International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-12 Issue-5, June 2023

A Novel High Computing Power Efficient VLSI Architectures of Three Operand Binary Adders

Shravan Chintam, Sai Krishna Marri, Saidulu Rapolu, Tejasvey Panchareddy, Sai Radha Krishan G

***Abstract:** Directly or indirectly adders are the basic elements in almost all digital circuits, three operand adders are the basic building blocks in LCG (Linear congruential generator) based pseudo-random bit generators. Elementary adders are fast, area and power efficient for small bit sizes. Carry save adder computes the addition in O(n) time complexity, due to its ripple carry stage. Parallel prefix adders such as Han-Carlson compute the addition in O(log(n)) time complexity but at the cost of additional circuitry. Hence new high-speed power-efficient adder architecture is proposed which uses four stages to compute the addition, which consumes less power, and the adder delay decreases to O(n/2). Even though it is not much faster than the High-speed Area efficient VLSI architecture of three operand adders (HSAT3), it computes the addition by utilizing less power. The proposed architecture is implemented using Verilog HDL in Xilinx 14.7 design environment and it is evident that this adder architecture is 2 times faster than the carry save adder and 1, 1.5, 1.75 times faster than the hybrid adder structure for 32, 64, 128 bits respectively. Also, power utilization is 1.95 times lesser than HSAT3, 1.94 times lesser than the Han-Carlson adder, and achieves the lowest PDP than the existing three operand techniques*

***Keywords:** Parallel Prefix Adders, Three Operand Binary Adders.*

I. INTRODUCTION

An adder or summer is a digital circuit that performs the addition of two or more numbers. Today the world is depending upon IOT devices, so it is necessary to provide security to each device which can be done by cryptographic and pseudo-random bit generator (PRBG) methods, in PRBG three operand binary Adders are the fundamental blocks so, efficient implementation of the adder is required and maintaining the trade-off between the delay, area, and power is also important.

In this paper, a new three-operand adder structure is proposed and is verified and simulated using Verilog HDL in Xilinx 14.7 design environment, when it is compared with the fundamental adders it performs well. Comparing the proposed architecture with the existing architectures in terms of area (in terms of LUTs), power (in terms of Watt) and Delay (in terms of the ns) is the main goal of this paper, all three parameters computed and analyzed with the Verilog HDL in Xilinx tool, and power is computed using power analyzer. The rest of this paper is organized as follows section-II deals with the literature overview, and section-III describes the various adder Architectures such as carry save adder, Han-Carlson adder, three operand adder, and hybrid adder, section-IV highlights the proposed adder VLSI architecture and synthesis results of the proposed adder along with other adders also reported in this section, finally section-V concludes the paper.

II. LITERATURE OVERVIEW

Any VLSI architecture or logic design becomes popular and successful when it meets the optimal performance of three constraints such as Area, Power, and Delay. But at a time dealing with all three constraints is difficult. As time passed many adder logics were proposed because adder is the basic building block in any digital circuit. Among all the adder logic no adder logic reached the optimal power, area, and delay at the same time. People will use a variety of adders depending on the application and purpose, one may require less power and the other may require less delay, and so on. We started our research with basic adder structures like Ripple Carry Adder, Carry save adder, carry Look ahead adder, carry skip adder and Carry select adder. Ripple carry adder (RCA)is the simplest adder, and it is very slow due to the propagation of the carry from one full adder block to another full adder block, the delay increases as the number of bits increases, it computes the addition of only two operands at a time, for the addition of three operands two stages of RCA is used, speed of the RCA depends on the carry propagation time. Carry-Save Adder (CSA) [1] is the most generally used three operand binary adder, in carry save adder as the number of bits increases the delay also increases, hence time complexity for computation of addition is O(n), it contains the two stages of array of full adder blocks, the first stage computes the sum and carry bits simultaneously at a time and the second stage resembles the ripple carry stage.

Manuscript received on 20 May 2023 | Revised Manuscript received on 31 May 2023 | Manuscript Accepted on 15 June 2023 | Manuscript published on 30 June 2023.
*Correspondence Author(s)
Chintam Shravan*, Department of ECE, RGUKT-Basar, Nirmal Telangana, India. Email: shravanchintam@gmail.com, ORCID ID: 0009-0005-5477-5084
Sai Krishna Marri, Department of ECE, RGUKT-Basar, Nirmal, Telangana, India. Email: saikrishnamarri@gmail.com, ORCID ID: 0009-0002-0969-1381
Rapolu Saidulu, Department of ECE, RGUKT-Basar, Nirmal, Telangana, India. Email: saidulurapolu@gmail.com, ORCID ID: 0009-0005-2926-475X
Panchareddy Tejasvey, Department of ECE, RGUKT-Basar, Nirmal Telangana, India. Email: panchareddyteja@gmail.com, ORCID ID: 0009-0007-0824-7867
Sai Radha Krishan G, Department of ECE, RGUKT-Basar, Nirmal Telangana, India. Email: radhakrishna9357@gmail.com, ORCID ID: 0000-0001-5195-4718

Carry Look Ahead (CLA) [2] is the most widely used adder to overcome the carry propagation problem, in CLA carry is computed in advance based on the input signals, carry is generated in two cases, one is when both the inputs are high and other is the XOR of inputs is high along with carry in, the circuitry will be more complex if the number of bits is more. Carry Skip Adder (CSKA) [3] is also known as a carry-bypass adder, it is used to improve the delay of the ripple carry adder it consists of RCA with a special speed up carry chain, the main idea here is, if all propagate bits are one the carry out will be equal to the carry of the first full adder block. Carry Select Adder (CSTA) [4] consists of two RCA blocks one block is fed with carry in zero and other block is fed with carry in one, thus both blocks can compute parallelly, multiplexers are used to select the correct one of both precalculated partial sums when actual carry in arrives to the block, resulting carry out is selected and propagated to the next carry select stage. And then we moved to parallel prefix Topologies [5], [6], [7], [8] such as Kogge-Stone Adder [9], Brent Kung Adder [10], Ladner Fischer Adder [11], Han-Carlson adder [12], Klinsky adder, and Knowles. These parallel prefix adders compute the addition in three stages

1. Preprocessing stage
2. carry computation stage
3. post processing stage

In preprocessing stage generate and propagate bits were calculated and in post processing stage sum bits are calculated. Let us understand the difference between linear and parallel prefix adders. Let **a, b, c,** and d be the numbers that are to be added. In basic adders first **(a + b)** is computed then the result is added to the next i.e., **((a + b) + c),** then **(((a + b) + c) + d)** is calculated, in this case, we can understand that computation requires more time, but whereas in parallel prefix structure **(a + b)** and **(c + d)** is computed at a time, and next individual results are added, just assume the same thing in bits level. From all this, one can understand that a major problem raising due to the propagation of the carry between stages. To transfer the carry from one stage to the next stage, different topologies were proposed and they are Kogge-Stone, Brent Kung, Ladner Fischer, and Han-Carlson. Among these Han -Carlson is fast in terms of speed of computation. Among the kogge-stone, brent kung, Sklansky, Han-Carlson, Knowles, and Ladner Fischer, the fundamental prefix adders are Kogge–stone, brent kung, and Sklansky. Brent kung adder uses a smaller number of logic gates i.e., nodes in the tree structure of carry computation stage which will results in the less area but depth of the circuitry will be more which leads to the slight increase in latency that implies calculation time is longer. Sklansky has the minimum depth (reduces delay) but at cost of high fanout nodes. Kogge–stone adder's tree structure of the carry computation stage has low depth and high node count (logic gates) which implies more area and complex circuitry means more wiring is needed and has the minimal fanout of 1 at each node which implies it achieves the better speed. Ladner Fischer has low depth and high fan out nodes Knowles are bound by the Lander Fischer and Brent-kung i.e., minimum depth and minimum fanout topologies, T. Han and D.A Carlson proposed a hybrid prefix adder which is derived from both kogge-stone and Brent-kung by choosing the best features of the low area from Brent-kung and best features of high speed from the Kogge-stone. Other topologies that resemble Ultra-Fast Adders are available [13], but they require more space and power to operate. Several writers have also covered the hybrid structures [14]. A hybrid adder is a mixture of two or more prefix adders; if the combination is of the same kind, it is referred to as a homogeneous hybrid adder otherwise it is heterogeneous hybrid adder. For example, our interest is to compute the sum of two 32-bit operands in hybrid parallel prefix adder the sum of the bits is computed by using Han-Carlson and next 16 bit by brent kung or first 8 bits by some adder next following bits by some other adders, hybrid adders are little slow because one adder has to wait for the carry of before adder block, but area consumption will be low if we use good combination, we can expect improvement in the speed. Whatever we have discussed till now they are two operand adders. A three-operand architecture is proposed in [15], it computes the addition in four stages, unlike three stages in the parallel prefix adders.

1. bit addition Stage
2. Base addition Stage
3. Propagate and generate
4. Sum logic

Today's world is adopting more IOT devices, and IOT applications require security it can be done by using a stream cipher. A stream cipher is an encryption technique that works byte by byte to transform plain text into code that is unreadable to anyone without the proper key pseudo random bit generator (PRBG) is primary in a stream cipher. LCG based PRBG methods are efficient among the existing PRBG methods and these are also suitable for the stream cipher LCG, MDCLCG is most random if the bit size is more than equal to 32. The linear congruential generator consists of three operand modulo 2 m adder, hence the delay, area, and power of the three operand adder will affect the performance of the LCG [16], among all the LCG based pseudo random bit generation algorithms MDCLCG [17] is the most secure one, it consists of two CLCG's(coupled linear congruential generator) and each block of CLCG [17] consist of two LCGs, therefore, they are four LCGs in the MDCLCG, let $\mathbf{x_1}$**,** $\mathbf{y_1}$**, and** $\mathbf{x_2}$**,** $\mathbf{y_2}$ are the outputs of the each LCG. if $\mathbf{x_1} > \mathbf{y_1}$ then $\mathbf{z_1}$=$\mathbf{x_1}$ otherwise $\mathbf{z_1}$=$\mathbf{y_1}$ and if $\mathbf{x_2} > \mathbf{y_2}$ then $\mathbf{z_2} = \mathbf{x_2}$ otherwise $\mathbf{z_2}$=$\mathbf{y_2}$, the exor operation of the $\mathbf{z_1}$**,** $\mathbf{z_2}$ gives the output of the MDCLCG hence the output is unpredictable and it will be mostly random.

III. VARIOUS ADDER STRUCTURES

3.1 Carry Save Adder

More generally used adder structures for computation of three operand addition is carry save adder (CS3A) computes the addition in two stages, the first stage consists of an array of full adders, to each full adder block one bit from first operand, one bit from second operand and one bit from third operand is given as inputs, and each full adder block in first stage results in one sum bit and one carry bit, with propagation delay equivalent to delay of one full adder block. Parallelly all full adder blocks give their output. And second stage resembles the ripple carry adder that means by using the first stage we reduced the three operands to two. Results of the first stage are given as input to the second stage i.e., RCA stage. The basic idea is to convert the three operands into two operands, for example, a, b, and c are the three numbers **a = 1469, b = 1470, c = 1471** respectively and a + b + c can be written as sum + carry when the addition is computed by hand the numbers will be aligned one below the other, and the sum of columns is computed and if overflow occurs it will be transferred to the next columns, here columns sum is stored in the sum ′ and instead of transferring the overflow to the next stage, it will be stored in the carry', for the getting the overall sum, the carry ′ is shifted by one bit to left hand side and addition of sum ′ and shifted carry ′ results in the actual sum and carry. **Sum′ = A ⊕ B ⊕ C, Carry′ = (A & B) ‖ (B & C) ‖ (C & A)**

A = 1 4 6 9
B = 1 4 7 0
C = 1 4 7 1
Sum′ = 3 2 0 0
Cy′ = 1 2 1 0
Sum = 4 4 1 0
Carry = 0

The circuit which is implemented in below Fig.1

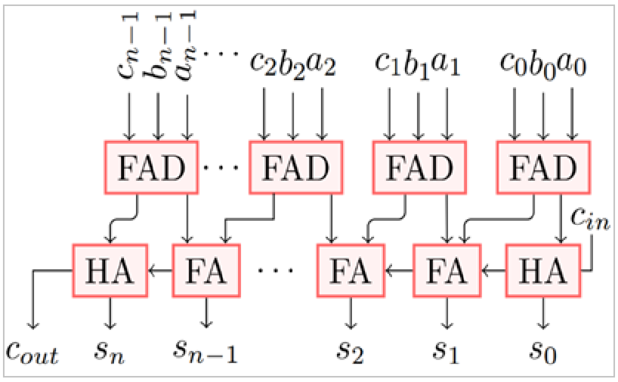

Fig.1 Carry Save Adder

3.2 Han-Carlson Adder

To compute the addition in less time, instead of linear adders we need to use parallel prefix adders, parallel prefix topologies will reduce the critical path delay. kogge-stone adder and brent-kung are the fundamental parallel prefix adder, it is the hybrid structure that uses the best features of less area requirement from the brent-kung and best features of high speed from the kogge-stone adder. As like every parallel prefix adder it also computes the addition in three stages, for the computation of the addition of three operands two blocks of Han-Carlson are required, the first block will take the first two operands and the result of the first along with the third operand is given to the second Han-Carlson block. The first stage of the Han-Carlson adder contains the array of half adders, which will compute sum bits and carry bits, next stage is the carry computation stage and which contains the gray and block cells, area, and delay depends on the number of gray and black cells, the last stage is post processing stage in this stage XOR operation of computed carry bits with propagate bits which resulted from the first stage will result in the sum bits.

Logic expression for each stage is given below:
Preprocessing stage:

Sum′ = a ⊕b
Carry′ = a & b
P=Sum′ G = Carry′

Gray cell:

G′ i = Gi + Pi * Gi − 1

Black cell:

G′ i = Gi + Pi * Gi − 1
P ′ i = Pi & Pi − 1

Post processing stage:

Sum = Pi ⊕G′ i

The circuit that will carry out the foregoing logic is depicted in Fig. 2:

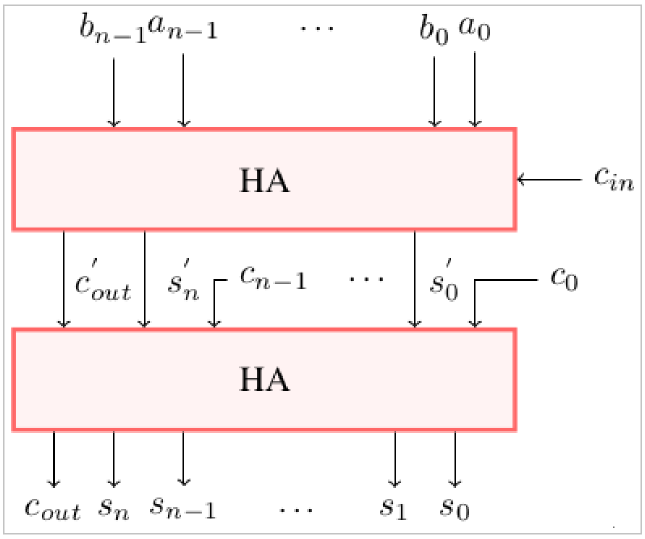

Fig.2 Han-Carlson Adder

3.3 Three Operand Binary Adders

Three Operand Adder also a parallel prefix adder unlike three stages in the parallel prefix adder it consists four stages 1. Bit addition logic 2. Base logic 3. Generate and propagate logic 4. Sum logic. Bit addition logic consists array of full adder blocks each full adder block computes sum and carry bits simultaneously, sum and carry bits of first stage are given to second stage i.e., base logic, from second it resembles the two-operand adder, that implies by using bit addition logic three operands converted to two operands. Base logic contains array of half adders which will computes sum and carry bits, i.e., propagate and generate bits (**p_i** and **g_i**), and third stage contains gray and black cells, and last stage exor operation of computed carry bits with propagate bits which resulted from the base logic stage will results in the sum bits.
Logic expression for each stage is given below:

Bit addition logic:

Sum′ i = ai ⊕bi ⊕ci
Carry′ i = (ai & bi) ‖ (bi & ci) ‖ (ci & ai)

Base logic:

Pi = sum′ i ⊕carry ′ i−1
Gi = sum′ i & carry′ i−1

Propagate and generate logic:

Black cell:

G′ i = Gi + Pi * Gi − 1
P′ i = Pi & Pi − 1

Retrieval Number:100.1/ijeat.E41880612523
DOI: 10.35940/ijeat.E4188.0612523
Journal Website: www.ijeat.org

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

Gray cell:

G′ i = Gi + Pi * Gi − 1

Sum logic:

Sum i = Pi ⊕G′

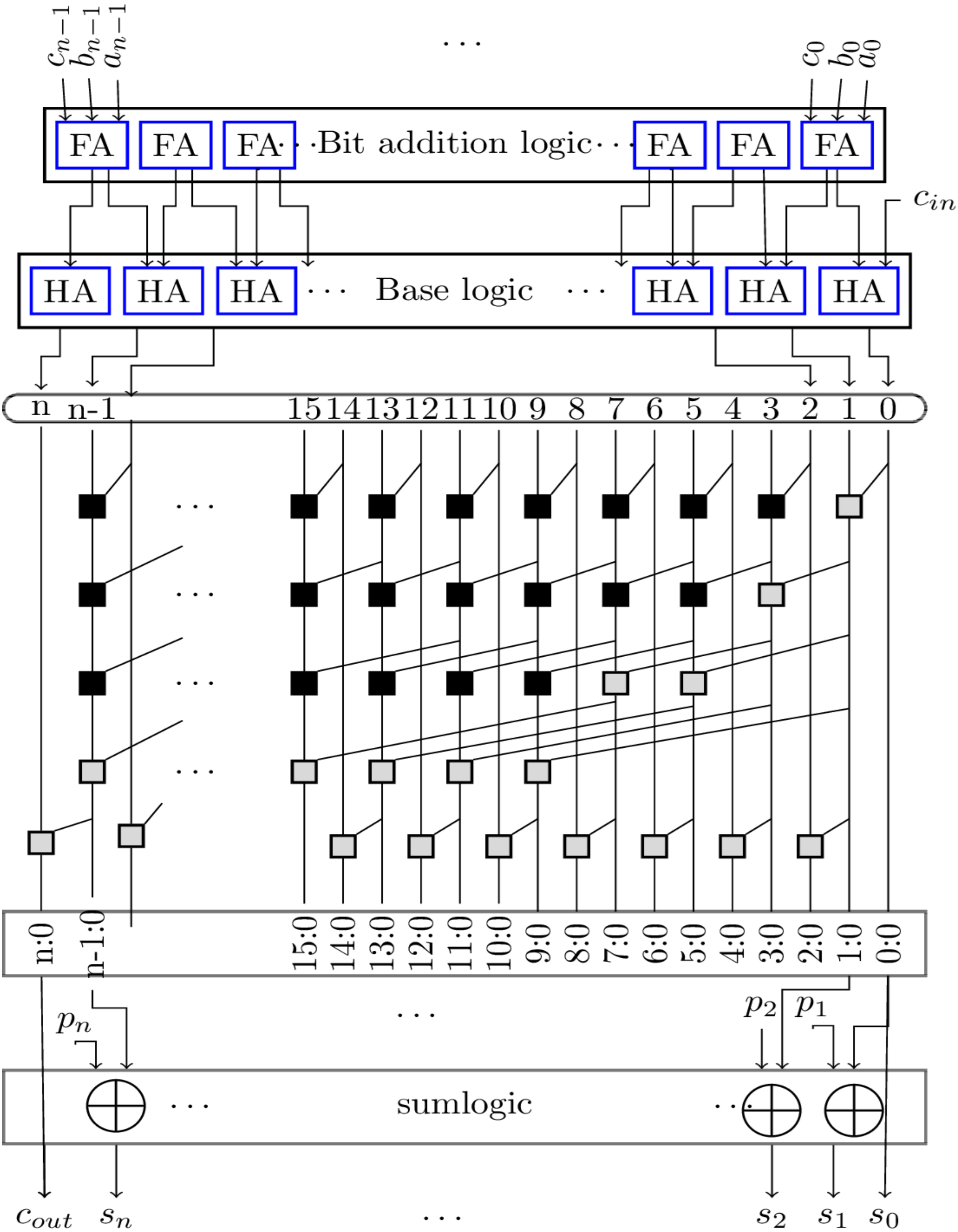

Fig.3 Three Operand Adder

Retrieval Number:100.1/ijeat.E41880612523
DOI: 10.35940/ijeat.E4188.0612523
Journal Website: www.ijeat.org

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

3.3.1 Hybrid Adder

In the carry computation stage i.e., in propagate and generate logic stage Han-Carlson carry computation tree is replaced with the combination of the four 8-bit Han-Carlson adder structures (for 32-operands), the first 8 bits of base logic are given to first 8-bit Han-Carlson block, next 8-bits to the second 8-bit Han-Carlson adder, and next 16 bit to another two Han-Carlson blocks. Carry of the first Han-Carlson is given to second Han-Carlson and the second to third and third to the fourth, this results in an increase in delay but the area will get reduced. for computation of 16-bit addition using Han-Carlson requires 17 black cells and 15 gray cells whereas in hybrid structure 10 black cells and 16 gray cells are enough.

3.3.2 Proposed Adder

It is also a parallel prefix adder; it computes the three-operand addition in four stages

1. bit addition logic
2. base logic
3. propagate and generate logic
4. sum logic

In a bit addition logic array of full adder blocks are present, each block will result in a sum bit and carry bit, along with the carry in this sum and carry bits given to the base logic. Base logic contains the array of half adder blocks, each half adder block resulting in sum and carry bits, this sum and carry bits are known as propagate and generate bits, and these propagate and generate bits are given to the propagate and generate stage. Propagate and generate logic contains black and grey cells, grey cells are used to generate the carry whereas black is used to both propagate and generate. The first stage of the propagate and generate logic will contain one grey cell and n-1 block cells, the second stage will contain two grey cells and (n/2-2) block cells, the third stage will contain four grey cells and (n/2-4) black cells, the fourth stage will contain eight grey cells and (n/2-8) black cells and so on the number of stages in the propagate and generate logic will depend on the bit size n, and last stage contains exactly n/2 grey cells. The final stage in the computation of addition is sum logic, in this logic XOR operation of propagate bits generated in the base logic and carry generated from the propagate and generate logic will result in the final sum. The critical path delay of this adder is less than the Carry Save Adder and the time complexity for computation of addition is O(n/2). Logic expression for each stage is given below:

Bit addition logic:

$Sum'i = ai \oplus bi \oplus ci$

$Carry'i = (ai \,\&\, bi) \,||\, (bi \,\&\, ci) \,||\, (ci \,\&\, ai)$

Base logic:

$Pi = sum'i \oplus carry'i{-}1$

$Gi = sum'i \,\&\, carry'i{-}1$

Propagate and generate logic:

Black cell:

$G\,i = G\,i + P\,i * G\,i{-}1$

$P\,i = P\,i \,\&\, P\,i{-}1$

Gray cell:

$G\,i = G\,i + P\,i * G\,i{-}1$

Sum logic:

$Sum\,i = P\,i \oplus G\,i$

The circuit that will execute the preceding logic is depicted in Fig. 4:

Retrieval Number:100.1/ijeat.E41880612523
DOI: 10.35940/ijeat.E4188.0612523
Journal Website: www.ijeat.org

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

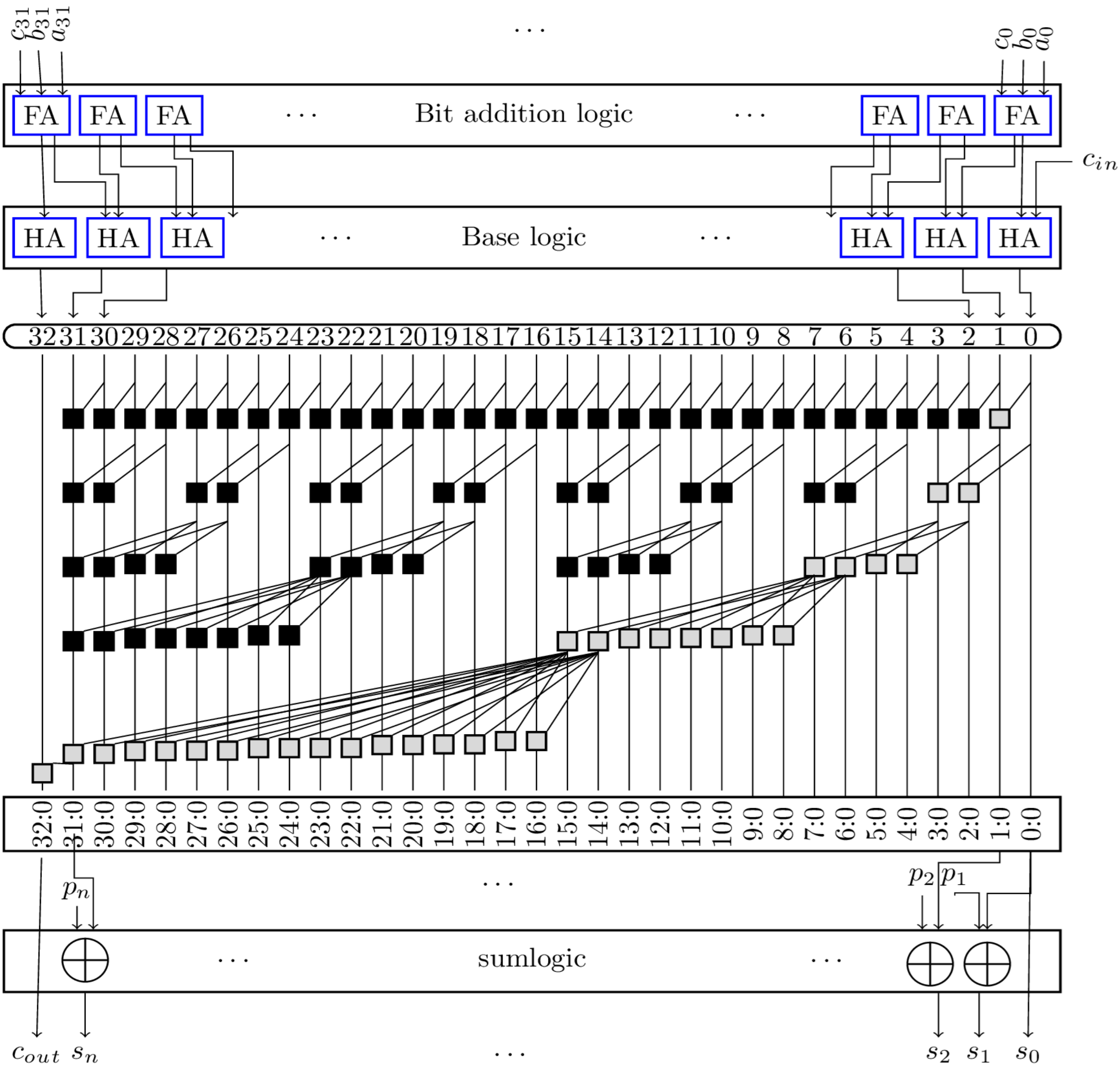

Fig.4 Proposed Adder

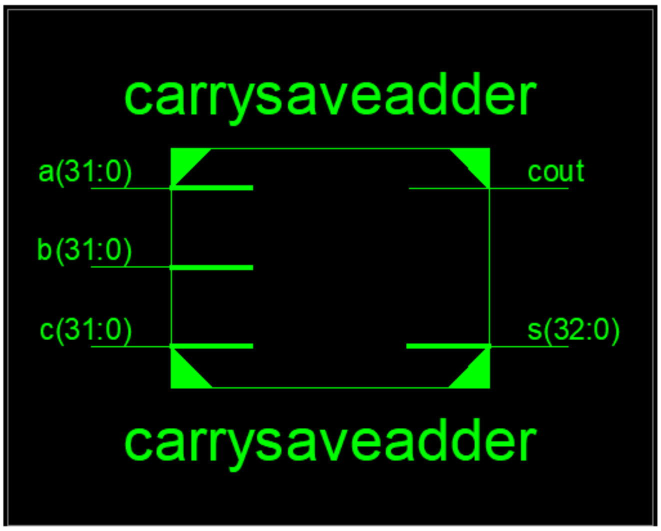

Fig.5 Carry Save Adder Module

Retrieval Number:100.1/ijeat.E41880612523
DOI: [10.35940/ijeat.E4188.0612523](https://doi.org/10.35940/ijeat.E4188.0612523)
Journal Website: www.ijeat.org

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

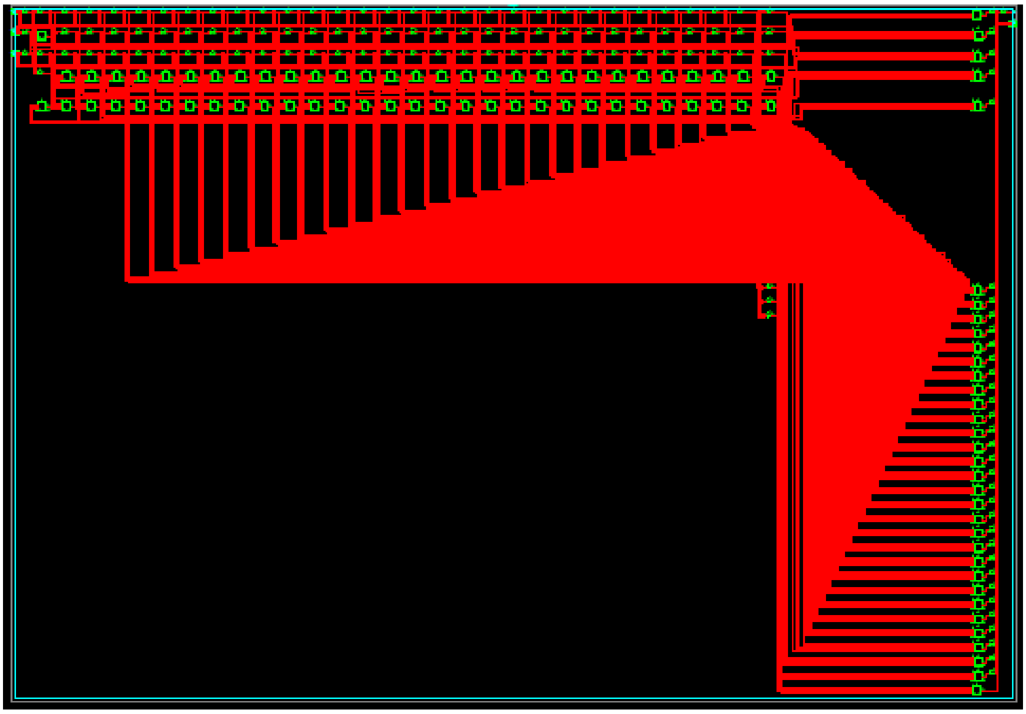

Figure.6 Carry Save Adder RTL Schematic

IV. RESULT AND DISCUSSION

Carry Save Adder: All the values are measured using Xilinx 14.7 ISE, Proposed and Carry Save Adder, and Han-Carlson adder, hybrid adder and existing three operand binary adder are implemented using Verilog HDL in Xilinx 14.7 ISE.

The simulation results of Carry Save Adder depicted in below figure.

Power Supply Summary			
	Total	Dynamic	Static Power
Supply Power (mW)	39.35	3.31	36.04

Fig.7 Carry Save Adder Power

On-Chip Power Summary				
On-Chip	Power (mW)	Used	Available	Utilization (%)
Clocks	0.00	0	---	---
Logic	0.01	62	27288	0
Signals	0.00	190	---	---
IOs	3.30	130	358	36
Static Power	36.04			
Total	39.35			

Fig.8 Carry Save Adder On-Chip Power

```
LUT5:I0->O          2   0.203   0.961  f216/carry1 (c2<18>)
LUT5:I0->O          2   0.203   0.961  f217/carry1 (c2<19>)
LUT5:I0->O          2   0.203   0.961  f218/carry1 (c2<20>)
LUT5:I0->O          2   0.203   0.961  f219/carry1 (c2<21>)
LUT5:I0->O          2   0.203   0.961  f220/carry1 (c2<22>)
LUT5:I0->O          2   0.203   0.961  f221/carry1 (c2<23>)
LUT5:I0->O          2   0.203   0.961  f222/carry1 (c2<24>)
LUT5:I0->O          2   0.203   0.961  f223/carry1 (c2<25>)
LUT5:I0->O          2   0.203   0.961  f224/carry1 (c2<26>)
LUT5:I0->O          2   0.203   0.961  f225/carry1 (c2<27>)
LUT5:I0->O          2   0.203   0.961  f226/carry1 (c2<28>)
LUT5:I0->O          2   0.203   0.961  f227/carry1 (c2<29>)
LUT5:I0->O          3   0.203   0.995  f228/carry1 (c2<30>)
LUT5:I0->O          1   0.203   0.579  f229/Mxor_sum_xo<0>1 (s_31_OBUF)
OBUF:I->O               2.571          s_31_OBUF (s<31>)
----------------------------------------
Total                     40.418ns (10.088ns logic, 30.330ns route)
                                   (25.0% logic, 75.0% route)
```

Fig.9 Carry Save Adder Delay

```
Slice Logic Utilization:
 Number of Slice LUTs:                    94  out of  27288     0%
    Number used as Logic:                 94  out of  27288     0%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:      94
   Number with an unused Flip Flop:       94  out of     94   100%
   Number with an unused LUT:              0  out of     94     0%
   Number of fully used LUT-FF pairs:      0  out of     94     0%
   Number of unique control sets:          0

IO Utilization:
 Number of IOs:                          130
 Number of bonded IOBs:                  130  out of    358    36%
```

Fig.10 Carry Save Adder Area

Han-Carlson Adder: The simulation results of Han-Carlson Adder depicted in below fig.11.

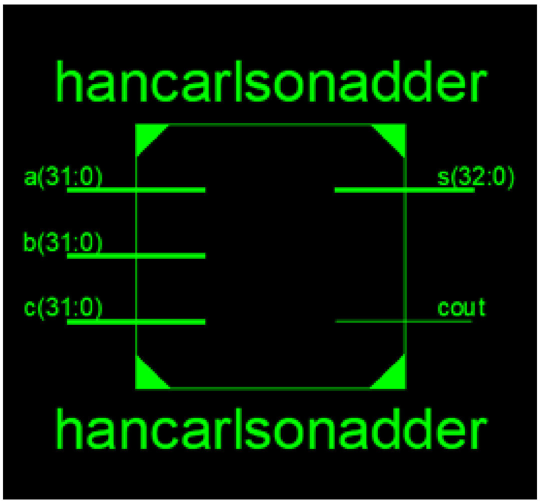

Fig.11 Han-Carlson Adder Module

Retrieval Number:100.1/ijeat.E41880612523
DOI: 10.35940/ijeat.E4188.0612523
Journal Website: www.ijeat.org

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

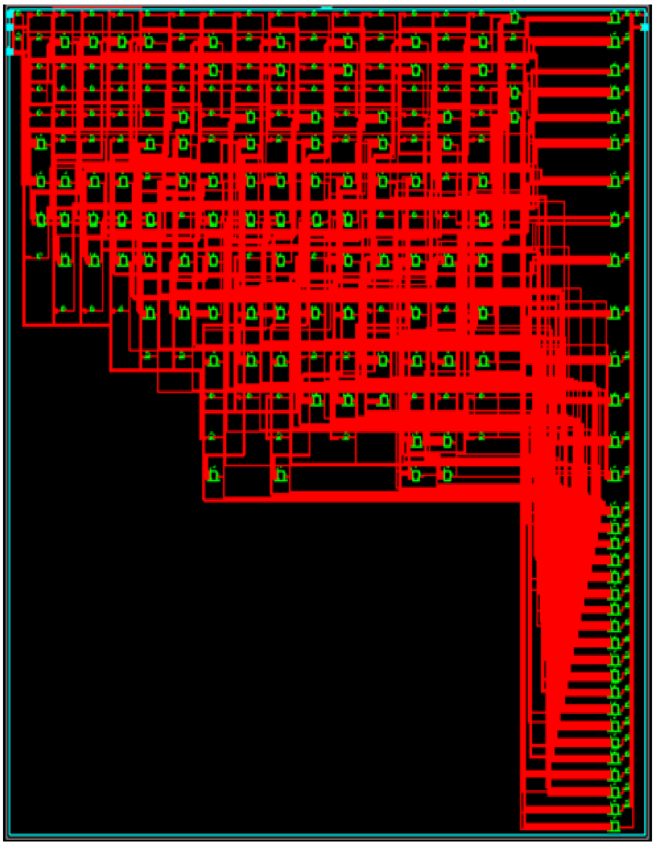

Fig.12 Han-Carlson Adder RTL Schematic

On-Chip	Power (mW)	Used	Available	Utilization (%)
Clocks	0.00	0	---	---
Logic	0.01	77	63400	0
Signals	0.00	220	---	---
IOs	1.10	130	210	62
Static Power	82.16			
Total	83.27			

Fig.13 Han-Carlson Adder On-Chip Power Summary

	Total	Dynamic	Static Power
Supply Power (mW)	83.27	1.11	82.16

Fig.14 Han-Carlson Adder Power

LUT5:I0->O	2	0.203	0.961	b0/e1 (g2<0>)
LUT5:I0->O	5	0.203	1.079	gr2/d1 (g2<1>)
LUT6:I0->O	1	0.203	0.000	gr5/d_G (N26)
MUXF7:I1->O	3	0.140	0.995	gr5/d (g3<2>)
LUT5:I0->O	2	0.203	0.845	gr9/d3 (g4<2>)
LUT5:I2->O	2	0.205	0.845	gr11/d1 (g4<4>)
LUT5:I2->O	3	0.205	0.898	gr13/d1 (g4<6>)
LUT6:I2->O	2	0.203	0.721	gr18/d2 (gr18/d1)
LUT5:I3->O	1	0.203	0.580	gr22/d1_SW3 (N19)
LUT6:I5->O	2	0.205	0.981	gr22/d1 (gr22/d1)
LUT6:I0->O	3	0.203	0.651	gr22/d (g5<6>)
LUT5:I4->O	1	0.205	0.580	gr26/d3_SW0 (N21)
LUT6:I5->O	2	0.205	0.617	gr26/d3 (gr26/d2)
LUT5:I4->O	2	0.205	0.845	gr26/d4 (g5<10>)
LUT6:I3->O	1	0.205	0.580	gr30/d_SW1 (N23)
LUT6:I5->O	1	0.205	0.580	gr30/d (g5<14>)
LUT5:I4->O	1	0.205	0.579	s30/Mxor_c_xo<0>1 (sum_31_OBUF)
OBUF:I->O		2.571		sum_31_OBUF (sum<31>)
Total		21.583ns (7.404ns logic, 14.179ns route) (34.3% logic, 65.7% route)		

Fig.15 Han-Carlson Delay

```
Slice Logic Utilization:
 Number of Slice LUTs:                 124  out of  63400    0%
    Number used as Logic:              124  out of  63400    0%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:   124
   Number with an unused Flip Flop:    124  out of    124  100%
   Number with an unused LUT:            0  out of    124    0%
   Number of fully used LUT-FF pairs:    0  out of    124    0%
   Number of unique control sets:        0

IO Utilization:
 Number of IOs:                        130
 Number of bonded IOBs:                130  out of    210   61%
```

Fig.16 Han-Carlson Adder Area

Three Operand Binary Adder: The simulation results of Three Operand Binary Adder depicted in below fig.17.

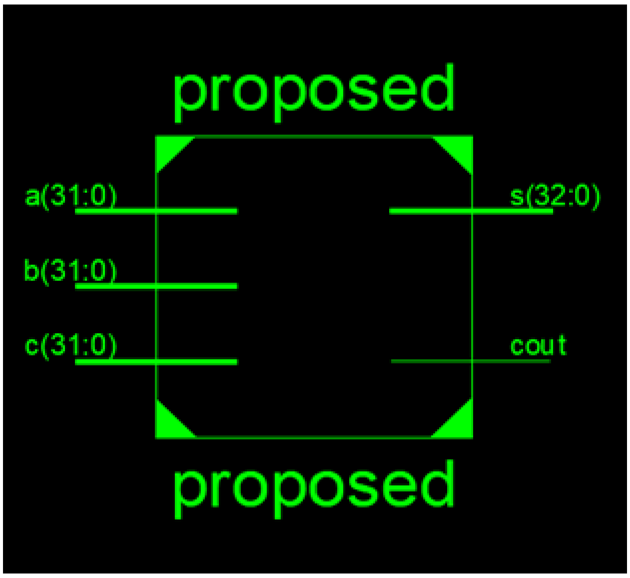

Fig.17 Three Operand Binary Adder Module

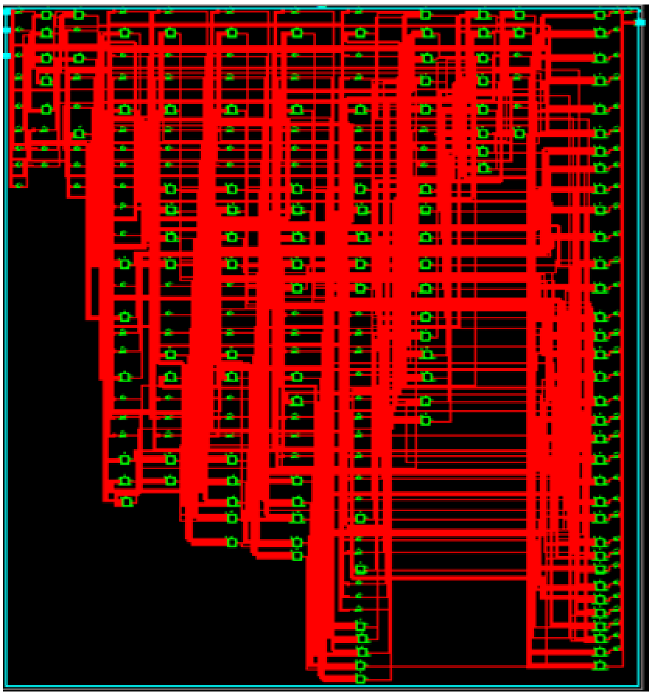

Fig.18 Three Operand Adder RTL Schematic

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

```
                               Gate    Net
Cell:in->out        fanout    Delay   Delay  Logical Name (Net Name)
----------------------------------------  ------------
  IBUF:I->O              3    0.001   0.703  b_0_IBUF (b_0_IBUF)
  LUT6:I0->O             3    0.097   0.521  ba0/carry1 (g1<1>)
  LUT5:I2->O             3    0.097   0.305  gr1/g1 (g4<0>)
  LUT6:I5->O             3    0.097   0.305  gr13/g11 (gr13/g1)
  LUT6:I5->O             5    0.097   0.575  g4<2>1 (g4<2>)
  LUT4:I0->O             2    0.097   0.561  gr14/g1 (g4<3>)
  LUT6:I2->O             3    0.097   0.566  gr16/g3 (g4<5>)
  LUT6:I2->O             3    0.097   0.566  gr18/g3 (g4<7>)
  LUT6:I2->O             3    0.097   0.566  gr110/g3 (g4<9>)
  LUT6:I2->O             3    0.097   0.521  gr112/g3 (g4<11>)
  LUT6:I3->O             2    0.097   0.688  gr114/g1 (gr114/g)
  LUT5:I0->O             1    0.097   0.556  gr114/g2 (g4<13>)
  LUT5:I1->O             1    0.097   0.279  su29/Mxor_sum_xo<0>1 (s_30_OBUF)
  OBUF:I->O                   0.000          s_30_OBUF (s<30>)
----------------------------------------
Total                         7.878ns (1.165ns logic, 6.713ns route)
                                      (14.8% logic, 85.2% route)
```

Fig.19 Three Operand Binary Adder Delay

```
Slice Logic Utilization:
 Number of Slice LUTs:                     128  out of  63400     0%
    Number used as Logic:                  128  out of  63400     0%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:       128
   Number with an unused Flip Flop:        128  out of    128   100%
   Number with an unused LUT:                0  out of    128     0%
   Number of fully used LUT-FF pairs:        0  out of    128     0%
   Number of unique control sets:            0

IO Utilization:
 Number of IOs:                            130
 Number of bonded IOBs:                    130  out of    210    61%
```

Fig.20 Three Operand Adder Area

On-Chip Power Summary				
On-Chip	Power (mW)	Used	Available	Utilization (%)
Clocks	0.00	0	---	---
Logic	0.03	100	63400	0
Signals	0.02	224	---	---
IOs	1.65	130	210	62
Static Power	82.16			
Total	83.86			

Fig.21 Three Operand Binary Adder On-Chip Power Summary

Power Supply Summary			
	Total	Dynamic	Static Power
Supply Power (mW)	83.86	1.70	82.16

Fig.22 Three Operand Binary Adder Power

Proposed Adder: The simulation results of Proposed Adder depicted in below fig.23.

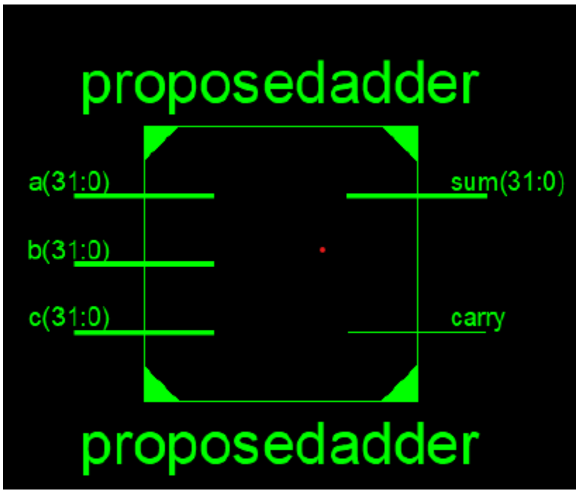

Fig.23 Proposed Adder Module

Fig.24 Proposed Adder RTL Schematic

Power Supply Summary			
	Total	Dynamic	Static Power
Supply Power (mW)	42.91	6.65	36.26

Fig.25 Proposed Adder Power summary

On-Chip Power Summary				
On-Chip	Power (mW)	Used	Available	Utilization (%)
Clocks	0.00	0	---	---
Logic	0.02	103	27288	0
Signals	0.03	230	---	---
IOs	6.60	129	218	59
Static Power	36.26			
Total	42.91			

Fig.26 Proposed Adder On-Chip Power

```
Slice Logic Utilization:
 Number of Slice LUTs:                     135  out of  27288     0%
    Number used as Logic:                  135  out of  27288     0%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:       135
   Number with an unused Flip Flop:        135  out of    135   100%
   Number with an unused LUT:                0  out of    135     0%
   Number of fully used LUT-FF pairs:        0  out of    135     0%
   Number of unique control sets:            0

IO Utilization:
 Number of IOs:                            129
 Number of bonded IOBs:                    129  out of    218    59%
```

Fig.27 Proposed Adder Area

```
 LUT5:I0->O            2   0.203   0.961  b0/e1 (g2<0>)
 LUT5:I0->O            5   0.203   1.079  gr2/d1 (g2<1>)
 LUT6:I0->O            1   0.203   0.000  gr5/d_G (N26)
 MUXF7:I1->O           3   0.140   0.995  gr5/d (g3<2>)
 LUT5:I0->O            2   0.203   0.845  gr9/d3 (g4<2>)
 LUT5:I2->O            2   0.205   0.845  gr11/d1 (g4<4>)
 LUT5:I2->O            3   0.205   0.898  gr13/d1 (g4<6>)
 LUT6:I2->O            2   0.203   0.721  gr18/d2 (gr18/d1)
 LUT5:I3->O            1   0.203   0.580  gr22/d1_SW3 (N19)
 LUT6:I5->O            2   0.205   0.981  gr22/d1 (gr22/d1)
 LUT6:I0->O            3   0.203   0.651  gr22/d (g5<6>)
 LUT5:I4->O            1   0.205   0.580  gr26/d3_SW0 (N21)
 LUT6:I5->O            2   0.205   0.617  gr26/d3 (gr26/d2)
 LUT5:I4->O            2   0.205   0.845  gr26/d4 (g5<10>)
 LUT6:I3->O            1   0.205   0.580  gr30/d_SW1 (N23)
 LUT6:I5->O            1   0.205   0.580  gr30/d (g5<14>)
 LUT5:I4->O            1   0.205   0.579  s30/Mxor_c_xo<0>1 (sum_31_OBUF)
 OBUF:I->O                 2.571          sum_31_OBUF (sum<31>)
----------------------------------------
Total                     21.583ns (7.404ns logic, 14.179ns route)
                                   (34.3% logic, 65.7% route)
```

Fig.28 Proposed Adder Delay

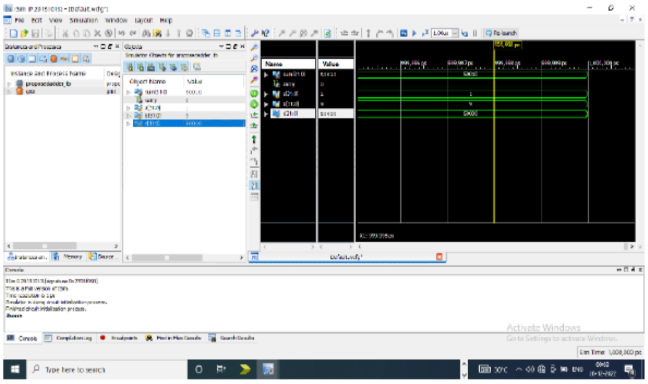

Fig.29 Proposed Adder Simulations

Table.1 Power, Delay, PDP and number of LUTs of various adders of 32-bit architecture

Architecture (32-bit)	Delay (ns)	Static Power (mW)	Dynamic Power (mW)	Total Power (mW)	Number of LUTs	PDP
Carry Save Adder	40.418	36.04	3.31	39.04	94	1577.91872
Han-Carlson Adder	10.442	82.16	1.11	83.27	124	869.505
HSAT3-32	7.878	82.16	1.70	83.86	128	660.64908
Proposed Adder	21.583	36.26	6.65	42.91	135	926.12651
Hybrid Adder	24.235	36.14	3.0	39.14	115	949.76965

V. CONCLUSION

In this paper, a new Three Operand Binary Adder architecture is proposed. The proposed architecture is a parallel prefix adder, unlike regular parallel prefix adders it computes the addition in four stages bit addition logic, base logic, propagate, and generate logic and, sum logic. The critical path delay of the proposed adder is less. When the performance of the proposed adder is compared with the carry save adder it is evident that with a slight increase in power utilization of about 9.04%, it computes addition 53% faster, carry save adder computes the addition in O(n) time complexity whereas proposed does in O(n/2) only, and PDP(power delay product) is 58.2% is less, and the area is slightly increases and when it is compared with the Han-Carlson adder proposed adder architecture is slow in terms of delay, but it utilizes 52% less power, and PDP is almost same, and when it is compared with existing three operand adder performance is slow in terms of delay but proposed architecture utilizes 49.4% less power, PDP is almost remained same. Therefore, proposed architecture is used for less power requirement applications. The suggested and carry save adders, Han-Carlson hybrid adders, and current Three Operand Binary adders were implemented using Verilog HDL in Xilinx 14.7 ISE, which is used for all measurements.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Rajiv Gandhi University Knowledge & Technologies (RGUKT)-Basar, for their constant support throughout this research work. I thank Electronics and Communication engineering Department, of Rajiv Gandhi University Knowledge & Technologies, Basar.

DECLARATION

Funding/ Grants/ Financial Support	Yes, This work was supported by the Rajiv Gandhi University Knowledge Technologies (RGUKT)- Basar https://www.rgukt.ac.in/
Conflicts of Interest/ Competing Interests	No conflicts of interest to the best of our knowledge.
Ethical Approval and Consent to Participate	No, the article does not require ethical approval and consent to participate with evidence.
Availability of Data and Material/ Data Access Statement	Not relevant.
Authors Contributions	All authors have equal participation in this article.

REFERENCES

1. R. M. a. T.Sasilatha, "A power efficient carry save adder and modified carry save adder using cmos technology," International Conference on Computational Intelligence and Computing Research, pp. 1-5, 2013.
2. K. R. a. M. Ahmadi, "Fast carry -look-ahead adder," Engineering Solutions for the Next MIllennium, vol. 1, pp. 529-532, 199.
3. D. R. a. R. S. K. a. D. S. a. M. V. R. a. V. S. a. S. S. A, "Design and Analysis of High-Performance Carry Skip Adder using Various Full Adders," 2021 Smart Technologies, Communication and Robotics (STCR), pp. 1-5, 2021.
4. O. J. Bedrij, "Carry-Select Adder," IRE Transactions on Electronic Computers, Vols. EC-11, pp. 340-346, 1962. [CrossRef]

5. S. a. P. N. a. S. M. a. S. R. a. T. T. a. M. M. K, "Certain Investigations on Adder Design for VLSI Signal Processing," 2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS), vol. 1, pp. 1409-1413, 2022.
6. B. a. M. N. a. J. M. O. a. J. P. R. Koyada, "A comparative study on adders," 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), pp. 2226-2230, 2017.
7. S. a. V. G. Dubey, "Analysis of Basic Adder with Parallel Prefix Adder," 2020 First IEEE International Conference on Measurement, Instrumentation, Control and Automation (ICMICA), pp. 1-6, 2020.
8. A. K. a. S. A. A. a. S. M. a. S. P. S. a. P. R. R, "Design and Implementation of 64-bit Parallel Prefix Adder," 2020 IEEE International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER), pp. 159-164, 2020.
9. A. a. S. S. K. Raju, "Design and performance analysis of multipliers using Kogge Stone Adder," 2017 3rd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), pp. 94-99, 2017.
10. P. P. a. J. V. D. Potdukhe, "Design of high speed carry select adder using brent kung adder," 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), pp. 652-655, 2016.
11. M. P. B. Sravanam Sravani, "Design of Efficient 32-Bit Parallel Prefix Ladner," International Journal of Advanced Trends in Engineering, Science and Technology (IJATEST), vol. 2, no. 2, 2017.
12. R. S. S. Gayathri, "Parallel Prefix Speculative Han-Carlson Adder," IOSR Journal of Electronics and Communication Engineering (IOSR-JECE), vol. 11, no. 3, pp. 38-43, 2016.
13. K. S. a. B. D. K. a. G. N. a. S. H. Pandey, "An Ultra-Fast Parallel Prefix Adder," 2019 IEEE 26th Symposium on Computer Arithmetic (ARITH), pp. 125-134, 2019.
14. S. a. C. K. P. a. S. E. E. Muthyala Sudhakar, "Hybrid Han-Carlson adder," 2012 IEEE 55th International Midwest Symposium on Circuits and Systems (MWSCAS), pp. 818-821, 2012. [CrossRef]
15. A. K. a. P. R. a. R. K. C. Panda, "High-Speed Area-Efficient VLSI Architecture of Three-Operand Binary Adder," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 67, no. 11, pp. 3944-3953, 2020. [CrossRef]
16. A. a. C. R. K. Kumar Panda, "A Coupled Variable Input LCG Method and its VLSI Architecture for Pseudorandom Bit Generation," IEEE Transactions on Instrumentation and Measurement, vol. 69, no. 4, pp. 1011-1019, 2020. [CrossRef]
17. A. K. a. R. K. C. Panda, "Modified Dual-CLCG Method and its VLSI Architecture for Pseudorandom Bit Generation," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 66, no. 3, pp. 989-1002, 2019. [CrossRef]

AUTHORS PROFILE

Chintam Shravan presently working as an Assistant Professor in the Department of Electronics and Communication Engineering, Rajiv Gandhi University of Knowledge Technologies, Basar, Telangana. He received B. Tech degree in Electronics and Communication engineering from Jawaharlal Nehru Technological University, Hyderabad, India in 2011 and the M. Tech degree in VLSI & Embedded System design from Kakatiya University, Warangal, India, in 2014 and he is pursuing a Ph.D. in VLSI Engineering at University college of Engineering, Osmania University. His research interest includes Memories, DRAM/SRAM and memory controllers, Low power VLSI, Energy converters.

Sai Krishna Marri presently working as a physical design engineer in Smart SOC solutions, Hyderabad Telangana. He received B. Tech degree in Electronics and Communication Engineering from Rajiv Gandhi University of Knowledge Technologies, Basar in 2023.His research interests includes Low power VLSI, latest trends in VLSI design, embedded systems.

Rapolu Saidulu presently working as an Assistant Professor in the Department of Electronics and Communication Engineering, Rajiv Gandhi University of Knowledge Technologies, Basar, Telangana. He received B. Tech degree in Electronics and Communication engineering from Jawaharlal Nehru Technological University, Hyderabad, India in 2011 and the M. Tech degree in VLSI System design from JNTU, Hyderabad, India, in 2013. His research interest includes Memories, Low power VLSI, Device modeling, Embedded Systems.

Panchareddy Tejasvey presently working as an Assistant Professor in the Department of Electronics and Communication Engineering, Rajiv Gandhi University of Knowledge Technologies, Basar, Telangana. She received B. Tech degree in Electronics and Communication engineering from Jawaharlal Nehru Technological University, Hyderabad, India in 2012 and the M. Tech degree from JNTU Hyderabad, India, in 2016 and she is pursuing a Ph.D. in Embedded System and IoT at GITAM University. Her research interest includes Embedded Systems, Internet of Things.

Sai Radha Krishan G presently working as an Assistant Professor in the Department of Electronics and Communication Engineering, Rajiv Gandhi University of Knowledge Technologies, Basar, Telangana. He received B. Tech degree in Electronics and Communication engineering from Jawaharlal Nehru Technological University, Hyderabad, India in 2011 and the M. Tech degree in wireless networks and applications from Amruta University, Kerala, India, in 2013 and he is pursuing a Ph.D. in wireless sensor networks at Amruta University, Kerala, India. His research interest includes wireless sensor networks, Machine Learning, Block Chain Technology, Low Power Energy converters.