Performance Analysis of High Speed and Low Power Binary Adders

Satyandra Bhati, Neeraj Sharma, Meetu Nag

Department of Electronics & Communication Engineering, Vivekananda Global University, Jaipur, Rajasthan,

neeraj.sharma@vgu.ac.in

Abstract. In all the arithmetic operations, addition is one of the most important and initial operations used in most of the mathematical equations. The operation is performed by many adders present in the digital world. These adders give us carries with preferred delay and power. The three main features like structure, logic, and compact circuit layout help design a better adder. This Paper aims to analyse and compare various additions for high-speed, low-power and fast calculation. The various adder designs seen in digital signal processing applications require computationally efficient adding and cumulative operations, so blocks with the required attributes must be carefully selected. Various techniques have been proposed to design efficient adders in terms of performance, low power consumption and area. This work focused on 16-bit implementations of highly optimized area-efficient Ripple Carry Adders (RCA) and Look Ahead Carry Adders (LAC), and carry select Adder. Finally, we can prove that the Carry look ahead Adder is very fast in all existing designs. These processes are simulated and synthesized in all ISE Xilinx 14.7 software.

Keywords: VLSI, Adder, Low Power, Xilinx

1. Introduction

Anyone can see that even some books on computer arithmetic contain many different circuit architectures that are widely used in practice and have different characteristics of interest. Many researchers have conducted research based on the proportional representation and analysis of data, even when they encounter the binary adder structure [1].

This treatise provides a qualitative assessment of classifying binary adder architecture. Among the vast members of the Adder, we wrote the VHDL code for Ripple carry, Carry select and Carry look to emphasize common recital attributes in the correct position in that class. The next section provides a brief description of the structural design of the Adder [2].

First-class based on very slow ripple carry Adder with minimal area and power. The second class has several levels of carry skip, the prerequisites for the carry select adder are small, and the computation time is reduced. The Carry look ahead Adder of the third class or parallel prefix adder of fourth class support fast addition methods with maximum area complexity [3].
1.1. Need for Low Power Design

There are various interpretations of Moore's Law, which predicts the growth rate of integrated circuits. According to one estimate, it doubles every 18 months. Others claim that device density increases 10-fold every seven years. I agree that all growth is fast, and there are no signs of a slowdown, regardless of the exact numbers. New generations of process technologies are being developed while current-generation devices are safe from underlying physical limitations. The need for low-power VLSI chips arose from such evolutionary forces in integrated circuits. The Intel 4004 microprocessor, developed in 1971, has 2300 transistors, consumes about 1 watt, and has a speed of 1 MHz [4].

Then came the Pentium in 2001, with 42 million transistors, about 65 watts of power, and clocked at 2.40 GHz [5]. As the power dissipation increases linearly over the years, the power density increases exponentially due to the size getting smaller and smaller out of Integrated Circuits. If this exponentially increasing power density were to continue to increase, a microprocessor designed a few years later would have a capacity equivalent to that of a nuclear reactor. Such high-power density leads to reliability problems such as electrical migration, thermal stress and device degradation caused by the hot carrier, resulting in performance loss. Another factor driving the demand for low-power chips is the increased market demand for battery-powered portable consumer electronics. The desire for smaller, lighter and more durable electronic products indirectly leads to lower power consumption. Battery life is emerging as a product differentiator in many portable systems. As the heaviest and largest component of many portable systems, batteries do not have the same rapid density growth rate compared to electronic circuits. The primary source of power dissipation in these high-performance battery-powered portable digital systems such as laptops, cell phones and personal digital assistants is gaining in importance. Low power consumption is a major concern for these systems, as it directly affects performance by affecting battery life [6]. In this situation, low power VLSI design has become important as an active and rapidly growing field. At the circuit design level, significant energy savings potential exists through the appropriate choice of type for the implementation of combinational circuits. This is because of all the important parameters governing power dissipation switching [7].

1.2. Logic Style Requirements for Low Power

Reducing each of these parameters results in a decrease in power dissipation. However, reducing the clock frequency is only possible at the architecture level, and at the circuit level, the frequency is generally considered constant in order to meet certain speed requirements. All other parameters are affected to some extent by the type of logic applied. Therefore, some general logic type requirements for implementing low power circuits can be stated at this point [8, 9].

- Switched Capacitance Reduction
- Supply Voltage Reduction
- Switching Activity Reduction
- Short-Circuit Current Reduction

2. Motivation

The need for low-power VLSI systems comes from two main factors. First, the stable expansion of operating frequency and processing power per chip, supply a bulky current, and appropriate cooling technology must remove the heat caused by large power consumption. Second, the battery life and low-power design of portable electronic devices frankly prolong the operating time of these convenient devices. [10]

This design implements additional logic to determine the Carry and increase additional speed without waiting for the total to complete. This is done by adding an intermediate signal called
propagate creation. The Signal generate describes the conditions for digit addition to generating a carry, regardless of the Cin signal. The propagate describes the condition of single addition propagating the Carry only if the Cin signal is a one.

2.1. Objective

Ripple-and-look-ahead architectures can be designed using an iterative approach [11]. This paper shows two novel methods of binary adders. The first approach presents a new Ripple and look ahead architecture design that simultaneously uses the Ripple and Look Ahead architectures to reduce the critical path and area. The following objectives of this work are given below:

- To design Ripple adder, carry select and look ahead Adder is an efficient manner.
- To reduce the area, power and delay to meet the fast operation of the logic.

2.2. Overview of Various Adder Architectures

Adders are one of the key components of any data path. Since any component in VLSI digital design, the alternative of adder architecture is constrained by the imperative factors of area, speed, power and delay. Amongst the various architectures of adders available, some of them are provisionally described in this section. Figure 1 shows technology decreasing day by day [13].

![Figure 1: Overview of Various Adder Architectures](image)

(a) Ripple Carry Adder

Ripple Carry Adder (RCA) is the simplest add-on tool for digital design. This Adder takes two binary inputs, N bit (where N is an integer) and produces (N + 1) the production bits (sum N bit and implementation 1bit. The RCA makes N full adders in cascade, and the FA bit connects to the next FA bit from analysis time. Figure 3.7 shows a representation of an N-bit RCA. [12] The input area or b, the execution of each FA is labeled counts as C_{out} (corresponding to the execution (C_{in}) of the next FA), or the sum bits are labelled as a sum [13]. Before counting, the operand and C_{in} need to be entered in each small amount. If we want to get close to the delay of the spread of this Adder, we need to study the delay of the worst case rather than the possible combination of double entry. This is also called the important path. The most significant amount can only be calculated when the previous FA administration is known. In the worst case (if the total lead bit is 1), this lead bit needs to be varied in the system from the smallest number of effective permutations to the most effective location Figure 3.8 has a black line representing the defined path. [14]
(i) Delay

Another important factor than the time between carry-in to carryout to sum is the turnaround time to implementation, as the carrier propagation chain will conclude the entire circuit delay until the Adder. Therefore, considering the worsening of the signal path above, we can write the following equation. For a k-bit RCA, the interrupted path in the worst case is:

\[ TRCA - k \text{ bit} = TFA(x_0, y_0, c_0) + (k-2) * TFA (Cin Ci) + TFA (Cin Sk) \]

Complexity and delay for n-bit RCA structure logic equations

\[ ARCA = O(n) = 7n \] ………………..……….1
\[ TRCA = O(n) = 2 \] ……………….….…2

Consequently, the time for this execution of the Adder where t RCA carry is the delay for the Carry out of a FA and t RCA sum is the delay for the sum of a FA. We can see that the delay is comparative to the length of the Adder [15]. An example of the worst-case propagation entry mode for a 4-bit addict is the conversion of inputs 1111 and 0000 to 1111 and 0001, causing the currency to change from 01111 to 10000. We have shown the results of the 1-bit ad. [16]. In terms of VLSI digital design, this is the easiest Adder to use. It only needs to design and order FA elements and then sequence those N elements to generate N-bit RCA. The concert in the FA band determines the overall RCA speed. Entering the pathway amplifying the FA (t RCA bearing) delay will reduce the RCA prompt. The application of the FA group can be selected to reduce the application delay [17].

![Figure 2: Critical Path for an N-bit Ripple Carry Adder](image)

(b) Carry Look Ahead Adder

From the critical path from the equations, the delay is linearly dependent on N, i.e. the length of particular Adder.

\[ Pi = Ai XOR Bi \] …………………….3
\[ Gi = AiBi \] Carry generate…..........4
\[ Si = Pi XOR Ci \] Sum………………….5
\[ Ci+1= Gi + PiCi \] Carry out ………………6

Equations 3 to 6 show that the carryout Signal may be thrown mainly on the delay. Algorithms that condense the time to compute carryout and condense N linear habits can significantly shorten the extra work. Equation 3 shows that the carry-in can be computed with g, p, and carry-in [18]. Signals g and p are not required for carry-in and can be computed when the two input operands are present. The CLA adder uses a partial full adder to calculate the resulting signal propagation signal required for the performance of the equation. Figure 2 shows the schematic of the added 4-bit CLA. The logic for each PFA block is shown in Figure 3. The CLA logic block implements the logic of equations 7 to 10, and the gate of this block is shown in Figure 3. For a 4-bit CLA adder, the 4th execution signal can also be considered the 5th total bit. Having single-level carry look adder logic for a long adder is impractical
but can be explained by adding other levels of CLA logic [19]. To achieve this, each adder block requires two additional signals: group creation and group propagation.

\[
\begin{align*}
C_1 &= G_0 + P_0 C_0 \\
C_2 &= G_1 + P_1 C_1 + (G_0 + P_0 C_0) = G_1 + P_1 G_0 + P_1 P_0 C_0 \\
C_3 &= G_2 + P_2 C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0 \\
C_4 &= G_3 + P_3 C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0
\end{align*}
\]

The equations for the two signals assuming a 4-bit adder block size are given in Equations 7 and 8. Group creation can occur when one Carry produces an adder block, or group propagation occurs when a carry-in to an adder block is propagated to a carryout. Multiple levels of CLA logic allow building carry prediction adders of arbitrary length. The size of the CLA adder block is 4 bits. This is a familiar factor for most word sizes, as there are practical limits to the gate size that can be executed. To show the use of other levels of CLA logic, Figure 4 has a schematic of a 16-bit CLA adder, creating a group of each 4-bit adder sub cell and acquiring and executing a group radio signal. This indicates that it is the second level of CLA logic that computes the Signal. If the Adder has several levels of CLA logic and not more than the final level, then a c4 signal should be generated. All other levels reinstate this c4 Signal with group creation or group radio waves.

\[
\begin{align*}
\text{ACLA} &= O(n) = 14n \\
\text{TCLA} &= O(\log n) = 4 \log n
\end{align*}
\]

Figure 3: 4-bit Carry Look-Ahead Adder

Figure 4: Schematic for a 16-bit CLA Adder
Complexity and Delay for n-bit CLA structure use third-level CLA logic and four blocks of 16 docs to create 64-bit docs. The CLA logic generates the $c_{16}$, $c_{32}$, and $c_{48}$ signals that are used as input to carry the 16-bit adder block and generates $C_{64}$ to the sum64 Signal. If the format requires a length of 32, the creator uses two 16-bit blocks and two load-bearing signals ($c_{16}$, $c_{32}$) in the third phase of the CLA logic. Free. A similar device to the CLA. Logic simplifies building a long ad using this system, and the adder block can be installed via the subunit. Defining the critical path of the CLA adder is difficult because the leading path gates have different nibs. If you want to have a rough idea, first think that the delays of all gates are the same. The delay of the 4-bit CLA adder requires a gate delay to count the waveform and generate the Signal and a two-gate delay to count the carrying signal and calculate the total Signal. Therefore, it is necessary to postpone the gate. It represents the postponement of the four gates. For 16-bit CLA servers, a single gate delay (from PFA) is used to calculate the frequency and output signal and a two-gate delay used to calculate the group and generation propagation is the first step in the logic rule. The delay of two gates is a gate of logic carrying a first level and a second level carrying a logic signal. Compared to the four-bit CLA ad, the second-level logic in the 16-bit CLA address provides two other gate delays, resulting in six gate complaints. Continue as follows (64 bits extension requires eight gate delays; 256 bits extension requires ten gate delays) The CLA address delay is use redundancy to add two numbers to further increase the rate of addition. That is, we can add two of any number of total bits.

2.3. Ripple carry algorithm

It successively adds the two least significant bits to produce a total bit of equal importance and one Carry output bit of higher importance. Post results after hours. Bits continuously $i = 1, 2, \ldots, N$ adds the two operand bits and the importance $i$ carry bit to produce a total bit of equal importance and one Carry bit of greater importance. The Post result changes the $Si$ time after switching the input operand that caused the change. Even if the Ripple carry algorithm references constant updates, the updates are rarely reflected in a source switch.

Figure 5 shows a prominent event designed to add two carry generations. The Carry propagates across the Adder, with the upper 4 bits executing the path through another path. More specifically, assuming that the adders all start with 0 carry, the first step in the algorithm is to generate the sum of heats 0 and 3 and the Carry in columns 1 and 4. The total calculation for the other columns will be recreated later. The second stage produces results such as columns 1 and 4. Three-step carry propagation is used to recalculate the Carry in column 3. This will have to start over because the calculations in the previous three columns will be unfounded. This will propagate the new Carry to column 4. Recalculating column 4 produces another

![Figure 5: Ripple Carry Addition](image)

The algorithm exits after step 5 because it totals a bit but does not generate another carry bit. Note that the correct most significant total bit was generated in step 3, but further two-step addition was not complete because the lower low bit was changed.
Look Ahead Adder

Carry look ahead logic uses the concept of generating and propagating Carry. It is most natural to think of creation and propagation in the context of binary addition in Carry look ahead adders, but the concept can be used more commonly than this. Carry look precedence depends on two points.

- If coming in from the right, calculate whether the position propagates the Carry for each digit position.
- You can combine these calculated values to quickly infer whether that group propagates the Carry coming from the right for each group of numbers.
- For all 1-bit adders, calculate the result. At the same time, the predictor performs the calculation.
- Assuming the Carry occurs in a particular group, the Carry will appear from the left edge of the group within a maximum of 5 gate delays and will begin propagating to the left through the group.
- If the Carry is continuously communicated through the next group, the predictor has already inferred this. Therefore, before the Carry comes out in the next group, the look ahead device can immediately notify the next group on the left (with one gate delay) that it will receive the Carry, and at the same time, notify the next group. The next predictor is on the left side, where the Carry is in progress.

3. Simulation and Results

3.1. Adder Implementation

Execution of 16-bit Ripple carry Adder and look ahead adder RCA has been done using Xilinx 14.1 and simulator has mention out by I-Sim 14.1e tool. Ripple carry Adder and look ahead Adder. Optimized Adder is meaningful that the worst-case delay of Ripple Carry Adder or look Ahead Adder. Delay among Carry in and carry out each full Adder; I investigate another optimized design of full Adder to see if it will get a better presentation. The top level of the Ripple carry Adder is the same, though the full adder schematic is different, as Figure 6. The intuition of this full adder optimization is to minimize the delay of Carry in and carry out of the full Adder, which is the answer direct of Ripple carry adder.

Figure 6: Ripple Carry Adder

Binary input, a(15:0) ,b(15:0) and c_in are the inputs which are going to (N-1 down to 0) s(15:0), are outputs shown in figure 6. The RTL view of the 16-bit binary coded decimal converter. RTL view of 16-bit Ripple carry Adder and look ahead adder input-output and RTL view.

RTL view of 16-bit Ripple carry Adder, RTL view of sequential output like counter and flip flop which count signal propagation and show I/O lines of mux and flip flops as shown in simulation. It generates indication bits for carry propagation and generation (P and G) of each full Adder and
calculates all the carries simultaneously. The drawing of the spread full Adder and carry look ahead component is as Figure 7. Carry look ahead logic uses concepts of generating or propagating carries.

![Figure 7: RTL View of 16-Bit Carry Look Ahead Adder](image)

A Carry Look Ahead Adder (CLA), or Fast Adder, is a type of Adder used in digital logic. Carry look ahead adder recover speed by reducing time obligatory to settle on the carry bit. The Ripple carry Adder requires the Carry to propagate to the gate level before getting a gun at the output terminal. An n bit ripple carry Adder has 2n gate levels. Propagation time can be a factor that hinders the speed of the circuit. One of the most widely used processes to reduce latency is to use the Carry look ahead mechanism.

3.2. Simulation and Synthesis Results

Run simulations and synthesis and predict results of all adders. Functional verification (simulation) and synthesis of all adders (where high-level techniques are converted to RTL) are run, summarized the results. Here is a summary of the 16-bit Carry look build results. Design and implement a circuit to build a 16-bit carry look ahead adder (how the carry look-a-head functions and how to derive the final circuit layout), builds and simulates the circuit using Xilinx. Accurate display of all three simulation bags and REAL time to completion of XST: 21.00 seconds Total CPU time to completion of XST: 21.27 seconds Number of slices: 18/5472,4 Number of Input LUTs: 32/10944 Few IOs: 50 Number of combined IOBs: 50 240

- Design Statistics: IOs: 50
  - BELS: 32
  - LUT3: 32
  - IO Buffers: 50
  - IBUF: 33
  - OBUF: 17
Every part of single-ended outputs in this plan are with slew rate imperfect productivity drivers. The hindrance on speed-critical single-ended outputs can be dramatically condensed by designating them as fast outputs.

![Simulation Wave Form of Carry-Look Ahead Adder](image)

Figure 8: Simulation Wave Form of Carry-Look Ahead Adder

Figure 8 shows the state and state_next Signal of the Simulation Wave Form of Carry-look ahead Adder shown in figure 12 simulation waveform of Simulation Wave Form of Carry-look ahead Adder.

### 3.3. Timing Analysis

It is important to realize the importance of timing analysis before taking the design to the next step. Timing analysis allows you to set time limits on rigorous designs to improve performance ultimately. The Carry look pre-adder allows addicts to exploit space to determine the number of associated logic gates. Logic gate limits fix space utilization.

![Simulation Wave Form of Ripple Adder with Input](image)

Figure 9: Simulation Wave Form of Ripple Adder with Input

Figure 9 gives an overview of the overall simulation of a 16-bit binary code decimal converter. Executed using Simulator (Isim) 14.7, described using a VHDL dataflow model that represents the delay and time of the configuration of all 16-bit Carry look pre-adders and ripple adders with the entire combinatorial logic of binary code decimal design. The 16-bit Carry look pre-adder, ripple adder, and multi-operand design are set so that all inputs have a 100% clock charge. Based on the proposed algorithm, a binary code decimal configuration was designed, and the 16-bit Carry looks pre-
adder and ripple adder of the proposed algorithm were replaced with the architectural structure for a fair comparison. The composite results show that there is a diminution in impediment power and area. This results in reduced power delay product or smallest duration: 2.600ns (maximum frequency: 384.645MHz) & smallest input arrival time before clock: 2.765ns & maximum output duration after clock: 4.714ns greatest combined path delay.

All the 16-bit Carry-look ahead Adder and Ripple Adder and adding and shifting arrangement were illustrated utilizing VHDL data flow modelling and imitation employing simulator (Isim) 14.7. The 16-bit Carry-look ahead Adder or ripple adder are mapped on RTL Compiler on 14.7 (Isim). All inputs are set to have a clock rate of 100%.

4. Conclusions and Future Work

All three models of Ripple Adder, Look-Ahead Adder, and Carry Select Adder are designed, and the results are compared in terms of delay implemented in VHDL using the Xilinx 14.7 ISE tool. The performance of the three adders on the delay is evaluated by implementing the logic using the predictive Adder in the adder part. The planned intend is optimized in terms of latency or hardware complexity. First, since the gate used here is a carry look ahead adder parity storage gate, Adder maintains full parity. Therefore, if no errors are detected, no intermediate checks are required. Total REAL time to completion of XST: 21.00 seconds Total CPU time to the conclusion of XST: 21.27 seconds, number of 4 input LUTs 3210,944. The number of occupied slices 245,472 Contains only related logic Slice 24 uses of 24, in the world of low power VLSI design, anything is possible, and in the future, all adder designs may be more accurate than all previous designs. This ultimately improves area efficiency because it can be implemented with fewer gates in the computer hardware. The proposed ripple adder forms the centerpiece of a multi-operand binary adder. The simulation results show the capabilities of the proposed ripple adder, carry select Adder, look ahead Adder in addition to the multi-operand decimal adder for conventional designs.

The Carry Select Adder, which is designed in this paper, is very suitable for implementing the VLSI circuits. The optimized delay power among the adders, power consumption, and reducing the transistors is carried out. The overall objective of the project is to construct a static and compact carry select adder that can be implemented for further purposes.

In an attempt to develop arithmetic algorithms and architecture-level optimization techniques for low-power, high-speed multiplier design, the method presented in this paper has achieved good results. However, there are limitations to this work, and several possible future research directions are as follows.

Possible directions are radix higher than base four recordings. radix4 recoding is being considered in the work of this paper because it is a simple and popular choice. Higher radix recoding allows power saving to reduce the number of PPs further. To improve the performance, you can use a higher-order deep-level pipeline architecture to improve speed. All designs can be identical at the circuit level where Vt can be changed for low power applications, made at the architectural level while keeping one common base for comparison. The pipeline technology is widely used to improve the performance of digital circuits. A simple calculation shows that a gate delay of around 33 is required to complete the multiplication of 32-bit operands without pipeline processing. In this number, nine gate delays belong to Wallace structure (4:2 compressor), ten gate delays belong to MBR, and 14 gate delays belong to 64-bit carry prediction adder. If you add three registers here, each will transmit one gate delay, resulting in a total of 36 gate delays. So, splitting this delay into three would require about 12 gate delays for each stage in the pipeline to complete.
5. References

1. J. L. Hennessy and D. A. Patterson and Morgan Kaufmann “Computer Architecture A Quantitative Approach” San Francisco:, 1990.
2. G. Ragunath, Viren Sugandh and R. Sakthivel, "Delay Optimized Binary to BCD Converter for Multi-operand Parallel Decimal Adder" International Conference on Vision Emerging Trends in Communication and Networking (VITECoN) Year: 2019
3. Bayadir A. Issa and Israa S. AL-Furati, "High Precision Binary Coded Decimal (BCD) unit for 128-bit addition" 2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE) Year: 2020
4. Radha N., M. Ajeeth, S. Akash and P. Muralikrishnan, "An Efficient Implementation of Decimal Adder Using Parallel Prefix Addition" 3rd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC) Year: 2019
5. Amit Kumar Panda, Rakesh Palisetty and Kailash Chandra Ray "High-Speed Area-Efficient VLSI Architecture of Three-Operand Binary Adder" IEEE Transactions on Circuits and Systems I: Regular Papers Year: 2020
6. Amr Mohammad, Mohammed E. Fouda, Lobna A. Said and Ahmed G. Radwan, "Memristor-CNTFET based Ternary Full Adders" IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS) Year: 2020
7. Avinash Jain, Somya Bansal, Shamim Akhter and Shaheen Khan, "Vedic-Based Squaring Circuit Using Parallel Prefix Adders" 7th International Conference on Signal Processing and Integrated Networks (SPIN) Year: 2020
8. S. Sridevi Sathy Priya, Benita Poul Raju, B. Benita and Dharani, "A Design of low power Adders" 4th International Conference on Devices, Circuits and Systems (ICDCS) Year: 2018
9. Jipping Fan, Yujie Gu, Masahiro Hachimori and Ying Miao, "Signature Codes for Weighted Binary Adder Channel and Multimedia Fingerprinting" IEEE Transactions on Information Theory Year: 2021
10. Junqi Huang, T. Nandha Kumar and Haider Abbas, "General Expressions for Performance Evaluation of Binary Adders" 17th International Conference on Electrical Engineering / Electronics, Computer, Telecommunications and Information Technology (ECTI-CON) Year: 2020
11. Aung Myo San and Alexey N. Yakunin, "Reducing the hardware complexity of a parallel prefix adder" IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIconRus) Year: 2018
12. Jaeyoon Park and Youngmin Kim, "Design and Implementation of Ternary Carry Lookahead Adder on FPGA" International Conference on Electronics, Information, and Communication (ICEIC) Year: 2021
13. Aung Myo Sana, lexey N. Yakunin, Han Myo Htun and Aung Thu, "Development and Result Verification of a Multi-Bit Binary Parallel-Prefix Adder-Subtractor" IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIconRus) Year: 2020
14. D Yaswanth, S. Nagaraj and R. Vishnu Vijeth, "Design and analysis of high speed and low area Vedic multiplier using carry select adder" International Conference on Emerging Trends in Information Technology and Engineering (ie-ETITE) Year: 2020
15. Sudhanshu Janwadkar and Rasika Dhavse, "Implementation and Performance Evaluation of Novel Line Adder Architecture for Portable Systems: A Vedic Mathematics" Approach 2020 IEEE REGION 10 CONFERENCE (TENCON) Year: 2020
16. Nafiz Hossain, Nowrin Hossain, Zarrin Tasnim Sworna and Mubin Ul Haque, “A Fast and Compact Binary to BCD Converter Circuit” IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE) Year: 2019
17. Anurag Chauhan, Lokesh Mahor and Piyush Tiwari, “Low Power Quaternary Adder Using CNFET” IEEE VLSI DEVICE CIRCUIT AND SYSTEM (VLSI DCS) Year: 2020
18. Sai Krishna Anumula and Xingguo Xiong, "Design and Simulation of 4-bit QCA BCD Full-adder" IEEE Long Island Systems, Applications and Technology Conference (LISAT) Year: 2019

19. Diksha Manikkule and Pravin Jaronde, "Encapsulation of Full Adder Using 180nm CNTFET" 9th International Conference on Emerging Trends in Engineering and Technology - Signal and Information Processing (ICETET-SIP-19) Year: 2019

20. Binta Lamba and Anurag Sharma, "Improved Pipeline Inner Product Computation Unit for Binary Numbers" 4th International Conference for Convergence in Technology (I2CT) Year: 2018

21. Neelam Somani, Chitrita Chaudhary and Sharad Yadav, "Reversible Adder Design for Ripple Carry and Carry Look Ahead (4, 8, 16, 32-bit)" International Conference on Computing, Communication and Automation (ICCCA) - 2016

22. M.-E. Hwang and K. Roy, "ABRM: Adaptive-Ratio Modulation for Process-Tolerant Ultradynamic Voltage Scaling," IEEE Trans. VLSI Syst., vol. 18, pp. 281-290, 2010.

23. Chetan Vudadha, Sai Phaneendra, Syed Ershad Ahmed, Sreehari Veeramachaneni, Moorthy Muthukrishnan and Srinivas M.B "An Improved Sum Computation Block for with High Sparseness", in 20th International Workshop on Logic & Synthesis (IWLS 2011), June 2011, San Diego, CA, USA.

24. J.-F. Li, J.-D. Yu and Y.-J. Huang, "A design methodology for hybrid Carry-lookahead / Carry Select Adders with reconfigurability", in Proc. IEEE International Symp. on Circuits and Systems (ISCAS), pp.77-80, May 2005.