International Journal for Modern Trends in Science and Technology
Volume 10, Issue 02, pages 362-368.
ISSN: 2455-3778 online
Available online at: http://www.ijmtst.com/vol10issue02.html
DOI: https://doi.org/10.46501/IJMTST1002048

A Novel Approach to reduce Energy/Power in a Parallel Decimal Multiplier

RH Naik, Challa Venkateswarlu, Konidela Siva Nagalakshmi, Chitteti Venkatesh, Dunna Raja Aseessu

Department of Electronics and Communications Engineering, Chalapathi Institute Of Technology,Guntur, Andhra Pradesh, India

To Cite this Article
RH Naik, Challa Venkateswarlu, Konidela Siva Nagalakshmi, Chitteti Venkatesh, Dunna Raja Aseessu, A Novel Approach to reduce Energy/Power in a Parallel Decimal Multiplier, International Journal for Modern Trends in Science and Technology, 2024, 10(02), pages. 362-368.https://doi.org/10.46501/IJMTST1002048

Article Info
Received: 30 January 2024; Accepted: 21 February 2024; Published: 26 February 2024.

ABSTRACT

Despite the fast and effective implementation of binary arithmetic functions, using decimal computation has been revived. This revitalization has been done for three main reasons; (1) the advances in VLSI technology, (2) the appearance of a large amount of decimal data in human-centric applications such as financial, commercial, scientific, and internet-based applications so that the software implementations. Decimal computation is highly demanded in many human-centric applications such as banking, accounting, tax calculation, and currency conversion. Hence the design and implementation of radix-10 arithmetic units attract the attention of many researchers. Among the basic decimal arithmetic operations, multiplication is not only a frequent operation but also has high complexity and considerable power consumption. Therefore, this paper concentrates on this issue and studies a general design methodology that reduces power/energy consumption via localizing switching activity without compromising target performance. This method decomposes a digit multiplier into smaller ones, like Karatsuba's algorithm, while the multiplicand and the multiplier can be partitioned into different sizes. We take advantage of various size partitions in two types of symmetric and asymmetric, which final designs provide specific characteristics. Evaluation results for 16-digit operands show that the proposed architectures have interesting area-delay figures compared to conventional binary multipliers and outperform the figures of previous alternatives for decimal multiplication. It will be developed using Verilog HDL. Xilinx ISE tool is used to perform the Simulation and Synthesis.

Keywords: Decimal computation, digital arithmetic, low power design, parallel multiplier

1. INTRODUCTION

Despite the fast and effective implementation of binary arithmetic functions, using decimal computation has been revived. This revitalization has been done for three main reasons; (1) the advances in VLSI technology, (2) the appearance of a large amount of decimal data in human-centric applications such as financial, commercial, scientific, and internet-based applications so that the software implementations do not satisfy the high-performance requirements [1], and finally (3) the lack of exact binary representation for some decimal fractions (e.g., 0.2). The former made hardware realization of complex functions possible, while the two others pushed the designers to use

hardware-implemented decimal arithmetic units for coping with the complexity of processing a massive amount of data with acceptable precision and time. Due to the importance of decimal arithmetic, the new feature of decimal representation and related operations is added to the latest revision of IEEE 754 standard for Floatingpoint arithmetic [2], [3]. Moreover, concerted activity in both industry and academia is progressing on decimal arithmetic. Several processors have been announced, such as IBM eServer z900 [4], IBM POWER6 [5], and IBM z10 [6], which are equipped with a dedicated decimal arithmetic unit. On the other hand, a considerable number of research papers have been published on decimal arithmetic algorithms and hardware units such as decimal addition [twooperand (e.g., [7]) and multi-operand (e.g., [8])], decimal multiplication [sequential (e.g., [9]) and parallel (e.g., [10])], decimal division [subtractive (e.g., [11]) and multiplicative (e.g., [12])], and other arithmetic functions (e.g., [13]). Among various operations, decimal multiplication is known as one of the most complex operations, which is high frequency, time-consuming, and power-hungry. In addition, it is iteratively used to implement other useful operations such as division, square root, and function evaluation circuits like radix-10 exponentiation and logarithm. Hence, high-speed decimal multiplication takes particular attention, and severalpublications have been published on this topic in less than ten years. The proposed methods of these articles are focused on latency and area as the main design parameters, while the power/energy consumption is neglected. Whereas, the cooling issue in high-performance processing systems and the limited power budget in embedded systems, as well as the effects of consumed power in the efficiency and reliability of digital circuits, make the power/energy consumption the most challenging parameter for nowadays hardware designers [14]. In [15], we provide a comparative study on the leakage and dynamic power consumption of released high-speed decimal multipliers and suggest some guidelines for EDA tools and hardware designers. In this paper, we target to reduce power/energy consumption in high-speed decimal multipliers (i.e., parallel decimal multipliers). To achieve the best result, we use the highest level of design abstraction for applying power reduction techniques, since power optimizations at the higher levels of design abstraction have the maximum influence. Our suggested method is based on partitioning (like Karatsuba's algorithm) and reduces power/energy consumption by localizing switching activity without any negative impact on performance. In our proposed method, we implement a digit multiplier (i.e., IEEE 754-2008 standard) via smaller ones in two different types of symmetric and asymmetric. In symmetric designs, all smaller multiplier units have the same size, while different sizes of multipliers may be used in asymmetric ones.

2. EXISTING MULTIPLICATION APPROACH

The concept of the carry-select adder is to compute alternative results in parallel and subsequently selecting the correct result with single or multiple stages. In carry-selectadders both sum and carry bits are calculated for the two alternatives: carry 0 and 1. Once the carry-in is fired, the correct computation is chosen using multiplexers to produce the desired output. Therefore, instead of waiting for the carry-in to calculate the sum, the sum is correctly output as soon as the carry-in gets there. The time taken to compute the sum is then avoided which results in a good improvement in speed.

Here, four 4X4 Vedic multiplier blocks and three carry select adders of 8 bits each are used. The arrangement of the carry select adders is made in a different way such that it requires less computation time. Some of the carry select adders are given with zero inputs, wherever required. The output of middle multipliers is added using first CSLA. The Output of first CSLA and first Vedic multiplier are added using second CSLA. Carry outputs from first two CSLAs are performed by OR operation and it is given as an input to the third CSLA to generate the final result.

The design of 16×16 block is a similar arrangement of 8×8 blocks in an optimized manner which is shown in Figure 1. The first step in the design of 16×16 block will be grouping the 8 bit (byte) of each 16-bit input.The LSB of two inputs will form vertical and crosswise product terms.Each input byte is handled by a separate 8×8 Vedic multiplier to produce sixteen partial product rows. These partial products rows are added in a 16-bit carry select adder optimally to generate final product bits. The schematic of a 16×16 block is designed by using the 8X8 Vedic multiplier. The partial products represent the Urdhva vertical and cross product terms. Then by using or gate, the final product is obtained.

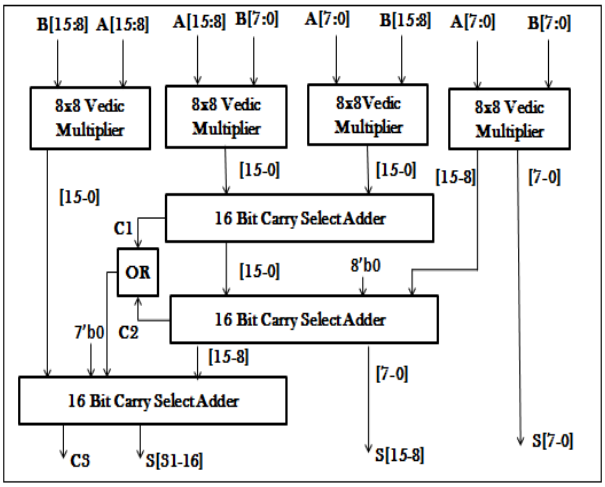


Figure 1: Block Diagram of 16X16 bit Vedic Multiplier

The use of carry select adder increases the hardware thereby more power consumption which is twice that of rippe carry adder.The use of multiplexer increases the chip area.But considerable increase in the speed is achieved. So, optimization of the carry select adder can beuseful inincreasing the speed of the multiplier.

3.PROPOSED MODEL

The Karatsuba algorithm is a well-known technique for reducing the delay of large numbers multiplication which is based on divide and conquer. According to this algorithm, several high-speed multipliers have been proposed. However, this paper focuses on power reduction of decimal multiplication via partitioning, like the Karatsuba technique. It uses smaller multipliers (i.e., multiplier cells) and provides appropriate granularity for localizing switching activity. Nevertheless, some issues like the size of utilized multiplier cells and the basic multiplication algorithm should be studied carefully.

3.1 Decimal Multiplication Via Partitioning

As mentioned above, the Karatsuba algorithm, via divide and conquer, accelerates the multiplication of large numbers. It was originally implemented for binary multiplication. Based on this algorithm in decimal multiplication, for computing $P = X \times Y$ the operands can be recursively divided into two partXH, XL and YH, YL, respectively. In this regard, XH and YH show the most significant parts, while XL and YL demonstrate the least significant portions. Thus, by assuming $k = 2n$ and equal partitioning, the operands can be rewritten in the form of Equation (2).

$$X = 10^n \sum_{i=0}^{n-1} x_{i+n} 10^i + \sum_{i=0}^{n-1} x_i 10^i = X_H 10^n + X_L$$
$$Y = 10^n \sum_{i=0}^{n-1} y_{i+n} 10^i + \sum_{i=0}^{n-1} y_i 10^i = Y_H 10^n + Y_L \quad (2)$$

After partitioning, based on the Karatsuba multiplication algorithm, the P D X x Y product is constructed by Equation (3).

$$P = (X_H 10^n + X_L)(Y_H 10^n + Y_L)$$
$$= 10^{2n} X_H Y_H + 10^n (X_H Y_L + X_L Y_H) + Y_L Y_L \quad (3)$$

Implementation of Equation (3) needs four nX$n$digit decimal multiplications and two $2n$-digit and $3n$-digit additions. This algorithm can be recursively continued till it achieves to 1X1 digit multiplication.

3.2 Symmetric Approach

For symmetric designs, we consider three different partitioning sizes of operands, namely, two, four, and eight. In the first design, multiplier and multiplicand are partitioned into two equal parts, so four 8 X 8 -digit multipliers are used to create a 16 X 16 -digit multiplier according to Equation (3).The output of multiplier cells must be aligned before the final reduction and conversion steps. This alignment is shown in Figure 1, where the eight bits of the least significant part, without any extra computation, form the least significant part of the final result. However, the 8th to 23rd positions need a multi-operand addition. Also, an increment operation is required for the eight bits of the most significant part (increment with carry out of multi-operand addition). It should be mentioned that the algorithm of multiplier cells is essential since it has a significant impact on the detail design and low-level architecture of the main multiplier. Thus, the discussion about the multiplier cells and the related influential characteristics of the algorithm

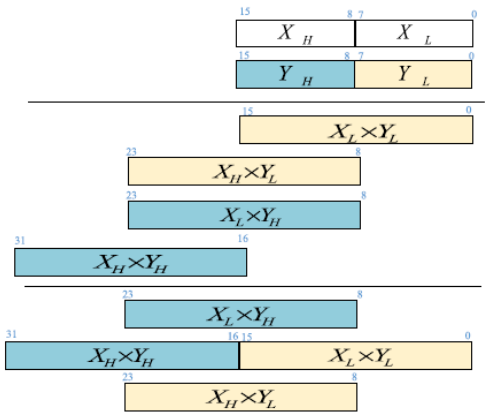


Figure2: The arrangement of the 16 X 16-digit multiplier by four 8 X 8 multipliers (mult.16-8)

The idea of divide and conquer can recursively be used to implement each multiplier cell with smaller ones. For example, a 4*n* X 4*n*-digit multiplier needs four 2*n* X 2*n*-digit multipliers, as mentioned above. By more partitioning, each 2*n* X 2*n*-digit multipliers can be implemented via four *n* X *n*-digit multipliers. According to these explanations, we can implement a 16 X 16-digit multiplier by using sixteen 4 X 4-digit multipliers, as shown in Figure 3.

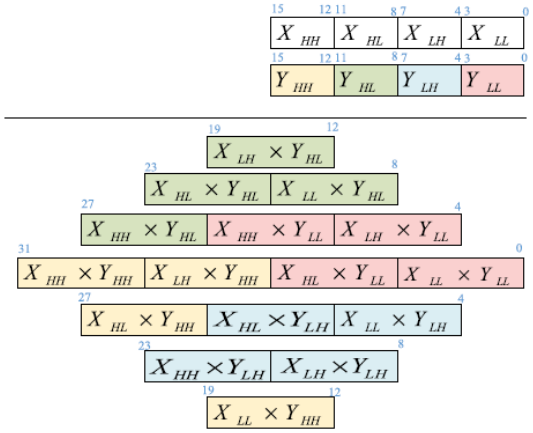


Figure3: The arrangement of the 16 X 16-digit multiplier by sixteen 4 X 4-digit multipliers (mult16-4)

The extreme point of partitioning provides a parallel multiplier which its partial products generate via digit-by digit multipliers. Since all the state-of-the-art multipliers use pre-computed multiplies (based on background information), we define the 2X2-digit multiplier as the smallest multiplier cell. According to this definition, we can partition operands into eight equal parts and construct a 16X16-digit multiplier by using sixty-four 2_2-digit multipliers, as shown in Figure 4.

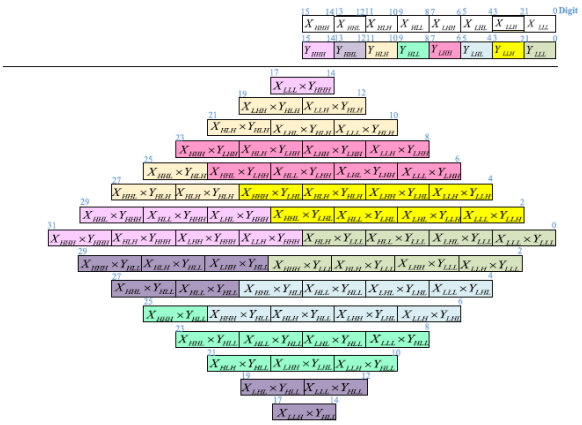


Figure 4: The arrangement of the 16 X 16-digit multiplier by sixty-four 2 _ 2-digit multipliers

By reviewing the aforementioned architectures, two important issues should be considered. The first is the depth of the multi-operand adder, and the second is the output format of multiplier cells. The depth of the multi-operand adder in Fig. 3, 4, and 5 are 3, 7, and 15, respectively. Obviously, using smaller multiplier cells provides more granularity. However, it increases the area consumption. So, a detailed analysis with power dissipation monitoring is required to compromise the size of multiplier cells and area consumption. To address the second issue, based on Fig. 2, the output of multiplier cells is in BCD format, which is provided after a redundant to non-redundant conversion. It causes an unnecessary time-consuming carry propagation that can be omitted since the output of multiplier cells feed to a multi-operand adder. Of course, this adder has to accept redundant inputs. This issue is related to the algorithm of multiplier cells.

3.3 Asymmetric Approach

As mentioned before, different sizes of multiplier cells are possible. Although the bigger multiplier cell has more delay, it may increase the total delay of the main multiplier due to imbalance delay paths. Moreover, an asymmetric method can show considerable superiority for an input pattern with special statistical properties (The experimental results are provided in Section 5). Since possible partitioning for asymmetric design is many, this category's advantage is shown by studying just a straightforward partitioning to illustrate a specific pattern of power consumption in our proposed method.

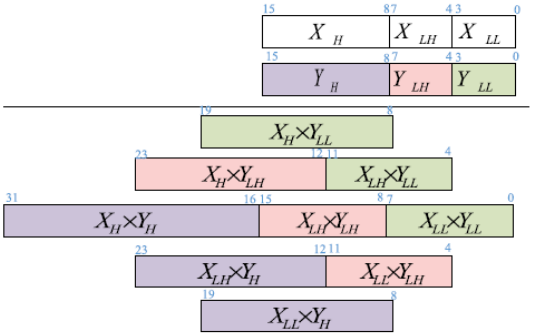


Figure5: The arrangement of the 16 X 16-digit multiplier with asymmetric partitioning (mult16-8-4)

In Figure 5, the arrangement of the 16_16-digit multiplier is shown, which is partitioned asymmetrically. This multiplier is composed of various multiplier cells (i.e., four 4 X 4- digit multipliers, two 4 X 8-digit multipliers, two 8 X 4-digit multipliers, and one 8 X8-digit multiplier).

3.4 Ripple Carry Adder (RCA)

Ripple Carry Adder is a basic adder circuit which contains individual full adder cells and the carry generated upon addition is propagated between the respective adder cells. The computation of result takes place only after the carry from the previous stage is applied as an input to present stage. Due to these propagation delays, the delay is bound to occur which is a major disadvantage. The architecture of the 32-bit RCA is shown in Figure 6.

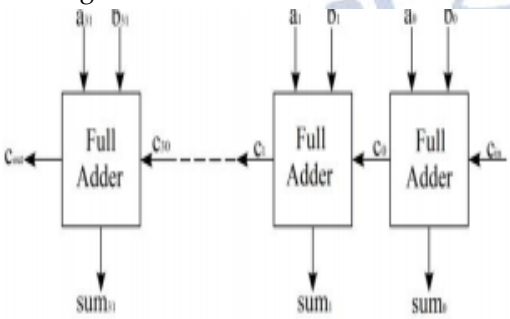
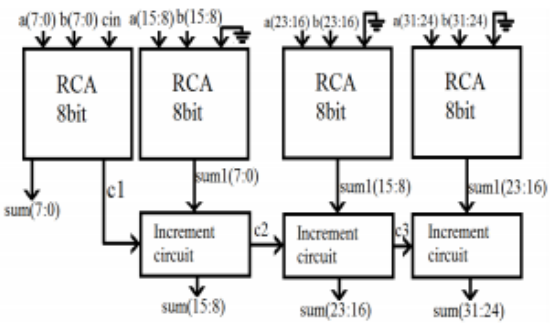


Fig.5 32-bit ripple carry adder

Figure 6: 32-bit RCA

3.5 Compute Add Increment (CAI) adder

In CAI adder, various RCA blocks are used to compute the results. The first RCA block is given carry-in as input along with addends to get carry(c1) and sum. For the rest RCA blocks, the carry-in is given as logic ⊚0⊚ to get temporary sum (sum1) and temporary carry which are given to increment circuit. Increment circuit consists of half adders which add temporary sum and carries to get the actual sum(sum) and carry (cy). The carry-out of increment circuit is obtained by performing OR operation between carry (cy) and carry-out of the previous stage. As the carry-in for the RCA stages is logic ⊚0⊚ and hence the carry propagation delay decreases. The architecture for the CAI 32bit is shown below.



4. RESULTS& DISCUSSION

Simulation results provide a comprehensive understanding of how the designed circuit behaves under different conditions. They are crucial for verifying the functionality, identifying and resolving issues, and ensuring that the circuit meets the desired specifications before physical implementation. Figure 7shows the simulation wave of the proposed multiplier.

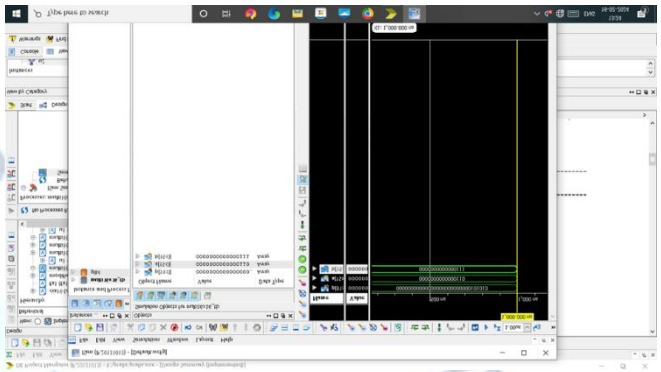


Figure 7: Simulation results of the proposed multiplier

The block diagram offers a high-level representation of the entire system, illustrating the functional blocks and their interconnections. It serves as a visual guide for system architecture, aiding designers in conceptualizing and communicating the design structure and functionality. Figure 8 shows the block diagram of the proposed multiplier.

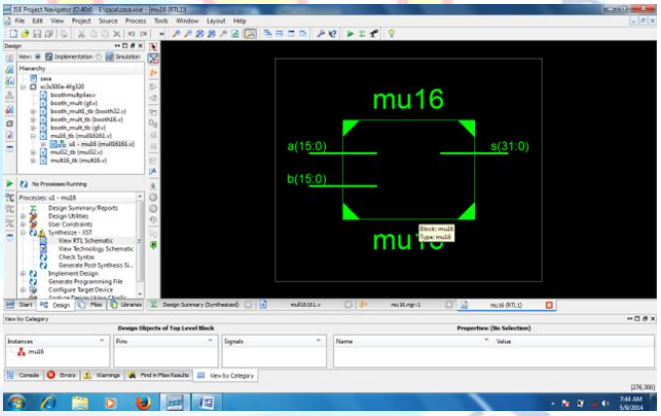


Figure 8: Block diagram of the proposed multiplier

RTL schematics depict the digital logic at a higher abstraction level, showing the flow of data between registers and logic elements. This representation is vital for understanding the data flow within the circuit, facilitating optimization, synthesis, and ensuring proper mapping of the design to hardware. Figure 9 shows the RTL schematic of the proposed multiplier.

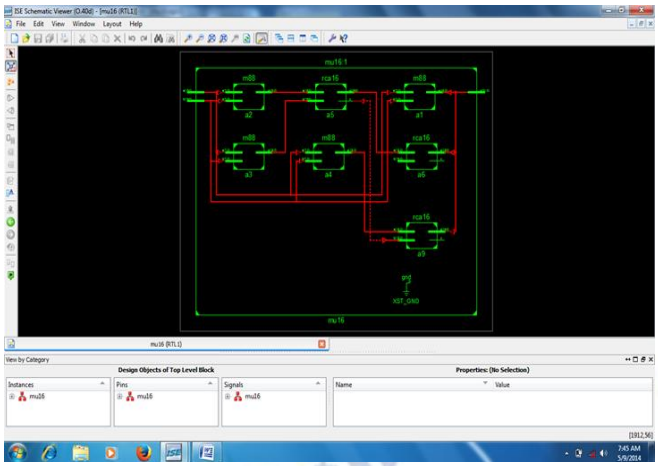


Figure 9: RTL Schematic of the poposed system

It shows the multiplier circuits and adders. Power consumption is a critical consideration in modern VLSI design. Estimating power consumption helps designers optimize the design for power efficiency, which is crucial for battery-operated devices and minimizing environmental impact. Power estimation also guides decisions on cooling mechanisms. Figure 10 presents the power estimation of the proposed system.The power consumed in the circuit is 0.368w.

Delay estimation is essential for ensuring that the designed circuit meets timing requirements. It helps identify and address timing issues such as setup and hold time violations, ensuring that signals propagate through the circuit within the specified time constraints. Figure 11 presents the delay estimation of the proposed system. The delay in the circuit is 42.24 ns.

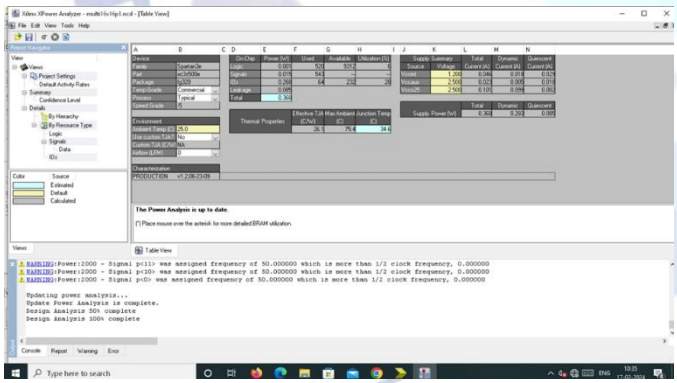


Figure 10: Power estimation of the proposed multiplier

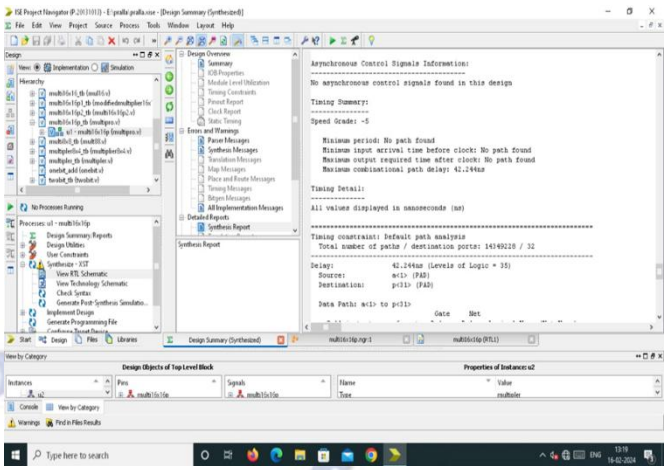


Figure 11:Delay estimation of the proposed Multiplier

Area estimation provides insights into the physical space occupied by the designed circuit on the semiconductor. It is crucial for optimizing the use of resources and determining the overall size of the chip. Efficient area utilization contributes to cost-effectiveness and manufacturability. Figure 12 presents the area estimation of the proposed system. The area report shows that it contains 644 look up tables.

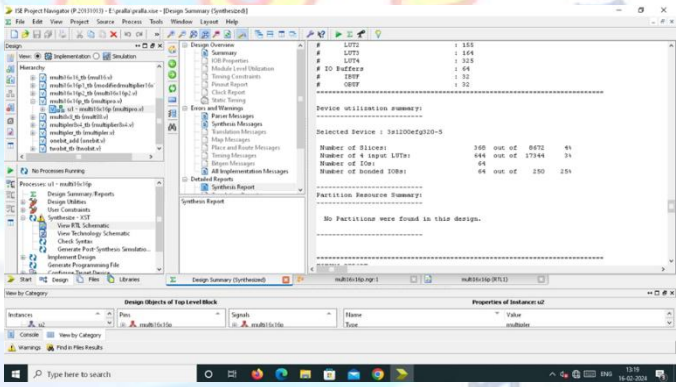


Figure 12: Device utilization summaryof the proposed multiplier

5. CONCLUSIONS

Due to the importance of decimal computation, in this paper, we study a design methodology for reducing the power consumption of decimal multipliers. The suggested method is based on partitioning, which constructs a large multiplier with the smaller ones. The proposed architectures and the original multiplier (which are used as multiplier cells) were implemented via structural Verilog and synthesized with Xilinx ise.

The experimental results demonstrate delay reduction when all the input signals are active.

Conflict of interest statement

Authors declare that they do not have any conflict of interest.

REFERENCES

[1] M. F. Cowlishaw, ''Decimal floating-point: Algorism for computers,'' in Proc. 16th IEEE Symp. Comput. Arithmetic, Jun. 2003, pp. 104–111.

[2] IEEE Standards Committee, Standard 754-2008, 2008, doi: 10.1109/IEEESTD.2008.4610935.

[3] Draft IEEE Standard for Floating-Point Arithmetic, Standard P754/D2.50, Apr. 2019.

[4] F. Y. Busaba, C. A. Krygowski, W. H. Li, E. M. Schwarz, and S. R. Carlough, ''The IBM z900 decimal arithmetic unit,'' in Proc. 25th Asilomar Conf. Signals, Syst. Comput., 2001, pp. 1335–1339.

[5] L. Eisen, J. Ward, H. Tast, and N. Mading, ''IBM POWER6 accelerators: VMX and DFU,'' IBM J. Res. Develop., vol. 51, no. 6, pp. 633–684, 2007.

[6] C. F. Webb, ''IBM Z10: The next-generation mainframe microprocessor,'' IEEE Micro, vol. 28, no. 2, pp. 19–29, Mar. 2008.

[7] A. Vazquez and E. Antelo, ''Conditional speculative decimal addition,'' in Proc. 7th Conf. Real Numbers Comput., Jul. 2006, pp. 47–57.

[8] R. D. Kenney and M. J. Schulte, ''High-speed multi operand decimal adders,'' IEEE Trans. Comput., vol. 54, no. 8, pp. 953–963, 2005.

[9] M. A. Erle and M. J. Schulte, ''Decimal multiplication via carry-save addition,'' in Proc. 14th IEEE Int. Conf. Appl.-Specific Syst., Archit., Processors, 2003, pp. 348–358.

[10] S. Gorgin and G. Jaberipur, ''Sign-magnitude encoding for efficient VLSI realization of decimal multiplication,'' IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 25, no. 1, pp. 75–86, Jan. 2017, doi: 10.1109/TVLSI.2016.2579667.

[11] A. Kaivani, A. Hosseiny, and G. Jaberipur, ''Improving the speed of decimal division,'' IET Comput. Digit. Techn., vol. 5, no. 5, pp. 393–404, 2011.

[12] A. Hosseiny and G. Jaberipur, ''Decimal Goldschmidt: A hardware algorithm for radix-10 division,'' Comput. Electr. Eng., vol. 53, pp. 40–55, Jul. 2016.

[13] V. Vázquez, J. Villalba-Moreno, E. Antelo, and E. L. Zapata, ''Redundant floating-point decimal CORDIC algorithm,'' IEEE Trans. Comput., vol. 61, no. 11, pp. 1551–1562, Nov. 2012.

[14] J. Rabaey, Low Power Design Essentials, 2009th ed. Springer, 2009.

[15] A. Malekpour, A. Ejlali, and S. Gorgin, ''A comparative study of energy/power consumption in parallel decimal multipliers,'' Microelectron. J., vol. 45, no. 6, pp. 775–780, Jun. 2014

[16] Ravikiran, D. N., & Dethe, C. G. (2018). Improvements in Routing Algorithms to Enhance Lifetime of Wireless Sensor Networks. International Journal of Computer Networks & Communications (IJCNC), 10(2), 23-32.

[17] Ravikiran, D. N., & Dethe, C. G. Fuzzy Rule Selection using LEACH Algorithm to Enhance Life Time in Wireless Sensor Networks. Advances in Wireless and Mobile Communications. ISSN, 0973-6972.

[18] Rajesh, G., Thommandru, R., & Subhani, S. M. DESIGN AND IMPLEMENTATION OF 16-BIT HIGH SPEED CARRY SELECT PARALLEL PREFIX ADDER.

[19] Polanki, K., Purimetla, N. R., Roja, D., Thommandru, R., & Javvadi, S. Predictions of Tesla Stock Price based on Machine Learning Model.

[20] Thommandru, R. A PROSPECTIVE FORECAST OF BRAIN STROKE USING MACHINE LEARNING TECHNIQUES.

[21] Rajesh, G., Raja, A., & Thommandru, R. OPTIMIZATION OF MINIATURIZED MICROSTRIP PATCH ANTENNAS WITH GA.

[22] Vellela, S. S., & Balamanigandan, R. (2022, December). Design of Hybrid Authentication Protocol for High Secure Applications in Cloud Environments. In 2022 International Conference on Automation, Computing and Renewable Systems (ICACRS) (pp. 408-414). IEEE.

[23] Vellela, S. S., & Balamanigandan, R. (2024). Optimized clustering routing framework to maintain the optimal energy status in the wsn mobile cloud environment. Multimedia Tools and Applications, 83(3), 7919-7938.

[24] Praveen, S. P., Sarala, P., Kumar, T. K. M., Manuri, S. G., Srinivas, V. S., & Swapna, D. (2022, November). An Adaptive Load Balancing Technique for Multi SDN Controllers. In 2022 International Conference on Augmented Intelligence and Sustainable Systems (ICAISS) (pp. 1403-1409). IEEE.

[25] Priya, S. S., Vellela, S. S., Reddy, V., Javvadi, S., Sk, K. B., & Roja, D. (2023, June). Design And Implementation of An Integrated IOT Blockchain Framework for Drone Communication. In 2023 3rd International Conference on Intelligent Technologies (CONIT) (pp. 1-5). IEEE.

[26] Vellela, S. S., & Balamanigandan, R. An intelligent sleep-awake energy management system for wireless sensor network. Peer-to-Peer Netw. Appl.(2023).

[27] Addepalli, T., Babu, K. J., Beno, A., Potti, B. M. K., Sundari, D. T., & Devana, V. K. R. (2022). Characteristic mode analysis of two port semi-circular arc-shaped multiple-input-multiple-output antenna with high isolation for 5G sub-6 GHz and wireless local area network applications. International Journal of Communication Systems, 35(14), e5257.

[28] Srija, V., & Krishna, P. B. M. (2015). Implementation of agricultural automation system using web & gsm technologies. International Journal of Research in Engineering and Technology, 04 (09), 385-389.

[29] Potti, D. B., MV, D. S., & Kodati, D. S. P. (2015). Hybrid genetic optimization to mitigate starvation in wireless mesh networks. Hybrid Genetic Optimization to Mitigate Starvation in Wireless Mesh Networks, Indian Journal of Science and Technology, 8(23).

[30] Potti, B., Subramanyam, M. V., & Prasad, K. S. (2013). A packet priority approach to mitigate starvation in wireless mesh network with multimedia traffic. International Journal of Computer Applications, 62(14).

[31] Potti, B., Subramanyam, M. V., & Satya Prasad, K. (2016). Adopting Multi-radio Channel Approach in TCP Congestion Control Mechanisms to Mitigate Starvation in Wireless Mesh Networks. In Information Science and Applications (ICISA) 2016 (pp. 85-95). Springer Singapore.