Design and Analysis of Compressor Based 16 Bit Multiplier

K. Radha[1], M. Charankumar[2], K. Vasu[3] and K. Vinay Teja[4]
[1]*ECE, Assistant Professor, Sir C R Reddy College of Engineering, India*
[2,3,4] *ECE, Sir C R Reddy College of Engineering, India.*

Abstract: **Now a days the technology is growing day by day with faster rate. Particularly the usage of electronics is increasing in wide range of ways depending on their intended purpose and preferences. In this regard multipliers are playing a vital role because they allow us to perform complex arithmetic operations involving large numbers more efficiently. Instead of performing a series of addition or subtraction operations, a multiplier allows us to perform the operation in a single step within no time that is the challenge of today's world. So, in addition to being more efficient, multipliers also have practical applications in fields such as engineering, computer science, and cryptography also used, for example, in the design of digital circuits and in the encryption and decryption of data. Overall, multipliers playing an important role in mathematics and its applications and are essential tools for performing complex computations efficiently.**

Compressors play a vital role in realizing the high speed multipliers. In error resilient applications such as Image processing, Multimedia and Matrix multiplication the approximate computing is used, which provides meaningful results faster with lower power consumption. The compressors are designed using the full adders which provides accurate results in the existing work. The 4:2 and 5:2 compressors are then introduced with delay reduction and ADP reduction. Now the further work concentrated on the implementation of 7:2 Compressor based multiplier, to further enhance the performance of multipliers. The proposed design will be provide maximum extent of reduction in area, delay or power consumption and achieves improvement in terms of speed as compared to the 4:2 and 5:2 compressor based multiplier.

Keywords: **Multiplier, Compressors, Half Adder, Full Adder, Area.**

1 INTRODUCTION

Multipliers are one of the most fundamental components in computer arithmetic and are frequently used in various digital signal processors, computer graphics, scientific calculations, image processing, and other applications. Three sequential phases are present in the process of multiplication. 1. Partial product generation 2.Partial product reductions. 3. Propagating addition is final computation. A significant number of compressors are used in multiplication to carry out the partial product addition in high-speed error resilient applications.

Typically, several different types of compressors, such as 3:2, 4:2 & 5:2 have been extensively used to achieve partial product addition and overall, the reduction of partial products during multiplication add mostly causes to the overall delay, power, and area. So, the latency of this stage is decreased by using compressors. Most of the time multipliers causes the overhead to the computer arithmetic units. So, multiplier unit consumes maximum power, and it occupies a more space in filter design. Therefore, the compressor-based multipliers were designed to minimize these effects. These multipliers will perform electrically more effectively in terms of area, power, and delay.

A compressor with the size x: y denotes that it has 'x' number of inputs and 'y' number of outputs. High-order compressors have the potential to simultaneously reduce both the vertical critical path and stage operations, resulting in improved power and speed performance. Exact and accurate calculations are not necessarily required in many signals or Image processing and Multimedia applications. Since these systems are error tolerant and generate output that is suitable for human perception. Compressors are used to improve the performance of circuits created for high-speed applications. Xilinx Vivado 2019.1 is the computer program used to simulate and synthesize multipliers and compressors. For the synthesis and analysis of HDL designs, Xilinx Vivado is a discontinued software package from the company. It was primarily used to generate embedded firmware for the FPGA (field programmable gate array) and CPLD (complex programmable logic device) product families as well.

2 EXISTING WORK

The multipliers using 4:2 compressors, 5:2 compressors and the exact 7:2 compressors are proposed earlier. Mostly used 3:2 compressors (full adders) need 5 steps in 8-bit multiplication, it is known that 4-2 compressor can reduce the number of steps in partial product reductions with efficient hardware costs for designing the fast arithmetic units. Similarly, the 5:2 Compressor based multipliers have better optimized performance as compared to the 4:2 Compressor based multipliers. These compressors are commonly implemented by using the full adders. The number of full adders that are used in implementing the compressorsdepends on the number of inputs that they have. The compressors which are designed using full adders i.e., 3:2 compressors are referred as exact compressors [4].

2.1 4:2 Compressor:

The general structure of a 4:2 compressor using full adders is shown in the figure 1. It consists of five inputs, three outputs and two cascaded full adders. A1, A2, A3, A4 and Cin are the inputs and Cout, Carry and Sum. On applying multipliers to 4:2 compressors. The area and time delay can reduced compared with normal multiplier.

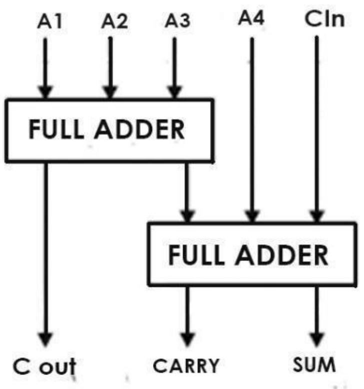


Figure 1. 4:2 compressor

2.2 5:2 compressor:

The basic structure of 5:2 Compressor using full adders is shown below figure 2. So this General block diagram of a 5: 2 compressor consists of three full adders are cascaded together. It has five inputs (X1, X2, X3, X4, and X5) along withcarries Cin1 and Cin2 and two outputs (Sum and Carry) along with Cout. The 5:2 compressor based 16 multiplier can reduce area time delay compared to 4:2 compressor based 16 multiplier [5].

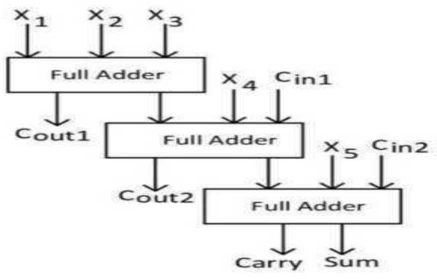


Figure 2. 5:2 compressor

2.1.2 Compressor:

To design a 7:2 compressor using full adders, we need to take the 7-bit input and compress it into a 2-bit output. The input bits will be divided into three groups, each consisting of three bits, two bits, and two bits. We will then use a combination of full adders and half adders to compress the input bits.

The 7:2 compressor consist of 7 inputs (A1, A2, A3, A4, A5, A6, and A7) and 2 carries from previous stage(Cin1, Cin2). It has 4 outputs consisting of one sum and three carries. The general 7:2 Compressor using full adders is shown in the below figure 3.

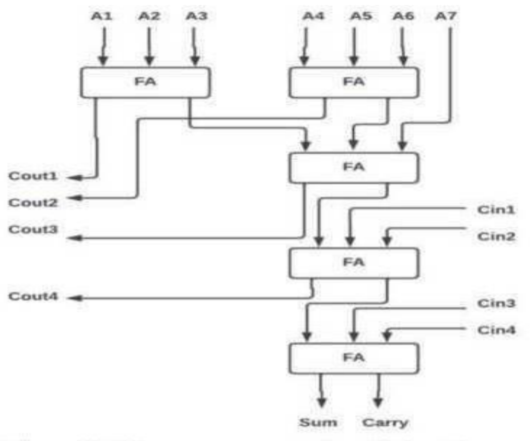


Figure 3. 7:2 compressor

2.2 Dadda multiplier Method

Dadda multiplier is of course a more efficient and faster method of multiplication than the traditional methods, such as the shift-and-add method or the booth algorithm. The Dadda multiplier is one type of a digital circuit that multipliestwo unsigned integers. Dadda multiplier uses less space andis faster than Wallace tree multiplier. The area reduction in this multiplier is caused by a general reduction in partial product levels by using Dadda compression technique often in digital multipliers.

The Dadda multiplier uses a tree-like structure to perform the multiplication operation. The tree is built

up of several stages, with each stage consisting of a number of parallel adders. The first stage of the tree consists of half adders, which take two bits as inputs and produce a sum and carry bitas outputs. The second stage consists of full adders, which take three bits as inputs and produce a sum and carry bit as outputs. Each subsequent stage has more full adders than the previous stage. To perform a multiplication using the Dadda multiplier, the two binary numbers are first represented as bitarrays of equal length. The two bit arrays are then placed at the top of the tree, with the bits flowing down the tree from the top to the bottom. At each stage of the tree, the bits are added together using the parallel adders, and the results are passed down to the next stage of the tree. The final result is obtained from the output of the lowest stage of the tree.
Wallace tree and Dadda multipliers both have the same number of partial product addition levels for 2 bit operand multiplication, but as the operand bit size increases for 4- bit 8 bit, and more, the partial product addition levels of the Dadda multiplier greatly decrease compared to the Wallace tree multiplier, and the Dadda multiplier is also faster [6].

Table 1: No. of reduction stages for Dadda Multiplier

Bits in Multiplier(N)	Number of Stages
3	1
4	2
5 < N< 6	3
7 < N < 9	4
10 <N< 13	5
14 <N< 19	6
20 < N < 28	7
29 < N < 42	8
43 < N < 63	9
63 < N < 94	10

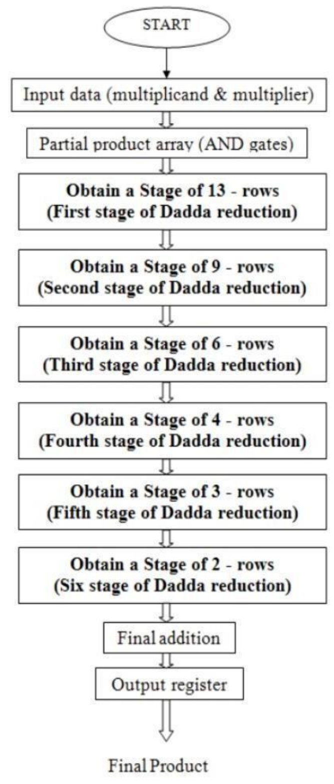
The reduction rules are as follows in terms of rules: Rule1: When three wires of identical weight are fed into a full adder, the output will consist of one wire of the same weight as the inputs and an additional wire of higher weight for every group of three input wires.

Rule2: If two wires of equal weight remain and the current count of output wires with that weight is divisible by 3 with aremainder of 2, then feed them into a half adder. Otherwise, simply forward them to the next layer.

Rule3: When there is only one wire remaining, it should be connected to the next layer. This process performs the required number of additions to maintain the output weights close to a multiple of 3, which is ideal when utilizing full adders as 3:2 compressors.

Rule4: If a layer contains a maximum of three input wires foreach weight, then it will be the final layer in the Dadda tree. In such cases, the Dadda tree will utilize half adders more extensively (though not to the same extent as in a Wallace multiplier) to guarantee that each weight has only two outputs.

Consequently, the second rule mentioned earlier undergoes a modification is as follows If there are two wiresof the same weight left, and the current number of output wires with that weight which is equal to 1 or 2 (nothing but modulo 3), input them into a half adder of the digital circuit.Otherwise, it will be pass to the next layer [9]. Some similar compressor circuits can be observed in [
The below figure 4 shows the 16 bit by 16 bit dada multiplier shows how the partial product reduction is followed stage by stage with the help of flowchart in 6 stagesconsecutively followed for getting the final addition which isstored in the output register and finally the output will be available as an end result is shown here in this flow chart.


START
Input data (multiplicand & multiplier)
Partial product array (AND gates)
Obtain a Stage of 13 - rows (First stage of Dadda reduction)
Obtain a Stage of 9 - rows (Second stage of Dadda reduction)
Obtain a Stage of 6 - rows (Third stage of Dadda reduction)
Obtain a Stage of 4 - rows (Fourth stage of Dadda reduction)
Obtain a Stage of 3 - rows (Fifth stage of Dadda reduction)
Obtain a Stage of 2 - rows (Six stage of Dadda reduction)
Final addition
Output register
Final Product
Figure 4 Flow chart of 16*16 Dadda multiplier

The Dadda multiplier algorithm efficiently reduces the number of partial products and performs the multiplication with fewer logic gates than traditional multiplication methods. This 16-bit Dadda multiplier takes 6 stages of reduction to obtain the final product. The partial product reduction is represented in terms of dot diagram as shown below figure 5.

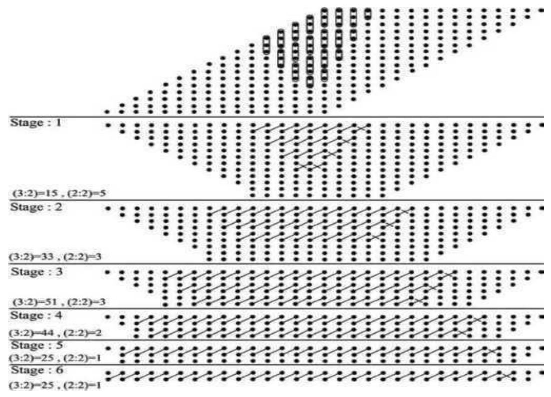


Figure 5: Dot diagram of 16*16 Dadda multiplier

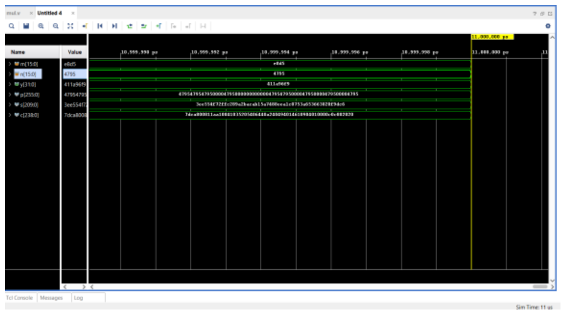
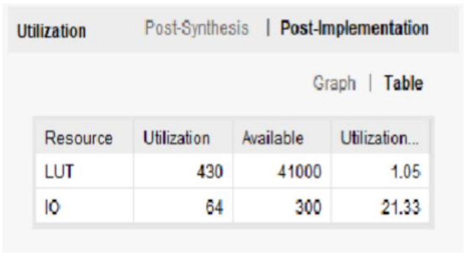


Figure 6: Simulation result for 16 bit dadda multiplier

The above figure shows the simulation result for 16bit dadda multiplier

In the above figure there are two inputs m=e8d5, n=4795 the inputs in Hexadecimal

Output y = 411a96f9 the output in Hexadecimal

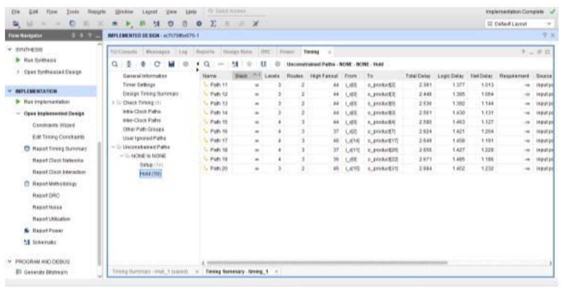


Utilization   Post-Synthesis | Post-Implementation

Graph | Table

Resource	Utilization	Available	Utilization...
LUT	430	41000	1.05
IO	64	300	21.33

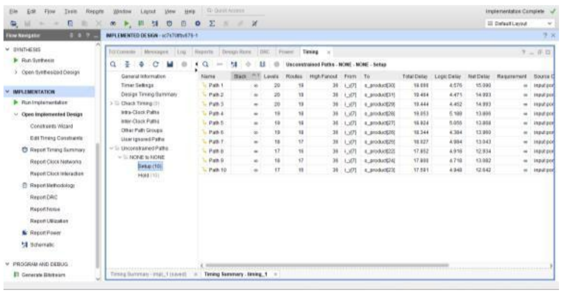
Figure 7: The above figure shows utilization report for 16 bit dadda multiplier .The 16 bit dadda multiplier utilizes the 430 out of 41000 LUT'S (1.05%).



Unconstrained Paths - NONE - NONE - Hold

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source
Path 11	∞	3	2	44	I_x[0]	c_product[2]	2.381	1.377	1.013	-∞	input p
Path 12	∞	3	2	44	I_x[0]	c_product[3]	2.448	1.395	1.054	-∞	input p
Path 13	∞	3	2	44	I_x[0]	c_product[5]	2.536	1.392	1.144	-∞	input p
Path 14	∞	3	2	44	I_x[0]	c_product[6]	2.561	1.430	1.131	-∞	input p
Path 15	∞	4	3	44	I_x[0]	c_product[4]	2.590	1.463	1.127	-∞	input p
Path 16	∞	4	3	37	I_x[2]	c_product[7]	2.624	1.421	1.204	-∞	input p
Path 17	∞	4	3	46	I_x[14]	c_product[17]	2.649	1.458	1.191	-∞	input p
Path 18	∞	4	3	37	I_x[11]	c_product[20]	2.656	1.427	1.229	-∞	input p
Path 19	∞	4	3	36	I_x[8]	c_product[22]	2.671	1.485	1.186	-∞	input p
Path 20	∞	3	2	46	I_x[15]	c_product[31]	2.684	1.452	1.232	-∞	input p

Figure 8: The above figure shows the setup timing report for 16 bit dadda multiplier.



Unconstrained Paths - NONE - NONE - Setup

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source
Path 1	∞	20	19	38	I_y[7]	s_product[30]	19.666	4.576	15.090	∞	input por
Path 2	∞	20	19	38	I_y[7]	s_product[31]	19.464	4.471	14.993	∞	input por
Path 3	∞	20	19	38	I_y[7]	s_product[29]	19.444	4.452	14.993	∞	input por
Path 4	∞	19	18	38	I_y[7]	s_product[28]	19.053	5.189	13.864	∞	input por
Path 5	∞	19	18	38	I_y[7]	s_product[27]	18.924	5.056	13.868	∞	input por
Path 6	∞	19	18	38	I_y[7]	s_product[26]	18.344	4.384	13.960	∞	input por
Path 7	∞	18	17	38	I_y[7]	s_product[25]	18.027	4.984	13.043	∞	input por
Path 8	∞	17	16	38	I_y[7]	s_product[22]	17.852	4.918	12.934	∞	input por
Path 9	∞	18	17	38	I_y[7]	s_product[24]	17.800	4.718	13.082	∞	input por
Path 10	∞	17	16	38	I_y[7]	s_product[23]	17.591	4.948	12.642	∞	input por

Figure 9: The above figure shows the Hold timing report for 16 bit dadda multiplier

$$\text{Time delay} = \frac{\text{Highest total delay of setup} + \text{Highest total delay of Hold}}{2}$$

Time Delay = 11.175ns

2.35:2 Compressor based 16 bit dadda multiplier

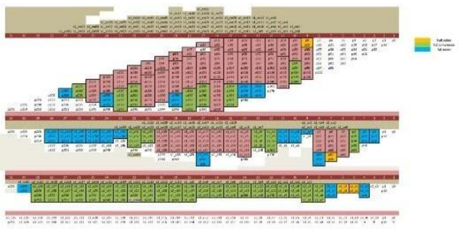


Figure 15: Diagram of 5:2 compressor based 16 bit dada multiplier

Usually, a multiplier is made up of three parts. To create partial products in the first section, AND gates are used. Compressors are used in the second stage to lower the PPM maximum height (partialproduct matrix).To get the final output, a carry propagation adder is employed in the third step. The PPM reductioncircuitry is therefore a key contributor to the multiplier's design complexity. The PPM reduction circuitry's optimization is the main emphasis of designs [1-6]. We present a 16*16 multiplier design in this part. According to their significance weights are divided into three categories: higher significance weights, medium significance weights, and lower significance weights. The number of higher significance weights, intermediate significance weights, and lower significance weights can all be changed by the designers to trade-off between power consumption and computation accuracy. Our PPM reduction circuitry uses significance driven logic compression, which consumes less power with a little amount of error While the lower weights employ inaccurate(5:2) area efficient compressors, the higher weights use exact (5:2) compressors, the middle weights use two stage approximate (5:2) compressors (OR-tree based approximation). In the second and third stages, significance weights are taken into consideration. When the second and third phases are complete, there are no more than two product phrases for each weight.

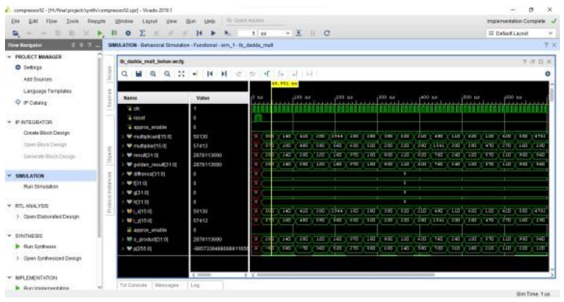
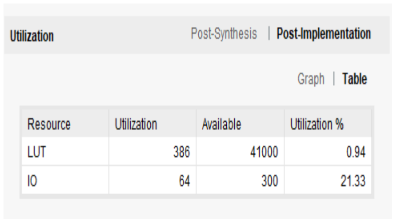


Figure 16: The above figure shows the simulation result for 5:2 compressor based 16 bit dadda multiplier. In the above result inputs m= 50130, n= 57413 inputs are in unsigned, Output = 2878113690 output is in unsigned.



Utilization | Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	386	41000	0.94
IO	64	300	21.33

Figure 17: The above figure shows utilization report for 5:2 compressor based 16bit dadda multiplier. The 5:2 compressor based 16 bit dadda multiplier utilizes the 386 out of 41000 LUT'S (0.94%).

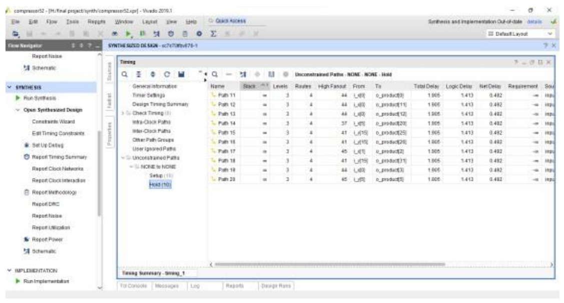


Figure 18: Above figure shows the setup timing report for 5:2 compressor based 16 bit dadda multiplier.

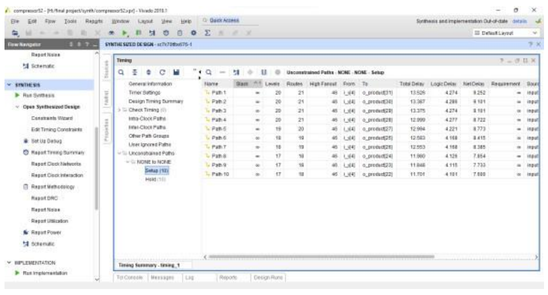


Figure 19: Above figure shows the Hold timing report for 5:2 compressor based 16 bit dadda multiplier.

Time delay = Highest total delay of setup + Highest total

$$\frac{\text{delay of Hold}}{2}$$

Time delay = 7.715ns

3 PROPOSED WORK

3 7:2 Compressor based 16 bit dadda multiplier

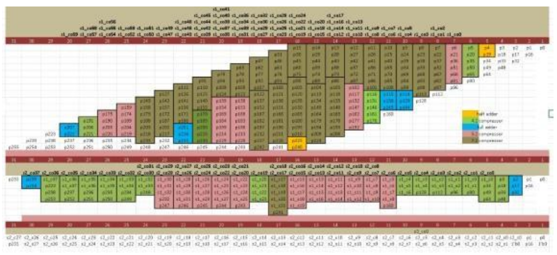


Figure 20: Diagram of 7:2 Compressor based 16 bit dadda multiplier.

The 7:2 compressor based Dadda multiplier is a form of digital multiplier circuit used to multiply two binary values together. It is based on the Dadda algorithm, a power full technique for multiplying binary numbers. The 7:2 compressor based Dadda multiplier's fundamental idea is to divide the multiplication problem into smaller subproblems each of which may be resolved using a straightforward 7:2 compressor circuit. A larger compressor circuit receives the output from each compressor circuit after which it mixes the outputs to produce the finished product. The fundamental benefit of the Dadda multiplier based on the 7:2 compressor is that it uses less adders and partial products than other multipliers, like the Wallace tree multiplier.

To develop a 16-bit Dadda multiplier based on a 7:2 compressor, follow these steps:

Step 1: The 16-bit multiplicand and multiplier are first divided into fourgroups of four bits each.

Step 2: Multiply the appropriate group of four multiplicand bits to generate the partial products for each multiplier bit.

Step 3: Organize the unfinished items into three rows of information of level 0, level 1, and level 2. Each level will have a different number of rows, and each row will contain a varied amount of partial products.

Step 4: Create the sum and carry signals for each row using the 7:2 compressor.

Step 5: Propagate the carry signals through the rows to

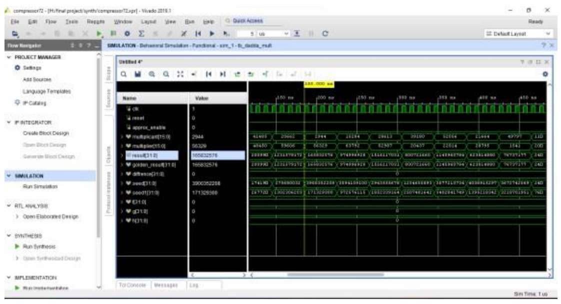
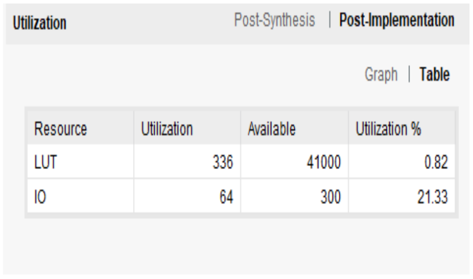


Figure 21: The above figure shows the simulation result for 7:2 compressor based 16 bit dadda multiplier

In the above result inputs m= 2944, n= 56329 inputs are in unsigned, Output = 165832576 output is in unsigned.



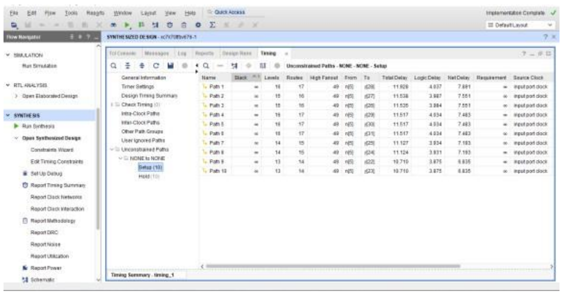
Utilization Post-Synthesis | Post-Implementation

Graph | Table

Resource	Utilization	Available	Utilization %
LUT	336	41000	0.82
IO	64	300	21.33

Figure 22: The above figure shows utilization report for 7:2 compressor based 16bit dadda multiplier.

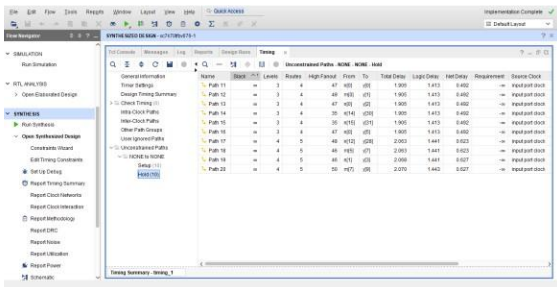
The 7:2 compressor based 16 bit dadda multiplier utilizes the 336 out of 41000 LUT’s (0.82%).



Unconstrained Paths - NONE - NONE - Setup

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock
Path 1	∞	16	17	49	n[5]	r[28]	11.928	4.037	7.891	∞	input port clock
Path 2	∞	15	16	49	n[5]	r[27]	11.538	3.987	7.551	∞	input port clock
Path 3	∞	15	16	49	n[5]	r[26]	11.535	3.984	7.551	∞	input port clock
Path 4	∞	16	17	49	n[5]	r[29]	11.517	4.034	7.483	∞	input port clock
Path 5	∞	16	17	49	n[5]	r[30]	11.517	4.034	7.483	∞	input port clock
Path 6	∞	16	17	49	n[5]	r[31]	11.517	4.034	7.483	∞	input port clock
Path 7	∞	14	15	49	n[5]	r[25]	11.127	3.934	7.193	∞	input port clock
Path 8	∞	14	15	49	n[5]	r[24]	11.124	3.931	7.193	∞	input port clock
Path 9	∞	13	14	49	n[5]	r[22]	10.710	3.875	6.835	∞	input port clock
Path 10	∞	13	14	49	n[5]	r[23]	10.710	3.875	6.835	∞	input port clock

Figure 23: Above figure shows the setup timing report for 7:2 compressor based 16 bit dadda multiplier.



Unconstrained Paths - NONE - NONE - Hold

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock
Path 11	∞	3	4	47	m[0]	y[0]	1.905	1.413	0.492	-∞	input port clock
Path 12	∞	3	4	49	m[3]	y[1]	1.905	1.413	0.492	-∞	input port clock
Path 13	∞	3	4	47	m[0]	y[2]	1.905	1.413	0.492	-∞	input port clock
Path 14	∞	3	4	35	m[14]	y[30]	1.905	1.413	0.492	-∞	input port clock
Path 15	∞	3	4	35	m[15]	y[31]	1.905	1.413	0.492	-∞	input port clock
Path 16	∞	3	4	47	m[0]	y[5]	1.905	1.413	0.492	-∞	input port clock
Path 17	∞	4	5	48	m[12]	y[28]	2.063	1.441	0.623	-∞	input port clock
Path 18	∞	4	5	46	m[5]	y[7]	2.063	1.441	0.623	-∞	input port clock
Path 19	∞	4	5	46	m[1]	y[3]	2.068	1.441	0.627	-∞	input port clock
Path 20	∞	4	5	50	m[7]	y[9]	2.070	1.443	0.627	-∞	input port clock

Figure 24: Above figure shows the Hold timing report for 7:2 compressor based 16 bit dadda multiplier.

$$\text{Time delay} = \text{Highest total delay of setup} + \frac{\text{Highest total delay of Hold}}{2}$$

Time delay = 6.6855ns

obtain the result,then combine the sum signals.

4 RESULTS

Comparison Table:

Multipliers	No of LUT’s	Time Delay
16 bit Dadda multiplier	430	11.175ns
4:2 compressors based 16 bit Dadda multiplier	400	9.527ns
5:2 compressors based 16 bit Dadda multiplier	386	7.715ns
7:2 compressors based 16 bit Dadda multiplier	336	6.685ns

Table 2: The above table shows the comparison of parameters between various multipliers

As from the table, it has been observed that the proposed design of 7:2 compressor based 16 bit dadda multiplier shows reduction in delay and device utilization as compared to the existed normal dadda multiplier and dadda multipliers using 4:2 and 5:2 compressors.

5 APPLICATION OF COMPRESSOR BASED DADDA MULTIPLIER

Some possible applications of the 7:2 compressor based 16 bit Dadda multiplier include:

Digital signal processing (DSP): Multipliers are a key component in many DSP applications, such as audio and video processing, speech recognition, and image processing.The 7:2 compressor based Dadda multiplier can be used in these applications to perform high-speed multiplication operations. Cryptography: Cryptographic algorithms, such asRSA and elliptic curve cryptography, rely heavily on large integer multiplication. The 7:2 compressor based Dadda multiplier can be used to implement these algorithms efficiently, reducing the computation time required for encryption and decryption Computer vision: Many computer vision algorithms, such as object recognition and tracking, require intensive matrix operations. The 7:2 compressor based Dadda multiplier can be used to perform these operations efficiently enabling real-time performance on embedded systems. Artificial intelligence (AI): Machine learning algorithms, such neural networks and deep learning, require a large number of multiplication operations. The 7:2compressor based Dadda multiplier can be used in these applications to speed up the computation time required for training and inference.

6. CONCLUSION

The proposed design is a flexible and effective circuit with a wide range of applications in artificial intelligence, processing. It is computer vision, and digital signal designed to reduce the area and power by reducing the count of adder units. According to the results, a remarkable improvement interms of delay and area is achieved. The delay occurred in multiplier while using 7:2 compressor is less when compared to 4:2 compressor and 5:2 compressor. So, the proposed design will be very effective in terms of speed of operation.

7. REFERENCES

[1] S. Venkatachalam and S.-B. Ko, "Design of power and area efficient approximate multipliers," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 25, no. 5, pp. 1782–1786, May2017.

[2] Z. Yang, J. Han, and F. Lombardi, "Approximate compressors for error-resilient multiplier design," in Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFTS), Oct. 2015, pp. 183–186.

[3] C.H. Lin and J.C. Lin, "High accuracy approximate multiplier with error correction," in Proc. IEEE 31st Int. Conf. Comput. Design (ICCD), Oct. 2013, pp.33-38.

[4] O. Akbari, M. Kamal, A. Afzali-Kusha, and M. Pedram, "Dual-quality 4:2 compressors for utilizing in dynamic accuracy configurable multipliers," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 25, no. 4, pp. 1352–1361, Apr. 2017.

[5] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," IEEE Trans. Comput.- Aided Design Integr. Circuits Syst., vol. 32, no. 1, pp. 124– 137, Jan. 2013.

[6] F. Ebrahimi-Azandaryani, O. Akbari, M. Kamal, A. Afzali-Kusha, and M. Pedram, "Block- based carry speculative approximate adder for energy-efficient applications," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 67, no. 1, pp. 137–141, Jan. 2020.

[7] D. Esposito, A. G. M. Strollo, E. Napoli, D. De Caro, and N. Petra, "Approximate multipliersbased on new approximate compressors," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 65, no. 12,pp. 4169– 4182, Dec. 2018.