seen in Table III.

Table I  
Comparison of Different Topologies of 4to2compressors

Type of 4 to2 Compressor	Transistor Count	Worst case Dynamic Power (nW)	Worst Case Delay (ps)	PDP (e-17)
Topology I Static Adder	56	110	131.7	1.45
Topology I PTL Adder	76	380	152	5.78
Topology I TG Adder	36	120.9	69.29	0.84
Topology II 10T XOR-XNOR	42	42.31	56.73	0.24

The 5to2 compressor results are compared in Table II. To have a better comparative done, the compressor topologies were implemented n the multiplier and compared which can be seen in table III

Table III  
Comparison of Different Topologies of 5to2compressors

Type of 5 to2 Compressor	Transistor Count	Worst case Dynamic Power (nW)	Worst Case Delay (ps)	PDP (e-17)
Topology I Static Adder	84	324.1	102	3.3
Topology I PTL Adder	114	833.9	243.6	20.3
Topology I TG Adder	54	267.6	43.88	1.2
Topology II 10T XOR-XNOR	54	320.1	41.34	1.3

The different multiplier topologies were compared with the first implementation of WTM with Static CMOS logic. In the Table III the various multiplier implementations are compared.

Table IIIII  
Comparison of Different Designs of Compressor based Multiplier

	5x5 multiplier with different 4to2 and 5to2 Compressor		Worst Dynamic Power ( $\mu$ W)	Worst Delay (ps)	Transistor Count	PDP (e-15)
	4to 2 comp.	5to 2 comp.				
1	Top. I Static Adder	Top. I Static Adder	2.144	211.4	678	0.45
2	Top. I PTL Adder	Top. I PTL Adder	3.393	208.8	858	0.71
3	Top. I TG Adder	Top. I TG Adder	1.845	204	498	0.37
4	Top. II	Top. II	1.698	211.5	498	0.36
5	WTM (Static )		2.058	186.0	660	3.83

It can be seen from the comparative Table III, the TG compressor and Topology II compressor based multiplier give a better power delay product (PDP) as compared to the Wallace tree with static CMOS logic. The Compressor based topologies not only have better PDP, but the transistor count is also lessened that is area is optimized. It can also be seen that compressor based reduces the layout complexity with a slight compromise in delay when compared to WTM.

## **5. CONCLUSIONS AND FUTURE SCOPE**

The multiplier at first was implemented with static CMOS logic. We then explored different topologies of compressors and implemented them at circuit level. These topologies were compared and then implemented in the multiplier.

The multiplier implemented was of 5x5bits which could be increased to higher order for the single precision IEEE floating point multiplier and used in it.

## **6. ACKNOWLEDGMENT**

We would like to acknowledge the VLSI design and Embedded Design Laboratory, KIIT University for allowing us to conduct our work.

## **7. REFERENCES**

- [1] Addanki Purna Ramesh, Rajesh Pattimi, High Speed Double Precision Floating Point Multiplier, , International Journal of Advanced Research in Computer and Communication Engineering Vol. 1, Issue 9, ISSN Print : 2319-5940, ISSN Online : 2278-1021, ,(November 2012) pp 647-650.
- [2] Ushasree G, R Dhanabal, Dr Sarat Kumar Sahoo , VLSI Implementation of a High Speed Single Precision Floating Point Unit Using Verilog, Proceedings of 2013 IEEE Conference on Information and Communication Technologies (ICT 2013), 978-1-4673-5758-6/13 (2013), pp 803-808
- [3] Neha Maheshwari, A Design of 4X4 Multiplier using 0.18 um Technology, International Journal of Latest Trends in Engineering and Technology (IJLTET), Vol. 2 Issue 1 ISSN: 2278-621X ( January 2013), pp 251-257
- [4] C.S Wallace, "Suggestion for a fast multiplier", IEEE Transactions on Electronic Computers, Vol.13, pp.14 – 17, 1964
- [5] Rabaey Et Al., Digital Integrated Circuits
- [6] Giuseppe Carso, Daniela Di Sclafani "Analysis of Compressor Architectures in MOS Current-Mode logic" ICES 2010, pp.13-16
- [7] Shahebaj Khan, Sandeep Kakde, Yogesh Suryanvanshi, "VLSI implementation of reduced Complexity Wallace Multiplier Using Energy Efficient CMOS Full Adder," IEEE International Conference on Computational Intelligence and Computing Research, 2013
- [8] V.G.Oklobdzija, D Villeger, S.S Liu, "A method for speed optimized partial product reduction and generation of fast parallel multipliers using an algorithmic approach," IEEE Transactions on Computers, Vol.45, pp.294 – 306, 1996
- [9] Al-Ashrafy, A.Salem, W. Anis, "An efficient implementation of Floating point Multiplier", Conference on Electronics, Communications and Photonics (SIEPCPC) 2011, pp 1-5
- [10] [www.cadence.com](http://www.cadence.com)