DIO: 30.0405/ijearst.2025.3575 https://ijearst.co.in/

ENGINEERING IN ADVANCED RESEARCH SCIENCE AND TECHNOLOGY

ISSN 2040-7467
Vol.03, Issue.02
Dec-2025
Pages: 1057-1068

Design and Performance Analysis of Various 64-bit Hybrid Adders using Verilog

[1]ATTILI AYYAPPA SWAMI, [2]Y. APPA RAO
[1]M. Tech, Dept. of ECE, V S M College of Engineering, Ramachandrapuram, A.P
[2]Associate Professor, Dept. of ECE, V S M College of Engineering, Ramachandrapuram, A.P

Abstract: This study focuses on the design and implementation of a 64-bit VLSI adder, incorporating advanced techniques for improved efficiency. Some of the adders in this study utilize a modified algorithm to enhance computational speed and reduce power consumption. Initially, this project focuses on the design and implementation of a 32-bit VLSI adder, incorporating advanced techniques for improved efficiency. Carry select adder (CSelA) and Kogge stone adder (KSA) are used to implement existing method in 64 bit. As an enhancement of this project 64 bit adder with CSelA_KSA is designed and again improved with proposed method like 64bit modified CSelA and HCA in order to reduce time and area parameters A hybrid 64-bit adders is proposed by combining different adder architectures and logic techniques. One such combination includes parallel prefix adders, such as the Han-Carlson Adder (HCA). The selection of the best 64-bit hybrid adder is based on the performance evaluation of 32-bit hybrid adders, identifying the most efficient design for extension to 64-bit.

Keywords: Hybrid Adder, Carry select adder, Kogge stone adder, Binary Excess Convertor, Han-Carlson Adder, Square Root.

Introduction: The challenge of the verifying a large design is growing exponentially. There is a need to define new methods that makes functional verification easy. Several strategies in the recent years have been proposed to achieve good functional verification with less effort. Recent advancement towards this goal is methodologies. The methodology defines a skeleton over which one can add flesh and skin to their requirements to achieve functional verification. This trend is expected to continue with very important implications on VLSI and systems design. One of the most important characteristics of information services is their increasing need for very high processing power and bandwidth. The other important characteristic is that the information services tend to become more and more personalized (as opposed to collective services such as broadcasting), which means that the devices must be more intelligent to answer individual demands and at the same time they must be portable to allow more flexibility/mobility.

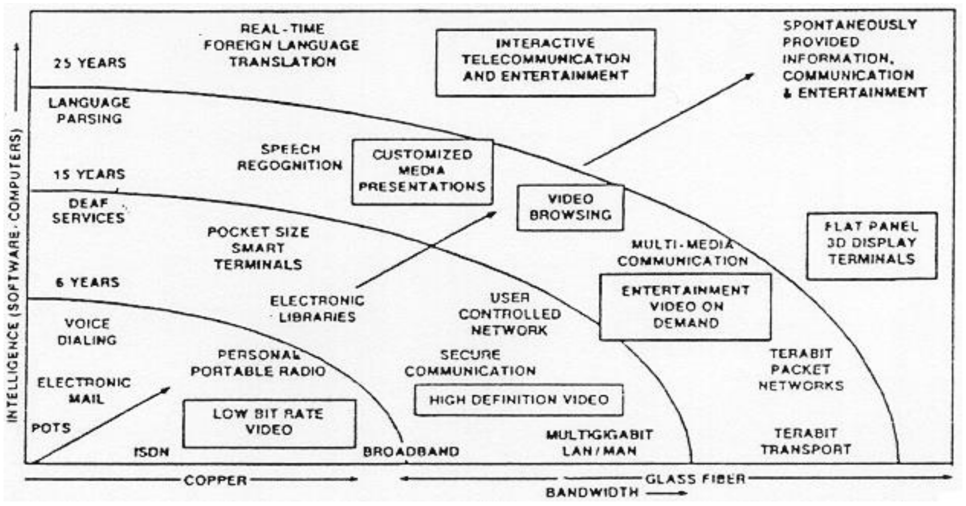


Fig1 : Overview of the prominent trends in information technologies

As more and more complex functions are required in various data processing and telecommunications devices, the need to integrate these functions in a small system/package is also increasing. The level of integration, as measured by the number of logic gates in a monolithic chip, has been steadily rising for almost three decades mainly due to the rapid progress in processing technology and interconnect technology.

Below Table shows the evolution of logic complexity in integrated circuits over the last three decades and marks the milestones of each era. Here, the numbers for circuit complexity should be interpreted only as representative examples to show the order-of-magnitude. A logic block can contain anywhere from 10 to 100 transistors, depending on the function. State-of-the-art examples of ULSI chips, such as the DEC Alpha or the INTEL Pentium contain 3 to 6 million transistors.

ERA (number of logic blocks per chip)	DATE	COMPLEXITY
Single transistor	1959	less than 1
Unit logic (one gate)	1960	1
Multi-function	1962	2-4
Complex function	1964	5-20
Medium scale integration	1967	20-200
Large scale integration	1972	200-2000
Very large scale integration	1978	2000-20000
Ultra large scale integration	1989	>20000

Table1: Evolution of logic complexity in integrated circuits

Existing Method:

Hybrid CSelA_KSA:

A hybrid adder tutorial explains how to design and implement a type of adder circuit that combines different logic styles to optimize performance. It focuses on techniques like hybrid pass logic with static CMOS (HPSC) to reduce transistors and power consumption while enhancing overall system speed.

CARRY SELECT ADDER:

Basically, this project can be classified into three major parts.

- Design of square root carry select adder
- Design of square root carry select adder with BEC (modified CSLA)
- Comparison of the two designs in terms of area, power consumption and time delay.

The above classification will be discussed individually in the following sessions.

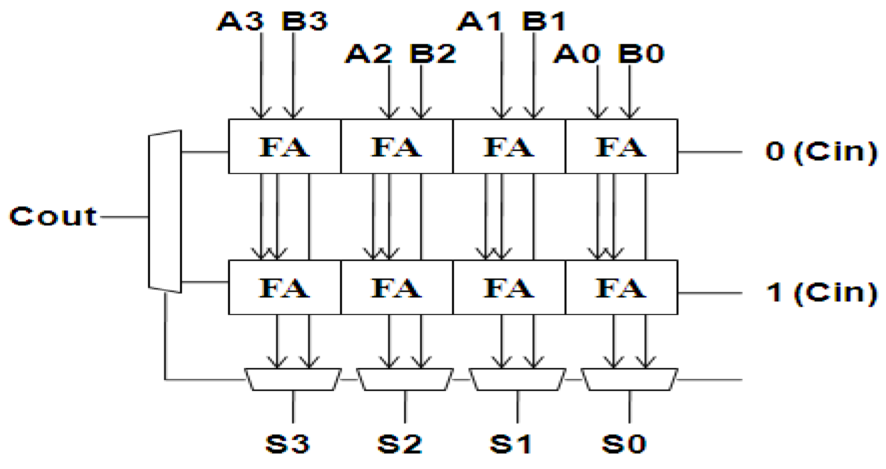


Fig2 : Basic building block of CSLA

The basic building block of the square root carry select adder of block size '4' is shown in the above figure.

Block diagram of 16-bit Carry Select adder:

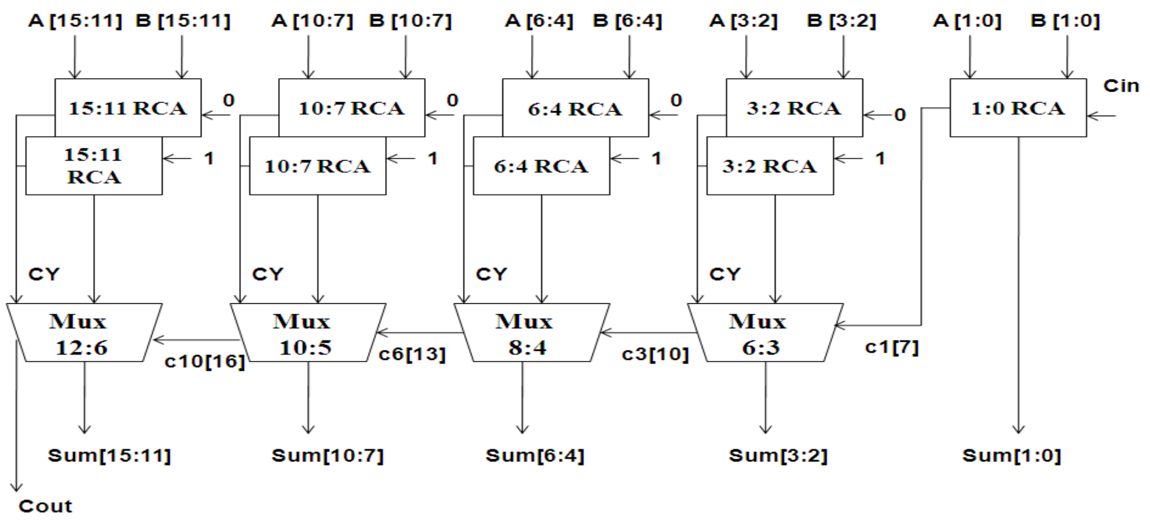


Fig3 : Existing system (Regular 16-bit Carry select adder)

The block diagram of the regular 16-bit square root CSLA is shown in the figure. This adder is a variable sized adder. The carryselect adder generally consists of two ripple carry adders and a multiplexer. Adding two n-bit numbers with a carry-select adder is done with two adders (therefore two ripple carry adders) in order to perform the calculation twice, one time with the assumption of the carry being zero and the other assuming one. After the two results are calculated, the correct sum, as well as the correct carry, is then selected with the multiplexer once the correct carry is known.

KOGGE STONE ADDER: The Kogge–Stone adder or KSA is a parallel prefix form of CLA (carry-lookahead adder). This adder uses more area to implement as compared to the Brent–Kung adder, although it has a low fan-out at every stage, which enhances the performance of typical CMOS process nodes. But, wiring congestion is frequently an issue for KSAs. Kogge Stone adder or KSA is a very fast adder utilized in various signal processing processors (SPP) to perform the best arithmetic function. So the operation speed of this adder can be restricted by carrying propagation from input to output. Generally, KSA is a parallel prefix adder that has the specialty of best addition depending on design time which is used for high-performance based arithmetic circuits within the industry.

Kogge Stone Adder Circuit Diagram

The Kogge-Stone Adder diagram is shown below. This type of adder is considered simply the fastest and most common architecture adder design mainly for high-performance adders within the industry. In this type of adder, carriers are very quickly generated by computing them in parallel at the increased area cost.

The Tree structures of carry propagate & generate signals are shown in the below diagram. In this adder, the Carry generation network is a very significant block that includes three blocks; Black cell, Grey cell, and buffer. So black color cells are used mainly in the calculation of both generate & propagate signals, Grey cells are mainly used in the calculation of generate signals which are required within the calculation of sum within the post-processing stage and Buffers are mainly used for balancing the loading effect.

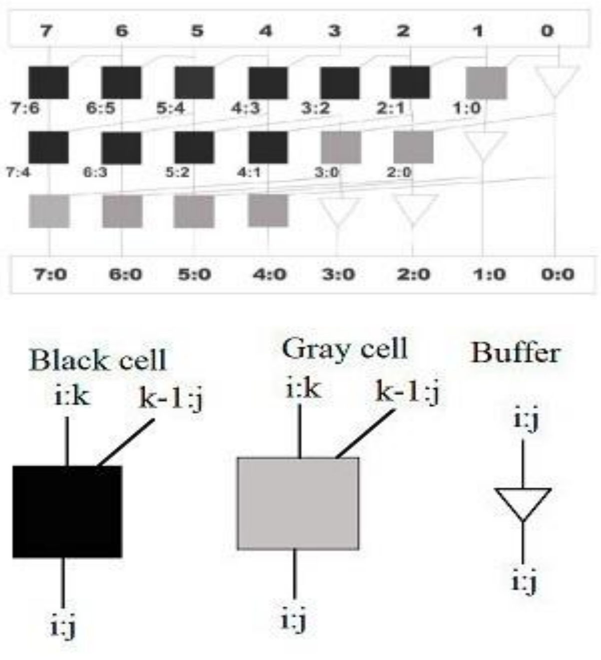


Fig4: KSA Tree Structure

Kogge Stone Adder working:

The Kogge-Stone adder tracks "generate" & "propagate" bits internally for spans of bits similar to all carry-lookahead adders. We begin with 1-bit spans, wherever a single column within the addition produces a carry bit when both inputs are 1 (logical AND) & a carry bit will propagate if precisely one input is 1 (logical XOR). Thus, Kogge-Stone Adder includes mainly three processing stages for calculating the sum bits; the Pre-processing stage, the Carry generation network, and the Post-processing stage. So these three steps are mainly involved in this adder operation. These three stages are discussed below.

Preprocessing Stage

This preprocessing stage involves the computation of both generated & propagated signals equivalent to every pair of bits within A and B.

Pi = Ai xor Bi

Gi = Ai and Bi

Carry Generation Network

In the carry generation stage, we calculate carries equivalent to every bit. So these operations execution can be carried out in parallel. After the carries computation in parallel, these are segmented into minor pieces. As intermediate signals, it utilizes carry propagate & generate signals which are specified by the below logic equations.

CPi:j = Pi:k + 1 and Pk:j

CGi:j = Gi:k + 1 or (Pi:k + 1 and Gk:j)

Post Processing

This post-processing stage is very common to all carry look-ahead family adders and it involves calculation of sum bits.

Ci – 1 = (Pi and Cin) or Gi

Si = Pi = x or Ci – 1

4-bit Kogge-Stone Adder

In the 4-bit Kogge-Stone adder, every vertical stage generates a “propagate” & a “generate” bit. The carries are generated in the final stage where these bits are XOR through the first propagate after the input within the square boxes to generate the sum bits.

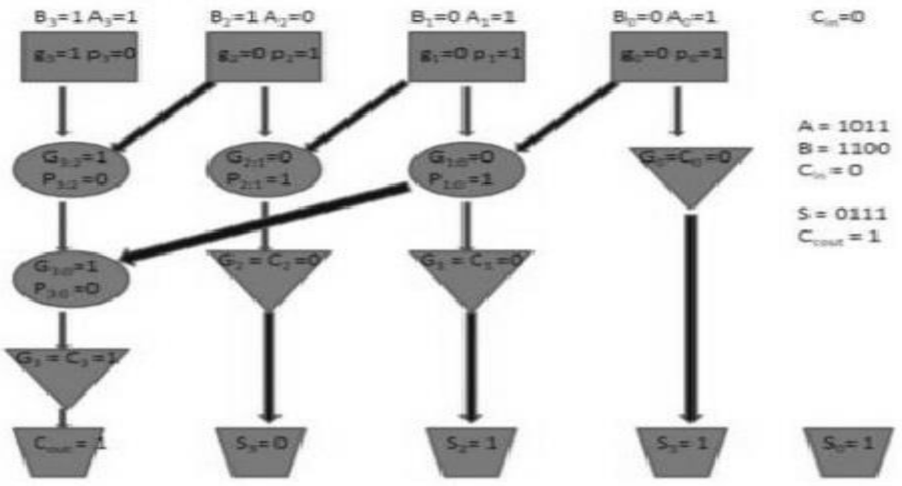


Fig: 5-bit Kogge Stone Adder

For instance; if the propagate is calculated by XOR when the A=1 & B=0 then it generates the propagate o/p as 1. Here, the generate value can be calculated with AND when the A = 1, B = 0, and the generate o/p value is 0.

Similarly, all the sum bits are calculated for Inputs: A = 1011 & B = 1100 Outputs then sum = 0111 and carry Cout = 1. In this adder proceed with the five outputs in the below expansion.

S0 = (A0 ^ B0) ^ CIN.

S1 = (A1 ^ B1) ^ (A0 & B0).

S2 = (A2 ^B2) ^ (((A1 ^ B1) & (A0 & B0)) | (A1 & B1)).

S3 = (A3 ^ B3) ^ ((((A2 ^ B2) & (A1 ^ B1)) & (A0 & B0)) | (((A2 ^ B2) & (A1 & B1)) | (A2 & B2))).

S4 = (A4 ^ B4) ^ ((((A3 ^ B3) & (A2 ^ B2)) & (A1 & B1)) | (((A3 ^ B3) & (A2 & B2)) | (A3 & B3))).

DRAWBACKS:

- More area overhead due to unwanted extra adders in carry select adder.
- Hude dynamic power consumption due to extra added hardware, which is used for adding 1 with inputs.
- Unnecessary leakage power
- Short circuit current

Proposed Method:

Hybrid MCSelA_HCA

Design of CSLA with BEC:

As stated above, the main idea of this work is to use BEC instead of the RCA with Cin=1 in order to reduce the area and power consumption of the regular CSLA. To replace the n-bit RCA, an (n+1)-bit BEC is required.

Structure of BEC:

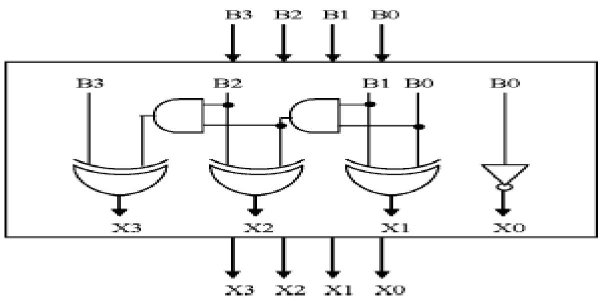


Fig6 : Structure of 4-bit BEC

The Boolean expressions of the 4-bit BEC are listed as below:

X0 = ~B0

X1 = B0 ^ B1

X2 = B2 ^ (B0 & B1)

X3 = B3 ^ (B0 & B1 & B2)

The above figure illustrates how the basic function of the CSLA is obtained by using the 4-bit BEC together with the mux. One input of the 8:4 mux gets its input from B3 B2 B1 B0 and another input to the mux is the BEC output. This produces the two possible partial results in parallel and the mux is used to select either the BEC output or the direct inputs according to the control signal Cin.

Block diagram of CSLA with BEC:

The structure of the proposed 16-b SQRT CSLA using BEC for RCAwithCin=1 to optimize the area and power is shown in the figure.

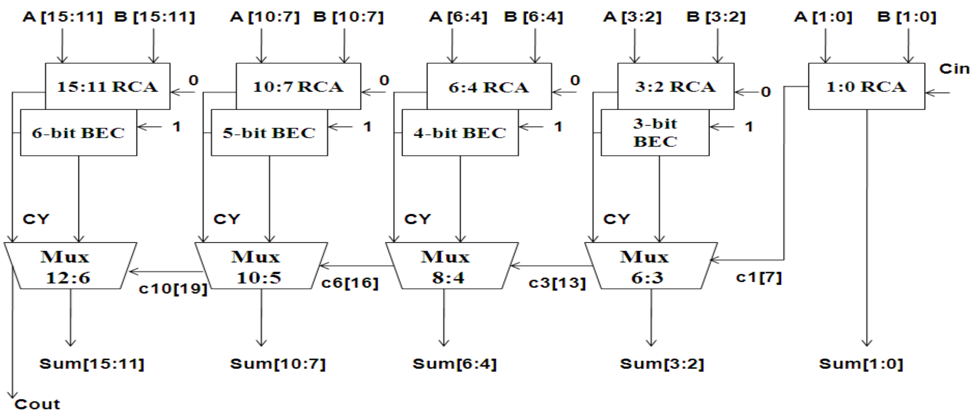


Fig7: Modified system (Modified 16-b SQRT CSLA)

Comparing the block diagram of the regular square root CSLA with the modified square root CSLA, it can be seen that the RCA with Cin=1 is replaced by BEC (binary to excess-1 converter). This is done to reduce the area consumption. This can be seen after evaluating the group delay and the number of gates required for the design.

HAN-CARLSON ADDER:

The Han-Carlson adder is a blend of the Brent-Kung and Kogge-Stone adders. It uses one Brent-Kung stage at the beginning followed by Kogge-Stone stages, terminating with another Brent-Kung stage to compute the odd numbered prefixes. It provides better performance compared to Kogge-Stone for smaller adders.

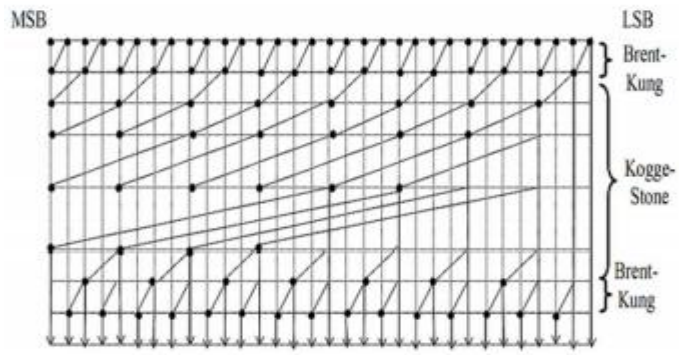


Fig8: GRAPICAL REPRESENTATION OF HAN CARLSON ADDER

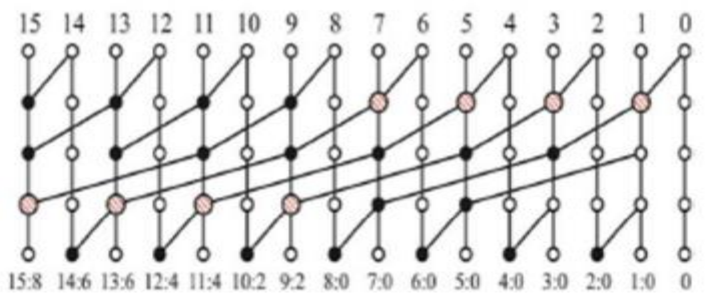
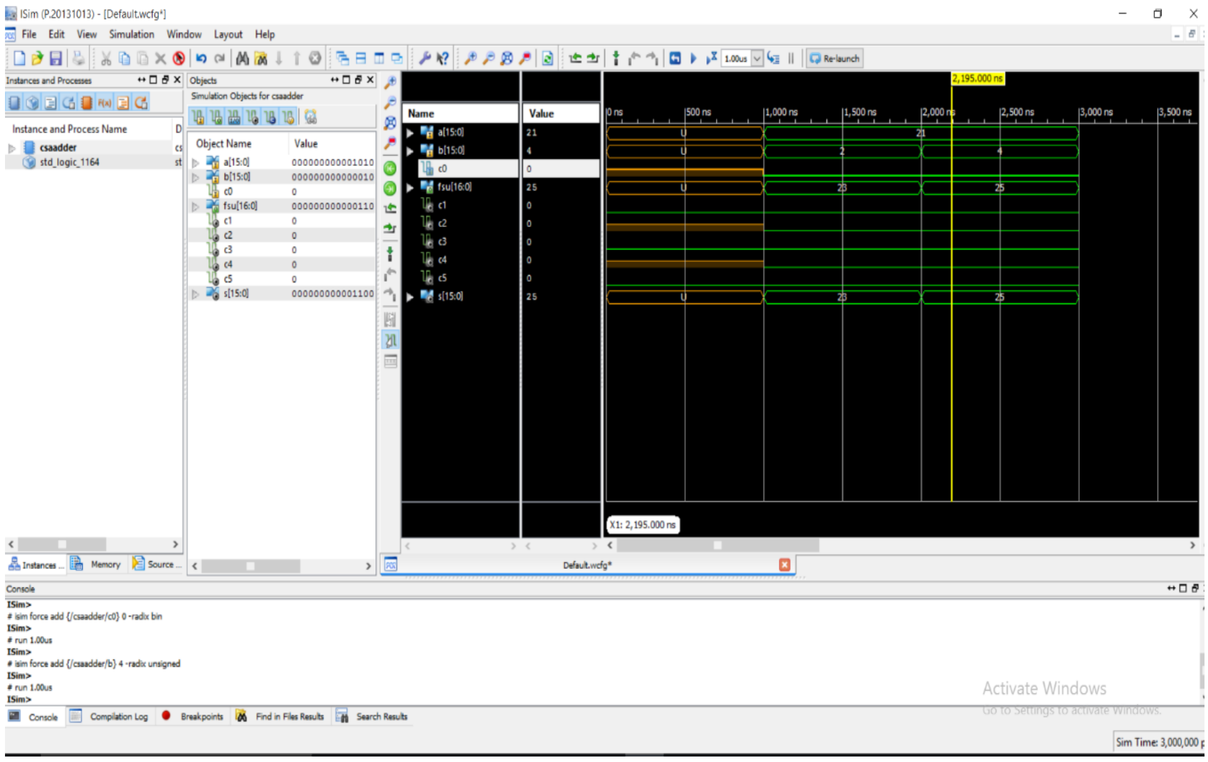


Fig9: HAN CARLSON ADDER CARRY LENGTH (K=16)

The Han-Carlson is the family of networks between Kogge-Stone and Brent-Kung. Han-Carlson adder can be viewed of Kogge-Stone adder. This adder is different from Kogge-Stone adder in the sense that these performs carry-merge operations on even bits and generate/propagate operation on odd bits. At the end, these odd bits recombine with even bits carry signals to produce the true carry bits. This adder has five stages in which the middle three stages resembles with the Kogge-Stone structure. The advantage of the adder is that it uses much less cells and its shorter. Thus there is a reduction in complexity at the cost of an additional stage for carry-merge path. We have generated a Han- Carlson Speculative Prefix Processing stage by deleting the last rows of the kogge stone adder. This yields a speculative stage with k=8=n/2^p, where p is the number of pruned levels. Parallel prefix adders are suitable for VLSI implementation since it differs from other adders, it can be used for large word sizes. The proposed design reduces the number of prefix operation by using more number of Brent-Kung stages and lesser number of Kogge-Stone Stages. This also reduces the complexity, silicon area and power consumption. Parallel Prefix Adder can be subdivided in the following stages: Pre-Processing, PostProcessing, Error Detection and Error Correction. The Error Correction Stage is Off the critical path, as it has two clock cycles to obtain the exact sum when speculation fails. The Pre-Processing and Post-Processing Stages of a Prefix adder involve only simple operations on signals to each bit location. Hence, adder performs mainly on Prefix operation. Therefore black dots represent the prefix operator, while white dots represent simple place holders.

RESULTS:


ISim (P.20131013) - [Default.wcfg*]
File Edit View Simulation Window Layout Help
Instances and Processes
Objects
Simulation Objects for csaadder
Re-launch
2,195.000 ns
X1: 2,195.000 ns
Default.wcfg*
Instances ... Memory Source ...
Console Compilation Log Breakpoints Find in Files Results Search Results
Sim Time: 3,000,000 p

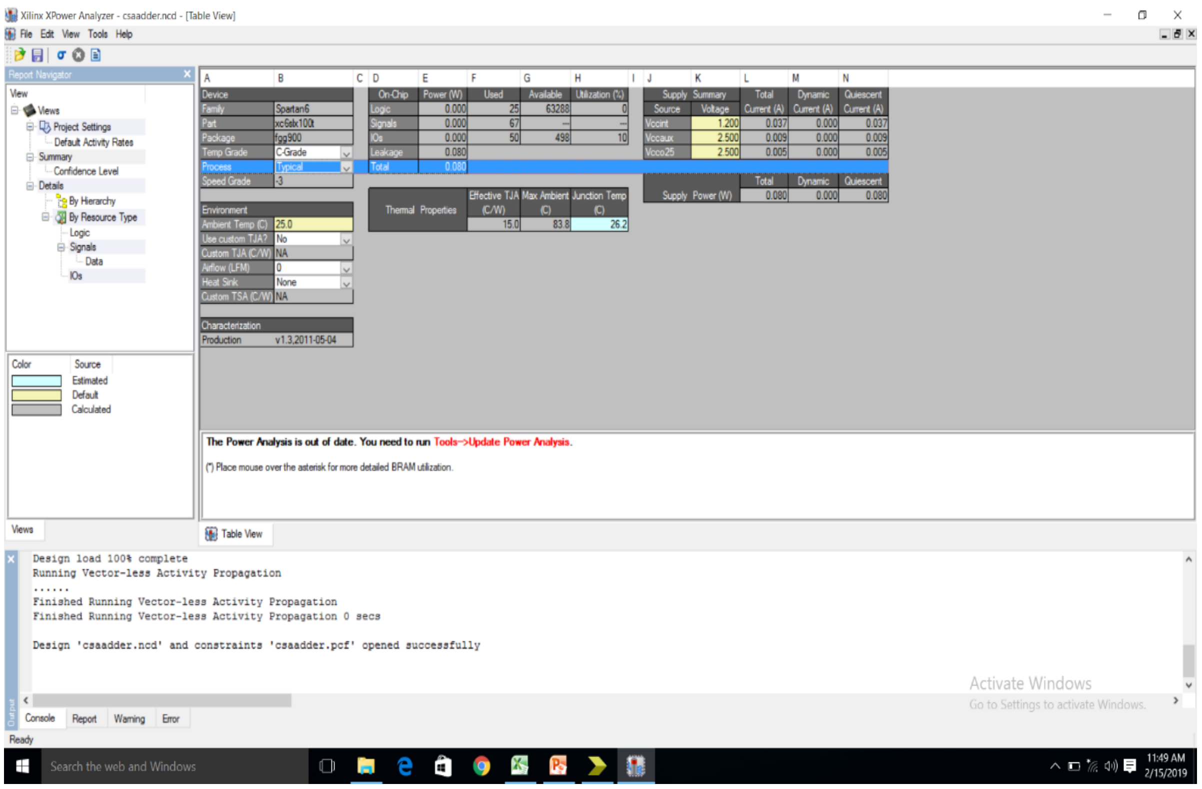
Instance and Process Name
csaadder
std_logic_1164

Object Name	Value
a[15:0]	000000000001010
b[15:0]	000000000000010
c0	0
fsu[16:0]	000000000000110
c1	0
c2	0
c3	0
c4	0
c5	0
s[15:0]	000000000001100

Name	Value
a[15:0]	21
b[15:0]	4
c0	0
fsu[16:0]	25
c1	0
c2	0
c3	0
c4	0
c5	0
s[15:0]	25

```
ISim>
# isim force add {/csaadder/c0} 0 -radix bin
ISim>
# run 1.00us
ISim>
# isim force add {/csaadder/b} 4 -radix unsigned
ISim>
# run 1.00us
ISim>
```

Fig a : Proposed Simulation results



A	B	C	D	E	F	G	H	I	J	K	L	M	N
Device			On-Chip	Power (W)	Used	Available	Utilization (%)		Supply Source	Summary Voltage	Total Current (A)	Dynamic Current (A)	Quiescent Current (A)
Family	Spartan6		Logic	0.000	25	63288	0		Vccint	1.200	0.037	0.000	0.037
Part	xc6slx100t		Signals	0.000	67	---	---		Vccaux	2.500	0.009	0.000	0.009
Package	fgg900		IOs	0.000	50	498	10		Vcco25	2.500	0.005	0.000	0.005
Temp Grade	C-Grade		Leakage	0.080									
Process	Typical		Total	0.080									
Speed Grade	-3										Total	Dynamic	Quiescent
									Supply	Power (W)	0.080	0.000	0.080

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	15.0	83.8	26.2

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	0
Heat Sink	None
Custom TSA (C/W)	NA

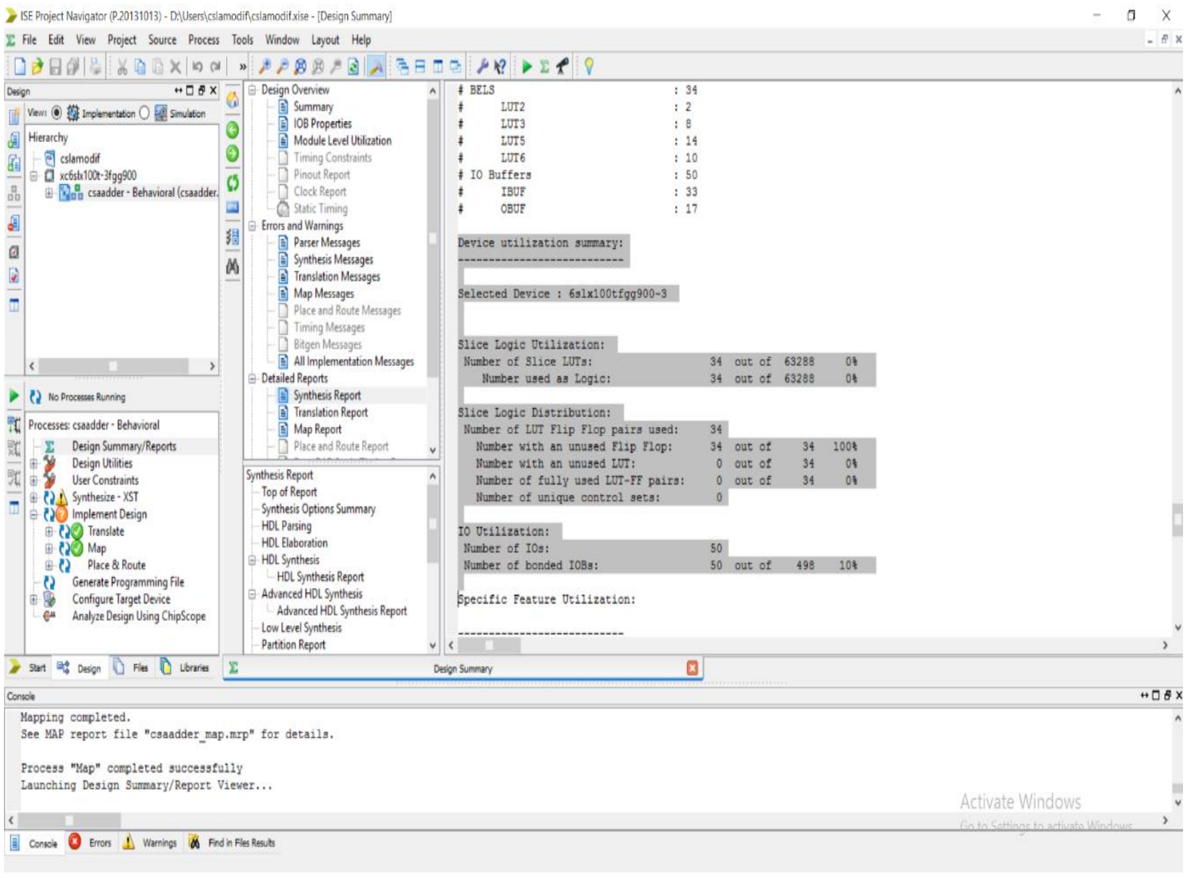
Characterization	
Production	v1.3,2011-05-04

The Power Analysis is out of date. You need to run Tools->Update Power Analysis.

(*) Place mouse over the asterisk for more detailed BRAM utilization.

```
Design load 100% complete
Running Vector-less Activity Propagation
......
Finished Running Vector-less Activity Propagation
Finished Running Vector-less Activity Propagation 0 secs

Design 'csaadder.ncd' and constraints 'csaadder.pcf' opened successfully
```

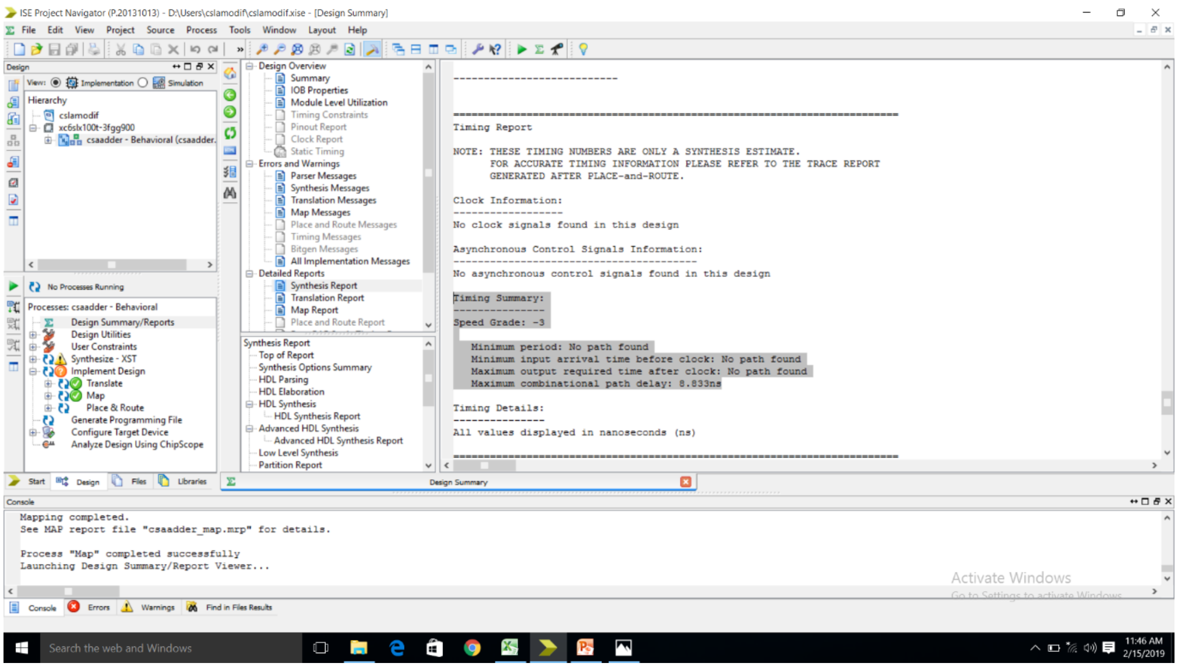
Fig b: Proposed power report



```
# BELS                                 : 34
#      LUT2                            : 2
#      LUT3                            : 8
#      LUT5                            : 14
#      LUT6                            : 10
# IO Buffers                           : 50
#      IBUF                            : 33
#      OBUF                            : 17

Device utilization summary:
---------------------------

Selected Device : 6slx100tfgg900-3


Slice Logic Utilization:
 Number of Slice LUTs:                  34  out of  63288     0%
    Number used as Logic:               34  out of  63288     0%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:    34
   Number with an unused Flip Flop:     34  out of     34   100%
   Number with an unused LUT:            0  out of     34     0%
   Number of fully used LUT-FF pairs:    0  out of     34     0%
   Number of unique control sets:        0

IO Utilization:
 Number of IOs:                         50
 Number of bonded IOBs:                 50  out of    498    10%

Specific Feature Utilization:

---------------------------
```

```
Mapping completed.
See MAP report file "csaadder_map.mrp" for details.

Process "Map" completed successfully
Launching Design Summary/Report Viewer...
```

Fig c: Proposed AREA Report



```
---------------------------


===========================================================================
Timing Report

NOTE: THESE TIMING NUMBERS ARE ONLY A SYNTHESIS ESTIMATE.
      FOR ACCURATE TIMING INFORMATION PLEASE REFER TO THE TRACE REPORT
      GENERATED AFTER PLACE-and-ROUTE.

Clock Information:
------------------
No clock signals found in this design

Asynchronous Control Signals Information:
----------------------------------------
No asynchronous control signals found in this design

Timing Summary:
---------------
Speed Grade: -3

   Minimum period: No path found
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: No path found
   Maximum combinational path delay: 8.833ns

Timing Details:
---------------
All values displayed in nanoseconds (ns)

===========================================================================
```

```
Mapping completed.
See MAP report file "csaadder_map.mrp" for details.

Process "Map" completed successfully
Launching Design Summary/Report Viewer...
```

Fig d: Proposed latency report

CONCLUSION & FUTURE SCOPE:

The hybrid adders are the fast adders which gives better Performance than the individual adder. It was observed as delay is Reduce speed is Increase along with power reduction. So for a perfect algorithm combination can be selected according to demand of Performance of integrated circuit. It was also observed that KCA along with MCSelA adder gives high performance. The area of all adders depends on the number of slices and RCA, CLA and PA have the least slice be counted. LUT's another manner has to have a significant range and right here in the work carried out MCSelA and HCA has the optimised LUT remember out of the all existing adders. Hence Hybrid MCSelA and HCA is nice amongst all the adders proven above. Further, this project can be enhanced by using QCA. Quantum Cellular automata are study of gate less structures for further improvement of gate delays and path losses. By using QCA technologies; power losses and propagation delays can be eliminated. Majority gate is basic element In QCA architectures.

References:

[1] N. Varshney and G. Arya,"Design and Execution of Enhanced Carry Increment Adder using Han-Carlson and Kogge-Stone adder Technique," IEEE. Conf. on Electronics Communication and Aerospace Technology, 2019.

[2] V.G. Oklobdzija, B.R. Zeydel and H.Q. Dao,"Comparison of high performance VLSI adders in the energy-delay space," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 13, no. 6, 2005.

[3] S. Pandu, A. Benerjee , B. Maji , and Dr. A.K. Mukhopadhya, " Power and delay comparison between different types of full adder circuits",in International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, vol. 1, no. 3, 2012.

[4] S. Ghosh , P. Ndai and K. Roy, " A novel low overhead fault tolerant Kogge-Stone adder using adaptive clocking," Design, Automation and Test in Europe IEEE, 2008 .

[5] G. G., S. S. Raju and S. Suresh,"Parallel Prefix Speculative Han Carlson Adder," in IOSR Journal of Electronics and Communication, vol. 11, no.3, 2016.

[6] B. Koyada, N. Meghana., Md.O. Jaleel. andP. R. Jeripotula,"A Comparative Study on Adders," in IEEE Conf. on Wireless Communication, Signal Processing and Networking , 2017.

[7] Y. Chen, H. Li, J. Li, and C.-K. Koh, "Variable-latency adder (VL-adder): New arithmetic circuit design practice to overcome NBTI," in Proc. ACM/IEEE Int. Symp. Low Power Electron. Design (ISLPED), Aug. 2007, pp. 195–200. [8] Y. Liu, Y. Sun, Y. Zhu, and H. Yang, "Design methodology of variable latency adders with multistage function speculation," in Proc. IEEE 11th Int. Symp. Quality Electron. Design (ISQED), Mar. 2010, pp. 824–830. [9] Y.-S. Su, D.-C. Wang, S.-C. Chang, and M. Marek-Sadowska, "Performance optimization using variable-latency design style," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 19, no. 10, pp. 1874–1883, Oct. 2011. [10] R. P. Brent and H. T. Kung, "A regular layout for parallel adders," IEEE Trans. Comput., vol. C-31, no. 3, pp. 260–264, Mar. 1982. [11] Darjn Esposito; Davide De Caro; Michele De Martino; Antonio G. M. Strollo, "Variable latency speculative Han-Carlson adders topologies" IEEE Conference Publications 2015 11th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME) [12] K. Golda Hepzibha; C P. Subha, "A novel implementation of high speed modified brent kung carry selectadder"IEEE Conference Publications 2016 10th International Conference on Intelligent Systems and Control (ISCO)