Journal of Engineering Sciences

Vol 13, Issue 06, June/2022
ISSN NO:0377-9254

DESIGN AND PERFORMANCE ANALYSIS OF 64-BIT PARALLEL PREFIX ADDERS FOR AREA EFFICIENT VLSI APPLICATIONS

[1]T.Gowri, [2]Ch.Bala Bharath, [3]M.AnvithA, [4]A.Mohana Ravali

#2,#3,#4, Student, Department of ECE, GITAM (deemed to be University), Gandhi nagar Rushikonda Visakhapatnam 530045 Andhra Pradesh, INDIA

#1 Assistant Professor, Department of ECE, GITAM (deemed to be University), Gandhi nagar Rushikonda Visakhapatnam 530045 Andhra Pradesh, INDIA

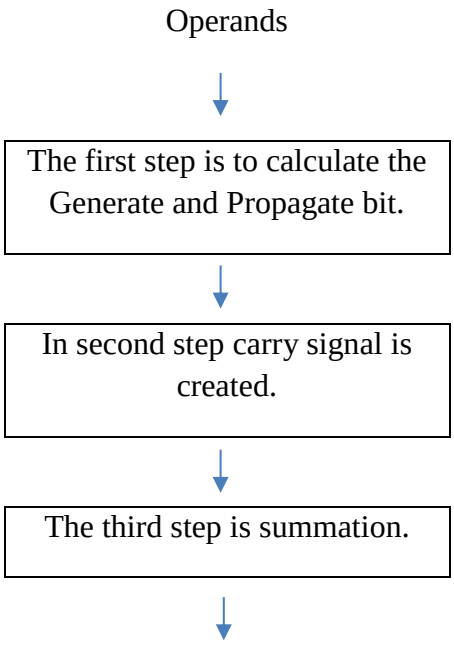
Abstract:

Every processor or data plane application relies on adders to function properly. In the development of high-performance processing units, high-speed adders with minimum power consumption are needed. Prefix that is similar to another Adder is a rapid adder used in a wide range of data processing applications. In this research, we show how to create and compare high-speed adders such the Brent Kung adder, Kogge stone adder, and Ladner Fischer adder. The proposed designs are validated by including 64-bit adder circuits. The proposed designs are evaluated in terms of space, latency, power consumption, and hardware overhead in this proposal. In this article, the number of slices (area occupied), power, and speed of the Kogge Stone Adder, Brent Kung Adder, and Ladner Fischer Adder were compared. The results are compared to existing fast adder designs to illustrate their effectiveness. Synthesis and simulation may be performed using the Xilinx ISE 14.7 version tool.

Keywords: **Verilog HDL, Parallel prefix adders like Brent Kung adder, Kogge stone adder and Ladner Fischer adder.**

1.INTRODUCTION

Since binary adders are the most important modules in computer arithmetic, they have been studied for 10 years. In the past, only a few conventional fast adders were available, such as the carry-skip adder, the carry-look-ahead adder, and the carry-select adder. In the design space, each fast adder has its own area-time tradeoff. Every rapid adder has its own ad-hoc structure. Parallel prefix adders may be made in a variety of methods, each with its own speed, size, and performance. The Brent-Kung approach is used to optimise the area. If we want to gather the most information in the quickest amount of time, we should employ the Kogge-Stone adder approach. When the system's fan out is high, the Lander and Fischer approach is applied. If we need to save space, we must employ a bitwise time constraint. Switching from one place to another takes less time using this, which improves performance and saves time. Electronic materials are quickly shrinking in everyday life, allowing them to be moved from one location to another. As a consequence, gadgets with low power consumption and great performance are in high demand. As a result, integrated circuits are the most widely used in the market. We'll need to utilise a simple bit addition approach to execute and optimise the device now. When constructing these algorithms, the size, performance, fan out, area, and complexity of the circuits are all taken into account. As a result, parallel prefix addition techniques are often regarded as the most exact form of total computation. Furthermore, the mechanism utilised in this approach is known as the Kogge-Stone method since it produces the least amount of fan out. When compared to others, the delay in terms of power and carrier creation is likewise rather short. The generate and propagate unit calculates the carry signal at each stage of parallel prefix addition. Figure 1 shows the three steps necessary to perform the addition.



Results

Figure 1: Parallel prefix addition stages

Parallel prefix addition is accomplished in three phases, as shown in the flowchart. We calculated two bits, the create bit and the propagate bit, in the first stage. The following stage involves using a prefix tree to produce a carry signal. Finally, the sum of two integers may be calculated using some of the equations previously discussed

2. OPERATIONS

There are primarily two operators used in parallel prefix addition. Black and grey operators are what they're called shown in Figure 2. Now, if we look at how both operators operate, we can see that the black operator uses two bits, the generate bit and the propagate bit, to create a single pair of generate and propagate bits.

The functions (1) may be constructed easily using the well-known formulae

Gi=Ai AND Bi and Pi=Ai XOR Bi.

Ci=Gi+(pi) AND C(i-1)

Si=Pi XOR Ci-1

Figure 2: operators

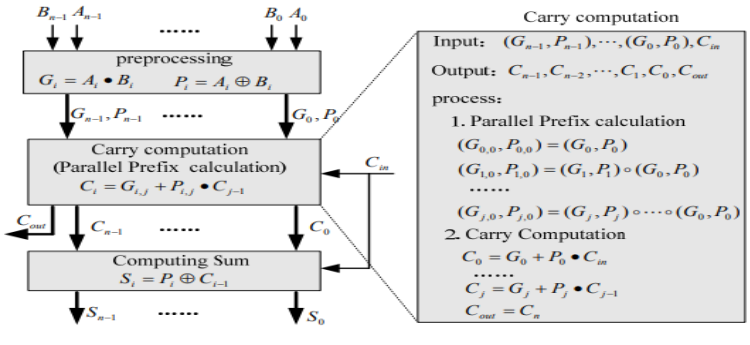


FIGURE 3: THE PARALLEL PREFIX ADDER'S WORK FLOW

3.Proposed Parallel Prefix Adders

Proposed 64 bit Kogge-Stone adder

Figure 4 illustrates the architecture of the 64-bit Efficient Kogge Stone adder [11]. The suggested 64-bit adder is built using a variety of valencies. We use design methodologies based on the valency of two, five, and two. Valency-3, valency-4, and valency-5 are included in the second step. The pipelined approach may provide a risk to the progress of the stated idea in order to increase the included requirements.

The notion of our suggested adder technique is identical to that of the standard Sparse Kogge-Stone adder, however the components are varying valency black cells. The carry is created for four bits, according to the design. This carry is assigned to a four-full-adder block that will propagate carry across it. When compared to the many adder techniques now in use, this design uses fewer gates.

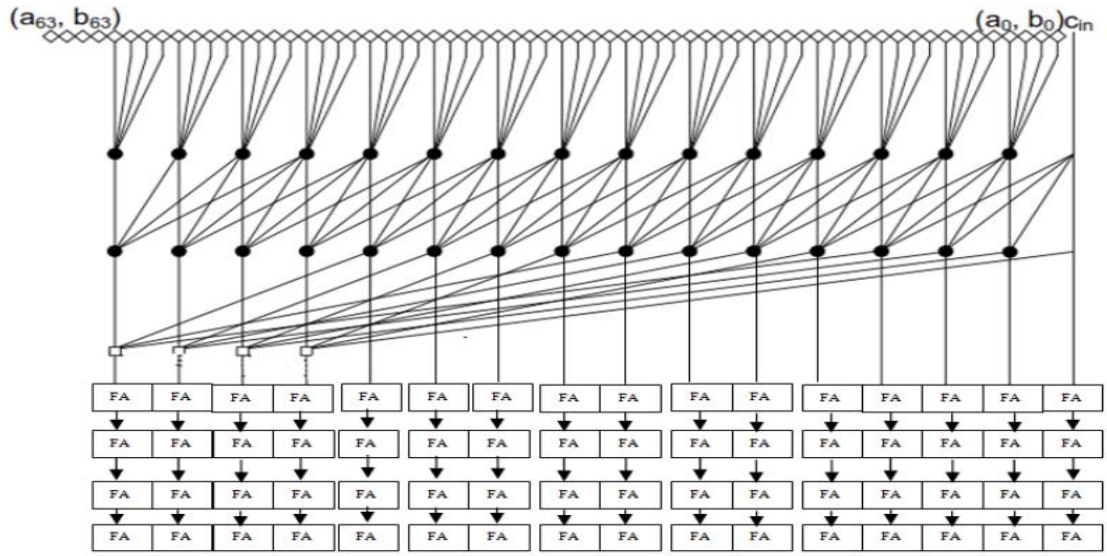


Figure 4: A 64-bit Kogge-Stone adder is proposed.

Brent Kung Adder

The efficient Brent-kung adder layout resembles a tree topology for high arithmetic performance, and it is a high-speed adder that concentrates on gate level logic. It is designed with a less number of gates. As a result, the architecture's latency and memory use are reduced. The input bits Ai and Bi use XOR and AND operations to produce and propagate information. The propagate and generates performs the black cell and grey cell operations, resulting in the carry Ci. That carry is XORed with the following bit's propagation, yielding sum.

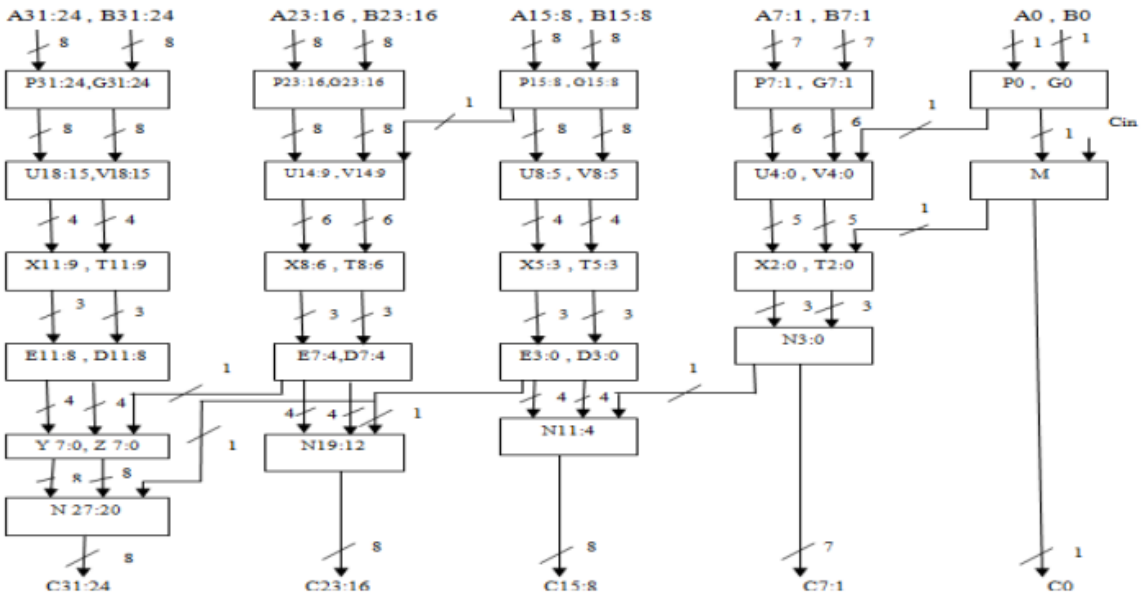


Figure 5: Efficient 64-bit Brent-kung Adder is a fictional character created by Brent-kung Adder

The operations' attributes are assessed in parallel with the trees overlapping, resulting in parallelization. For the operation of 16-bit addition, the design of the Efficient Brent-kung adder provides reduced latency and memory.

Figure 5 illustrates the architecture of the 64-bit Efficient Brent kung adder [10]. To determine the ans, or sum of N-bit values, the logical circuit uses multiple adders. A carry input (Cin) is the preceding bit carry output for each addition operation (Cout). The development of gadgets is invigorated by research on binary addition. The literature of parallel addition is described by several parallel prefix networks. Adders are used in large-scale integrated circuits and digital signal processors because of their rapid and precise performance.

LADNER-FISCHERADDER

In 1980, R. Ladner and M. Fischer created the Ladner-Fischer parallel prefix adder. The mediated structure of the Brent-Kung prefix tree and the Sklansky prefix tree is represented by this tree structure.

It computes prefixes for odd number bits in the first stage, then propagates the odd number bits to the even places in a second step. This adder has a shallow logic depth but a lot of fan-out.

n/2 is the maximum fan-out.

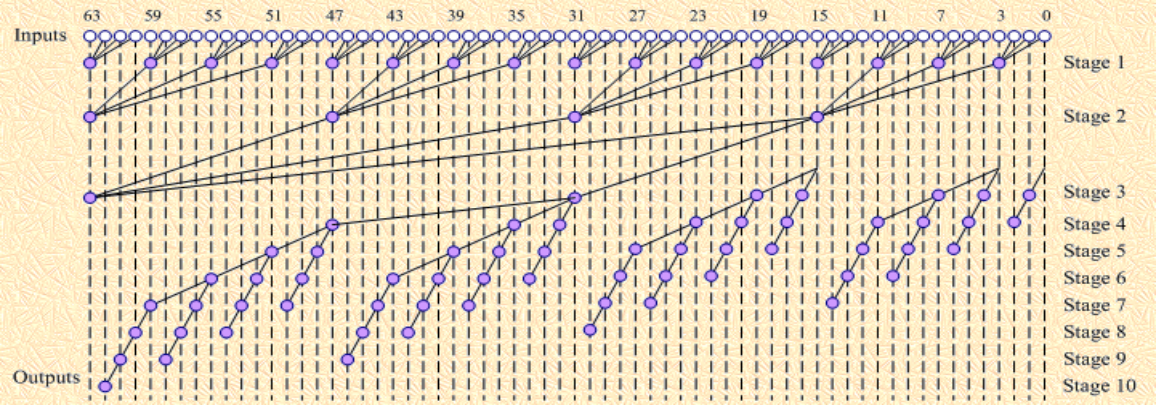


Figure 6: 64-bit Ladner-Fischer adder.

Figure 6 illustrates the architecture of the 64-bit Efficient Ladner fisher adder adder [12] utilising just grey cells is shown above, which minimises the latency and memory requirements when compared to prior adders. The adders that utilise the black cell do not need to wait for the carry from the grey cell in this adder. The grey cell's carry is presumed to be 0 and 1, and the addition operation is performed at the same time.

A multiplexer with two inputs, one from 0 cases and the other from 1 case, is used to pick the outcome from both cases. The carry formed from the grey cell, which was previously supposed to be 0 and 1, is the selection line used for selecting the appropriate output. The latency may be considerably decreased by utilising CSLA instead of RCA, but the memory is increased due to the higher use of logic gates. BEC is introduced to replace CSLA in order to reduce memory use.

Simulation Results:

In this simulation, we used all three kinds of 64-bit parallel prefix adders (KS, BK, and LF) that were previously mentioned. For synthesis, Verilog project navigator 14.2i was used to design all of the PPAs (Xilinx version). Area and delay are used to verify simulation findings. The waveforms and comparative findings for all three parallel prefix adders are also provided.

Figures 7, 8, and 9 depict the test bench waveform, time delay, and area usage of KSA, respectively. Figures 10, 11, and 12 depict the test bench waveform, time delay, and area usage of BKA, respectively. Figures 13, 14, and 15 depict the test bench waveform, time delay, and area usage of LFA, respectively.

Journal of Engineering Sciences

Vol 13, Issue 06, June/2022
ISSN NO:0377-9254

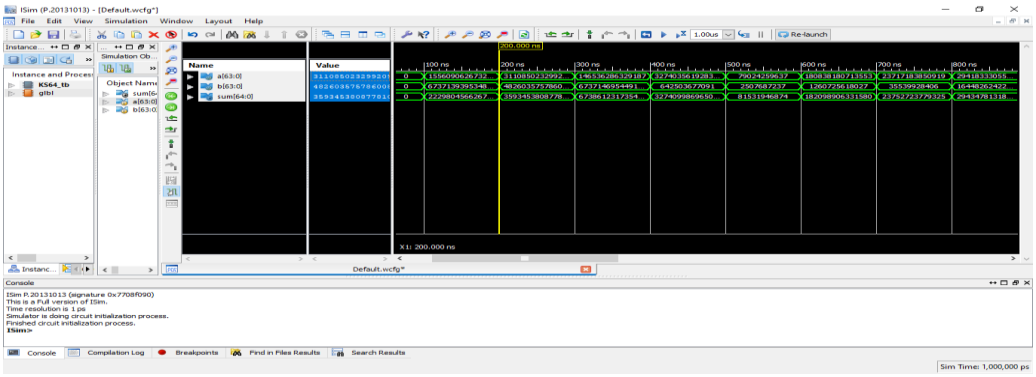


Figure 7: The kogge stone adder simulation result

```
--------------------------------------------------------------------------------
Delay:               10.369ns (Levels of Logic = 26)
  Source:            a<2> (PAD)
  Destination:       sum<64> (PAD)

  Data Path: a<2> to sum<64>
                                Gate     Net
    Cell:in->out      fanout   Delay    Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O             2    0.000    0.554  a_2_IBUF (a_2_IBUF)
     LUT6:I0->O            3    0.043    0.353  f31/out1 (g2<0>)
     LUT5:I3->O            3    0.043    0.353  f47/out1 (g2<0>)
     LUT5:I3->O            5    0.043    0.362  f50/out11 (f50/out1)
     LUT3:I1->O            2    0.043    0.347  f50/out2 (g3<3>)
     LUT5:I3->O            2    0.043    0.294  f66/out (g4<3>)
     LUT5:I4->O            2    0.043    0.347  f68/out1 (g4<5>)
     LUT5:I3->O            5    0.043    0.569  f70/out1 (g4<7>)
     LUT6:I0->O            2    0.043    0.293  f89/out (g5<7>)
     LUT5:I4->O            1    0.043    0.428  f93/out_SW0 (N36)
     LUT6:I3->O            2    0.043    0.347  f93/out (g5<11>)
     LUT5:I3->O            2    0.043    0.347  f95/out1 (g5<13>)
     LUT5:I3->O            4    0.043    0.357  f97/out1 (g5<15>)
     LUT5:I3->O            3    0.043    0.353  a63/out1 (g2<16>)
     LUT5:I3->O            3    0.043    0.353  a47/out1 (g3<16>)
     LUT5:I3->O            5    0.043    0.308  a50/out11 (a50/out1)
     LUT3:I2->O            2    0.043    0.348  a50/out2 (g3<19>)
     LUT5:I3->O            2    0.043    0.293  fa66/out (g4<19>)
     LUT5:I4->O            2    0.043    0.348  fa68/out1 (g4<21>)
     LUT5:I3->O            6    0.043    0.573  fa70/out1 (g4<23>)
     LUT6:I0->O            1    0.043    0.542  fg89/out1 (g5<23>)
     LUT5:I0->O            1    0.043    0.289  fg93/out4_SW0 (N46)
     LUT6:I5->O            2    0.043    0.347  fg93/out4 (g5<27>)
     LUT5:I3->O            2    0.043    0.347  fg95/out1 (g5<29>)
     LUT5:I3->O            1    0.043    0.279  fg97/out1 (sum_64_OBUF)
     OBUF:I->O             0.000          sum_64_OBUF (sum<64>)
    ----------------------------------------
    Total                      10.369ns (1.032ns logic, 9.337ns route)
                                       (10.0% logic, 90.0% route)
```

Figure 8: The 64-bit kogge stone adder's delay report

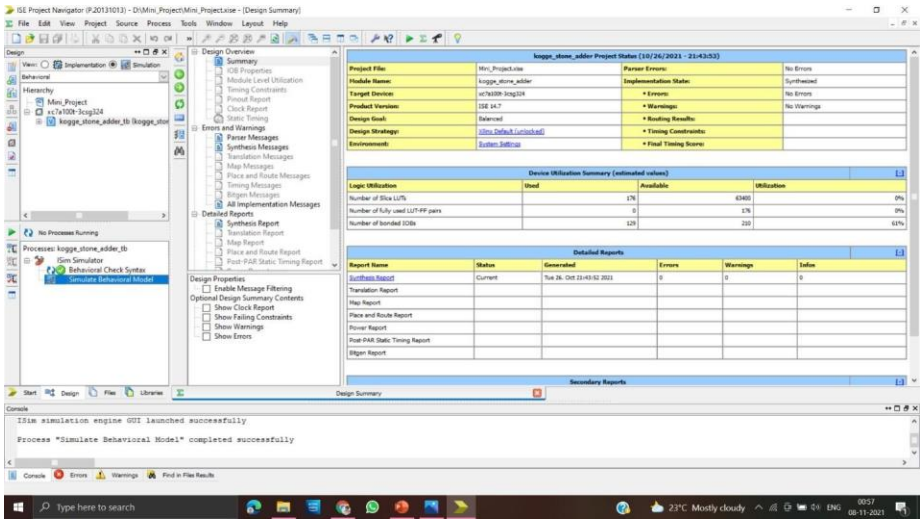


Figure 9: Area for Kogge stone adder

Journal of Engineering Sciences

Vol 13, Issue 06, June/2022
ISSN NO:0377-9254

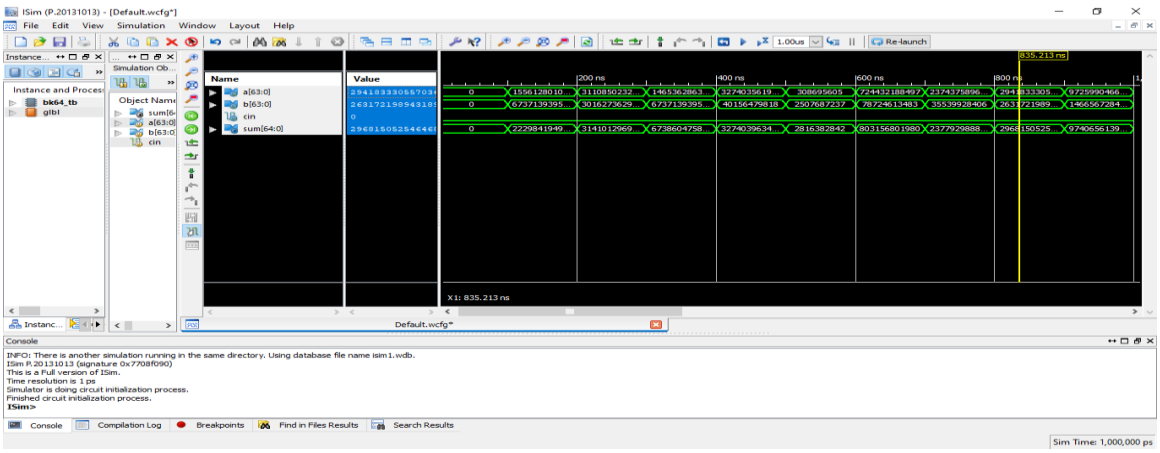
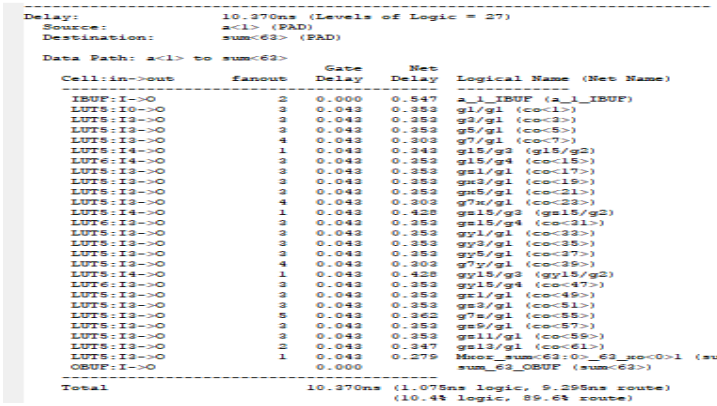


Figure 10: shows the outcome of the 64-bit Brent Kung Adder simulation.



```
--------------------------------------------------------------------------------
Delay:               10.370ns (Levels of Logic = 27)
  Source:            a<1> (PAD)
  Destination:       sum<63> (PAD)

  Data Path: a<1> to sum<63>
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O              2   0.000   0.547  a_1_IBUF (a_1_IBUF)
     LUT5:I0->O             3   0.043   0.353  g1/g1 (co<1>)
     LUT5:I2->O             3   0.043   0.353  g3/g1 (co<3>)
     LUT5:I2->O             3   0.043   0.353  g5/g1 (co<5>)
     LUT5:I3->O             4   0.043   0.303  g7/g1 (co<7>)
     LUT5:I4->O             1   0.043   0.343  g15/g3 (g15/g2)
     LUT6:I4->O             2   0.043   0.353  g15/g4 (co<15>)
     LUT5:I3->O             3   0.043   0.353  gr1/g1 (co<17>)
     LUT5:I3->O             3   0.043   0.353  gr3/g1 (co<19>)
     LUT5:I3->O             3   0.043   0.353  gr5/g1 (co<21>)
     LUT5:I2->O             4   0.043   0.303  g7x/g1 (co<23>)
     LUT5:I4->O             1   0.043   0.428  gr15/g3 (gr15/g2)
     LUT6:I3->O             2   0.043   0.353  gr15/g4 (co<31>)
     LUT5:I2->O             2   0.043   0.353  gy1/g1 (co<33>)
     LUT5:I3->O             3   0.043   0.353  gy3/g1 (co<35>)
     LUT5:I3->O             3   0.043   0.353  gy5/g1 (co<37>)
     LUT5:I2->O             4   0.043   0.303  g7y/g1 (co<39>)
     LUT5:I4->O             1   0.043   0.428  gy15/g3 (gy15/g2)
     LUT6:I3->O             3   0.043   0.353  gy15/g4 (co<47>)
     LUT5:I2->O             3   0.043   0.353  gr1/g1 (co<49>)
     LUT5:I3->O             3   0.043   0.353  gr3/g1 (co<51>)
     LUT5:I2->O             5   0.043   0.362  g7z/g1 (co<55>)
     LUT5:I3->O             3   0.043   0.353  gr9/g1 (co<57>)
     LUT5:I3->O             3   0.043   0.353  gr11/g1 (co<59>)
     LUT5:I3->O             2   0.043   0.347  gr13/g1 (co<61>)
     LUT2:I1->O             1   0.043   0.275  Mxor_sum<63:0>_xo<0>1 (sum_63_OBUF)
     OBUF:I->O              0   0.000          sum_63_OBUF (sum<63>)
    ----------------------------------------
    Total                     10.370ns (1.075ns logic, 9.295ns route)
                                       (10.4% logic, 89.6% route)
```

Figure 11: The 64-bit Brent Kung Adder's Delay Report

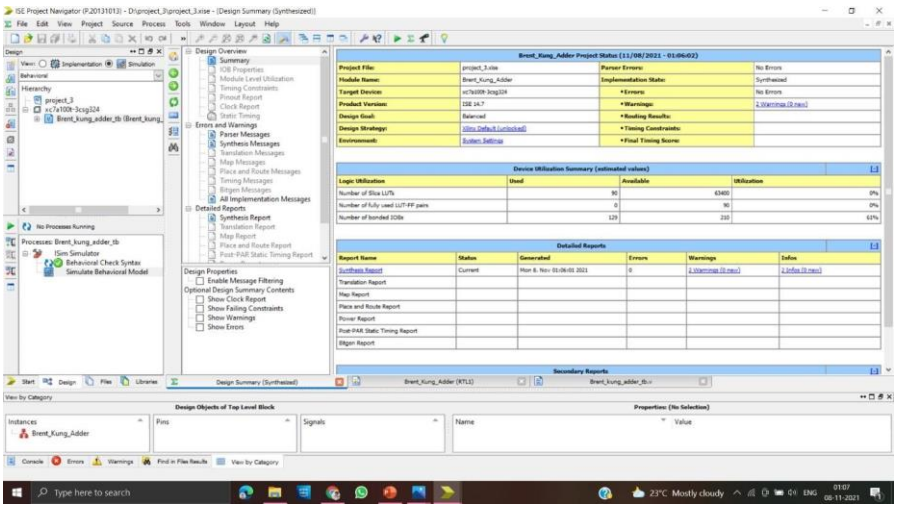


Figure 11: Area for Brent kung adder

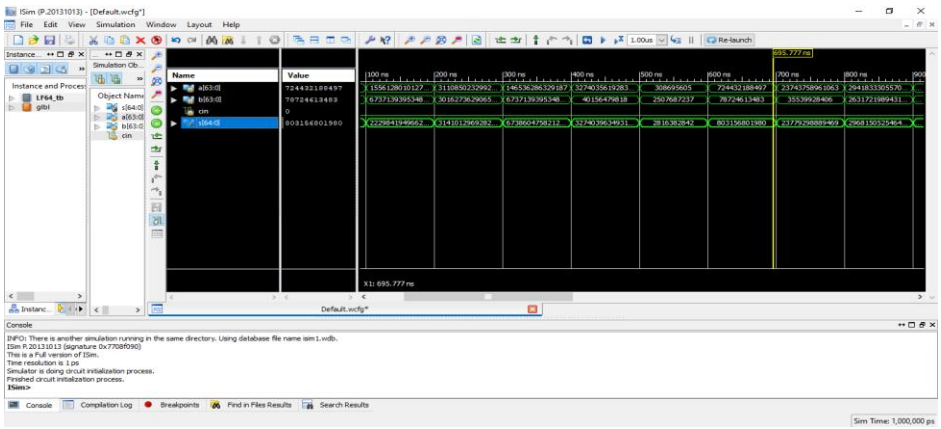


Figure 12: shows the outcome of the ladner fischer adder simulation.

```
--------------------------------------------------------------------------------
Delay:               7.652ns (Levels of Logic = 17)
  Source:            a<2> (PAD)
  Destination:       s<32> (PAD)

  Data Path: a<2> to s<32>
                                Gate     Net
    Cell:in->out      fanout   Delay    Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O             2    0.000    0.618  a_2_IBUF (a_2_IBUF)
     LUT6:I0->O            3    0.043    0.417  n1/gc17/G1 (n1/c<2>)
     LUT3:I1->O            3    0.043    0.534  n1/gc2/G1 (n1/c<3>)
     LUT5:I1->O            3    0.043    0.417  n1/gc4/G3 (n1/c<7>)
     LUT5:I3->O            3    0.043    0.362  n1/gc5/G2 (n1/c<9>)
     LUT5:I4->O            3    0.043    0.417  n1/gc6/G5 (n1/c<11>)
     LUT5:I3->O            4    0.043    0.422  n1/gc7/G1 (n1/c<13>)
     LUT5:I3->O            2    0.043    0.410  n1/gc8/G1 (n1/c<15>)
     LUT5:I3->O            3    0.043    0.362  n1/gc9/G2 (n1/c<17>)
     LUT5:I4->O            3    0.043    0.417  n1/gc10/G5 (n1/c<19>)
     LUT5:I3->O            3    0.043    0.417  n1/gc12/G11 (n1/gc12/G1)
     LUT5:I3->O            3    0.043    0.417  n1/gc12/G1 (n1/c<23>)
     LUT5:I3->O            3    0.043    0.417  n1/gc13/G1 (n1/c<25>)
     LUT5:I3->O            4    0.043    0.539  n1/gc14/G1 (n1/c<27>)
     LUT5:I1->O            2    0.043    0.500  n1/gc16/G3 (s1<32>)
     LUT3:I0->O            1    0.043    0.339  n2/sum<0>1 (s_32_OBUF)
     OBUF:I->O                  0.000            s_32_OBUF (s<32>)
    ----------------------------------------
    Total                       7.652ns (0.645ns logic, 7.007ns route)
                                        (8.4% logic, 91.6% route)
```

Figure 13: The Ladner Fischer adder delay report

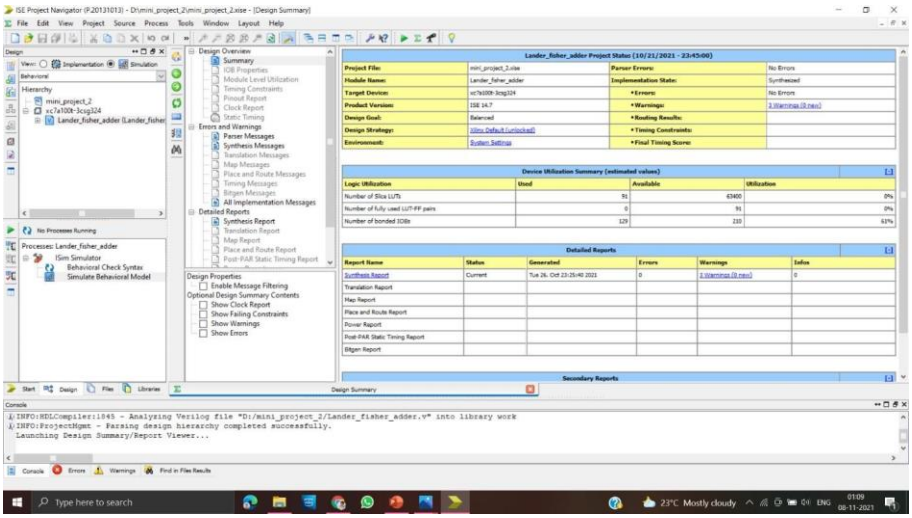


Figure 14: Area for Ladner Fisher adder

Table 1 shows the 64-bit parallel prefix adders in comparison.

Types of adder	Area(LUT's)	Delay(ns)
Kogge-stone adder	167	10.369
Brent-kung adder	96	10.370
Ladner-Fischer adder	94	7.652

4.CONCLUSION

In this project, we are implementing the design of fast adders like Parallel prefix adders like Brent Kung Adder, Kogge Stone Adder, and Ladner-Fischer Adder. And we evaluating the adders like Kogge Stone Adder, Brent Kung Adder, Kogge stone Adder & Brent Kung Adder will give the better performance in terms of Area, Delay . By comparing Ladner Fischer Adder with Brent Kung and Kogge stone, Brent Kung and Kogge stone Adder will give the better performance in terms of Area. By comparing Ladner Fischer Adder with brent kung adder, Ladner Fischer Adder will give the better performance in terms of area and delay. The synthesis results reveal that among the proposed adders, LFA is better in both area and delay in three adders. The proposed designs are synthesized and simulated in Xilinx ISE 14.7 version tool.

REFERENCES

[1] P. Patronik, S. J. Piestrak, "Design of RNS Reverse Converters with Constant Shifting to Residue Datapath Channels" Journal of Signal Processing Systems, 2017. https://doi.org / 10.1007/s11265-017-1238-6

[2] D. Kunjan, Shinde, S. Badiger, "Analysis and comparative study of 8-bit adder for embedded application" International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2015. https://doi.org /10.1109/iccicct.2015.7475290

[3] S. Gedam, Pravin Zode, and Pradnya Zode, "FPGA implementation of hybrid Han-Carlson adder" in 2nd International Conference on Devices Circuits and Systems (ICDCS), 2014. https://doi.org/10.1109/icdcsyst.2014.6926185

[4] H. Moqadasi, M. B. Ghazvani-Ghoushchi, "A new parallel prefix adder structure with efficient critical delay path and gradded bits efficiency in CMOS 90nm technology" in 23rd Iranian Conference on Electrical Engineering (ICEE), 2015. https://doi.org /10.1109/iraniancee.2015.7146426

[5] P. Kumar and M. Kiran, "Design of optimal fast adder" in International Conference on Advanced Computing and Communication Systems, 2013. https://doi.org /10.1109/icaccs.2013.6938692

[6] A. Zarandi, A. Molahosseini, M. Hosseinzadeh, S. Sorouri, S. Antão, and L. Sousa, "Reverse Converter Design via Parallel-Prefix Adders: Novel Components, Methodology, and Implementations" IEEE transactions on very large scale integration (VLSI) systems, 2014. https://doi.org/10.1109/tvlsi.2014.2305392

[7]. M.Indu & D. Somasundareswari, Energy Efficient Carry Skip Adder Using Skip Logic In Various Voltage Levels. International Journal of Management, Information Technology and Engineering (BEST: IJMITE) ISSN (P): 2348-0513, ISSN (E): 2454-471X, Vol. 3, Issue 10, Oct 2015, 145-150.

[8] N. Shanil Mohamed, T. Y. Siby, "16-bit Velocious Fault Lenient Parallel Prefix Adder" in International Conference on Electronics, Communication and Computational Engineering (ICECCE), 2014. https://doi.org/10.1109/icecce.2014.7086612

[9] A. Baliga, "Design of High Speed Adders Using CMOS and Transmission Gates in Submicron Technology: A Comparative Study" in Fourth International Conference on Emerging Trends in Engineering & Technology, Nov 2011. https://doi.org/ 10.1109/icetet.2011.25

[10] B. Mounika, and Dr. A. Raj Kumar "Design of Efficient 32-Bit Parallel Prefix Brent Kung Adder" Advances in Computational Sciences and Technology ISSN 0973-6107 Volume 10, Number 10 (2017) pp. 3103-3109 © Research India Publications http://www.ripublication.com.

[11] Gangula Thirupathi Reddy1, Dr. M. K Charan "64 Bit Pipelined Hybrid Sparse Kogge-Stone Adder Using Different Valance" International Journal of Research Studies in Science, Engineering and Technology Volume 2, Issue 12, December 2015, PP 22-28 ISSN 2349-4751 (Print) & ISSN 2349-476X (Online).

[12] M. Mahaboob Basha, Dr. K. Venkata Ramanaiah, Dr. P. Ramana Reddy, B. Lokeshwar Reddy "An efficient model for design of 64-bit High Speed Parallel Prefix VLSI adder" International Journal of Modern Engineering Research (IJMER) www.ijmer.com Vol. 3, Issue. 5, Jul - Aug. 2013 pp-2626-2630 ISSN: 2249-6645.