VLSI Implementation of Parallel Prefix Subtractor using Modified 2's Complement Technique and BIST Verification using LFSR Technique

Malti Kumari[1] , Vipin Gupta[2], Gaurav K Jindal[3]

[1]M.Tech Student Suresh Gyan Vihar University, Jaipur, Rajasthan, India

[2]Assistant Professor, Suresh Gyan Vihar University, Jaipur, Rajasthan, India

[3]Design Engineer, Priganik Technologies, Jaipur, Rajasthan, India

Abstract: ***Parallel prefix Subtractor is the most flexible and widely used for binary addition/subtraction. Parallel Prefix Subtractor is best suited for VLSI implementation. No any special parallel prefix Subtractor structures have been proposed over the past years intended to optimize area. This paper presents a new approach to new design the basic operators used in parallel prefix architectures which subtract the unit by using modified technique of 2's complement. Verification will also be done using LFSR technique so we don't need to apply any manually input to perform the subtraction process. We can analysis and create the difference in terms of area between parallel prefix Subtractor and BIST architecture of parallel prefix Subtractor. The number of multiplexers contained in each Slice of an FPGA is considered here for the redesign of the basic operators used in parallel prefix tree. The experimental results indicate that the new approach of basic operators make some of the parallel Prefix Subtractor architectures faster and area efficient.***

Keywords: Parallel Prefix Adder, 2s complement, Optimize Area, BIST architecture, LFSR Approach.

1. Introduction

1.1 Parallel-Prefixaddition Basics

The parallel prefix is the discriminating component in most computerized circuit plans including advanced sign processors (DSP) and microchip information way units. Thusly, broad exploration keeps on being centered around enhancing the force delay execution of the viper. In VLSI usage, parallel-prefix adders are known to have the best execution. Reconfigurable rationale, for example, Field Programmable Gate Arrays (FPGA) has been picking up in notoriety as of late in light of the fact that it offers enhanced execution regarding speed and control over DSP-based and chip based answers for some handy outlines including versatile DSP and information transfers applications and a noteworthy decrease being developed time and cost over Application Specific Integrated Circuit (ASIC) plans. The force point of interest is particularly imperative with the developing ubiquity of portable and versatile hardware, which make far reaching utilization of DSP capacities. Then again, in view of the structure of the configurable rationale and directing assets in Fpgas, parallel-prefix adders will have an alternate execution than VLSI executions. Specifically, most advanced Fpgas utilize a quick convey chain which advances the convey way for the basic Ripple Carry Adder (RCA). In this paper, the viable issues included in planning and actualizing tree-built adders in light of Fpgas are. A proficient testing technique for assessing the execution of these adders is examined. A few tree-based viper structures are actualized and described on a FPGA and contrasted and the Ripple Carry Adder (RCA) and the Carry Skip Adder (CSA). At last, a few conclusions and proposals for enhancing FPGA plans to empower better tree-based viper execution are given.

2. Related Work

Xing and Yu noted that postpone models and expense investigation for viper outlines produced for VLSI engineering don't delineate to FPGA plans . They thought about the outline of the swell convey viper with the convey lookahead, convey skip, and convey select adders on the Xilinx 4000 arrangement Fpgas. Just an enhanced type of the convey skip viper performed better than the swell convey snake when the viper operands were over 56 bits. An investigation of adders actualized on the Xilinx Virtex II yielded comparable results [9]. In [10], the creators considered a few parallel prefix adders actualized on a Xilinx Virtex 5fpga. It is observed that the basic RCA viper is better than the parallel prefix outlines on the grounds that the RCA can exploit the quick convey chain on the FPGA. Kogge-Stone The Kogge-Stone tree [22] Figures 1- 5 accomplishes both log2n stages and fan-out of 2 at each one stage. This takes on at the expense of long wires that must be directed between stages. The tree additionally contains more PG cells; while this may not affect the range if the viper design is on a consistent lattice, it will expand power utilization. Regardless of these expense, Kogge-Stone viper is for the most part used for wide adders because it shows the lowest delay among other structures.

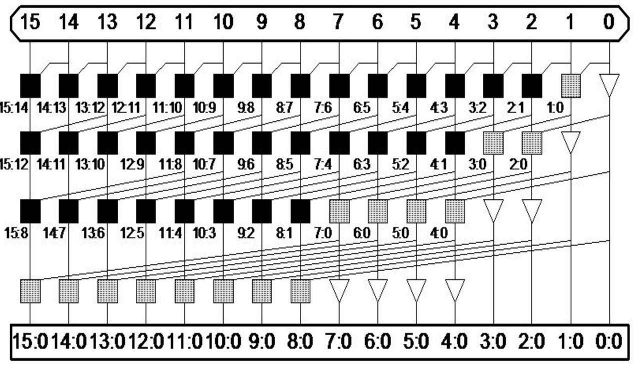


Figure 1: The Parallel Prefix addition

An alternate convey tree snake known as the crossing tree convey look ahead (CLA) viper is likewise inspected [6]. Like the inadequate Kogge-Stone viper, this outline ends with a 4-bit RCA. As the FPGA utilizes a quick convey chain for the RCA, it is fascinating to contrast the execution of this snake and the scanty Kogge-Stone and consistent Kogge-Stone adders. Likewise of enthusiasm for the spreading over tree CLA is its testability characteristics [7].

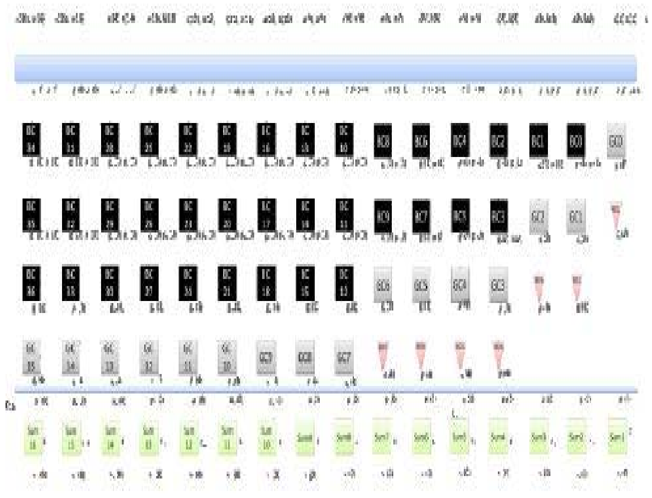


Figure 2: 128-bit Kogge-Stone adder

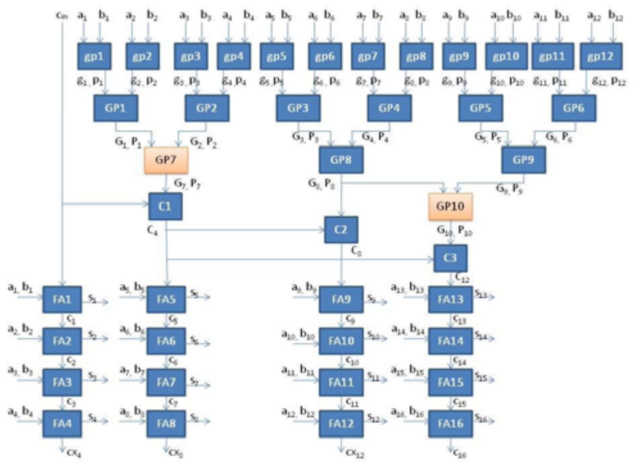


Figure 3: Spanning Tree Carry Lookahead Adder (16 bit)

3. New Approch 2'S Compliment

Let us consider the multiplier data A to be used with the negative partial product factors. To calculate the 2's complement first is to inverse all the bits of the data A denoting them as Abar. Now perform "Exclusive OR" (XOR) operation on Abar(0) with 1'b1, Abar(1) xor Abar(0), Abar(2) xor Abar(1) and so on till a 1'b0 is found while traversing the data bits A(i). Once 1'b0 is arrived keep the remaining bits as it is without any change.

Lets us consider an example where A=10101000, then 2's complement of A be denoted as A2_c_bar, then
Step 1: Abar=01010111.
Step 2:
A2_c_bar (0) = 1 xor 1 = 0
A2_c_bar (1) = 1 xor 1 = 0
A2_c_bar (2) = 1 xor 1 = 0
A2_c_bar (3) = 1 xor 0 = 1
A2_c_bar (4) = A'4 = 1
A2_c_bar (5) = A'5 = 0
A2_c_bar (6) = A'6 = 1
A2_c_bar (7) = A'7 =0

4. Parallel Prefix Subtractor

Given figure 4 represents the subtraction part using parallel prefix Subtractor using modified approach of 2's compliment. Output analysis of this approach will be explained in detail in this paper in result section.

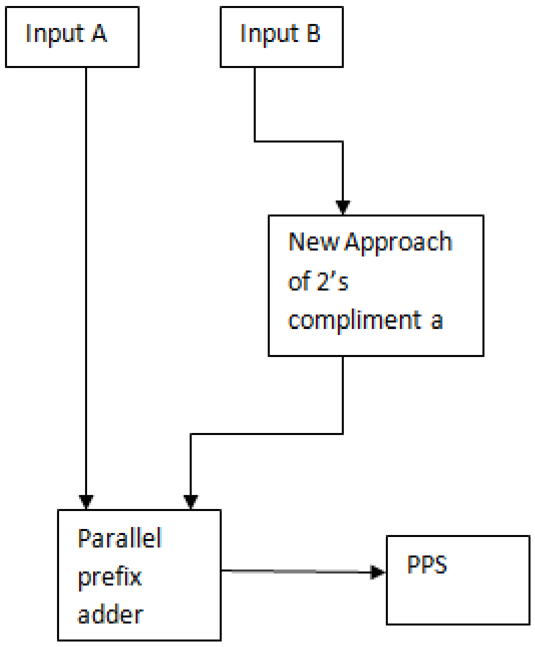


Figure 4: Parallel Prefix Subtractor

5. BIST Approach

Built in self test architecture, which analysis the on chip verification of Circuit under Test (CUT). We do not need to apply any input for any input drivers. Figure 5 & Figure 6 represent the logical architecture bist capability using LFSR technique for any circuit in vlsi design.

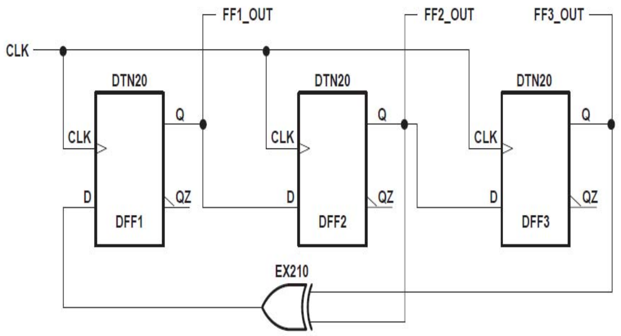


Figure 5: LFSR Circuit

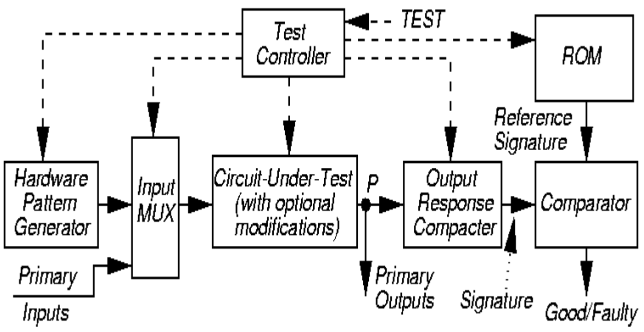


Figure 6: BIST approch

6. BIST approach of Parallel prefix Subtractor (PPS)

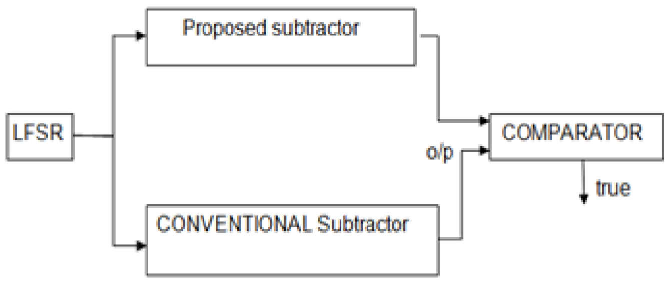
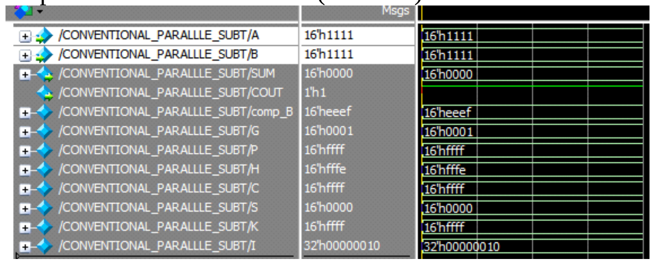


Figure 7: BIST with PPS

7. Result and Analysis

7.1 Conventional parallel prefix Subtractor

Input a= 0001000100010001(4’h1111)
Input b= 0001000100010001(4’h1111)
Output: =0000000000000000(4’h0000)



```
/CONVENTIONAL_PARALLLE_SUBT/A        16'h1111       16'h1111
/CONVENTIONAL_PARALLLE_SUBT/B        16'h1111       16'h1111
/CONVENTIONAL_PARALLLE_SUBT/SUM      16'h0000       16'h0000
/CONVENTIONAL_PARALLLE_SUBT/COUT     1'h1
/CONVENTIONAL_PARALLLE_SUBT/comp_B   16'heeef       16'heeef
/CONVENTIONAL_PARALLLE_SUBT/G        16'h0001       16'h0001
/CONVENTIONAL_PARALLLE_SUBT/P        16'hffff       16'hffff
/CONVENTIONAL_PARALLLE_SUBT/H        16'hfffe       16'hfffe
/CONVENTIONAL_PARALLLE_SUBT/C        16'hffff       16'hffff
/CONVENTIONAL_PARALLLE_SUBT/S        16'h0000       16'h0000
/CONVENTIONAL_PARALLLE_SUBT/K        16'hffff       16'hffff
/CONVENTIONAL_PARALLLE_SUBT/I        32'h00000010   32'h00000010
```

Figure 8: Functional simulation waveform

RTL VIEW of Conventional Design

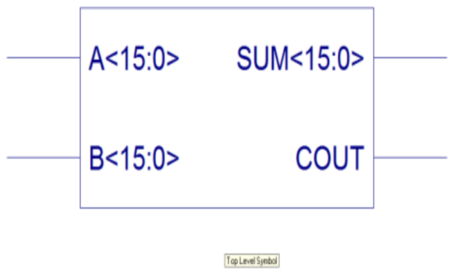


Figure 9: RTL view of conventional (Top design)

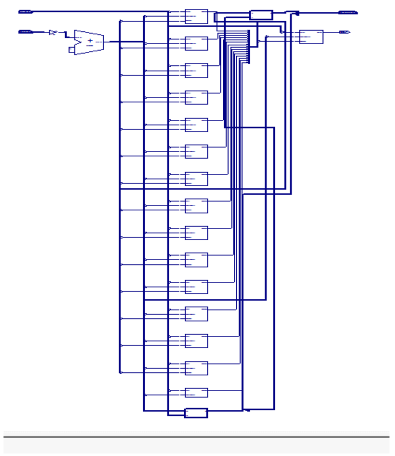
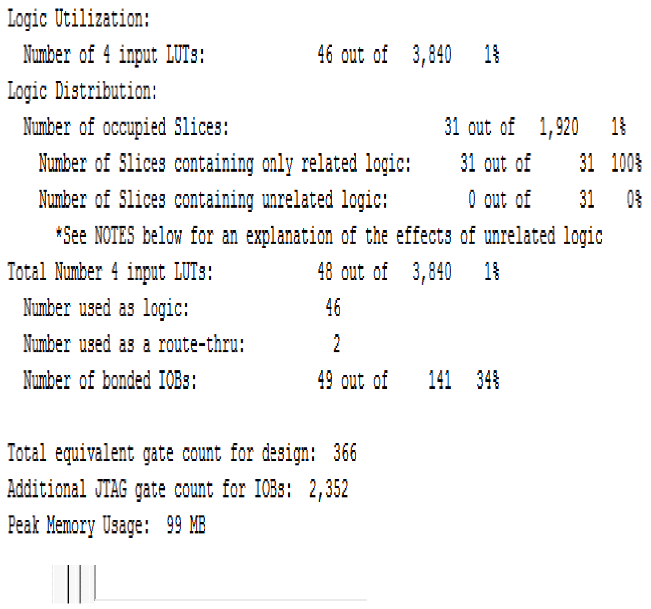


Figure 10: RTL view of conventional (Internal design)

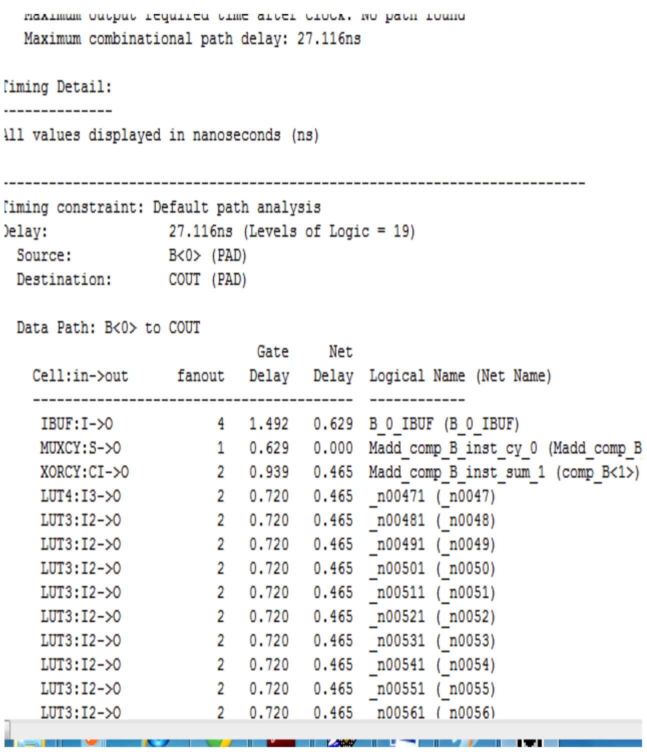
Area analysis



```
Logic Utilization:
  Number of 4 input LUTs:              46 out of   3,840    1%
Logic Distribution:
  Number of occupied Slices:                         31 out of   1,920    1%
    Number of Slices containing only related logic:    31 out of      31  100%
    Number of Slices containing unrelated logic:        0 out of      31    0%
      *See NOTES below for an explanation of the effects of unrelated logic
Total Number 4 input LUTs:             48 out of   3,840    1%
  Number used as logic:                 46
  Number used as a route-thru:           2
  Number of bonded IOBs:               49 out of     141   34%

Total equivalent gate count for design:  366
Additional JTAG gate count for IOBs:  2,352
Peak Memory Usage:  99 MB
```

Figure 11: Area description of conventional design

Timing analysis

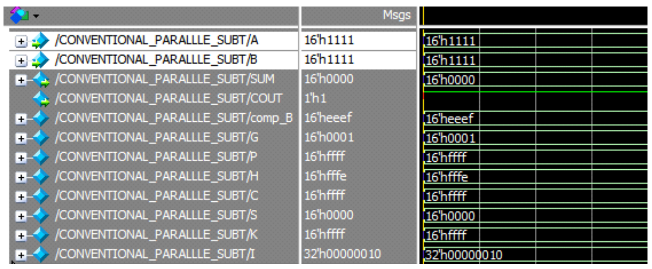


```
   Maximum output required time after clock: No path found
   Maximum combinational path delay: 27.116ns

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

================================================================================
Timing constraint: Default path analysis
Delay:               27.116ns (Levels of Logic = 19)
  Source:            B<0> (PAD)
  Destination:       COUT (PAD)

  Data Path: B<0> to COUT
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O             4   1.492   0.629  B_0_IBUF (B_0_IBUF)
     MUXCY:S->O            1   0.629   0.000  Madd_comp_B_inst_cy_0 (Madd_comp_B
     XORCY:CI->O           2   0.939   0.465  Madd_comp_B_inst_sum_1 (comp_B<1>)
     LUT4:I3->O            2   0.720   0.465  _n00471 (_n0047)
     LUT3:I2->O            2   0.720   0.465  _n00481 (_n0048)
     LUT3:I2->O            2   0.720   0.465  _n00491 (_n0049)
     LUT3:I2->O            2   0.720   0.465  _n00501 (_n0050)
     LUT3:I2->O            2   0.720   0.465  _n00511 (_n0051)
     LUT3:I2->O            2   0.720   0.465  _n00521 (_n0052)
     LUT3:I2->O            2   0.720   0.465  _n00531 (_n0053)
     LUT3:I2->O            2   0.720   0.465  _n00541 (_n0054)
     LUT3:I2->O            2   0.720   0.465  _n00551 (_n0055)
     LUT3:I2->O            2   0.720   0.465  _n00561 (_n0056)
```

Figure 12: Timing description of conventional design

7.2 Proposed parallel prefix Subtractor

Input a= 0001000100010001(4’h1111)
Input b= 0001000100010001(4’h1111)
Output: =0000000000000000(4’h0000)



```
                                          Msgs
/CONVENTIONAL_PARALLLE_SUBT/A        16'h1111        16'h1111
/CONVENTIONAL_PARALLLE_SUBT/B        16'h1111        16'h1111
/CONVENTIONAL_PARALLLE_SUBT/SUM      16'h0000        16'h0000
/CONVENTIONAL_PARALLLE_SUBT/COUT     1'h1
/CONVENTIONAL_PARALLLE_SUBT/comp_B   16'heeef        16'heeef
/CONVENTIONAL_PARALLLE_SUBT/G        16'h0001        16'h0001
/CONVENTIONAL_PARALLLE_SUBT/P        16'hffff        16'hffff
/CONVENTIONAL_PARALLLE_SUBT/H        16'hfffe        16'hfffe
/CONVENTIONAL_PARALLLE_SUBT/C        16'hffff        16'hffff
/CONVENTIONAL_PARALLLE_SUBT/S        16'h0000        16'h0000
/CONVENTIONAL_PARALLLE_SUBT/K        16'hffff        16'hffff
/CONVENTIONAL_PARALLLE_SUBT/I        32'h00000010    32'h00000010
```

Figure 13: Waveform of Proposed Design

RTL VIEW of proposed design

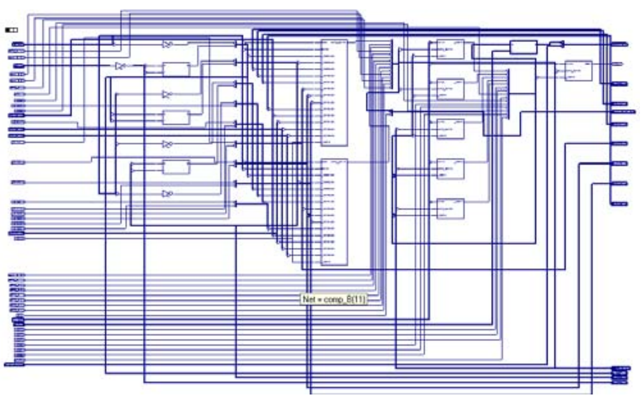
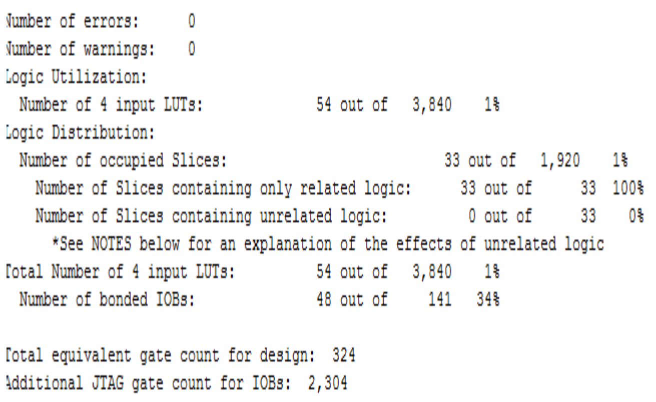


Figure 14: Bottom level design

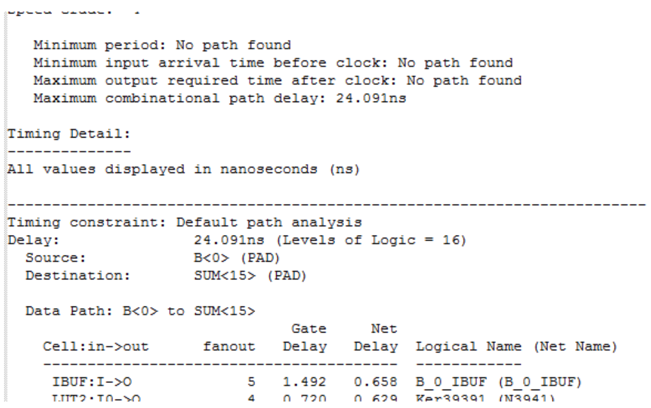
Area Analysis



```
Number of errors:       0
Number of warnings:     0
Logic Utilization:
  Number of 4 input LUTs:              54 out of   3,840    1%
Logic Distribution:
  Number of occupied Slices:                         33 out of   1,920    1%
    Number of Slices containing only related logic:    33 out of      33  100%
    Number of Slices containing unrelated logic:        0 out of      33    0%
      *See NOTES below for an explanation of the effects of unrelated logic
Total Number of 4 input LUTs:          54 out of   3,840    1%
  Number of bonded IOBs:               48 out of     141   34%

Total equivalent gate count for design:  324
Additional JTAG gate count for IOBs:  2,304
```

Figure 15: Bottom level design

Timing analysis



```
   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: 24.091ns

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

================================================================================
Timing constraint: Default path analysis
Delay:               24.091ns (Levels of Logic = 16)
  Source:            B<0> (PAD)
  Destination:       SUM<15> (PAD)

  Data Path: B<0> to SUM<15>
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O             5   1.492   0.658  B_0_IBUF (B_0_IBUF)
     LUT2:I0->O            4   0.720   0.629  Ker39391 (N3941)
```

Figure 16: Waveform of Proposed Design

BIST result

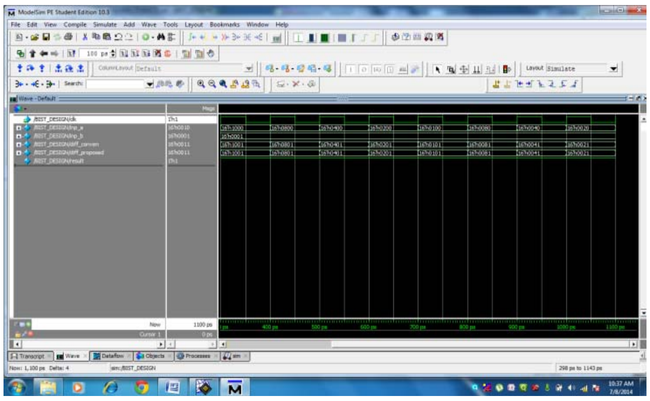

ModelSim PE Student Edition 10.3
File Edit View Compile Simulate Add Wave Tools Layout Bookmarks Window Help
Wave - Default
Msgs
Now
Cursor 1

Figure 17: Waveform of bist architecture

Waveform of bist architecture: result is done and set on 1 while expected output and proposed output is same

8. Comparison Table of PPS (Parallel Prefix Subtractor)

Table 1: Comparison Table

Design	*Area(gate Count)*	*Delay*
Conventional	366	27.116ns
Proposed	324	24.091ns

9. Conclusion

In this paper we approach Parallel prefix Subtractor using prefix algorithm. Proposed design based on modified 2's compliment method whereas area reduced by approx 12% and delay reduced by 13%. BIST architecture also introduced for on chip verification using LFSR technique. In future this work can be extend to multiplication part for modulo arithmetic operation.

References

[1] Skalansky Conditional sum additions logic, IRE Transactions, Electronic Computers, vol. EC – 9, pp, 226 -231, June 1960
[2] Kogge P and Stone H. A parallel algorithm for the efficient solution of a general class Recurrence relations, IEEE Trans. Computers, Vol.C-22, No.8, pp 786-793, Aug.1973
[3] Brent R and Kung H. A regular layout for parallel adders. IEEE Trans, computers, Vol.C-31, No.3, pp 260-264, March1982
[4] Koren Computer arithmetic algorithms, A K Peters, Ltd. 2002
[5] 5 R. P. Brent and H. T. Kung A regular layout for parallel adders, IEEE Trans. Computers, Vol. 31, No.3, pp. 260–264, March 1982. Srinivasasamanoj. R et al., International Journal of Wireless Communications and Network Technologies, 1(1), August-September 2012, 4 - 99@ 2012, IJWCNT All Rights Reserved
[6] P. M. Kogge and H. S. Stone, A parallel algorithm for the efficient solution of a general class of recurrence equations, IEEE Trans. Computers**, Vol. 22, No. 8 pp. 786–793, August 1973**
[7] J. Sklansky Conditional sum addition logic, IRE Trans. Electron. Compute, Vol. 9, No. 6, pp. 226–231, 1960
[8] J. Liu, S. Zhou, H. Zhu, and C.-K. Cheng. An algorithmic approach for generic parallel adders, in ICCAD, November 2003, pp. 734–730
[9] R. Zimmermann. Non-heuristic optimization and synthesis of parallel-prefix adders, International Workshop on Logic and Architecture Synthesis, December 1996, pp. 123–132
[10] T. Matsunaga and Y. Matsunaga. Timing-constrained area minimization algorithm for parallel prefix adders, IEICE Transactions on Fundamentals, Vol. E90-A, No. 12, pp. 2770–2777, December 2007
[11] T. Matsunaga, S. Kimura, and Y. Matsunaga.
[12] Power-conscious syntheses of parallel prefix adders under bitwise timing constraints, Proc. the Workshop on Synthesis And System Integration of Mixed Information technologies(SASIMI), Sapporo, Japan, October 2007, pp. 7–14
[13] R. Bryant. Graph-based algorithms for Boolean function manipulation, IEEE Trans. Computers, Vol. 35, No. 8, pp. 677–691, August 1986.
[14] M. Pedram. Power minimization in ic design:
[15] Principles and applications, ACM Trans. on Design Automation of Electronic Systems, Vol. 1, No. 1, pp. 3–5, January 1996.