International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2015): 6.391

Design and Analysis of Advanced Booth Dadda Multiplier Using Approximate Compressors

Aswathy V Nair[1], Manjusree S[2]

[1] PG Student, Embedded Systems, Dept. of Electronics and Communication, Sree Buddha College of Engineering, Pattoor

[2] Assistant Professor, Dept. of Electronics and Communication, Sree Buddha College of Engineering, Pattoor

Abstract: ***Processors processes on the basis of arithmetic operations. Among the basic arithmetic operations, the most complex is the multiplication as it consumes more time for generation of partial products. So a faster multiplier is required to speed up the processor. A faster multiplier can be accomplished by the reduction of partial products. Here in this paper, Radix 4 booth algorithm is used for partial product reduction and Dadda multiplier is used to reduce the number of adders. This multiplier can be called as Advanced Booth Dadda Multiplier as two methods are clubbed together for different operations. Approximate compressors are utilized in the reduced partial products to minimize the delay and no. of transistors. Two different designs for utilizing the proposed approximate compressors are proposed and analyzed for Booth Dadda Multiplier. Detailed simulation results are provided. The utilization of proposed approximate compressors in Dadda Multiplier and its use in advanced Booth Dadda Multiplier is analyzed and is compared with the normal multiplier. The results show that there is a significant reduction in the delay and number of transistors when compared with the normal multiplier.***

Keywords: Radix 4 booth algorithm, Dadda multiplier, Approximate Compressors, Booth Dadda Multiplier, Advanced Booth Dadda Multiplier

1. Introduction

Multiplication is the basic arithmetic operation for several microprocessors and digital signal processing applications. Multipliers are utilized within the arithmetic logic units of microprocessors and to implement DSP algorithms such as convolution and filtering, multipliers are required in the digital signal processing systems. Multipliers lies directly within the critical path in most systems, due to the demand for high speed multiplication. For higher speeds, column compression multipliers are most popular. Wallace introduced the first column compression multiplier. L. Dadda modified the Wallace multiplier and it is slightly faster than the Wallace multiplier. Also the hardware requirement is less. Both of these multipliers consist of three stages. In the first stage, the partial product matrix is formed. This partial product matrix is reduced by half in the second stage. In the final stage, by using a carry propagating adder, these two rows are combined. In the first stage, the multiplicand and the multiplier are multiplied bit by bit to generate partial product. To reduce the partial product to half and to generate all partial products in parallel, a modified Booth algorithm has been used in this stage. The most important stage is the second stage as it is complicated and determines the overall speed of the multiplier. For high speed design, Dadda multiplier is used to add the partial products in order to produce two rows of partial products that can be added in the last stage. The addition of the partial products can be done using 4:2 compressors.

At nanometric scales, approximate computing is an attractive option for digital processing. It is interesting for computer arithmetic designs. Approximate computation makes the existing design process of digital circuits and systems by taking advantage of a decrease in complexity and cost with potential increase in performance and power efficiency. Approximate computing uses this property to design simplified yet approximate circuits operating at higher performance and or lower power compression compared with precise logic circuits.

This paper is arranged as follows: First section contains the related works and next section contains the compressor architecture and design and analysis of two novel approximate 4-2 compressors. When compared with the exact compressors, it is obtained that these simplified compressors have reduced delay and area. These approximate compressors are then used in the partial product module of a Dadda multiplier. For inexact multiplication, two different schemes are proposed. Delay and area are analyzed for the two different schemes of the inexact Dadda multiplier. The next section describes the Booth Algorithm and its implementation in the Dadda Multiplier. Then the next provides the design of Advanced Booth Dadda Multiplier using the Approximate Compressors. Extensive simulation results are also provided. There is significant advancements in the delay and number of transistors.

2. Related Works

Computing systems which are increasingly becoming embedded and mobile are mainly designed on the basis of energy efficiency. Various application such as media processing, recognition and data mining depend on the computational tasks. In these applications, the common factor is that perfect result is not necessary and an inexact or less than optimal result is sufficient. That is, these applications can tolerate errors and imprecision in computation and still providing acceptable and useful results. Accurate and precise computations are not necessary in these applications as its results mainly focus on delay, area and power consumption. So, when compared with the precise

logic circuits approximate computing provides better results by designing the system by decreasing the complexity and operating at higher performance and or lower power consumption [1].

Since addition and multiplication are the widely used computations in the computer arithmetic designs, it is necessary to speed up these operations. Full adder cells have been extensively used for approximate computing of addition [2]-[4]. But the design of approximate multiplier has got less attention. This is because it would be very inefficient in terms of precision and performance. [4] [5] [6] [7] proposes several approximate multipliers. In [4] an imprecise array multiplier is designed that can be utilized in neural network applications by removing some adders in the array. This can be achieved by omitting some of the least significant bits in the partial products. A truncated multiplier is proposed in [5]. This multiplier calculates the sum of the n+k most significant columns of the partial products and truncates the other n-k columns. Larger multiplier arrays are designed using a simplified and thus inaccurate 2X2 multiplier block are proposed in [7].

Column compression multipliers are most popular for high speed operations. The two well-known fast multipliers are those presented by Wallace [8] and Dadda [9]. Both these multipliers consist of three stages. Partial products are reduced as soon as possible in the Wallace method. But in the Dadda method, does the minimum reduction necessary at each level to perform the reduction in the same number of levels as required by a Wallace Multiplier. And it is proved that the Dadda Multiplier is faster than the Wallace Multiplier [10].

The speed of the multiplier can be increased by using the Booth algorithm for the reduction of partial products. This reduced partial products is utilized in the Dadda Multiplier. Compressors can be widely used for the partial product reduction tree to speed up the processor and to decrease power consumption [8]-[10]. Several compressor designs are proposed in [8], [11]-[16]. [17], [18] consider the compression for approximate multiplication also. The design for approximate compressors are described in [18] but not explaining multiplication using these compressors.

3. Compressor Architecture

Figure 1 shows the 4-2 compressor architecture. It has five inputs, X1, X2, X3, X4, Cin and three outputs, Cout, Carry, and Sum. The following equations give the outputs and table I provides the truth table.

$$Sum = X1 \oplus X2 \oplus X3 \oplus X4 \oplus Cin \quad (1)$$

$$Cout = (X1 \oplus X2)X3 + \overline{(X1 \oplus X2)}X1 \quad (2)$$

$$Carry = (X1 \oplus X2 \oplus X3 \oplus X4)Cin + \overline{(X1 \oplus X2 \oplus X3 \oplus X4)}X4 \quad (3)$$

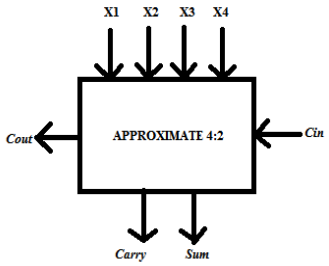

Figure 1: 4:2 Compressor

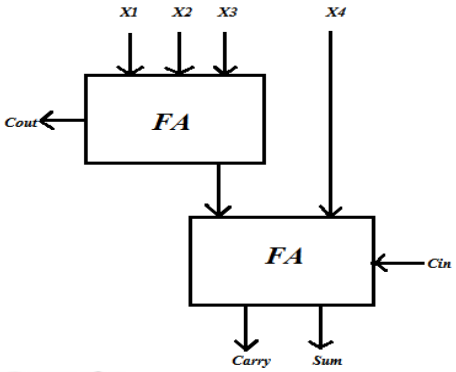

Figure 2: Implementation of 4:2 Compressor

In [8] gives the common implementation of 4:2 compressor using full-adder (FA) cells (figure 2).

Table 1: Truth Table of 4:2 compressors

Cin	X4	X3	X2	X1	Cout	Carry	Sum
0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	1
0	0	0	1	0	0	0	1
0	0	0	1	1	1	0	0
0	0	1	0	0	0	0	1
0	0	1	0	1	1	0	0
0	0	1	1	0	1	0	0
0	0	1	1	1	1	0	1
0	1	0	0	0	0	0	1
0	1	0	0	1	0	1	0
0	1	0	1	0	0	1	0
0	1	0	1	1	1	0	1
0	1	1	0	0	0	1	0
0	1	1	0	1	1	0	1
0	1	1	1	0	1	0	1
0	1	1	1	1	1	1	0
1	0	0	0	0	0	0	1
1	0	0	0	1	0	1	0
1	0	0	1	0	0	1	0
1	0	0	1	1	1	0	1
1	0	1	0	0	0	1	0
1	0	1	0	1	1	0	1
1	0	1	1	0	1	0	1
1	0	1	1	1	1	1	0
1	1	0	0	0	0	1	0
1	1	0	0	1	0	1	1
1	1	0	1	0	0	1	1
1	1	0	1	1	1	1	0
1	1	1	0	0	0	1	1
1	1	1	0	1	1	1	0
1	1	1	1	0	1	1	0
1	1	1	1	1	1	1	1

4. Approximate Compressors

In order to reduce the delay, number of transistors and power consumption thereby improving the efficiency it is necessary to design an approximate design for the 4:2 compressors. To

design an approximate compressor, it is possible to substitute the full adder cells in figure 2 by approximate full-adder cells. But it produces significant error rate [2].So to reduce the error rate two designs are proposed, and it is shown that there is a significant reduction in the error rate and performance improvement compared to the exact compressor module.

4.1. Design 1

From the Table 1, it is clear that the value of the carry output in the exact compressor is same for the Cin input in 24 out of 32 states. Thus, an approximate design must consider this feature. By changing the value of carry for the other 8 states, design 1 can be made.

$$Carry' = Cin \quad (4)$$

The change in the value of *Carry* may produce a difference value of two in the output since the *carry* output has higher weight of the binary bit. Even though this difference is not acceptable, by simplifying the *Cout* and *Sum* signals the problem can be compensated or can be reduced. This can be accomplished by reducing the value of *Sum* to zero as shown in the second half of Table II. Simplifying the value of *sum* to zero reduces the difference between the approximate and exact compressor outputs, also the complexity of the design.

$$Sum' = \overline{Cin}(\overline{X1 \oplus X2} + \overline{X3 \oplus X4}) \quad (5)$$

Table II: Truth Table of the First Approximate 4-2 Compressor

Cin	*X4*	*X3*	*X2*	*X1*	*Cout'*	*Carry'*	*Sum'*	*Difference*
0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	1	0
0	0	0	1	0	0	0	1	0
0	0	0	1	1	0	0	1	-1
0	0	1	0	0	0	0	1	0
0	0	1	0	1	1	0	0	0
0	0	1	1	0	1	0	0	0
0	0	1	1	1	1	0	1	0
0	1	0	0	0	0	0	1	0
0	1	0	0	1	1	0	0	0
0	1	0	1	0	1	0	0	0
0	1	0	1	1	1	0	1	0
0	1	1	0	0	0	0	1	-1
0	1	1	0	1	1	0	1	0
0	1	1	1	0	1	0	1	0
0	1	1	1	1	1	0	1	-1
1	0	0	0	0	0	1	0	1
1	0	0	0	1	0	1	0	0
1	0	0	1	0	0	1	0	0
1	0	0	1	1	0	1	0	-1
1	0	1	0	0	0	1	0	0
1	0	1	0	1	1	1	0	1
1	0	1	1	0	1	1	0	1
1	0	1	1	1	1	1	0	0
1	1	0	0	0	0	1	0	0
1	1	0	0	1	1	1	0	1
1	1	0	1	0	1	1	0	1
1	1	0	1	1	1	1	0	0
1	1	1	0	0	0	1	0	-1
1	1	1	0	1	1	1	0	0
1	1	1	1	0	1	1	0	0
1	1	1	1	1	1	1	0	-1

The change of value of *Cout* in some states may simplify the design by reducing the error rate provided by the approximate *Carry* and *Sum*.

$$Carry' = \overline{(\overline{X1X2} + \overline{X3X4})} \quad (6)$$

Even though the simplifications of carry and sum increase the error rate, complexity of the design and delay is decreased.

Table II provides the truth Table for the first approximate compressor along with the difference value which indicates the variation from the exact output to the compressed outputs. From the Table II, it is clear that the proposed design has 12 incorrect outputs out of 32 outputs.

The logic expressions for the outputs of the proposed approximate designs are given in equations (4)-(6).

4.2. Design 2

To further increase the performance as well as reducing the error rate a second design is proposed. The proposed equations for the approximate carry and Cout in the previous part can be interchanged, since the carry and Cout outputs have the same weight. In this new design, carry uses the result for Cout and Cout is always equal to Cin; since it is obtained as Cin zero in the first stage, in all stages, Cout and Cin will be zero. So, it is acceptable to ignore Cin and Cout in the hardware design. The block diagram of this approximate 4:2 compressor is shown in figure (7) and the desired equations are described below,

$$Sum' = (\overline{X1 \oplus X2} + \overline{X3 \oplus X4}) \quad (7)$$

$$Carry' = (\overline{X1X2} + \overline{X3X4}) \quad (8)$$

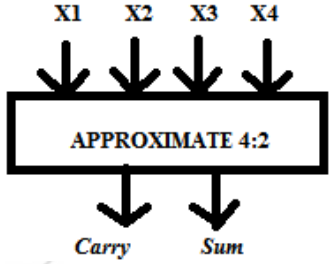

Figure 3: Approximate 4:2 Compressor

It is understood that (8) is same as (6) and (7) is (5) for Cin=0.

Table III: Truth table of second proposed 4:2 compressors

X4	X3	X2	X1	Carry'	Sum'	Difference
0	0	0	0	0	1	1
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	1	-1
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	1	0	0
0	1	1	1	1	1	0
1	0	0	0	0	1	0
1	0	0	1	1	0	0
1	0	1	0	1	0	0
1	0	1	1	1	1	0
1	1	0	0	0	1	-1
1	1	0	1	1	1	0
1	1	1	0	1	1	0
1	1	1	1	1	1	-1

The truth table of the second approximate design for a 4:2 compressor is shown in Table III. This table also shows the difference between the exact decimal value of the addition of the inputs and the decimal value of the outputs produced by the approximate compressors.

When it comes to the second design it is obtained that the delay and number of transistors are considerably reduced.

5. Multiplication

Here, the effect of using proposed compressors in the fast multiplier is investigated. A multiplier usually consists of three parts,

- Partial products generation
- For the addition of only two operands, a carry save adder tree is used to reduce the partial products matrix.
- For the final computation of the binary result, a carry propagation adder is used.

The second module plays an important role in deciding the delay, number of transistors and power consumption. To achieve fast and low-power operation, compressors have been widely used to speed up the CSA tree and decrease its power dissipation. Approximate multiplier is the result of using the approximate compressors in the CSA tree of a multiplier.

Initially, the impact of using the approximate compressors in the restoration module of an 8X8 unsigned Dadda Multiplier is designed and analysed. Then the design of advanced Booth Dadda Multiplier and use of approximate compressors are analysed.

5.1. Dadda Multiplier

The refinement of the parallel multipliers presented by Wallace paved the way for L. Dadda to design a new multiplier. Since it was designed by L. Dadda, it is known as Dadda multiplier. Both the multiplier methods consist of three stages. Using the N^2 AND gates, in the first stage partial product matrix is formed. In the second stage, this partial product matrix is reduced to a height of two by using suitable half adders and full adders. The final stage consist of obtaining the final result by utilizing the carry propagating adder.

As the time consuming module in the multiplier is the partial product reduction stage, it is suitable to use compressors in the generated partial products thus reducing the delay and number of transistors.

Figure (4) shows the utilization of exact 4:2 compressors in the restoration module of the Dadda Multiplier.

In the given figure the reduction circuitry uses full adders, half adders and 4:2 compressors.in the first stage, to reduce the partial product into at most four rows, 2 full adders, 2 half adders and 8 compressors are utilized. In the second or final stage, 1 half-adder, full-adder and 10 compressors are used to compute the two final rows of partial products. Therefore, in total three half-adders, three full-adders and 18 compressors are needed in the reduction circuitry of an 8X8 Dadda Multiplier.

For designing an approximate multiplier, two cases are considered.

- Design 1 is utilized in the first case for all 4:2 compressors
- Design 2 is considered for the 4:2 compressors in the second case. Since the design 2 does not have Cin and Cout, Multiplier 2 uses 6 half adders, 1 full adder and 17 compressors.

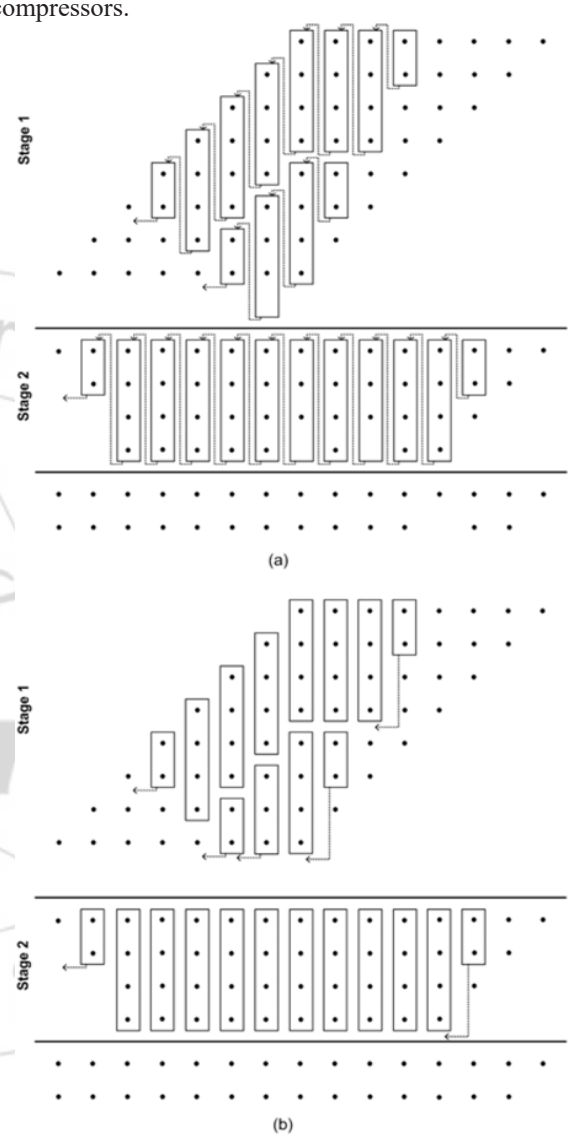

Figure 4: Reduction circuitry of an 8X8 Dadda multiplier, (a) using Design 1 compressors, (b) using Design 2 compressors.

5.2 Booth Multiplier

Using Booth algorithm, multiplication becomes faster as the partial products gets reduced. Modified Booth algorithm does the reduction of partial products by scanning three bits instead of two bits.

By using the technique of radix-4 Booth recoding, it is possible to reduce the number of partial products by half. Here, in the LSB of the multiplier, 0 is appended and then it is grouped in overlapping groups of three. To obtain the final

product, multiplicand is altered according to the operations mentioned in the recoding table (Table IV). By combining the Booth algorithm and Dadda multiplier, also by using the approximate compressors, number of partial products as well as number of adders can be reduced, thereby reducing the delay of operation.

Table IV: Recoding Table of Radix 4 Multiplication

M(i+1)	M(i)	M(i-1)	OPERATION
0	0	0	0
0	0	1	1xMultiplicand
0	1	0	1XMultiplicand
0	1	1	2XMultiplicand
1	0	0	-2XMultiplicand
1	0	1	-1XMultiplicand
1	1	0	-1XMultiplicand
1	1	1	0

6. Proposed Multiplier

In this paper, advanced Booth Dadda multiplier is proposed where radix 4 Booth multiplier is used for partial product formation and Dadda multiplier is used to add those partial products using the approximate compressor designs. Figure (5) shows the block diagram of the proposed system.

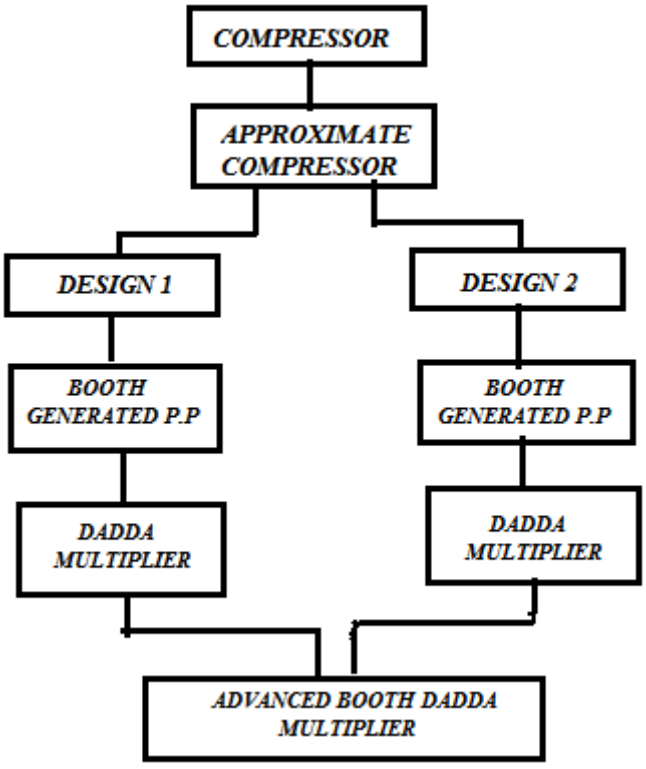

Figure 5: Proposed System

Radix 4 booth multiplier is used for the formation of partial products and Dadda multiplier is used to add those partial products.

7. Simulation Results and Comparison

7.1 Approximate Compressors

Verilog implementation of the Approximate Compressors and its utilization in the Dadda multiplier and advanced Booth Dadda Multiplier is analysed.

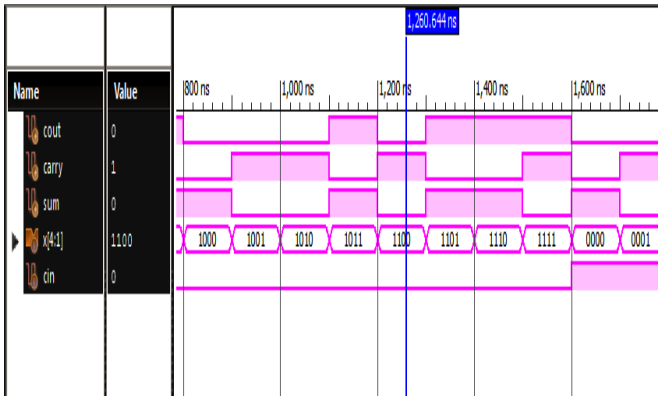

Figure 6: Exact Compressor

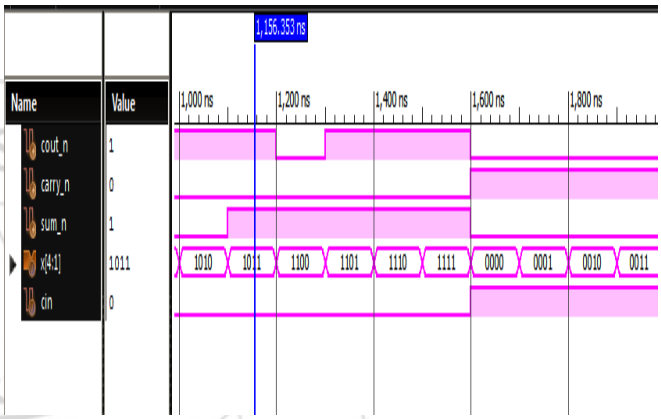

Figure 7: Approximate Compressor- Design1

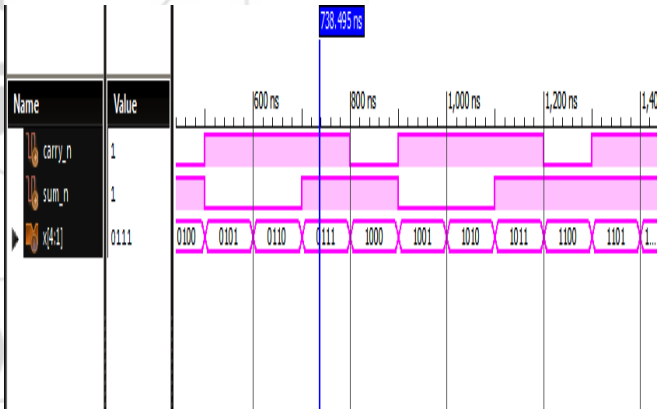

Figure 8: Approximate Compressor- Design2

Table V: Analysis of Compressors

Parameters	Exact Compressors	Design 1	Design 2
No.of LUTs	4 of 4896	3 of 4896	2of 4896
No.of slices	2 of 2448	2 of 2448	1of 2448
Delay	6.920ns	6.125ns	5.776ns

The simulation results for the exact compressor, approximate compressor using design 1, and design 2 is shown in the figure (6), (7) and (8).

Also analysis of the two designs of approximate compressors is shown in table V. No. of LUTs, no. of slices and delay are analysed and it is clear from the table V that the area and delay are considerably reduced when it comes to design 2 compared with the exact compressor.

7.2 Dadda multipliers

Here, the previously designed approximate compressors are used in the Dadda multiplier. Initially, exact compressor is used in the partial product module of the Dadda multiplier. Then its area and delay are analysed. Then it is compared with the multiplier where two designs of approximate compressors are used. Figure (9) shows the result obtained when exact compressor is utilized in the Dadda Multiplier.

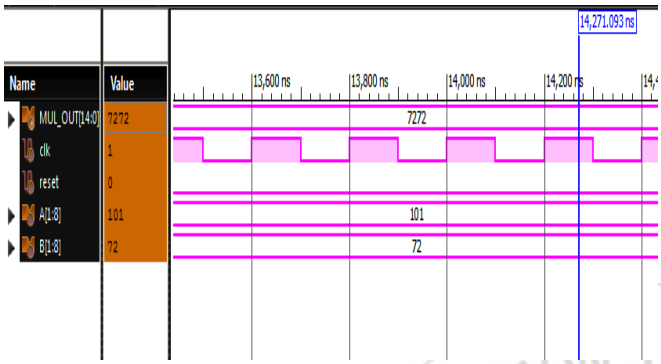

Figure 9: Normal Multiplier

Figure (10) and (11) shows the multipliers using approximate compressors of design 1 and design 2 respectively. Table VI indicates the analysis of these multipliers.

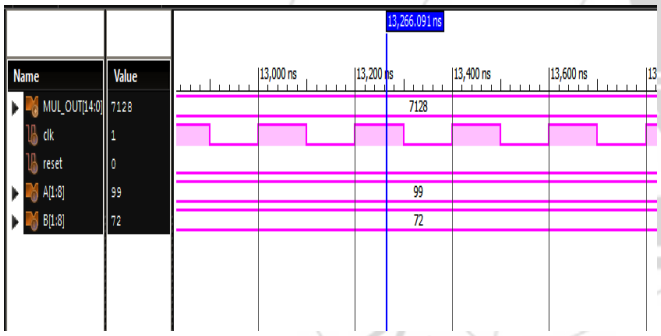

Figure 10: Multiplier using design 1

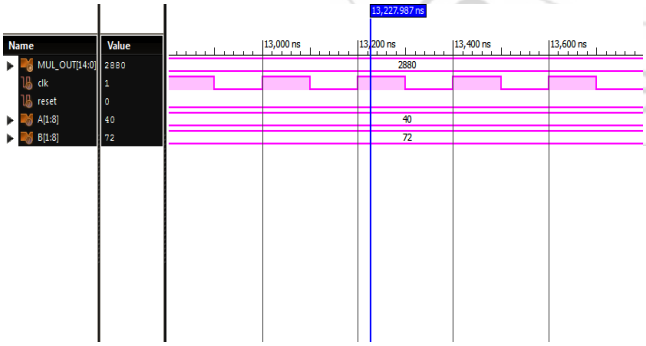

Figure 11: Multiplier using design 2

Table VI: Analysis of Dadda Multiplier Using Approximate Compressors

Parameters	Normal Multiplier	Multiplier using Design 1	Multiplier using Design 2
No.of LUTs	316 of 4896	347 of 4896	284 of 4896
No.of slices	185 of 2448	196 of 2448	179 of 2448
Delay	11.943 ns	13.608 ns	9.638 ns

From the table it is clear that, when compared with the normal multiplier, multiplier using design 2 have got much reduced delay and number of slices.

7.3. Advanced Booth Dadda Multiplier

Instead of Dadda multiplier, here the simulation results and analysis of advanced Booth Dadda multiplier is done. Figure (12), (13) and (14) shows the simulation results of advanced Booth Dadda multiplier using exact compressor, design 1 and design 2 respectively. Table VII shows the analysis.

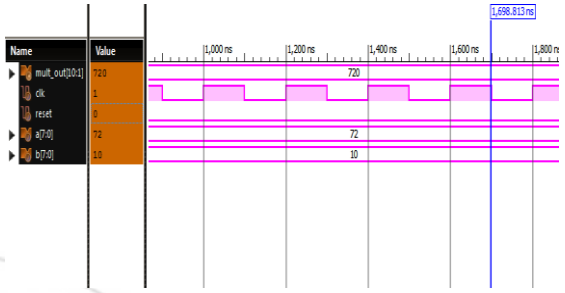

Figure 12: Proposed multiplier using exact compressor

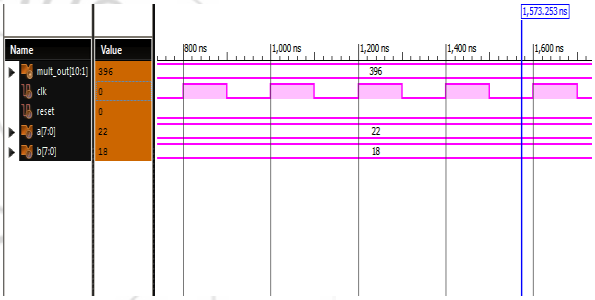

Figure 13: Proposed multiplier using design 1

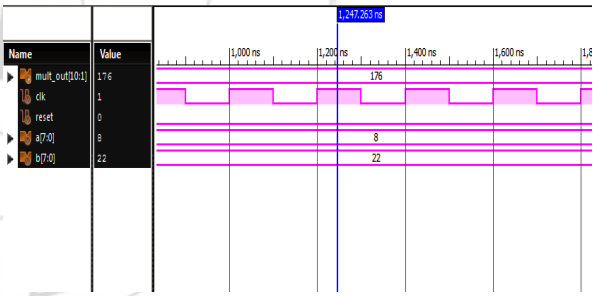

Figure 14: Proposed multiplier using design 2

Table VII: Analysis of Advanced Booth Dadda Multiplier

Parameters	Normal Multiplier	Multiplier using Design 1	Multiplier using Design 2
No.of LUTs	160 of 4896	173 of 4896	155 of 4896
No.of slices	87 of 2448	93 of 2448	84 of 2448
Delay	12.128 ns	13.997 ns	10.718ns

From the table VII, it is clear that when compared with the Dadda multiplier area is much reduced in the Advanced Booth Dadda multiplier.

8. Conclusion

Multiplication is an inevitable operation for the arithmetic designs of processors. This paper has presented the novel designs of the approximate compressors. Two designs proposed here that has better efficiency when compared with

the exact compressors on delay and number of transistors. These compressors are then utilized in the partial products in the Dadda multiplier. Dadda multiplier is used for high speed multiplication. When analysed the Dadda multiplier that used approximate compressors have better delay and number of transistors than those used with exact compressors. As the time consuming module of the multiplier is the operation of partial products, it is necessary to reduce the partial products. Booth Algorithm is used for reducing the partial products generated by half. This Booth generated partial products are the used in Dadda Multiplier along with the approximate compressors designed. This implementation provides better result in reducing the delay and number of transistors.

References

[1] Amir Momeni, Student Member, IEEE, Jie Han, Member, IEEE, Paolo Montuschi, Fellow, IEEE, and Fabrizio Lombardi, Fellow, IEEE, "Design and Analysis of Approximate Compressors for Multiplication" IEEE Transactions On Computers, Vol. 64, No. 4, April 2015

[2] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and Probabilistic Adders," IEEE Trans. Computers, vol. 63, no. 9, pp. 1760–1771, Sep. 2013.

[3] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, "IMPACT: IMPrecise adders for low-power approximate computing," in Proc. Int. Symp. Low Power Electron. Design, Aug. 2011, pp. 409–414.

[4] S. Cheemalavagu, P. Korkmaz, K. V. Palem, B. E. S. Akgul, and L. N. Chakrapani, "A probabilistic CMOS switch and its realization by exploiting noise," presented at the IFIP Int. Conf. Very Large Scale Integ., Perth, Australia, Oct. 2005.

[5] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," IEEE Trans. Circuits Syst. I: Reg. Papers, vol. 57, no. 4, pp. 850–862, Apr. 2010.

[6] M. J. Schulte and E. E. Swartzlander Jr., "Truncated multiplication with correction constant," in Proc. Workshop VLSI Signal Process. VI, 1993, pp. 388–396.

[7] E. J. King and E. E. Swartzlander Jr., "Data dependent truncated scheme for parallel multiplication," in Proc. 31st Asilomar Conf. Signals, Circuits Syst., 1998, pp. 1178–1182.

[8] C. S. Wallace," A Suggestion for a Fast Multiplier" IEEE Trans. on Computers, vol.13, pp.14-17, 1964.

[9] L. Dadda, "Some Schemes for Parallel Multipliers", Alta Frequenza, vol.34, pp.349-356, 1965.

[10] Whitney J. Townsend, Earl E. Swartzlander, Jr., Jacob A Abraham, "A Comparison of Dadda and Wallace Multiplier Delays" Computer Engineering Research Center, the University of Texas at Austin.

[11] C. Chang, J. GU, and M. Zhang, "Ultra low-voltage low- power CMOS 4-2 and 5-2 compressors for fast arithmetic circuits," IEEE Trans. Circuits Syst., vol. 51, no. 10, pp. 1985–1997, Oct. 2004.

[12] D. Radhakrishnan and A. P. Preethy, "Low-Power CMOS pass logic 4-2 compressor for high-speed multiplication," in Proc. IEEE 43rd Midwest Symp. Circuits Syst., 2000, vol. 3, pp. 1296–1298.

[13] Z. Wang, G. A. Jullien, and W. C. Miller, "A new design technique for column compression multipliers," IEEE Trans. Comput., vol. 44, no. 8, pp. 962–970, Aug. 1995.

[14] J. Gu and C. H. Chang, "Ultra low-voltage, low-power 4-2 Compressor for high speed multiplications," in Proc. 36th IEEE Int. Symp. Circuits Syst., Bangkok, Thailand, May 2003, pp. v-321–v-324.

[15] M. Margala and N. G. Durdle, "Low-power low-voltage 4-2 compressors for VLSI Applications," in Proc. IEEE Alessandro Volta Memorial Workshop Low-Power Design, 1999, pp. 84–90.

[16] B. Parhami, Computer Arithmetic; Algorithms and Hardware Designs, 2nd ed. London, U.K.: Oxford Univ. Press, 2010.

[17] K. Prasad and K. K. Parhi, "Low-power 4-2 and 5-2 compressors," in Proc. 35th Asilomar Conf. Signals, Syst. Comput., 2001, vol. 1, pp. 129–133.

[18] J. Ma, K. Man, T. Krilavicius, S. Guan, and T. Jeong, "Implementation of high performance multipliers based on approximate compressor design," presented at the Int. Conf. Electrical and Control Technologies, Kaunas, Lithuania, 2011.