Design of a Novel BCD Adder using Parallel Prefix Technique

Hima Bindu Challa^{1*}, Srujana Gollapalli^{2}, Dr.M.Varaprasada Rao^{3}

^{1}Scholar, ^{2}Asst. Professor, ^{3}Professor & Dean,

GIET, Rajahmundry, Andhra Pradesh 533296 India.

**Corresponding Author*

***Abstract* -** Recently, many advances are being made for designing nano-scale electronic circuits for satisfying commercial needs and at the same time meeting the low power and delay requirements. One such circuit that is mostly used for decimal operations is decimal adder. Design of multi-bit adders using QCA(Quantum-dot Cellular Automata) technique has been the ongoing trend. BCD adders that are designed based on QCA technique are being smartly exploited for further improvements.Various clocking schemes have been practiced to observe the performance of the adder circuits.This paper discusses about Parallel Prefix technique applied for the designing of efficient adders by performing more than one operation at a time. This aids in reducing the total area occupied and overall delay without compromising issues like performance and power consumption.

***Keywords* -** BCD adders, Nanoscale electronics, Parallel prefix adders, Quantum-dot cellular automata.

I. INTRODUCTION

Among the numerous digital circuits, decimal adders hold great importance considered their applications in microprocessors, signal processing etc. "The system performance is greatly influenced by the performance of the adder"[1]. The most important characteristic that determines the pace of a adder is carry propagation. The carry at one stage depends on the inputs given at that stage and the carry out from previous stage that propagates as carry in to the present stage. As the number of stages increases, the carry propagation through the stages also gets delayed. Therefore, such adders are being designed using different architectures displaying various performance characteristics. BCD addition is one type of binary addition. Known to all is that in BCD format the decimal numbers from 0 to 10 are represented using a four bit code. In BCD addition, "first the two 4-bit BCD numbers are added in binary format and then it is converted in to BCD format by adding either a binary 6(if the sum is greater than 9) or a binary 0(if the sum is less than or equal to 9)"[1]. For this purpose, the basic binary adder circuit is followed by a correction logic circuit to convert the obtained binary sum into BCD sum. Binary adders are generally classified into two types namely Serial adders & Parallel adders. Serial adders perform addition in a bit by bit fashion and due to this, computation time increases. While on the other hand parallel adders perform addition in a parallel way by taking all the available inputs at a time. By the implementation of parallel adders, the speed of the circuit has greatly increased. Many architectures supporting parallel processing such as ripple carry adder, carry look ahead and carry save adder etc., have been proposed in the past. The base for our work is "QCA based BCD adder"[2] which produces an average delay of 43.142ns. In this brief, we propose a new adder, implemented by using the parallel prefixing of the carry technique in the place of conventional adders and yields output within 32.981nswhich is less than the prior one.

II. BACKGROUND &RELATED WORK

The delay factor in binary adders is banked up on how quickly the carry moves from one bit to another bit. So the carry propagation is a hindrance in the design of binary adders. If the number of input bits increases then the carry propagation through all the bits also increases which corresponds to the delay. The binary adders and BCD adders have undergone many transformations owing to the development in the VLSI technology. A number of methods have been proposed for reducing the carry path throughout the adder. One such method proposed is "Parallel Prefix adder"[3]. A number of algorithms have been proposed for promoting PPA adders by amending one or the other parameter such as quickness, power, space etc."Sklansky (1960) came up with tree prefix algorithm"[4], where computation of intermediate signals is done by tree type structure. In the year 1973, "Kogge and Stone proposed a scheme which uses recursive doubling property and the properties of the prefix tree were determined by the minimum logic depth, structure and unity fan-out"[5].
But the problem here is more number of wires are required between the consecutive stages and the number of gates also increases which causes high power dissipation. In 1980, "Ladner and Fischer introduced minimum depth prefix graph"[6] with higher fan-out for driving larger capacitive loads. Due to this, additional buffers have to be added to the circuit which increases the cost and delay in the circuit. Ling (1981) introduced a set of new carry generation equations where one propagate term is used to simplify the group

generate function and thus reducing the burden on the first level prefix tree."Brent and Kung (1982) presented a prefix computation graph"[7] which is similar to Kogge and Stone but has less number of internal wirings. However, having appealing structure did not help it as the logical depth had increased. In 1987, "Han and Carlson came up with a new structure which is a blend of both Brent-Kung and Kogge-Stone adders"[8]. It has got comparatively less number of nodes with a minute increase of depth in logic. "Dimitrakopoulos and Nikolos (2005) presented an innovative approach where one logic level of implementation can be avoided when compared to the traditional binary adders"[9].

III. PRELIMINARIES

If we consider a full adder with inputs a_i,b_i and c_i, then the outputs obtained are sum and carry out. the sum and carry expressions for a binary adder when expressed in terms of Boolean laws can be given by

$$S_i = a_i \text{ (xor) } b_i \text{ (xor) } c_i \qquad (1)$$
$$C_{i+1} = (a_i \,.\, b_i) + (a_i \,.\, c_i) + (b_i \,.\, c_i) \qquad (2)$$

This c_{i+1} acts as the carry input to the next bit and the process goes on until all the bits present in the input sequence are added and the final output is obtained. If suppose there are n input bits in both a and b, then the final sum and carry can only be obtained only after the n^{th} bit position receives its input carry from the $n\text{-}1^{th}$ bit position. Every stage should wait for the carry input from it's prior stage. This causes more and more delay as the width of the input bits increases. To fix this issue in the binary adders, two terms called the "generate and the propagate"[10] are introduced by which the higher order carry's are calculated much before itself. This reduces the propagation time as the higher order bit positions need not wait for the carry input. Another literal called the temporary sum is also computed. The following are the terms:

$$G_i = a_i \,.\, b_i \qquad (3)$$
$$P_i = a_i + b_i \qquad (4)$$
$$t_i = a_i \text{ (xor) } b_i \qquad (5)$$

Thus with the help of the above mentioned literals, computation of sum and carry at each bit position has become much easy as shown below

$$C_{i+1} = G_i + (P_i \,.C_i) \qquad (6)$$
$$S_i = t_i \text{ (xor) } C_i \qquad (7)$$

In this way, the higher order sum and carry are calculated based on the generate and propagate terms which can be obtained by using the lower order carry and sum. This is the same mechanism implemented in "Carry Look ahead adder"[11]. The name itself says, that the aftermath carry's are predicted far before itself by using the lower order carry(s).

A. Parallel Prefix Technique - Coming to our concept of Parallel Prefix Adders, it's operation is similar to Carry Look-ahead adder but it differs in the way the carry is generated and propagated. Parallel prefix adders are one unique class of adders that effectively exploit the generate and propagate signals so that the adder circuit operates with less delay. The following figure shows the stages involved in the PPA adder.

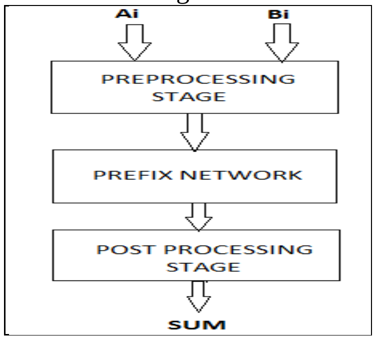


Figure1: Structure of Parallel prefix adder

This depicts that the "entire operation depends up on how the group generate and group propagate terms are calculated"[12]. There are certain other aspects such as radix, fan-out logical depth and wire tracks that effect the overall performance of the adder.

B. Topologies - Various topologies were proposed in the past which employ different logic methods for the effective utilization of PPA adders. The most popular which are also considered the best are mentioned below-

1. Kogge Stone adder
2. Brent Kung adder
3. Han Carlson adder

The above mentioned techniques just differ by the factor of logical depth. The stages of operation are same in all the three.

Kogge Stone Adder: It is one of the prefix type of carry look ahead adder, proposed by "M.Kogge and Harold.S.Stone"[5]. It offers better performance than the traditional carry look ahead adders. The total time taken by a Kogge and Stone adder to generate the output is $O(\log_2 N)$. It is considered as one of the fastest adder design with a large area and minimum fan out. It's simpler architecture helps in designing it easily and quickly. If the basics are strictly followed the architecture can be easily extended to higher orders. "The only disadvantage here is KSA adder it occupies a lot of space"[13]. The below figure shows a 16-bit Kogge-Stone adder..

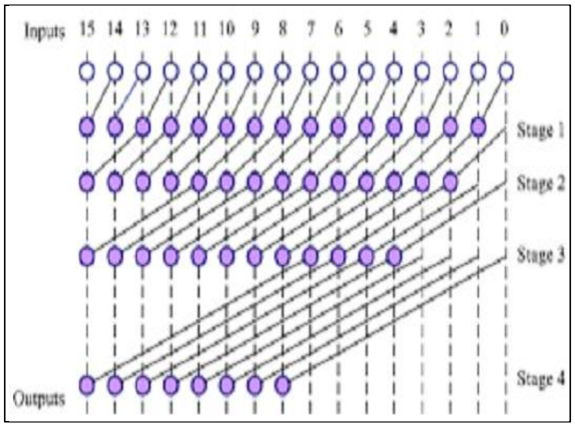


Figure 2: Kogge-Stone Adder[14]

Brent Kung Adder: "Brent kung adder"[7] is put forward by two persons Brent and Kung in the year 1982. BK Adder is mainly used when the input is bigger/have more number of bits, something which is not possible in Kogge Stone Adder as it's space occupancy increases. Brent Kung adder has got a very low fan-out from each of its prefix tree. But the critical path of Brent Kung adder is very long and the circuit structure doesn't support a fast operation. "The number of cells in Brent and Kung adder are calculated by using 2(n-1)-Log2^n"[15] and as mentioned earlier it's maximum fan-out is 2. In spite of low speed operation, Brent Kung adder is one efficient parallel prefix adder as it can accommodate more number of input bits in much less space and occupies very less space. Also it is well known for its high logical depth implementation along with minimum area attributes. The delay and area of the Brent Kung adder are given by the following expressions.

$$\text{Delay}= (2*\log_2 n)-2 \qquad (8)$$
$$\text{Area}= (2*n)-2-\log_2-n \qquad (9)$$

If there are N number of input bits, then the total number of logic levels in a Brent Kung Adder is given by 2($\log_2$ N-1). The 16-bit Brent Kung adder is shown below.

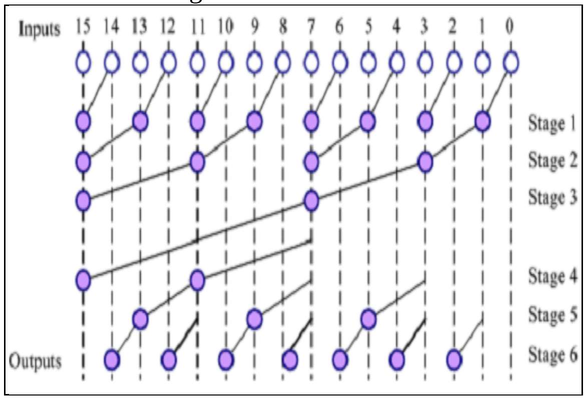


Figure3: Brent Kung Adder[14]

Han Carlson Adder: "Han Carlson adder combines both Kogge Stone and Brent Kung carry merge operations"[16]. It implements less number of cells and wiring tracks than the Kogge Stone adder. Han Carlson prefix tree can be easily yielded by performing some modifications to the pseudo code of Kogge Stone adder. Han Carlson adder differs from the Kogge Stone adder in the sense that it performs two different operations on even and odd bits. Here the even bits undergo carry merge operation and the odd bits undergo carry propagate/generate operation. The generate and propagate bits travel through the prefix tree and finally unite with carry merge bits to give the final accurate carry. It's fan-out is same as that of the Kogge Stone adder.But the fan out of Han Carlson adder may increase if an error detection and correction stage is attached to the adder. The structure of a 16-bit Han Carlson adder is given below-

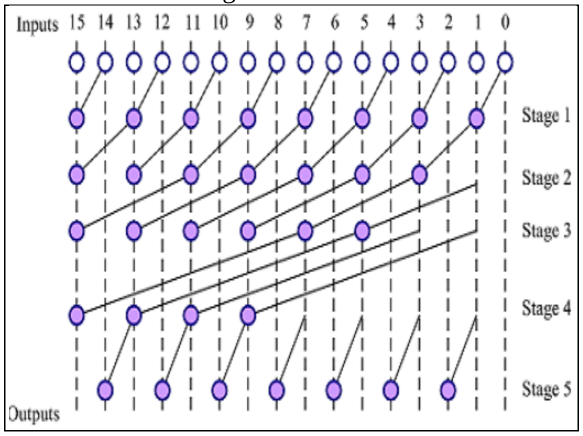


Figure 4: Han Carlson Adder[14]

The above are the commonly used topologies for the parallel prefix adders. Also there are many other ways by which the binary sum is computed. One such way is the QCA technique which is also the base for our current work.

C. Existing Technology - The existing technology deals with the computation of binary sum using a network of Full adders. They put forth a novel adder which computes sum of two BCD numbers. It employs the technique of "Quantum dot Cellular Automata"[17], where majority gates were used instead of normal logic gates. Majority gates perform the operation of AND and OR on the given bits such as M= a.b + b.c + c.a. "The generate and propagate terms are computed by using these Majority gates"[18]. By employing the concept of the majority gates in the Carry look ahead adders and other Ripple carry adder topologies the binary sum is computed here. Along with QCA technique, an innovative approach of "2D clocking technique"[19] is also implemented here. Here the total clock cycle is divided in to four parts namely Switch, Hold, Release and Relax. Accordingly, the operations are carried out simultaneously so that the delay can be avoided. The n digit binary adder is given below:

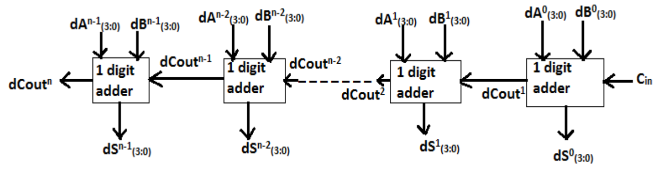
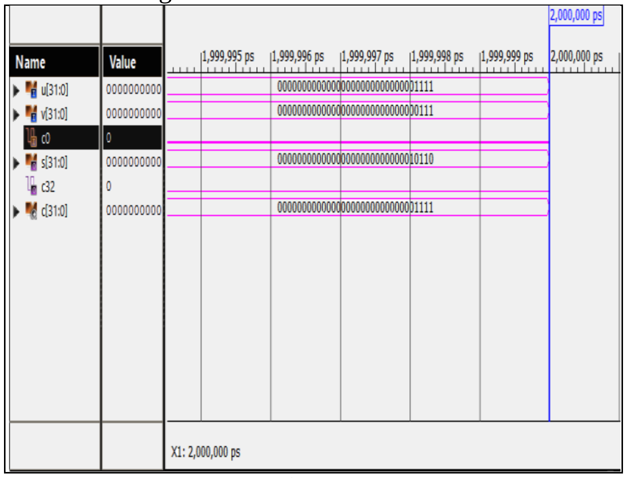


Figure 5: N digit BCD Adder

For obtaining N digit adder, N number of one digit adders are attached one after the other as shown the figure, where N represents the number of stages. Each one digit adder module receives two bit inputs in the form of $dA^{i}_{(3:0)}$and $dB^{i}_{(3:0)}$ and third input in the form of carry in and here i=0,1,2,....n-1. These inputs are furnished and two outputs dS and dCout are computed. The $dCout^{1}$ from the first adder has to travel through n-2 stages to be received by the final stage adder and thereby yield the final sum. In this case, the worst case delay T_w can be calculated as follows

$$T_w = T_g + (n-2)T_p + T_a \qquad (10)$$

where, T_w- Worst case delay for an N digit adder, T_g and T_p are the time required for the generation and propagation of dCout from one stage to another stage and T_a is the delay with which the most significant bit receives the carry in. If this N is taken as 32 (0-31), i.e., for 32-bit adder the delay obtained is approximately 43.142ns which is obtained after it's simulation in Xilinx. The total space consumption/memory usage for this 32-bit adder is about 248252 kilobytes. The simulation results of this adder are given below:



2,000,000 ps
Name
Value
1,999,995 ps
1,999,996 ps
1,999,997 ps
1,999,998 ps
1,999,999 ps
2,000,000 ps
u[31:0] 0000000000 00000000000000000000000000001111
v[31:0] 0000000000 00000000000000000000000000000111
c0 0
s[31:0] 0000000000 00000000000000000000000000010110
c32 0
c[31:0] 0000000000 00000000000000000000000000001111
X1: 2,000,000 ps

Figure 6: Simulation results for the Existing QCA based adder

This worst case delay can never be completely eliminated. But it can be reduced if changes are made at the logic level by implementing the parallel prefix technique where all the intermediate carry's are calculated simultaneously instead of waiting for the prior carrys to visit the stages. Here we propose a Novel BCD adder which is built using the PPA technique by implementing black and gray cells for reducing the delay to some extent.

IV. PROPOSED TECHNOLOGY

In the proposed technology, Parallel prefix technique is implemented in the QCA technique instead of a network of full adders and majority gates. It can be considered as modified version of a Brent Kung Adder. It lessens the complexity, the silicon area, delay and power consumption. As discussed prior, "a parallel prefix structure has three stages namely Pre-processing, Prefix network and Post-Processing stages"[20]. Another stage of error detection and correction can be added to this network for detecting and correcting the errors that may occur during the process of pre-computation of the carry(s). Thus the structure of the adder is as follows:

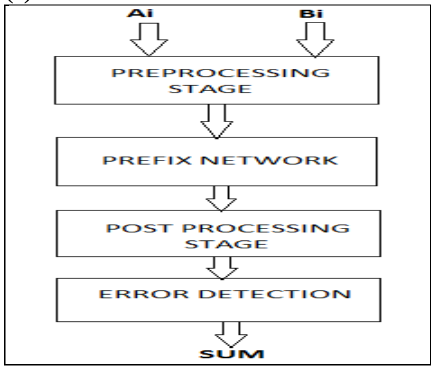
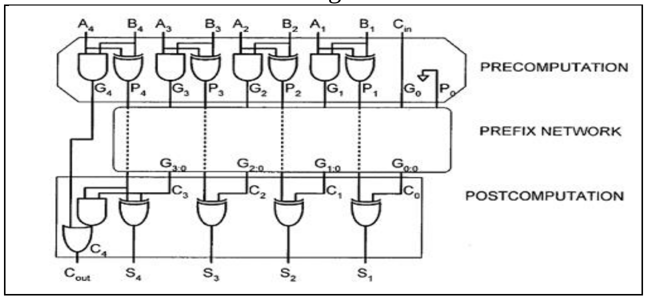


Figure.7: Block Diagram of Proposed PPA Adder

Being mentioned about the factor that Brent Kung adder operates effectively with less number of stages than the basic Kogge Stone adder and with much more logical depth, Brent Kung Adder is taken here and clear modifications are done to it to overcome the drawbacks of the traditional adder. The Brent Kung adder primarily computes prefixes for 2-bit groups. These 2-bit groups then generate the 4-bit prefixes, which in turn help in computing 8-bit prefixes and the process goes on. "These generated prefixes are then applied along with the group generate and group propagate terms to obtain the final sum and carry"[3]. The below figure shows the prefix tree structure and the internal logic:



A_4 B_4 A_3 B_3 A_2 B_2 A_1 B_1 C_{in}
G_4 P_4 G_3 P_3 G_2 P_2 G_1 P_1 G_0 P_0
PRECOMPUTATION
PREFIX NETWORK
$G_{3:0}$ $G_{2:0}$ $G_{1:0}$ $G_{0:0}$
C_3 C_2 C_1 C_0
POSTCOMPUTATION
C_4
C_{out} S_4 S_3 S_2 S_1

Figure 8: Anatomy of the Prefix adder[21]

Here is a representation of the proposed adder for 16-bit addition about how the carry bits are propagated through the stages and how an efficient path is brought out.

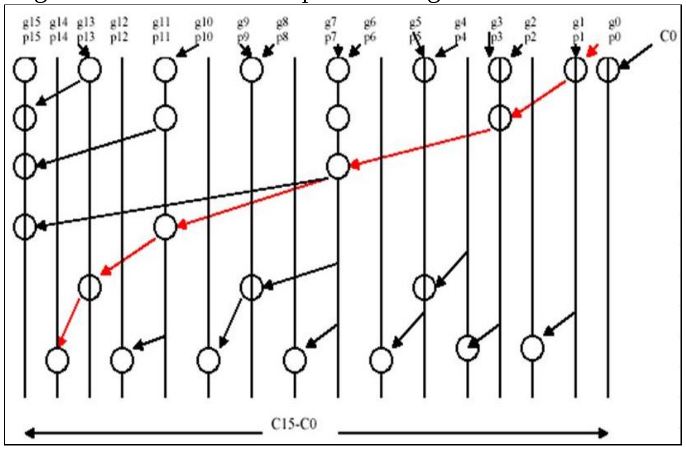


Figure 9: Proposed Brent Kung Adder

Pre-Computation Stage: In the first stage called as the Pre-computation stage/pre-processing stage, the individual generate and propagate signals of all the available input bits are calculated using the equations:

$$P_i = A_i \text{ XoR } B_i \qquad (11)$$
$$G_i = A_i \text{ AND } B_i \qquad (12)$$

where $0 \le i \le m-1$, here m represents the number of input bits.

Prefix Network Stage: In the second stage, the most crucial operation of carry merging is performed.As mentioned prior, the terms Propagate and Generate play a key role in the computation of the sum. As a part of this, group generate and group propagate signals are computed for general i^{th} bit and j^{th} bit using the following equations:

$$G_{i:j} = G_{i:k} + (G_{k-1:j}\, P_{i:k}) \qquad (13)$$
$$P_{i:j} = P_{i:k}\, P_{k-1:j} \qquad (14)$$

where, $i > k > j$

The group generate and propagate terms are furnished by using two operators called as "Black cell and the Gray cell"[22].

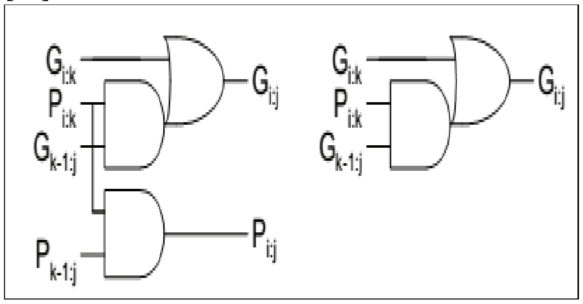


Figure10: Black and Gray Cell logics with in prefix network[23]

As mentioned prior, the majority gates and full adder tree structures of the existing adder structure are replaced the PG blocks that furnish the generate and propagate bits. The number of PG blocks depend up on the number of input bits. The figure above depicts the structure of Black and Gray cells. Black cell does complex logic operation and gives two outputs namely Group Propagate and Group Generate signals. On the other hand the Gray cells give only the Group Generate output. The outputs of pre-processing stage, i.e., the individual generate and propagate bits, are given as inputs to these blocks for the generation of group generate and group propagate terms. These two signals are then passed on to the post computing stage.

Post Computing Stage: In the last stage, i.e. Post processing/post computing stage, the final sum and carry are calculated with the aid of Ex-OR gates, AND gates and OR gates. The method of computation of sum and carry is given below:

$$SUM = P_i \text{ XOR } C_{i-1} \qquad (15)$$
$$C_{i-1} = (P_i \text{ and } C_{in}) \text{ or } G_i \qquad (16)$$

In the final stage, error detection and correction are done. As many circuits are operated simultaneously, errors may occur which may lead to inefficiency of the adder circuit. The main reasons for the occurrence of errors are the "delay and the clock cycle differences"[24] that arise when a particular operation is taking place. These errors are to be detected and corrected; otherwise the circuit will be declared inefficient. To serve this purpose, the network of "Razor Flip-flops"[25] are used."Razor flip-flop is used to determine the exact time period within which a specific operation can be accomplished"[26]. Also it checks whether the final sum obtained is in BCD format or not. The number of razor flip-flops required is judged by the number of input bits. A 1-bit razor flip-flop is shown below:

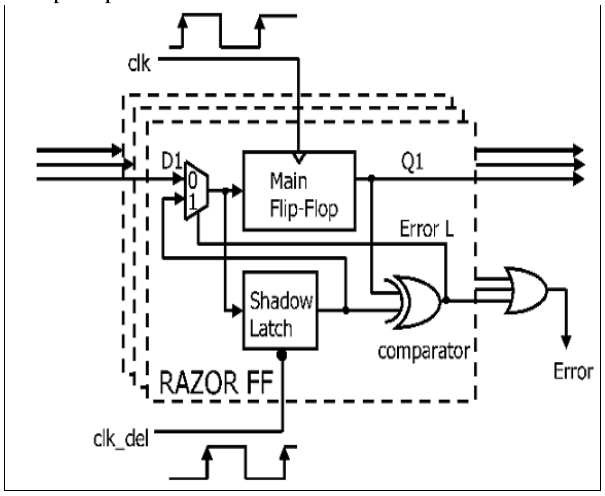
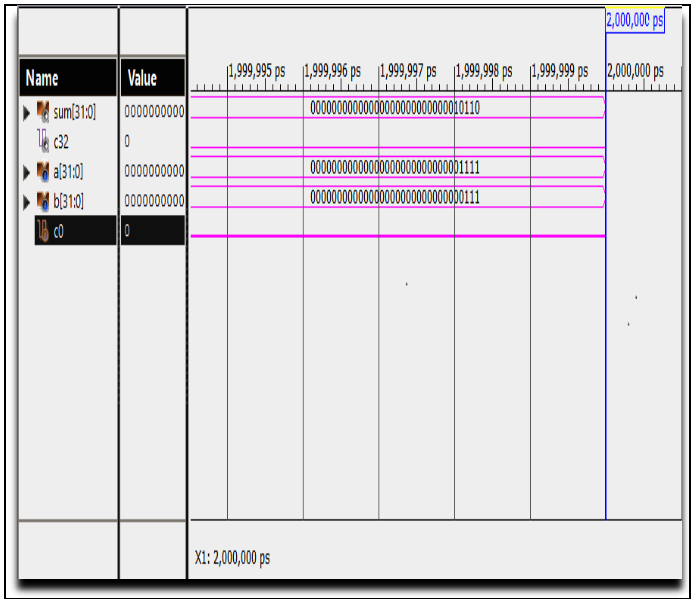


Figure11: 1 Bit Razor Flip-flop[25]

The razor flip-flop is built with a main flip-flop, a shadow latch, Ex-OR gate and a Multiplexer. The main flip-flop which works with the main clock cycle receives the output sum bits. On the other hand, the shadow latch works with delayed clock and it also receives the same output bits. The error presence can be observed when the outputs of the flip-flop and the latch differ. This drives us to a conclusion that the sum and carry generation process is still intact and it needs more clock cycles for it's completion. The shadow latch sets the error bit to 1 indicating that an error has popped up and it needs to be corrected. Even though it's a tedious job to correct the error, as the circuit performance cannot be compromised.

V. RESULTS

The simulation results of the proposed Novel Brent Kung Adder for the computation of Binary Sum and Carry are given below:



```
Name       Value        1,999,995 ps  1,999,996 ps  1,999,997 ps  1,999,998 ps  1,999,999 ps  2,000,000 ps
sum[31:0]  0000000000   00000000000000000000000000010110
c32        0
a[31:0]    0000000000   00000000000000000000000000001111
b[31:0]    0000000000   00000000000000000000000000000111
c0         0

X1: 2,000,000 ps
```

Figure12: Simulation Results of the Proposed Binary Adder

The overall operation of the proposed 32-bit adder can be accomplished with in 32.981ns, which is quite less than the existing adder's delay of 43.142ns. Also when attached by the error detection and correction, the efficiency gets built up for the circuit. One more important point here is the effective utilization of the available cells and LUT's. Even though the number of available 4-input LUT's available are same for both the adders, the utilization of these cells and LUTs effectively is what matters. The utilization of the four input LUTs and slices has been increased up to 4% and 5% respectively. The memory usage for this implementation is 248272 kilobytes which is very slightly more than the existing technology. The below table displays the comparison of the proposed adder with other adder structures.

Table.1: Comparison of Delay, utilization of slices and LUTs

Adder/Method	Delay in nano sec	Number of slices utilized	Number of LUTs utilized
Conventional KS Adder	38.487	45	82
Existing Adder	43.142	37	64
Proposed BK Adder	32.981	49	87

VI. CONCLUSION

The Novel BK adder proposed here operates effectively on inputs of large width. The decimal adder provides a higher logical depth which proves about the critical logic operations that take place inside. The delay of the existing QCA based BCD adder is 43.142ns which is reduced to 32.981ns in the suggested novel adder. This is possible because of the innovative logic of parallel prefixing implemented here. This shows that having such logical depth didn't prevent it from possessing less delay. The utilization of the available slices has increased from 37 to 49, and available LUTs has improved from 64 to 87 which proves the efficiency and superiority of the proposed adder over the existing counterpart. Even though the Kogge Stone adder and Han Carlson adder are efficient to some extent, they cannot be employed for inputs with more number of bits, as they are said to consume more space when the number of input bits increase. In addition to it, the power consumption also gets increased. Finally it can be concluded that the suggested adder has surpassed the precedent adder circuits in many aspects. This adder design can be further exploited with the help of upcoming technologies enriched with enhanced logics in such a way to provide much less area consumption. Improvements in logic with the help of innovative current day technologies may aid in providing less delay along with more efficiency.

VII. REFERENCES

[1]. Kenney, R.D. and Schulte, M.J., 2005. High-speed multioperand decimal adders. IEEE Transactions on Computers, 54(8), pp.953-963.

[2]. Shah, N.A., Khanday, F.A. and Bangi, Z.A., 2012.Quantum cellular automata based efficient BCD adder structure. Communications in Information Science and Management Engineering, 2(2).

[3]. Beaumont-Smith, A. and Lim, C.C., 2001. Parallel prefix adder design. In Computer Arithmetic, 2001. Proceedings. 15th IEEE Symposium on (pp. 218- 225). IEEE.

[4]. Sklansky, J., 1960. Conditional-sum addition logic. IRE Transactions on Electronic computers, (2), pp.226-231.

[5]. Kogge, P.M. and Stone, H.S., 1973. A parallel algorithm for the efficient solution of a general class ofrecurrence equations. IEEE transactions on computers, 100(8), pp.786-793.
[6]. Ladner, R.E. and Fischer, M.J., 1980. Parallel prefix computation. Journal of the ACM (JACM), 27(4), pp.831-838.
[7]. Brent, R.P. and Kung, H.T., 1982. A regular layout for parallel adders. IEEE transactions on Computers, (3), pp.260-264.
[8]. Radder, P. and Saptalakar, B.K., 2017. PERFORMANCE AND ANALYSIS OF SIGNED HAN-CARLSON ADDER USING VLSI. International Journal of Research in Management & Social Science, p.73.
[9]. Basha, M.M., Ramanaiah, K.V. and Reddy, P.R., Modified Reverse Converter Design With intervention of Efficacious Parallel Prefix Adders
[10]. Zimmermann, R., 1998. Binary adder architectures for cell-based VLSI and their synthesis. Hartung-Gorre.
[11]. Doran, R.W., 1988. Variants of an improved carry look-ahead adder. IEEE Transactions on Computers, 37(9), pp.1110-1113.
[12]. Oklobdzija, V.G., 2000. High-speed VLSI arithmetic units: adders and multipliers. Design of High-Performance Microprocessor Circuits. IEEE Press, Los Alamitos, 232.
[13]. Han, T. and Carlson, D.A., 1987, May. Fast area-efficient VLSI adders. In Computer Arithmetic (ARITH), 1987 IEEE 8th Symposium on (pp. 49-56). IEEE.
[14]. Daphni, S. and Grace, K.V., 2017, December. A review analysis of parallel prefix adders for better performnce in VLSI applications. In Circuits and Systems (ICCS), 2017 IEEE International Conference on (pp. 103-106). IEEE.
[15]. R. P. Brent and H. T. Kung, "A regular layout for parallel adders", IEEE trans, computers, Vol.C-31,pp. 260-264,.March 1982.
[16]. Prakash, P. and Saxena, A.K., 2009, October. Design of Low Power High Speed ALU Using Feedback Switch Logic. In Advances in Recent Technologies in Communication and Computing, 2009. ARTCom'09. International Conference on(pp. 899-902). IEEE.
[17]. Snider, G.L., Orlov, A.O., Kummamuru, R.K., Ramasubramaniam, R., Amlani, I., Bernstein, G.H. and Lent, C.S., 2001. Quantum-dot cellular automata. MRS Online Proceedings Library Archive, 696.
[18]. Pudi, V. and Sridharan, K., 2011. Efficient design of a hybrid adder in quantum-dot cellular automata. IEEE Trans. VLSI Syst., 19(9), pp.1535- 1548.
[19]. Vankamamidi, V., Ottavi, M. and Lombardi, F., 2006, June. Clocking and cell placement for QCA. In Nanotechnology, 2006. IEEE-NANO 2006. Sixth IEEE Conference on (Vol. 1, pp. 343-346). IEEE.
[20]. Kunz, H. and Zimmermann, R., 1996. High-performance adder circuit generators in parameterized structural VHDL. ETH.
[21]. Rani, G. and Kumar, S., 2014. Delay analysis of parallel-prefix adders. International Journal of Science and Research, 3(6), pp.2339-2342.
[22]. Harris, D. and Sutherland, I., 2003, November. Logical effort of carry propagate adders. In Signals, Systems and Computers, 2004. Conference Record of the Thirty-Seventh Asilomar Conference on (Vol. 1, pp. 873-878). IEEE.
[23]. Dave, V., Oruklu, E. and Saniie, J., 2006, May. Performance evaluation of flagged prefix adders for constant addition. In Proc. IEEE Int. Conf. Electro/inf. Technol (pp. 415-420).
[24]. Cho, H. and Swartzlander Jr, E.E., 2009. Adder and multiplier design in quantum-dot cellular automata. IEEE Transactions on Computers, 58(6), pp.721-727.
[25]. Ernst, D., Kim, N.S., Das, S., Pant, S., Rao, R., Pham, T., Ziesler, C., Blaauw, D., Austin, T., Flautner, K. and Mudge, T., 2003, December. Razor: A low-power pipeline based on circuit-level timing speculation. In Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture (p. 7). IEEE Computer Society.
[26]. Ernst, D., Das, S., Lee, S., Blaauw, D., Austin, T., Mudge, T., Kim, N.S. and Flautner, K., 2004. Razor: circuit-level correction of timing errors for low- power operation. IEEE Micro, 24(6), pp.10-20.