International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-12 Issue-4, March 2023

Delay-Optimistic Multiplier Design using Parallel Prefix Adder with Compressors

J. V. R Ravindra, Chava Chaitanya, Kasam Pranya, Vaddem Sahiti

Abstract: ***This article provides an illustration of the design process for 5-2 and 7-2 compressors operating at extremely high speeds. When compared to the prior designs, the new approach significantly reduced the gate-level delay while maintaining an appropriate overall transistor and gate count. With the help of 7:2 and 5:2 compressor infusion, when compared to earlier designs, the gate-level latency has been significantly decreased. At the same time, the overall transistor and gate counts have remained within acceptable bounds. The technique was initially developed for the 5-2 compressor and subsequently expanded for the 7-2 design, which exhibits enhanced speed performance in these architectures. To increase performance in terms of latency, we can replace the ripple-carry adder used in the last addition with a parallel-prefix adder. Additionally, careful design considerations were made to keep other factors, such as power consumption and activity levels, within reasonable bounds. The best-reported circuits have also undergone redesigns, and the parasitic components of those circuits have been eliminated using the same method to produce a fair comparison. Using a common 16 × 16-bit multiplier, the performance of the built compressor blocks has also been assessed.***

Keywords: 5-2 and 7-2 Compressors, Ripple Carry Adder, Parallel Prefix Adder, Parallel Multiplier, Increased Performance and Speed.

I. INTRODUCTION

Approximate computing, which relaxes the requirement for precise computation to make considerable performance advantages in terms of power, speed, and size, is an emerging paradigm in digital design. Embedded and mobile systems are increasingly relying on this tactic due to their stringent energy and speed constraints. For many applications that can tolerate errors, approximation computing is helpful. Examples include multimedia processing, data mining, and machine learning. For microprocessors, digital signal processors, and embedded systems, multipliers are essential subsystems.

Manuscript received on 23 February 2023 | Revised Manuscript received on 27 February 2023 | Manuscript Accepted on 15 March 2023 | Manuscript published on 30 March 2023.

*Correspondence Authors

Dr. J.V.R. Ravindra, Principal and Professor, Vardhaman College of Engineering, Hyderabad (Telangana), India. Orcid ID: https://orcid.org/0000-0001-7636-1791

Chava Chaitanya*, Department of Electronics and Communication Engineering, Vardhaman College of Engineering, Hyderabad (Telangana), India. E-mail: ch.chaitanya194@gmail.com, Orcid ID: https://orcid.org/0009-0006-8937-8537

Kasam Pranya, Department of Electronics and Communication Engineering, Vardhaman College of Engineering, Hyderabad (Telangana), India. Orcid ID: https://orcid.org/0009-0000-4399-0903

Vaddem Sahiti, Department of Electronics and Communication Engineering, Vardhaman College of Engineering, Hyderabad (Telangana), India. Orcid ID: https://orcid.org/0009-0003-3275-6435

They are used for a wide range of applications, from filtering to convolutional neural networks. Unfortunately, multipliers are among the most power-hungry digital blocks due to their intricate logic architecture. The creation of approximation multipliers has garnered considerable interest in recent years. A multiplier is composed of three essential building blocks: carry-propagate addition, partial product generation, and partial product reduction[1]. Any of these blocks may introduce approximations. Sections of the partial products are not generated, for example, in the partial product truncation technique, and the truncation error is minimised by using the appropriate correction functions.

For variation tolerance and error correction, free computing requires guard bands and redundancies at various levels of the design hierarchy, which adds significant energy overhead. A prospective alternative, known as approximation computing, has garnered considerable support in both academia and industry due to the aforementioned problems. Approximation computing relaxes the numerical equivalence between the design and implementation of error-tolerant systems, offering significant energy and delay savings by introducing "acceptable defects" into the calculation process. However, it is not always necessary to use accurate or high-precision computation. Instead, a few minor mistakes can cancel each other out or have a negligible impact on the computation's results. As a result, approximation computing (AC) has emerged as a novel method for designing structures that are both energy-efficient and that may increase performance while sacrificing some accuracy. An emerging trend in digital design is approximate computing, which trades off the need for perfect calculation for increased speed and efficiency. This study proposes new approximate compressors with 8-bit and 16-bit multiplier designs to evaluate the performance of the proposed compressors.

The design of high-performance, delay-optimistic circuits and systems is now being rethought using the conceptual model of approximate computing [2][3]. Understanding a design or approximation strategy for a specific application is now crucial for the various approximate arithmetic circuits that have been developed to enhance performance and energy efficiency while sacrificing the least amount of precision. The purpose of this paper is to offer a complete examination and comparative evaluation of newly constructed approximation arithmetic circuits under diverse design constraints. As a result, the fundamental issue in developing a binary adder is the carry sequence, which creates the carry bit. The previously described 32-bit basic carry adders, on the other hand, have considerable delay values in higher-order bits because each level of the adder must wait for the previous carry result.

PPA is a well-suited adder in the current technical environment for high-speed addition operations with lower latency in Digital systems due to the aforementioned difficulty with 32-bit basic existing carry adders. The PPA is a typical design that delivers a good mix of speed, power, and available space. Low-order PPAs were developed before the development of 8-bit and 16-bit processors.

II. EXISTING METHOD

Figure 1 demonstrates the 5-2 compressor architecture's seven inputs and four outputs. We're working with the inputs I1, I2, I3, I4, and I5. The values for the secondary inputs, Cin1 and Cin2, are given by the compressor next to it, which has a lower binary bit order significance. Each of the seven inputs is given equal weight. The Carry, Cout1, and Cout2 outputs are one binary bit higher in order than the other bits, but the Sum output has the same weight as the inputs. Cout1 and Cout2 outputs are sent to the more crucial neighbouring compressor block.

$$I1 + I2 + I3 + I4 + I5 + Cin1 + Cin2 = Sum + 2 \times (Carry + Cout1 + Cout2) \quad \ldots\ldots(1)$$

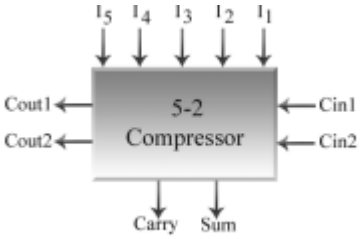

Figure 1: General Architecture of 5:2 Compressor

This structure's critical path is composed of several cascaded sets of six logic gates. Consequently, six XOR logic gates provide an optimistic estimate of the gate-level latency. This is the key factor impeding the arrangement's efficacy. Suppose the impact of the carry-rippling issue between horizontally connected compressor blocks is disregarded. In that case, the gate-level delay on the redesigned structure of Fig. 2 is decreased to four XOR logic gates.

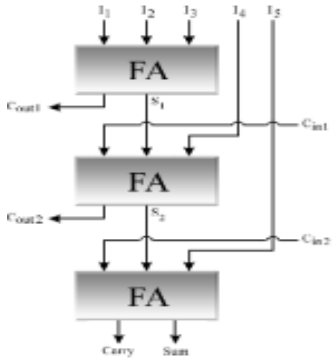

Figure 2: Conventional Implementation of 5:2 compressor

The typical 5-2 compressor is changed to build the new circuit. Based on the logic state of Cin1 and Cin2, there will be three separate tables. To allow all conceivable I1, I2, and I3 input combinational states, the hatching areas divide each table into four sections. To boost speed performance in a simplified design, Cout1 and Cout2 should be independent of Cin1 and Cin2. Because Cin1 and Cin2 are formed from close compressor cells, it also decreases carry rippling during Cout1 and Cout2 generation. As a result of this, we now have

$$Cout1 = I1 \cdot I2 + I1 \cdot I3 + I2 \cdot I3 \quad \ldots\ldots(2)$$

Also, Cout2 can be obtained using the following equation:

If I4=I5:

$$Cout2 = I4 \text{ or } I5 \quad \ldots\ldots(3)$$

Else:

$$Cout2 = I1 \oplus I2 \oplus I3 \quad \ldots\ldots(4)$$

To account for all conceivable combinations of Cin1 and Cin2, three distinct states must be taken into account for output creation. As shown in Fig. 3, if Cin1 = Cin2 = 1, the Carry output is always equal to the logic "1" value. When Cin1 = Cin2 = 0, the Carry output is always equal to logic 0. But if Cin1 and Cin2 are not equal.

Three phases make up the multiplication procedure in a parallel multiplier. Initial partial products (PPs) are produced. Once there are just two rows left, these products are added, and the procedure is repeated until they are added at the end. To effectively sum the products in the second stage and following the generation of PPs, a partial product reduction tree (PPRT) is typically used. The cornerstone for a broad variety of architectural styles is the full adder (FA), which is the main component in many multiplication combinations.

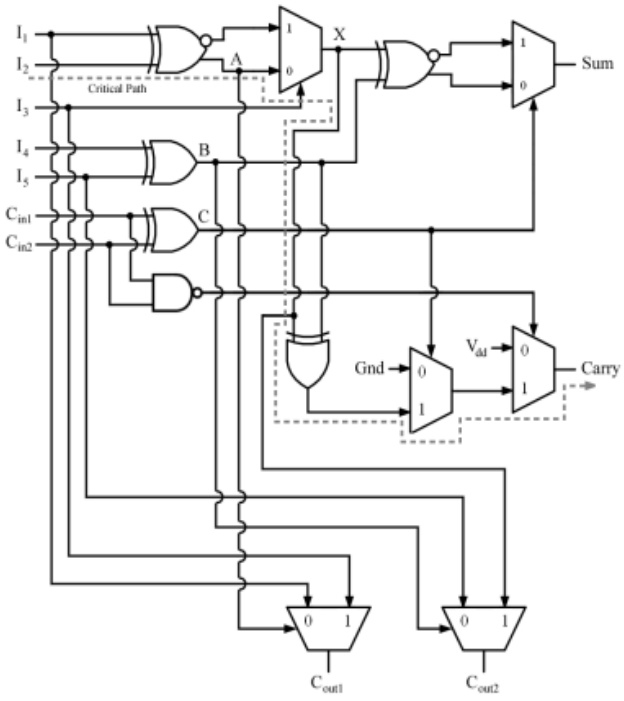

Fig. 3: 5-2 Compressor

Fig. 3 illustrates the newly suggested circuit for the 5-2 compressor. As the carry rippling concludes in the second compressor block, two neighbouring 5-2 compressor cells must be coupled together for a better simulation of the delay in various pathways from inputs to outputs. The section on simulation discusses this subject. In light of this reality, as shown in Fig. 3, the development of Carry output will be the essential path.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

A. 7-2 Compressor:

For 7-2 compressor circuits, design optimization was less critical. Furthermore, the carry-rippling problem persists for at least three periods. To overcome the ripple issue, the essential path contains approximately six cascaded XOR gates due to a negligent design. The 7-2 compressor's new architecture has been proposed utilising the same method as the 5-2 compressor. Carry rippling between adjacent blocks has been decreased by one level, but speed performance has improved over previous designs[13]. Figure 4 depicts the developed approach, which reduces the time between inputs and outputs to no more than four XOR logic gates. To compute the critical route delay, it is evident that the critical path corresponds to the path that starts at I3 and terminates at I4. As a result of creative design and a one-stage decrease in the carry rippling problem[4], speed will be significantly increased.

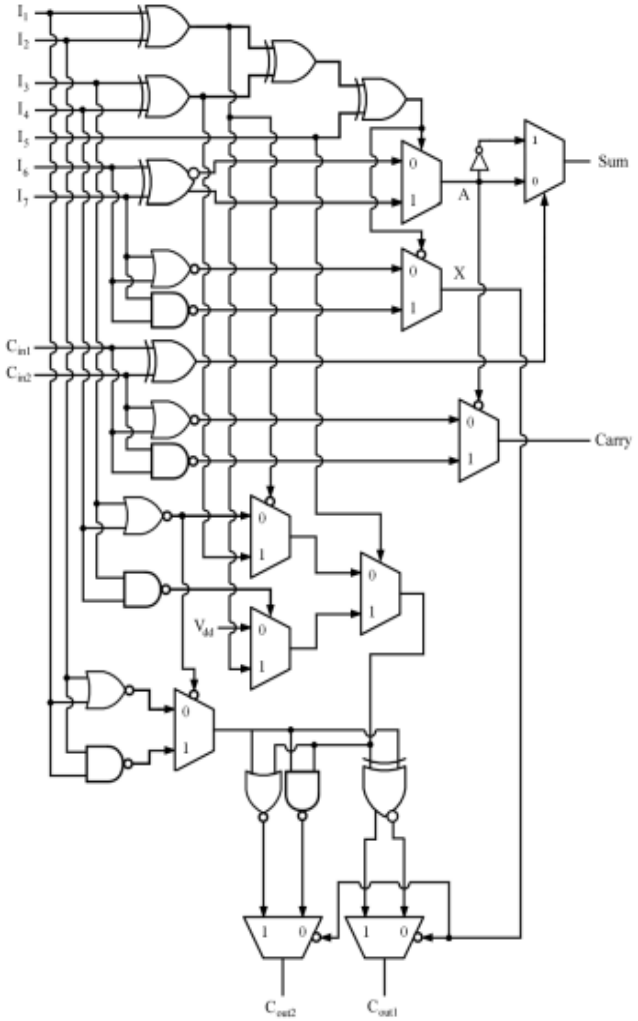

Fig. 4: Proposed 7-2 compressor.

To illustrate the value of the recommended compressor blocks by giving an example of a common application platform, a 16 x 16-bit multiplier architecture has been created [19]. This decision was made to enable the proper use of compressor blocks 5-2 and 7-2. The PPs are built utilising AND gates in their simplest form. The PPs will then be divided into two rows using a two-step procedure. To achieve the result, a 32-bit ripple-carry adder (RCA) was utilised. The PPRT of the multiplier was initially built using the blocks we came up with[5], and it was subsequently modified using previously published works for the 5-2 and 7-2 structures. This allowed for a fair comparative study.

The multiplier is implemented by using the above-designed 5-2 & 7-2 compressors. The architecture of the multiplier is shown below:

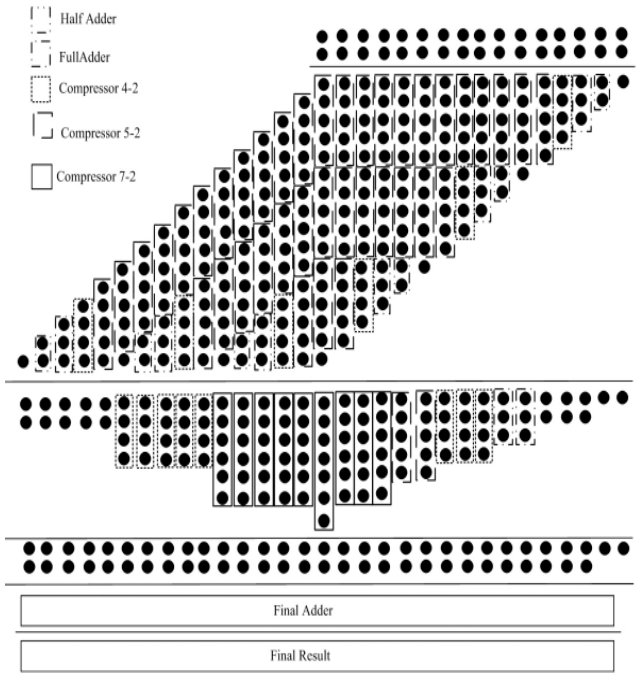

Fig. 5: 16X16 Multiplier Architecture

III. PROPOSED METHOD

The sole metrics utilised to evaluate the adder's performance in the device are the processor's or system's high speed and precision. The Ripple Carry Adder (RCA), Carry Propagate Adder (CPA), and Carry Look Ahead Adders, three 32-bit carry adders with various addition durations (delay), space requirements, and power requirements, were formerly utilised in CPUs (CLA). The postponement of any binary adder is calculated by the speed with which the carry arrives at each bit location. The carry chain, which creates the carry bit, is thus the most challenging barrier to overcome when constructing a binary adder[14][15]. The aforementioned 32-bit basic carry adders, on the other hand, exhibit significant delay values in the higher-order bits, as each level of the adder must wait for the previous carry. Because of the previously described difficulty with 32-bit basic existing carry adders, a processor or PPA is a well-suited adder in the current technological context for high-speed addition operations with decreased latency in VLSI technology[6][7][8]. The PPA design is also one of the most popular options, yielding good results. The PPA is a popular design that offers an excellent balance of speed, power, and available space. Before the creation of 8-bit and 16-bit computers, low-order PPAs were constructed. By the conclusions of the 64-bit PPA investigation, the Ladner Fischer Adder performs best among adders with low power and minimal response time[20]. The most fundamental function of every digital system is the addition.

The construction of more complicated structures, such as the arithmetic logic units of microprocessors, depends heavily on the efficient operation of adders. While circuit levels address the area and other elements, arithmetic operations are covered by logical levels[16]. The following sections analyse several adder designs with an emphasis on each device's size, number of components, and speed.

It is crucial to select the adder topology in VLSI design that results in the required performance. However, it won't be until the design is complete that a topology's performance can be determined. Therefore, the question of whether or not a higher performance is obtained or if a better VLSI adder architecture exists remains. In most cases, the answers to such questions are unknown. The community of computer mathematicians does not currently employ a reliable and realistic method for speed estimates. The majority of algorithms rely on antiquated techniques to count the number of logic gates in the critical path, which leads to erroneous and deceptive results[12]. Different circuit approaches and algorithms have been applied to the implementation of parallel prefix adders under various limitations. The adder is an excellent case study for examining design techniques due to its broad range of implementations. The goal of this study is to present a list of energy-efficient methods that any prefix calculation algorithm may use. To acquaint the reader with the principles of adder circuits, the adder topologies are first emphasised[11].

The delay caused by the ripple effect of the carry bits is the primary drawback of RCA. The wait will be over when the Adder appears (CLA). The CLA functions on the assumption that it examines the input's lower-order bits and adds whenever a high-order carry results. The carry propagation delay is proportionately shorter for CLA than RCA because CLA has fewer gates[9]. To quickly identify whether each group of digits will propagate a carry that comes from the right, it is necessary to compute whether each digit position will do so if a carry arrives from the right and combine these calculated values.

A carry generator, a sum generator, and a propagate/generate block are the three generators. Equation 1 permits the generated block to be realised.

$$Gi = Ai.Bi \text{ for } i = 0,1,2,3\ldots\ldots\ldots\ldots\ldots\ldots..(5)$$

Similarly, Equation 2 could be employed to implement the propagate block:

$$Pi = Ai \oplus Bi \text{ for } i = 0,1,2,3\ldots\ldots\ldots\ldots\ldots..(6)$$

The carry output of the $(i-1)^{th}$ stage is obtained from Equation 3:

$$Ci(\text{ cout }) = Gi + Pi.C(i-1) \text{ for } i=0,1,2,3.. \ldots\ldots.(7)$$

Equation 4 could be utilized to figure out the total output:

$$Si = Ai \oplus Bi.C(i-1) \text{ for } i=0,1,2,3.. \ldots\ldots..(8)$$

One of the quickest adders ever created and developed is known as the Parallel Prefix Adder (PPA). Parallel prefix adders were found to be the most efficient circuits for binary addition. These adders, also known as carry tree adders, have indeed been found to be more efficient in VLSI architectures[10].

A. Parallel Prefix Adders

The notion behind parallel prefix adders is first to compute a small group of transitional prefixes, then locate the large group prefixes, and so on until all of the carry bits have been evaluated[17]. To produce the final sum operation, parallel prefix adders are divided into three stages:

1. Pre-processing.
2. Generation of carry.
3. Final processing.

B. Pre-Processing Stage

The computation of produced and propagated signals corresponding to each pair of bits in A and B is done during the pre-processing stage.

$$Pi = AiBi\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(9)$$

$$Gi = Ai.Bi\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(10)$$

C. Carry Generation Stage:

The carriers for each bit are calculated at this stage. Simultaneous execution occurs. They are fragmented into smaller pieces after the simultaneous computation of carries. There are two AND gates and one OR gate in the carry operator. To create intermediate signals for propagation, Equations 3 and 4 have been used.

$$P(i:k) = P(i:j) \bullet P(j-1:k)\ldots\ldots\ldots\ldots\ldots\ldots\ldots(11)$$

$$G(i:k) = G(i:j) + G(j-1:k) \bullet P(i:j) \ldots\ldots\ldots\ldots....(12)$$

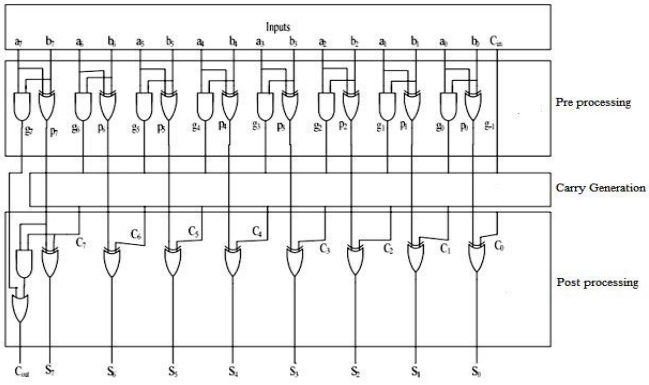

Fig.6: Overall architecture of Parallel prefix addition

This carry is created by utilising three separate cell architectures known as the Grey cell, Black cell, and Buffer cell. Using these cell structures, which are the same for all parallel prefix adders, we can compute the final carry. However, the carry generation design differs depending on the type of parallel prefix adder.

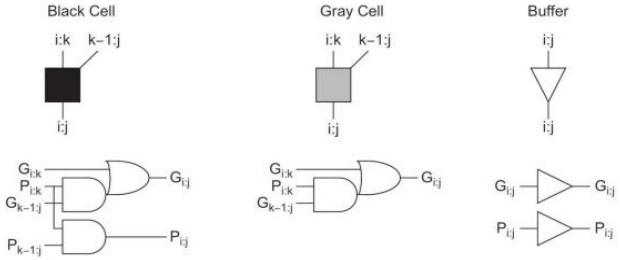

Fig. 7: Buffer cells, Black cells, and Grey cells are engaged during the carry generation stage.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

D. Post-Processing Stage

At this stage, the final sum is generated based on the propagate and carry values generated from previous stages. The Boolean expression is shown in equation 5.

$$si = pi \oplus ci-1 \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (13)$$

In VLSI design, a parallel prefix adder offers the best performance. Each bit performing an addition operation in a ripple carry adder waits for the bit preceding it to complete its addition operation[18]. In an effective Ladner-Fischer adder, changes are made at the gate level to increase speed and decrease area without waiting for the previous bit addition operation to complete.

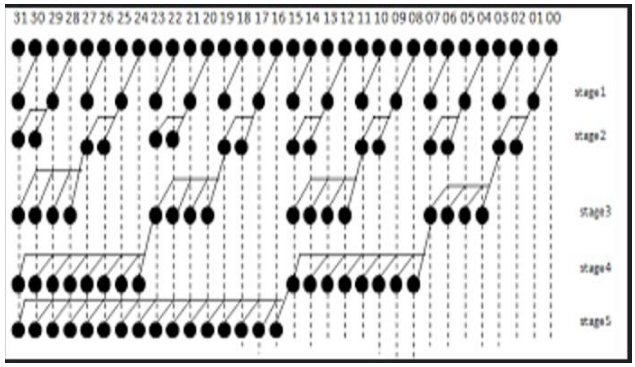

Fig. 8: 32-bit Ladner-Fisher adder

IV. EXPERIMENTAL RESULTS, COMPARISON AND DISCUSSION

As a result, in the final addition, a parallel prefix adder was used, which has lower latency than a ripple carry adder. Among all the Ladner-Fischer adders, numerous distinct types of parallel prefix adders were chosen for their ability to utilise less delay. Furthermore, when the decreased carry ripple issue between consecutive cells is compared to the overall transistor and gate count, the PDP-recommended architectures are less complex than the earlier designs.

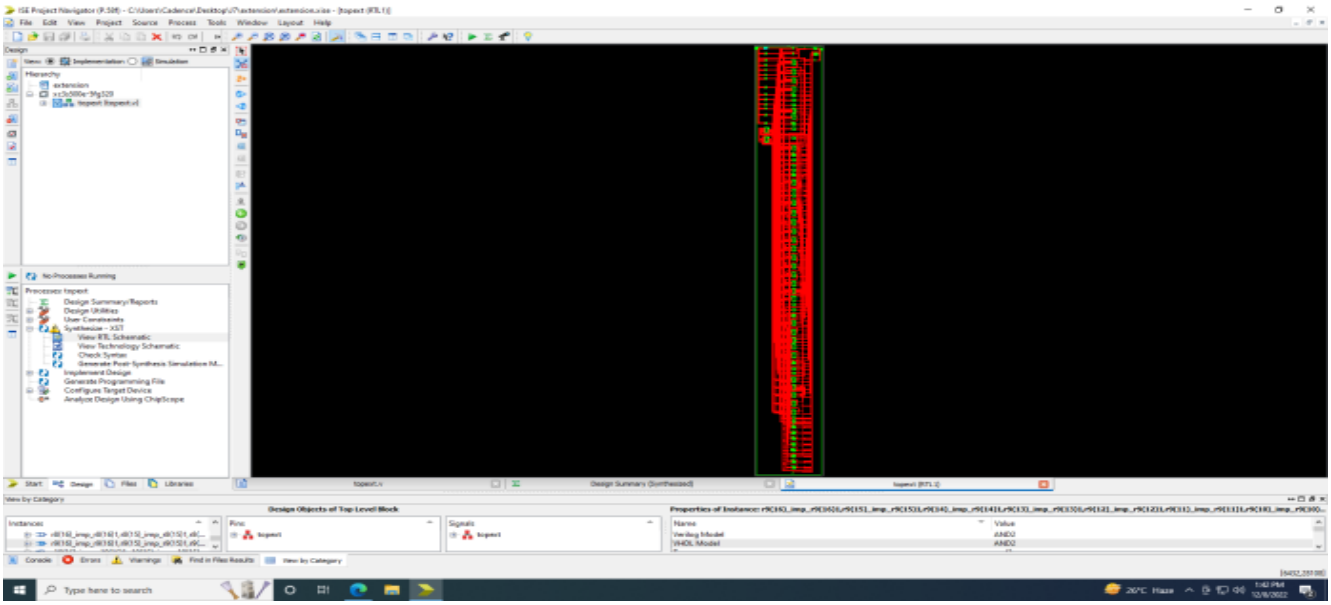

Fig. 9: RTL Schematic

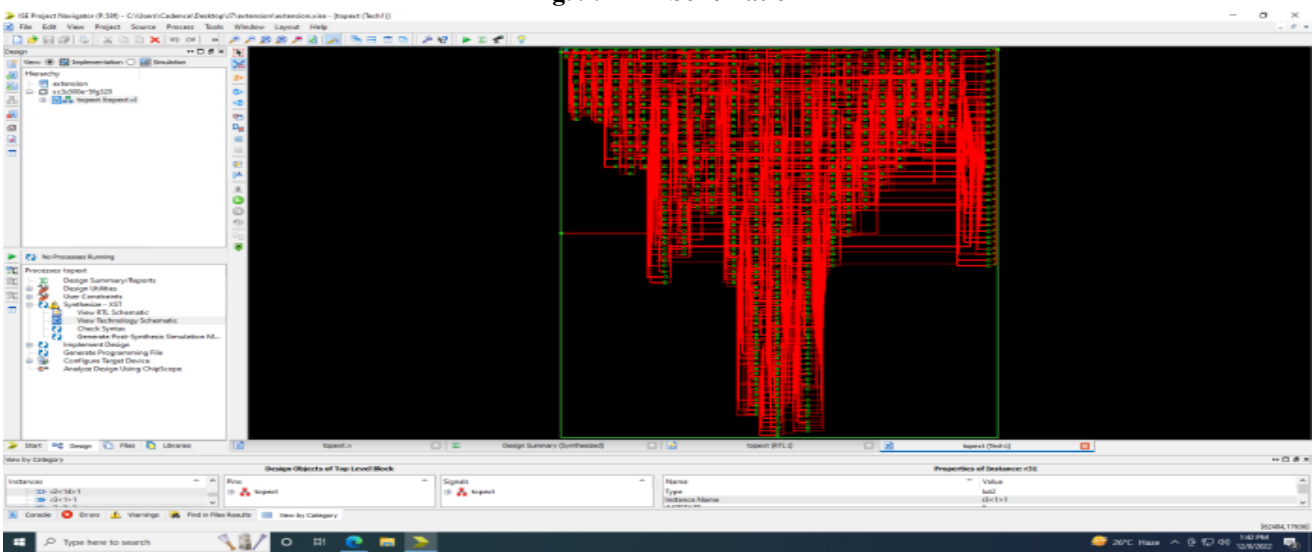

Fig. 10: Technology Schematic

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

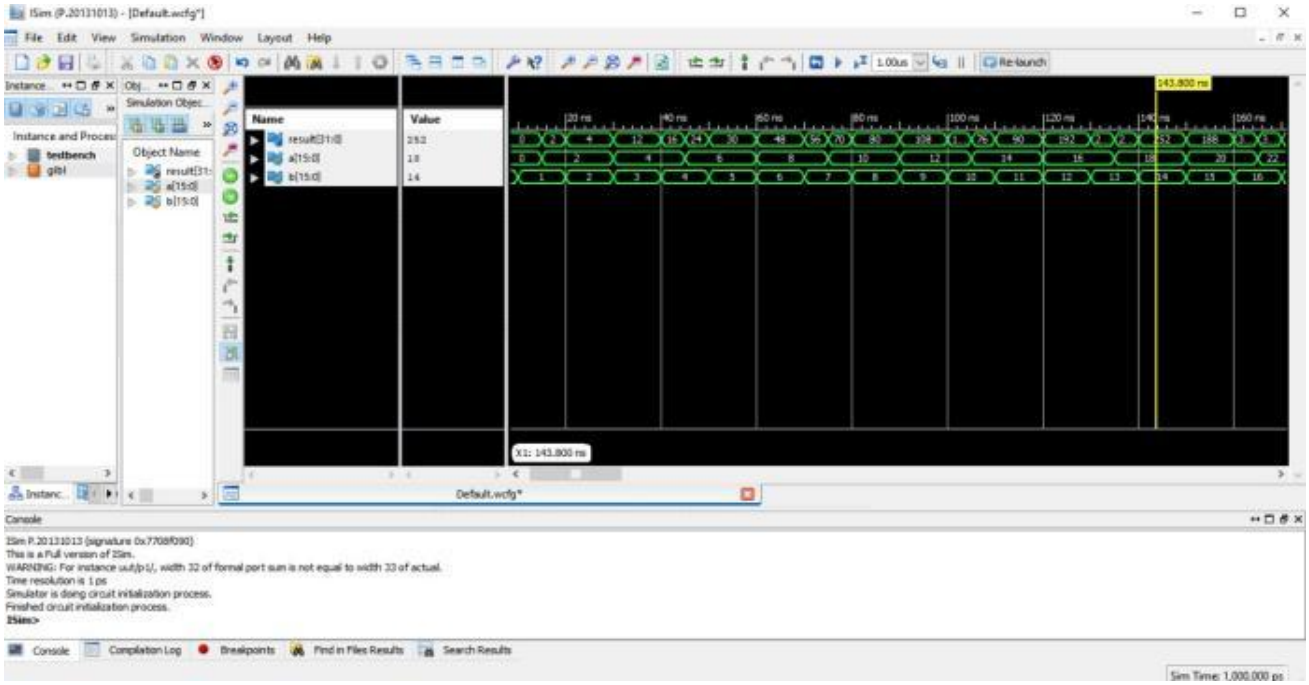

Fig. 11: Simulation Results

```
Slice Logic Utilization:
 Number of Slice LUTs:                  506  out of  204000     0%
    Number used as Logic:               506  out of  204000     0%
```

Fig. 12: Area

```
Timing constraint: Default path analysis
  Total number of paths / destination ports: 368252 / 32
-------------------------------------------------------------------
Delay:               7.736ns (Levels of Logic = 17)
  Source:            a<6> (PAD)
  Destination:       result<31> (PAD)
```

Fig. 13: Delay

```
2.3.  Power Supply Summary
-------------------------------------------------------------
|                  Power Supply Summary                     |
-------------------------------------------------------------
|                     | Total  | Dynamic | Static Power |
-------------------------------------------------------------
| Supply Power (mW)   | 157.46 | 14.46   | 143.01       |
-------------------------------------------------------------
```

Fig. 14: Power

Comparison Between Existing and Proposed Methods:

Table 1: Evaluation table for Area, Delay of Proposed and Existing methods.

	Area	Delay (NSEC)	PDP (NJ)
Existing	466/63400	19.639	1.343
Proposed	506/63400	7.736	1.218

V.CONCLUSION

To improve speed performance, this paper proposes a novel design method for parallel prefix adders and n2 compressor structures. As a consequence, high-speed compressors with PPA of the Ladner Fischer adder type were put into use. The key benefit of the planned constructions is the significantly shorter delay when compared to earlier reported activities. Starting with a ripple-carry adder, it proceeds to a carry-lookahead adder, followed by a parallel-prefix adder, which exhibits improved speed compared to prior published studies. When the carry ripple issue between cells in a row is taken into account, as well as the overall transistor and gate count, it is obvious that the PDP-presented architectures are less complicated than the preceding designs [19]. As a result, high-performance architectures well-suited for high-speed multipliers can be developed, making them preferable.

DECLARATION

Funding/ Grants/ Financial Support	The institution provides a licensed version of Xilinx for this project's development.
Conflicts of Interest/ Competing Interests	To the best of our knowledge, there are no conflicts of interest.
Ethical Approval and Consent to Participate	No, the article does not require ethical approval or consent to participate, as it presents evidence that is not subject to interpretation.
Availability of Data and Material/ Data Access Statement	Yes, it is relevant. Most of the data are collected from various papers and journals, as mentioned in the references.
Authors Contributions	All authors have equal participation in this article.

REFERENCES

1. A. Fathi, S. Azizian, K. Hadidi, and A. Khoei, "A novel and very fast 4-2 compressor for high-speed arithmetic operations," IEICE Trans. Electron, vols. E95, no. 4, pp. 710–712, 2012. [CrossRef]
2. I. S. Abu-Khater, A. Bellaouar, and M. I. Elmasry, "Circuit techniques for CMOS low-power high-performance multipliers," IEEE J. Solid-State Circuits, vol. 31, no. 10, pp. 1535–1546, Oct. 1996. [CrossRef]

3. O. Kwon, K. Nowka, and E. E. Swartzlander, "A 16-bit × 16-bit MAC design using fast 5:2 compressors," in Proc. IEEE Int. Conf. Appl.-Specific Syst., Archit., Processors, Jul. 2000, pp. 235–243.
4. W.-C. Yeh and C.-W. Jen, "High-speed booth encoded parallel multiplier design," IEEE Trans. Comput., vol. 49, no. 7, pp. 692–701, Jul. 2000. [CrossRef]
5. A. M. Shams, T. K. Darwish, and M. A. Bayoumi, "Performance analysis of low-power 1-bit CMOS full adder cells," IEEE Trans. Very Large Scale Integration. (VLSI) Syst., vol. 10, no. 1, pp. 20–29, Feb. 2002. [CrossRef]
6. P. J. Song and G. De Micheli, "Circuit and architecture trade-offs for high-speed multiplication," IEEE J. Solid-State Circuits, vol. 26, no. 9, pp. 1184–1198, Sep. 1991. [CrossRef]
7. V. G. Oklobdzija, D. Villeger, and S. S. Liu, "A method for speed optimized partial product reduction and generation of fast parallel multipliers using an algorithmic approach," IEEE Trans. Comput., vol. 45, no. 3, pp. 294–306, Mar. 1996. [CrossRef]
8. C.-H. 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. I, Reg. Papers, vol. 51, no. 10, pp. 1985–1997, Oct. 2004. [CrossRef]
9. A. Fathi, S. Azizian, K. Hadidi, A. Khoei, and A. Chegeni, "CMOS implementation of a fast 4-2 compressor for parallel accumulations," in Proc. IEEE Int. Symp. Circuits Syst., May 2012, pp. 1476–1479. [CrossRef]
10. K. Prasad and K. K. Parhi, "Low-power 4-2 and 5-2 compressors," in Proc. Conf. Rec. 25th Asilomar Conf. Signals, Syst. Comput., Nov. 2001, pp. 129–133. [CrossRef]
11. P. Saha, P. Samanta, and D. Kumar, "4:2 and 5:2 decimal compressors," in Proc. 7th Int. Conf. Intell. Syst., Modelling Simulation (ISMS), Jan. 2016, pp. 424–429 [CrossRef]
12. D. Balobas and N. Konofaos, "Low-power high-performance CMOS 5-2 compressor with 58 transistors," Electron. Lett., vol. 54, no. 5, pp. 278–280, Mar. 2018. [CrossRef]
13. A. Najafi, S. Timarchi, and A. Najafi, "High-speed energy-efficient 5:2 compressor," in Proc. 37th Int. Conv. Inf. Commun. Technol., Electron. Microelectron., Opatija, Croatia, May 2014, pp. 80–84. [CrossRef]
14. M. Tohidi, M. Mousazadeh, S. Akbari, K. Hadidi, and A. Khoei, "CMOS implementation of a new high-speed, glitch-free 5-2 compressor for fast arithmetic operations," in Proc. 20th Int. Conf. Mixed Design Integr. Circuits Syst. (MIXDES), Jun. 2013, pp. 204–208.
15. G. Caruso and D. Di Sclafani, "Analysis of compressor architectures in MOS current-mode logic," in Proc. 17th IEEE Int. Conf. Electron., Circ. Syst., Dec. 2010, pp. 13–16. [CrossRef]
16. W. Ma and S. Li, "A new high compression compressor for the large multiplier," in Proc. 9th Int. Conf. Solid-State Integrated-Circuit Technol., Oct. 2008, pp. 1877–1880.
17. M. Rouholamini, O. Kavehie, A.-P. Mirbaha, S. J. Jasbi, and K. Navi, "A new design for 7:2 compressors," in Proc. IEEE/ACS Int. Conf. Comput. Syst. Appl. (AICCSA), May 2007, pp. 474–478. [CrossRef]
18. S. Mehrabi, K. Navi, and O. Hashemipour, "Performance comparison of high-speed high-order (n:2) and (n:3) CNFET-based compressors," Int. J. Model. Optim., vol. 3, no. 5, pp. 432–435, Oct. 2013. [CrossRef]
19. C. Pan, Z. Wang, and C. Sechen, "High speed and power efficient compression of partial products and vectors," J. Algorithms Optim., Oct., vol. 1, no. 1, pp. 39–54, 2013.
20. G. Yang, S.-O. Jung, K.-H. Baek, S. Hwan Kim, S. Kim, and S.-M. Kang, "A 32-bit carry-lookahead adder using dual-path all-N logic," IEEE Trans. Very Large.

AUTHORS PROFILE

Dr. J.V.R. Ravindra received a Bachelor's degree in Electronics and Communication Engineering from Osmania University, Hyderabad, India, and a Master's degree in Systems and Signal Processing from the same university. He received a PhD degree from the Centre for VLSI and Embedded Systems Technologies of the International Institute of Information Technology (IIIT-H), Hyderabad, India. He is currently working as a Principal and Professor at Vardhaman College of Engineering, Hyderabad, India. His research interests include device-circuit co-design for nanoscale silicon and non-silicon technologies, low-power electronics for portable computing and wireless communications, and new computing models enabled by emerging technologies. He has published over 100 papers in peer-reviewed journals and conferences and holds two patents. He is a senior member of IEEE and ACM, and a fellow of IETE.

Chava Chaitanya is currently pursuing her Bachelor's degree in Electronics and Communication Engineering from Vardhaman College of Engineering, Hyderabad, India. She is currently in her senior year. She is quick at grasping things and is enthusiastic about learning and exploring new things. During her scholastic education and through online learning platforms, she learned many principles of Verilog HDL and CMOS designs. She has proficiency with tools such as Xilinx ISE, Cadence, NI Multisim, and MATLAB. Her areas of research interest include approximation computing methods, mixed-signal VLSI design, physical design automation, low-power and high-speed VLSI design, and different VLSI technologies.

Kasam Pranya is currently pursuing her Bachelor's degree in Electronics and Communication Engineering from Vardhaman College of Engineering, Hyderabad, India. She is currently in her final year of study. She studied CMOS design and Verilog HDL fundamentals while attending college, as well as utilising web-based learning resources. She began to comprehend many concepts and gain knowledge in this field as her interest in VLSI grew. She also gained knowledge in physical design automation, low-power and high-speed VLSI design, and mixed-signal VLSI design. She is skilled in using applications such as MATLAB, Cadence, NI Multisim, and Xilinx ISE. Her work entails high-speed and efficient VLSI design.

Vaddem Sahiti is currently pursuing her Bachelor's degree in Electronics and Communication Engineering from Vardhaman College of Engineering, Hyderabad, India. She is currently in the final year of her graduation. While attending college, she used interactive learning methods to learn CMOS design fundamentals and Verilog HDL. As her interest in VLSI expanded, she began to grasp numerous concepts and expand her expertise in this field. She also learned about mixed-signal VLSI design, low-power and high-speed VLSI design, and physical design automation. Applications such as MATLAB, Cadence, NI Multisim, and Xilinx ISE are among those with which she is skilled. Her work interests include fast computational circuits.