VOL. 20, NO. 3, FEBRUARY 2025 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

DESIGN AND ANALYSIS OF PARTITION TECHNIQUE BASED DADDAMULTIPLIER ARCHITECTURE

D. Kalaiyarasi[1], T. Suguna[1], C. Padma[2], Chekuri Nalini[3], M. Sundar Raj[4] and G. Nagarajan[4]
[1]Department of Electronics and Communication Engineering, Panimalar Engineering College, Chennai, India
[2]Department of Electronics and Communication Engineering, Sri Venkateswara College of Engineering, Tirupati, India
[3]Department of Electronics and Communication Engineering, Mohan BabuUniversity(MBU), Tirupati, India
[4]Department of Mathematics, Panimalar Engineering College, Poonamallee, Chennai, India
E-Mail: kalaiccarthi@gmail.com

ABSTRACT

This paper combines two design strategies to speed up column compression multiplication using the Dadda algorithm: decomposing partial products into two sections so that they can be compressed individually in parallel columns and added more quickly using a ripple carry adder and a Binary to Excess-1 converter. This paper also proposes multiplexer based full adder and a half adder designs to optimize power dissipation and propagation delay. The proposed Dadda multiplier of size 8,16,32 and 64 is designed by employing proposed adder designs and is simulated and synthesized using Altera Quartus II with EP2S15F484C3 device for 90nm technology of supply voltage of 1.2V and observed the performance parameters such as delay, power, Maximum Usable Frequency (MUF), Power Delay Product (PDP) and area respectively. It is observed that the proposed partitioned Dadda multiplier in this work,on an average, was able to minimize ALUTs utilization by 73.16%, delay by 51.57%, power by 47.38%, PDP by 73.89% and MUF in an average is increased by 51.57% when compared to existing works. It is concluded that the suggested multiplier can be employed in applications where power and propagation delay are of major concerns since the proposed multiplier design has substantially optimized for all the performance parameters than the conventional Dadda multiplier.

Keywords: column compression, dadda multiplier, mux based full adder, mux based half adder, binary to excess-1 convertor.

Manuscript Received 8 November 2024; Revised 4 February 2025; Published 21 March 2025

INTRODUCTION

VLSI systems are increasingly being utilized in multi-standard wireless receivers, portable and mobile devices, and bio-medical applications [1, 2]. Digital signal processing is one of the many electronic applications that use multipliers to accomplish algorithms like Finite Impulse Response and Infinite Impulse Response, among others. For high-performance applications like signal, image, video processing, etc. The most fundamental and commonly used mathematical operations are addition and multiplication of two binary values. The performance of these applications relies on the performance of the adder and multiplier circuits, since these circuitslie in the critical path of the system. Hence, for high-performance digital systems, high performance multiplication is an essential need. Column compression multipliers are becoming more and more popular for high-speed calculations due to their higher rates because their total delay is related to the logarithm of the operand word length. [3], [4]. In 1964, Wallace invented the first column compression multiplier [5]. The author lowered N rows partial products by splitting them into three and two row groups, which can be implemented using full adder and half adder circuits respectively. Dadda refined Wallace's approach of multiplier in 1965 with the precise placement of the full and half adder in the critical path of the multiplier [6]. In the 2000s, an in-depth analysis of the Wallace and Dadda multipliers demonstrated that the Wallace multiplier is marginally less effective than the Dadda multiplier and also has higher hardware requirements [7] [8].The mentioned strategies are used in the Dadda multiplier since they are faster and perform better than the conventional Dadda multiplier [9] [10].

Three primary components comprise the multiplier's whole delay: the partial product generator block (PPG), the partial product summation tree section (PPST), and the final adder block. [11]. The PPST and final adder contribute a major part of the multiplier's delay. The PPG has extremely low latency. As a result, shortening the time between the PPST and the multiplier's penultimate adder step could result in a large speed increase. As a result, the Adder is one of the most important components of a multiplier. [15] [16], A Carry Look-Ahead Adder (CLA), a Carry Selects Adder (CSLA), a Ripple Carry Adder (RCA), or any other type of adder can be used in its design. Conversely, using the proper multiplier enhances the overall performance of the system.

A multiplier with a larger latency and area that utilizes a CLA and a CSLA was proposed by V. Vijayalakshmi*et al*. [17]. HasanKrad and AwsYousif Al-Taie [18] presented a multiplier using a Carry Look-Ahead Adder and a Ripple Carry Adder with a longer delay. As a result, to improve the performance of a multiplier [19][20][21][22], an efficient adder is still needed. This work reduces the PPST delay by dividing partial products

into two distinct groups. Furthermore, for the proposed Ripple Carry adder with Binary to Exces-1 convertor, a significant improvement in computational speed for generating final products is observed [25][26][27].

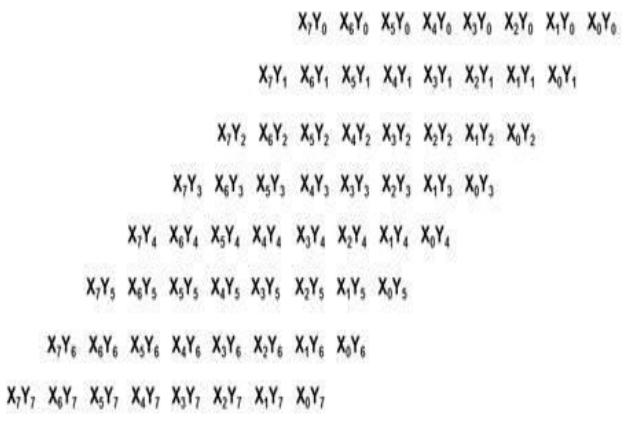
This paper is structured as follows: Sections II and III, respectively, cover the design of the PPST's parallel and final adder structures. The different blocks of Dadda Tree multiplier blocks are explained in Section IV. Section V reports on the design implementation and simulation findings. It is assumed that the multiplier and multiplicand have the same number of bits across the entire document.

DESIGN OF MULTIPLIER PARALLEL STRUCTURES

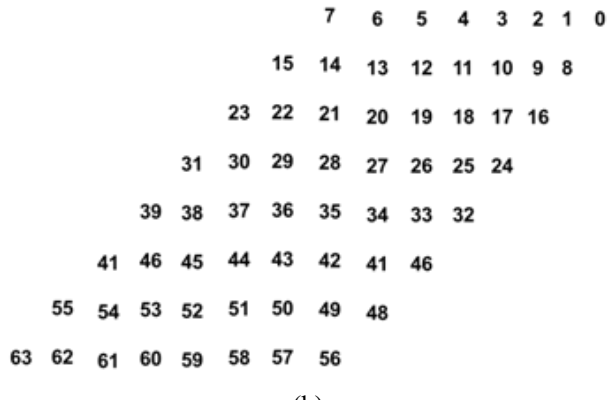
An array of AND gates is used at the beginning of the multiplication operation to generate each partial product simultaneously. The segmentation of partial products as well as their reduction technique are the key significant steps in the design process. Each of these processes is explained in much more detail in the subsections that follow.

Division of Partial Products

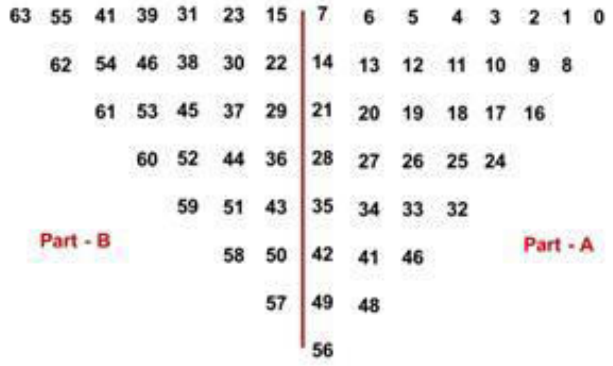
The suggested approach takes into account two M-bit operands for the M-by-M Multiplier: $x_{M-1}x_{M-2}...x_2x_1x_0$ and $y_{M-1}y_{M-2}...y_2y_1y_0$. In the Baugh-Wooley multiplier, the partial products of two N-bit values are x_iy_j, where i,j vary from 0 to M-1. The partial products, as illustrated in Figure 1(a), create a matrix with M rows and 2M-1 columns. As shown in Figure-1(a), assign a number to each partial product. For example, x_0 y_0 is assigned a 0 index, x_1y_0 is assigned $a_1$and so on. Figure-1(b) shows the restructuring of remaining partial products. The highest delay in the PPST is due to a longest column in the centre of the incomplete products. As a result, PPST is divided into two halves in this study, as shown in Figure-1(c), with M columns in each Part-A and Part-B. Then each column is added from the two halves. The summation technique employed in the proposed work is explained in the following section.



(a)



(b)



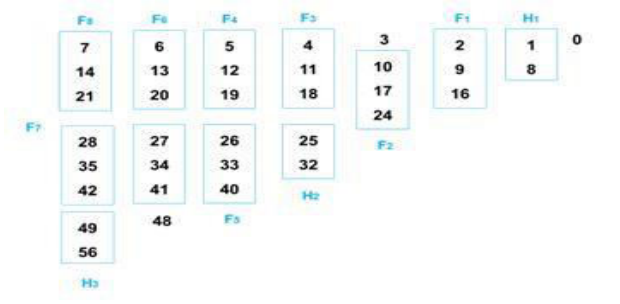
(c)

Figure-1. Partioning of Partial products for 8*8 Multiplier (a) Array diagram for Partial products (b) Alternate Representation (c) Part-A and Part –B Partitioned structure of multiplier.

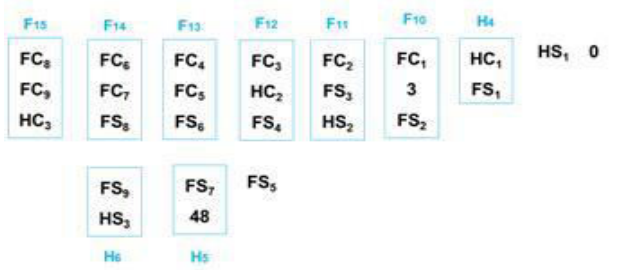
Daddatree Algorithm Based Reduction

Several methods for reducing propagation latency during AND Gate addition of incomplete products have been developed. The Dadda algorithm is widely regarded as one of the most effective multipliers. The height of the tree in Figure-1(a) is eight since the suggested 8*8 multiplier and has a total of 64 partial products. Using the Dadda Algorithm, the tree's height was reduced from eight to two levels. If the Dadda method is not employed, the subsequent stage will use the carry from the previous stage, increasing the propagation latency, therefore it is necessary to wait until the previous stage to be replicated. The ripple carry adder, which uses less power but has a little bit more latency, is used in this proposed work for this straightforward technique. Using the multiplier Dadda approach, the total propagation delay has been minimized. This is the best technique for reducing total multiplier design latency since it does not rely on the preceding step at the start. As a result, the first stage will go ahead regardless of the others. To begin, we made a tree consisting of our partial items.

The partial products of each section are then reduced into two rows using the full adder and half adder based on the traditional Dadda reduction method, as shown in Figures-2 and 3, respectively. Where S and C denote the partial sum and carry, respectively, and 3-bit and 2-bit groupings indicate full and half adders, respectively. Part -A and B partial product reduction, which are shown in Figure - 2 and 3, employs a half adder (HA1) to add bits 1 and 8 to produce sum HS1 and carry HC1. HC1 is transferred to the next column and added to the full adder's (FA1) sum FS1, which adds 2, 9, and 16. The FA1 carries FC1 is added to the column below. As shown in Figure-2 and 3, the last two rows of each segment are merged to form the partial final products, which have a height of one-bit in each column.

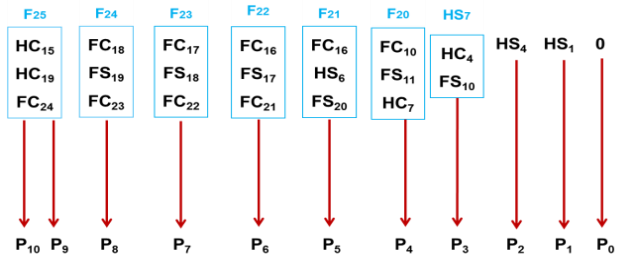


(a)



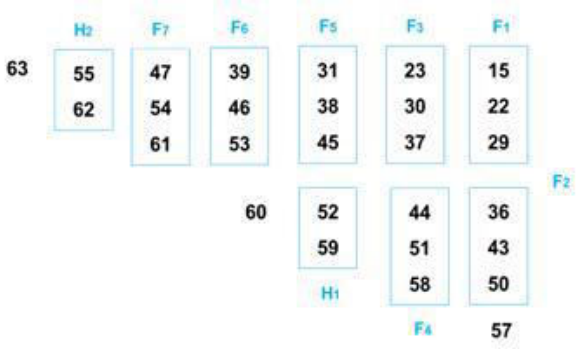
(b)

(c)



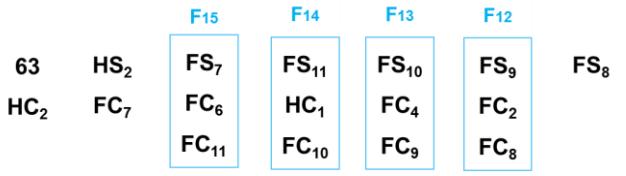
(d)

Figure-2. Part-A Partial Product reduction of Dadda Tree (a) Dadda stage- 1 (b) Dadda Stage -2 (c)Dadda stage-3 (d) Dadda Last stage.

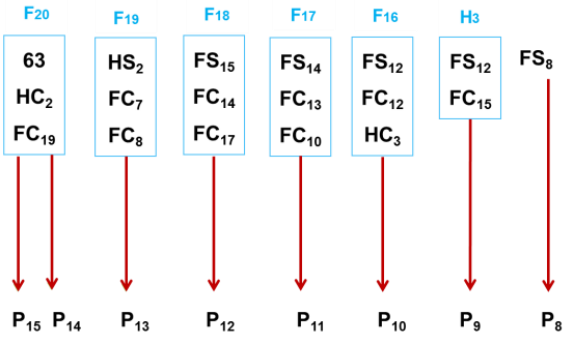


(a)

(b)



(c)



(d)

Figure-3. Part-B Partial Product Dadda Tree Reduction: (a) Dadda stage- 1 (b) Dadda Stage -2 (c) Dadda stag-3 (d) Dadda Last stage.

Figure-4 and 5 employ the Dadda technique to illustrate the two parallel structures presented in Figure 2 and 3 respectively. HA, FA and PA represent half adder, full adder, partial final product of Part-A and Part-B, structures respectively. The ordering of partial products is denoted by the HA and FA numerals. The suggested adder structure, which is covered in the following section, can be used to replace the carry look ahead adder found in typical multipliers to further improve performance.

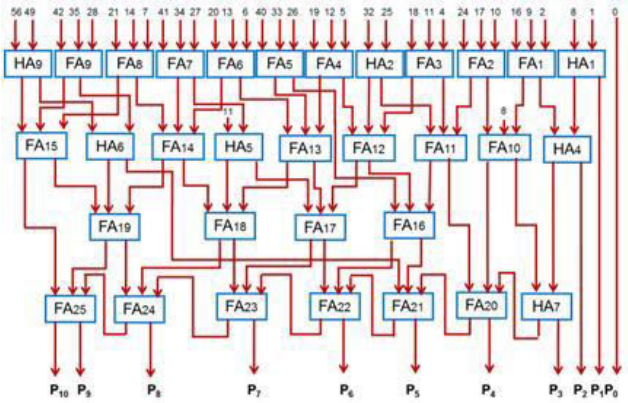


Figure-4. Dadda based implementation of part-a.

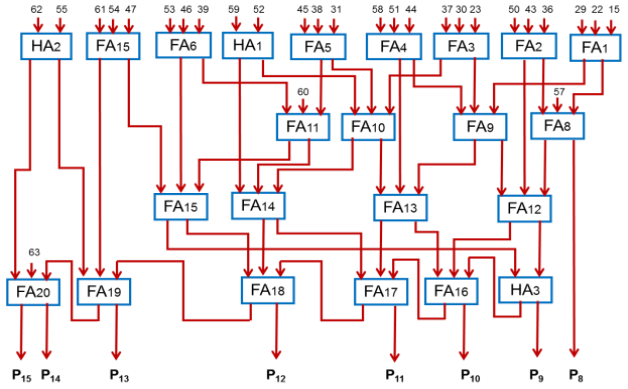


Figure-5. Dadda based implementation of part-b.

Design of Proposed Adder

In previous studies [11-13], the Carry Select Adder (CSLA) and the Adder (CLA) were employed to increase parallel multiplier performance. The CSLA uses more chip area than traditional adders as a result of its design. In the current work, a Ripple Carry Adder and multiplexers-based Binary to Excess-1 Converters (MBEC) are used to enable rapid summing of differential inputs from the PPST [14]. In terms of performance, the MBEC adder outperforms the CSA and CLA adders. It also takes up less area and power when compared with CSLA.

The following are the final partial products: Before Part-A and Part-B outputs are computed in parallel, individual portions of partial products are minimized one bit column height and concatenation is done by using the suggested high speed adder block.

Part-A PartialProducts: P_{10} P_9 P_8 P_7 P_6 P_5 P_4 P_3 P_2 P_1 P_0

Part-B Partial Products:P_{15} P_{14} P_{13} P_{12} P_{11} P_{10} P_9 P_8

Part-A's carry bits are PA[10:8], while Part-B's carry bits are PB[15]. Part-A of PA[7:0] is assigned directly to the end products P[7:0].Thesuggested method for determining the remaining P[15:8] employs Multiplexers with BEC and an RCA as shown in Figure-6.

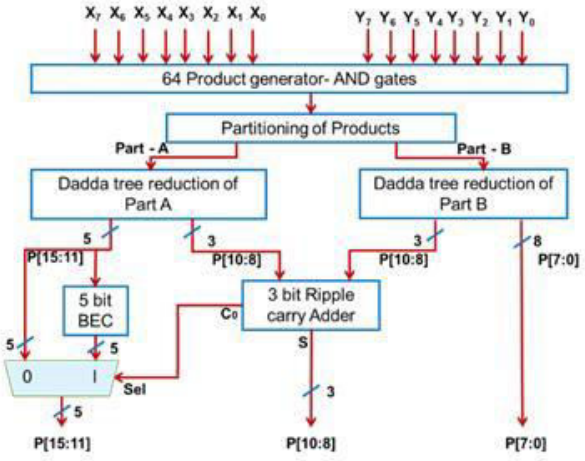


Figure-6. Structure of proposed 8*8 multiplier.

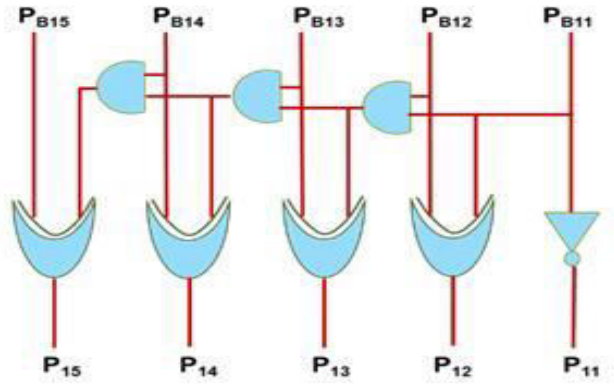


Figure-7. Structure of 5-bit binary to excess-1 converter.

P [10:8] is the result of integrating parts A and B of P [10:8] using 3-bit RCA. Two partial results are generated when PB [15:11] is provided as the input for the 5-bit MBEC, resulting in the remaining P [15:11], where Cin='0' for P [15:11] and Cin='1' for the 5-bit BEC output. Figure-7 shows the precise structure of the 5-bit BEC without carry. The multiplexer generates the final P [15:11] based on the Cout of the RCA, avoiding the propagation of carry along PB.

Building Blocks of Dadda Tree

The proposed block employs a modified and gate, half and full adder with MUX in the multiplier structure, as shown in Figure-8(a), (b), and (c) respectively. This implementation effectively realizes Sum and Carry in the Dadda stage of multiplier design using 2:1 MUX.

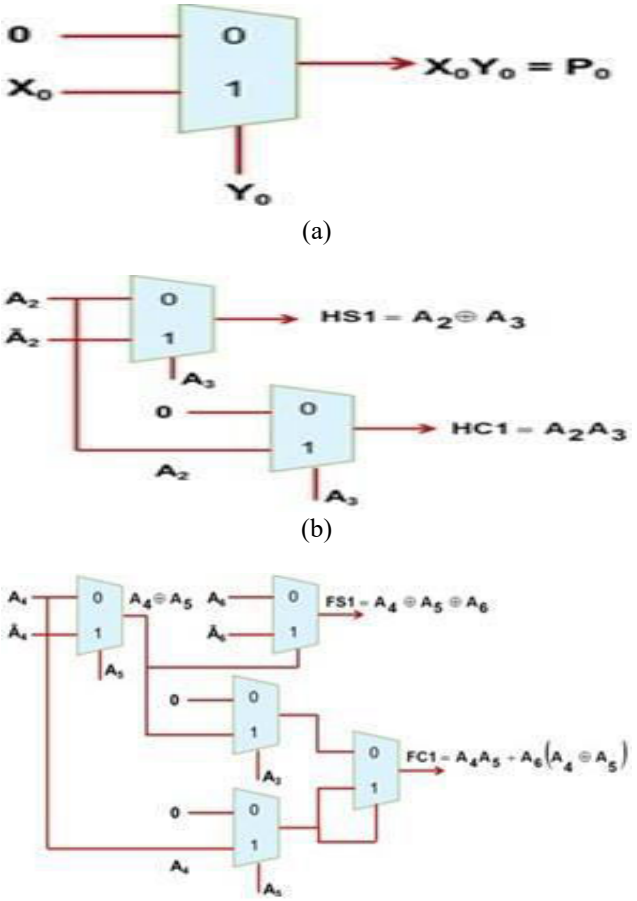


Figure-8. Building Blocks of Dadda Tree (a) AND Gate (b) Half adder (c) Full adder.

RESULTS AND DISCUSSIONS

This section discusses the outcomes of parallel processing to design the proposed 8-bit Partitioned Dadda multiplier as well as higher order multipliers of 16, 32, and 64 bits respectively by employing the Dadda algorithm and a BEC-based Ripple carry adder. The EP2S15F484C3 chip and Altera Quartus II are used to simulate and synthesize the suggested multiplier design for 90nm Technology for a supply voltage of 1.2V.

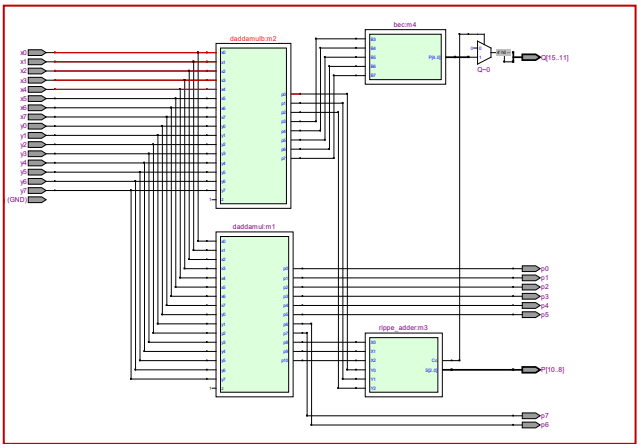


Figure-9. RTL view of proposed 8 * 8 Partitioned Dadda multiplier.

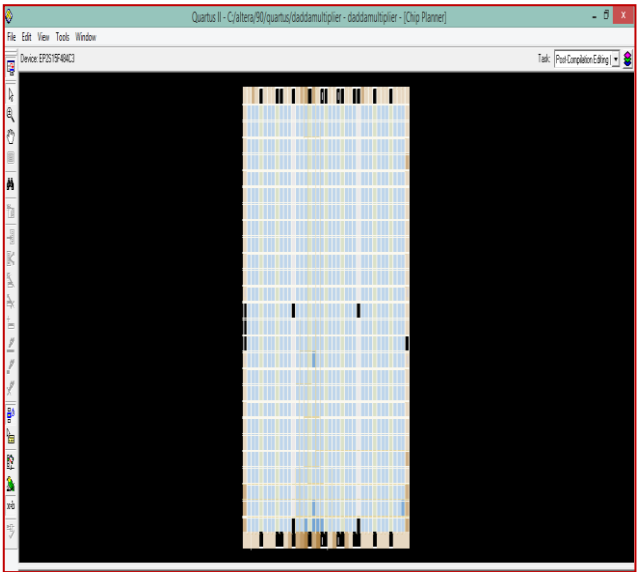


Figure-10. Chip layout of proposed 8 * 8 Partitioned Dadda multiplier.

Figures 9 and 10present the proposed 8 * 8 multiplier's RTL schematic view and chip layout respectively. Performance metrics of proposed multiplier for 8,16,32 and 64 bits are shown in Table-1. Figure-11 gives the graphical comparison of PDP for the proposed multiplier for 8,16,32, and 64 bits respectively.

Table-1. Performance of proposed multiplier.

Multiplier N bit	No. of ALUT's	Delay (ns)	Power (mW)	MUF (MHz)	Power Delay Product(pJ)
8	55	7.77	1.21	128.75	9.4
16	110	8.98	2.37	111.39	21.27
32	220	11.99	4.62	83.41	55.38
64	440	12.52	9.32	79.84	116.72

Table-2. Performance comparison of proposed 32-bit partitioned Dadda multiplier existing works.

32-bit Array Multiplier	No. of ALUT's	Delay (ns)	Power (mW)	MUF (MHz)	Power Delay Product(fJ)
Conventional CSLA (Work 1)[17]	3949	94.147	447.81	10.62	42.159
BEC based CSLA (Work 2)[23]	3783	90.582	377.42	11.03	34.187
Optimized CSLA (Work3)[24]	3680	72.412	271.88	13.81	19.687
Proposed Multiplier (Proposed Work)	1020	40.956	184.4	24.41	7.55

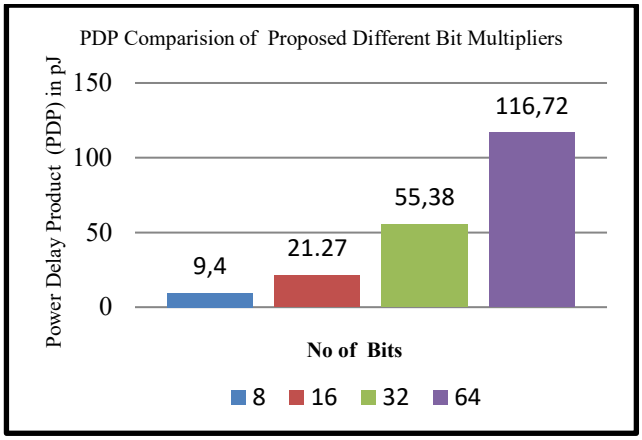


Figure-11. Comparison of power delay product of proposed different bit multipliers.

Table-2 compares the performance of the proposed 32-bit partitioned Dadda multiplier with the works in the literature, for the mentioned performance metrics. Figure-14 shows the pictorial comparison of MUF of the proposed multiplier with existing works respectively. From Figure-12 it is observed that the Maximum Usable Frequency (MUF) is increased by 56.49%, 54.81%, and 43.42% when the proposed multiplier is compared with mentioned works respectively and is represented in Figure-14 with purple line.

From Table-2 it is observed that ALUTs utilization is reduced to 74.17%, 73.04%, and 72.28% when the proposed 32-bit multiplier is compared with work 1, work 2, work 3, respectively, and is represented in Figure-14 with a blue line.

When the proposed 32-bit multiplier is compared with work 1, work 2, work 3 for delay it is minimized by 56.49%, 54.79% and 43.44% respectively and represented with red line in Figure-16. Power consumption is reduced by 58.82%, 51.14% and 32.18% for the proposed multiplier when compared with work 1, work 2 and work 3 respectively and represented in Figure-16 with green line.

Figure-13 shows the Power Delay Product (PDP) comparison of proposed multiplier with different existing works in the literature and it is observed that PDP is reduced by 82.09%, 77.92% and 61.65% when the proposed 32-bit multiplier is compared with work 1, work 2 and work 3 respectively and represented with sky blue line in Figure-14.

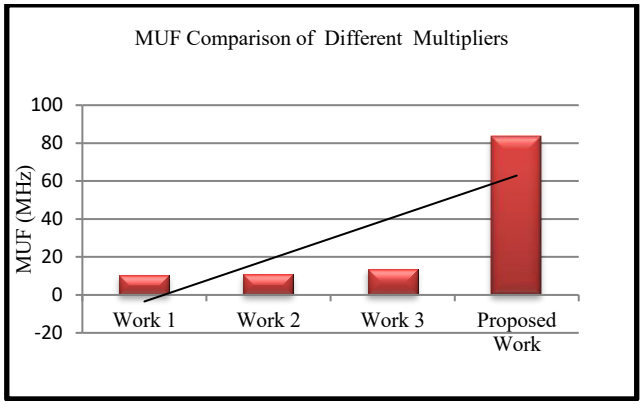


Figure-12. Comparison of maximum usable frequency for different multipliers.

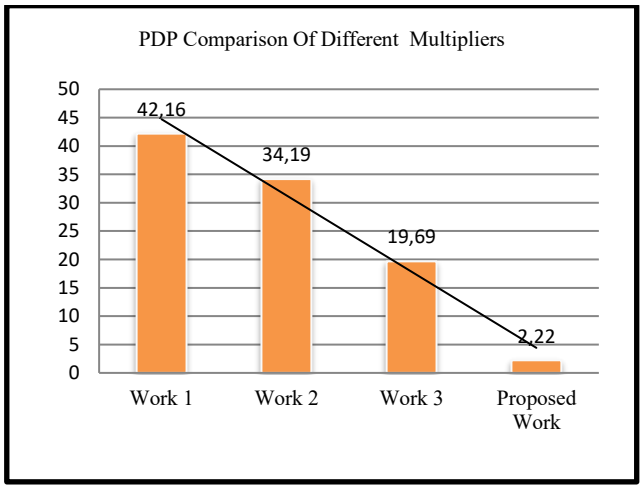


Figure-13. Comparison of Power Delay Product (PDP) of proposed work with existing multipliers.

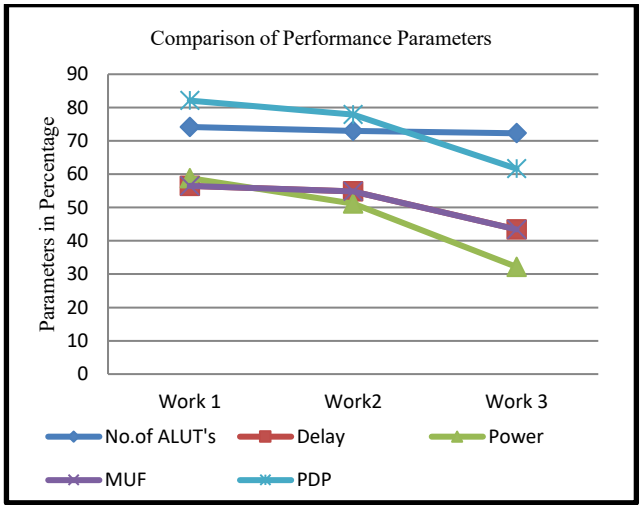


Figure-14. Performance parameters comparison of proposed multiplier with existing multipliers.

CONCLUSIONS

Dadda algorithm is used to reduce the multiplier's area, number of stages, and propagation time, furthermore, by altering the full and half adder designs in each of the Dadda tree algorithm stages. This research paper develops an area-efficient, low-power, high-speed Dadda tree multiplier by combining two design strategies: To perform independent parallel column compression, partial products are first divided into two pieces. Secondly, a BEC-based adder structure is used for the final stage addition. The proposed partitioned Dadda tree technique employs AND gate, half adders, and full adders, which are designed by using 2:1 MUX. Using Altera Quartus II, the suggested multiplier design was simulated and implemented with EP2S15F484C3 device for 90nm Technology for a supply voltage of 1.2V for 8,16,32 and 64 bit respectively. Then the performance parameters such as area in terms of number of combinational ALUTs, delay, power, Maximum Usable Frequency (MUF) and Power Delay Product (PDP) of proposed multiplier were compared with the existing multipliers in the literature. It is observed that the proposed partitioned Dadda multiplier in this work on an average was able to minimize ALUTs utilization by 73.16%, delay by 51.57%, power by 47.38% and PDP by 73.89% when compared to existing works. And also, MUF in an average is increased by 51.57% when compared with existing works. Hence it is concluded that the approach can be employed in any complex circuit appropriate for designing of low-power VLSI and DSP applications.

REFERENCES

[1] Parhi K. K. 1998. VLSI Digital Signal Processing, Wiley: New York. NY.

[2] Chandrakasan P., Verma N., Daly D. C. 2008. Ultralow-power electronics for biomedical applications:Annu. Rev. Biomed. Eng. 10: 247-274.

[3] Parhami B. 2000. Computer Arithmetic: Oxford University Press.

[4] Swartzlander E. E. Jr. and Goto G. 2002. Computer arithmetic. The Computer Engineering Handbook. Oklobdzija,V. G, ed., Boca Raton, FL: CRC Press.

[5] Wallace C. S. 1964. A Suggestion for a Fast Multiplier.IEEE Transactions on Electronic Computers. EC-13: 14-17.

[6] Luigi Dadda. 1965. Some Schemes for Parallel Multipliers.Alta Frequenza.34: 349-356.

[7] Bickerstaff K. C., Swartzlander E. E., Schulte M. J. 2001. Analysis of column compression multipliers. Proceedings of 15th IEEE Symposium on Computer Arithmeitc.

[8] Townsend W. J., Earl E., Swartzlander, Abraham J. A. 2003. A comparison of Daddaand Wallace multiplier Delays. Advanced Signal Processing Algorithms, Architectures and Implementations XIII. Proceedings of the SPIE. 5205: 552-560.

[9] Cappello P. R., Steiglitz K. 1983. A VLSI layout for a pipe-linedDaddamultiplier.ACM Transactions on Computer Systems.pp. 157-174.

[10] Bickerstaff K. C. 2007. Optimization of Column Compression Multipliers. Doctoral Dissertation, Dept. of Electrical and Computer Engineering, University of Texas at Austi., Austin, Texas.

[11] Oklobdzija V. G., Villeger D.1995. Improving Multiplier Design by Using Improved Column Compression Tree and Optimized Final Adder in CMOS Technology. IEEE transactions on Very Large Scale Integration (VLSI) systems.3(2).

[12] Paul F. Stelling.1996. Design strategies for optimal hybrid final adders in parallel multiplier. Journal of VLSI Signal Processing. 14: 321-331.

[13] Sabyasachi Das, Sunil P. Khatri. 2007. Generation of the Optimal Bit-Width Topology of the Fast Hybrid Adderin a Parallel Multiplier.International Conference

on Integrated Circuit Design and Technology (ICICDT).

[14] Ramkumar B., Harish M. Kittur, Mahesh Kannan P. 2010. ASIC Implementation of Modified Faster Carry Save Adder. European Journal of Scientific Research. 42(1).

[15] Wakerly J. F. 2006. Digital Design-Principles and Practices. 4th Edn. Pearson Prentice Hall, USA.ISBN: 0131733494.

[16] William Stallings. 2008. Computer Organization and Architecture Designing for Performance. 7th Edn. Pearson Prentice Hall, USA. ISBN: 0-13-185644-8.

[17] V. Vijayalakshmi V., Seshadri R., Ramakrishanan S. 2013. Design and implementation of 32-bit unsigned Multiplier using CLAA and CSLA. IEEE Proceedings on Circuits, Devices and Systems.

[18] Hasan Krad, Aws Yousif.2008. Performance Analysis of a 32-Bit Multiplier with a Carry-Look-Ahead Adder and a 32-bit Multiplier with a Ripple Adder using VHDL. Journal of Computer Science.4(4): 305-308.

[19] Bharathi S. L., Pritha N. Srividhya G., Padma Priya D. 2018. A Frame of Multiplier using Compactor. International. Journal of Innovative Science and Research Technology. ISSN: 2456-2165,3(3).

[20] Kalaiyarasi D., Saraswathi M. 2018. Design of an Efficient High Speed Radix-4 Booth Multiplier for both Signed and Unsigned Numbers.International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB) 27-28.

[21] Kalaiyarasi D., Pritha N. Srividhya G., Padma Priy A. D., 2021. Low Power, High Speed MUX based Area Efficient Dadda Multiplier. Fourth Internal conference on intelligent Technology(ICONIC).

[22] Srividhya G., Kalaiyarasi D., Pritha N. Bharathi S. L. 2021. Analysis of Vedic, Wallace tree and array multipliers. Fourth Internal conference on intelligent technology(ICONIC).

[23] Nithiyanandham P., Balamurugan V. 2015. Design of modified 32-bit booth multiplier for high-speed digital circuits. ARPN Journal of Engineering and Applied Sciences. 10(10): 4546-4549.

[24] Elakkiya J., Mathan N. 2015. Highly reliable low power MAC unit using Vedic multiplier.ARPN Journal of Engineering and Applied Sciences. 10(10): 4557-4562.

[25] Shoba M., Nakkeeran R. 2016. Energy and area efficient hierarchy multiplier architecture based on Vedic mathematics and GDI logic. Engineering Science and Technology, an International Journal. 20(1): 321-331.

[26] Afreen N., Basha M., Das S. 2017. Design and implementation of area-delay-power efficient CSLA based 32-bit array multiplier. IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT). pp. 1578-1582.

[27] Suguna T., Janaki Rani M. and Anand M. 2020. Efficient Charge Recovery Adiabatic Logic Design and Implementation of a Novel 32-bit Vedic Multiplier for DSP Applications. Journal of Advanced Research in Dynamical and Control Systems. 12(6): 1709-1722.