DESIGN OF FIXED-WIDTH BOOTH MULTIPLIER USINGDATA SCALING METHOD FOR DSP COMPUTATIONS

A. NAGAMALLI [1], P. GNANA SAI LAKSHMAN [2], MUDDETI. SRIKANTH [2], BUDDABATHUNI. UMA SUPRAJA [2], TALLURU. HARSHAVARDHAN [2], MANNEM. PRADEEP [2]

[1] *Associate Professor, ECE department, PBR Visvodaya Institute Of Technology & Science, Kavali, Andhra Pradesh, India.*
[2] *UG students, ECE department, PBR Visvodaya Institute Of Technology & Science, Kavali, Andhra Pradesh, India.*

ABSTRACT

Truncation errors lead to the adverse effects in obtaining the high-speed performance with in the DSP. Computation circuits to overcome these issues Data Scaling Technology (DST) is used. The difference between a truncated value and the actual value is known as Truncation error. This truncation error can be reduced by doing repeated iteration using iterative methods.

Data Scaling Technology (DST) is used in a low error Fixed –Width Booth Multipler (FWBM) to reduce truncation in DSP circuits. The proposed DST architecture uses the redundant bits of the multiplicand to more efficiently obtain bits for low FWBMs. More specifically, a Data Scaling Technology (DST) for use in a low-error Fixed-Width Booth Multiplier (FWBM) to reduce truncation errors. The proposed DST uses the redundant bits of the multiplicand to more efficiently obtain bits low-error FWBMs. To reduce the truncated partial products, which are used to estimate the compensation bias. At the cost of increased computations in the FWBM. The accuracy of the FWBM can be improved by adding the proposed DST circuit with in the architecture by reduced area overhead and enhanced accuracy.

Expected results indicate that the use of the proposed DST increases the accuracy of low error FWBMs with a small area overhead by estimating delay and speed. The effectiveness of the proposed method is designed using Verilog HDL program in Xilinx 14.7 environment.

1. INTRODUCTION:

The complexity of VLSI is being designed and used today makes the manual approach to design impractical. Design automation is the order of the day. With the rapid technological developments in the last two decades, the status of VLSI technology is characterized by the following

A steady increase in the size and hence the functionality of the ICs:

• A steady reduction in feature size and hence increase in the speed of operation as well as gate or transistor density.

• A steady improvement in the predictability of circuit behavior.

• A steady increase in the variety and size of software tools for VLSI design.

The above developments have resulted in a proliferation of approaches to VLSI design.VLSI DESIGN FLOW:

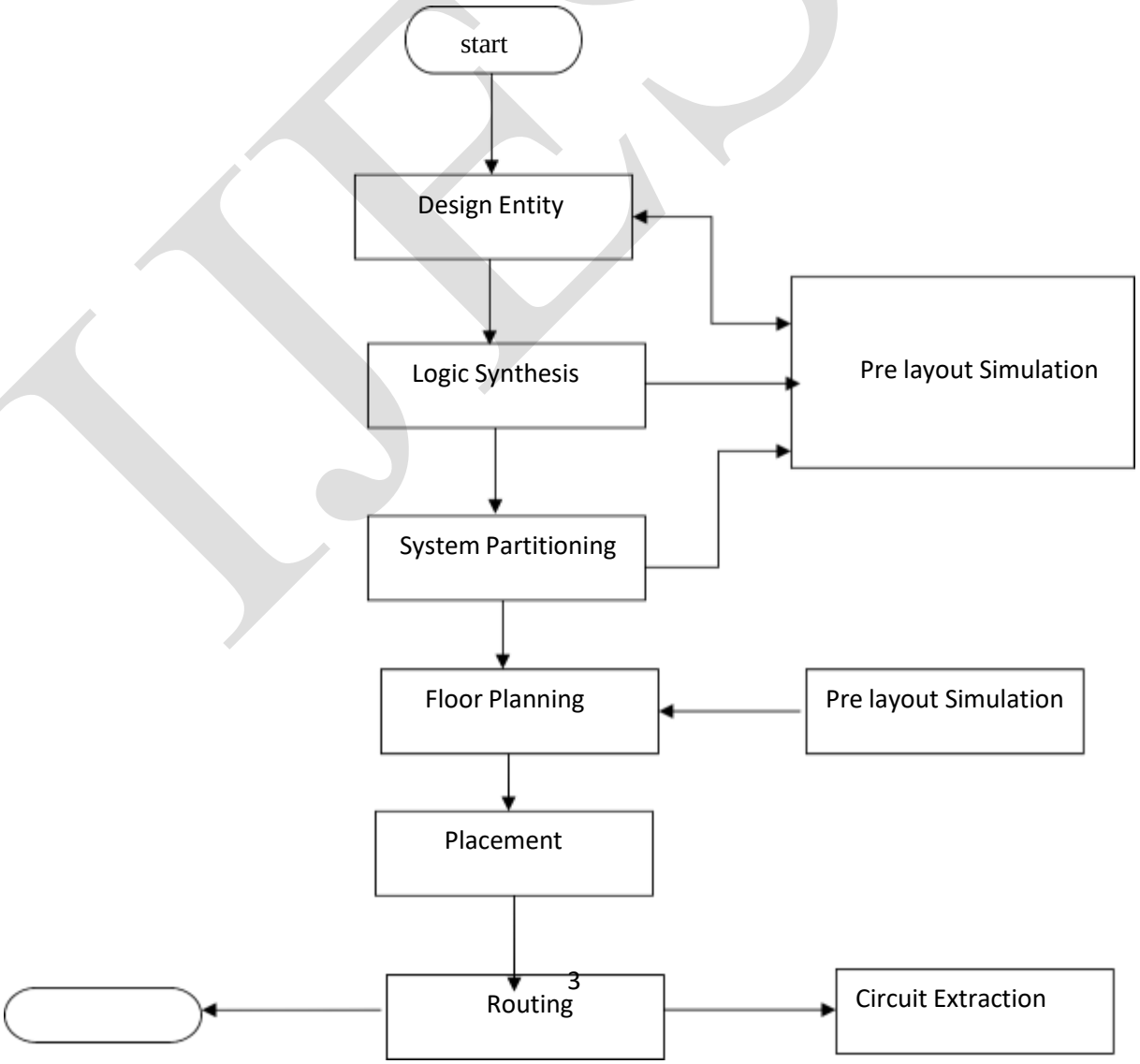

Fig 1.1 VLSI Design Flow

2. MULTIPLERS

Multipliers play an important role in today's digital signal processing and various other applications. With advances in technology, many researchers have tried and are trying to design multipliers which offer either of the following design targets

1. High speed,
2. Low power consumption,
3. Regularity of layout and hence less area or even combination of them in one multiplier thus making them suitable for various high speed,
4. Low power and compact VLSI implementation.

The common multiplication method is "add and shift" algorithm. In parallel multipliers number of partial products to be added is the main parameter that determines the performance of the multiplier. To reduce the number of partial products to be added, with increasing parallelism, the amount of shifts between the partial products and intermediate sums to be added will increase which may result in reduced speed, increase in silicon area due to irregularity of structure and also increased power consumption due to increase in interconnect resulting from complex routing. On the other hand, "serial-parallel" multipliers compromise speed to achieve better performance for area and power consumption.

The selection of a parallel or serial multiplier actually depends on the nature of application. In this lecture we introduce the multiplication algorithms and architecture and compare them in terms of speed, area, power and combination of these metrics. AND gates are used to generate the Partial Products (PP). If the multiplicand is N-bits and the Multiplier is M-bits then there is N* M partial product.

2.1 HISTORY OF MULTIPLIERS

The early computer systems had what are known as Multiply and Accumulate units to perform multiplication between two binary unsigned numbers. The Multiply and Accumulate unit was the simplest implementation of a multiplier. The basic block diagram of such a system is given below fig.2.1.

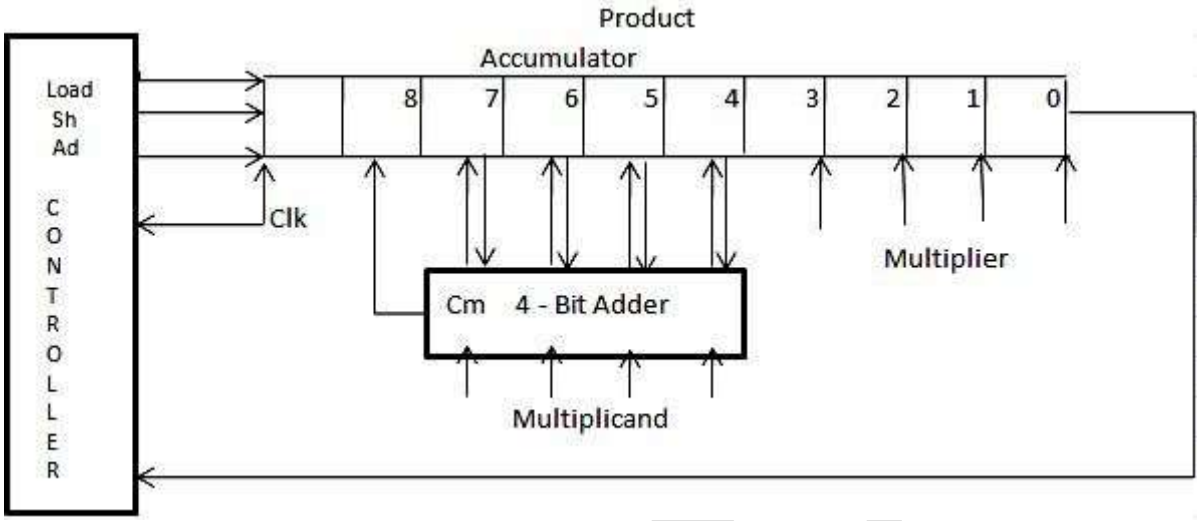

Fig.2.1 Multiplier Block Diagram

2.1.1 IMPLEMENTATION:

The MAC unit requires a 4-bit multiplicand register, 4-bit multiplier register, a 4-bit full adder and an 8-bit accumulator to hold the product. In the figure above the product register holds the 8-bit result. In a typical binary multiplication, based on the multiplier bit being processed, either zero or the multiplicand is shifted and then added.

Following the same process would require an 8-bit adder. Instead, in the above design the contents of the product register are shifted right by one position and the multiplicand is added 5 to the contents. This multiply and accumulate block is also known by the name serial-parallel multiplier as the multiplier bits are processed serially but the addition takes place in parallel. The second type of multiplier is the parallel array multiplier.

The desire to speed up the rate at which the output is generated resulted in the development of this category of multiplier. In a serial-parallel multiplier discussed above, it takes one clock cycle to process one bit of the data input at any given time. Therefore, when working on an N-bit input it would take at least N clock cycles to generate the final output. In a parallel array multiplier the result is obtained as soon as inputs are presented to the multiplier. This is mainly because of the use of AND array structure to compute

the partial product terms. Once the partial product terms are generated the only delay in generating the output is contributed by the adders which sum the partial product terms column wise to generate the result. The figure below represents a parallel array multiplier with N=8 bit inputs.

In Figure 2.1.1 block A stands for an AND gate. Block AHA stands for AND GATE and HALF ADDER structure and AFA stands for AND GATE and FULL ADDER structure. FA stands for full adder. The partial product terms are added along the diagonal (as shown by the arrows along the diagonal) to generate the product bits P. The carry from each block is passed onto to the next column and this is shown by vertical arrows. The gate level representation of an AND gate, HALF ADDER and FULL ADDER is given below fig. 2.1.1.

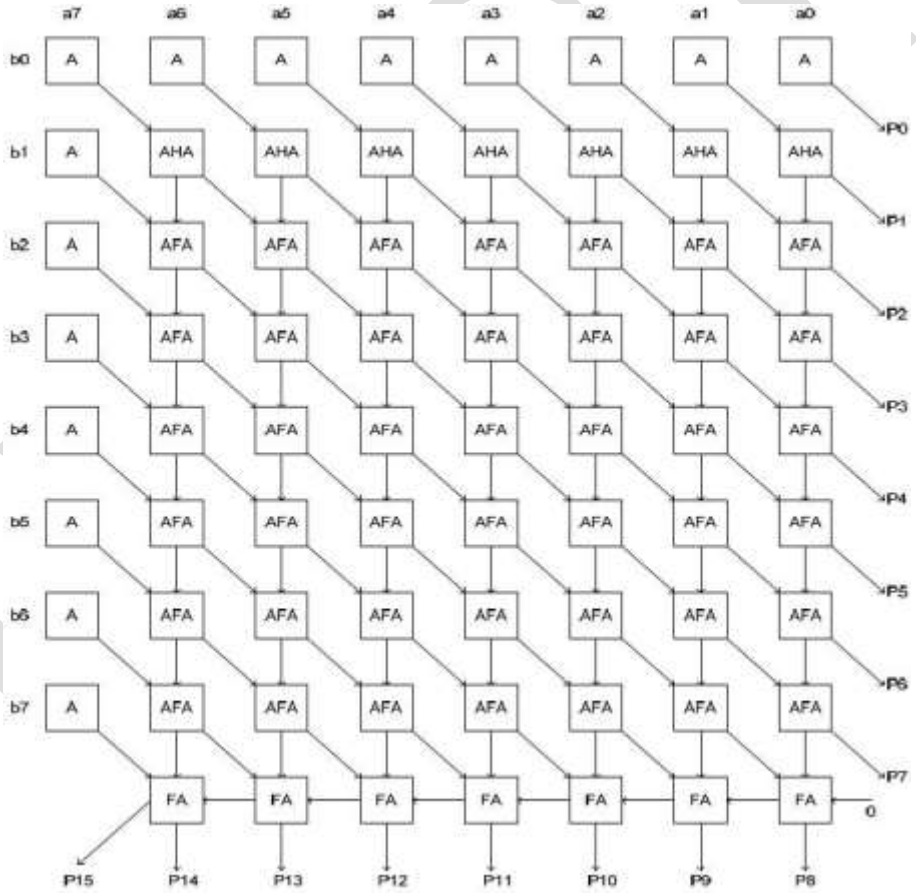

Fig.2.1.1 Parallel array multiplier for N=8 bits

.

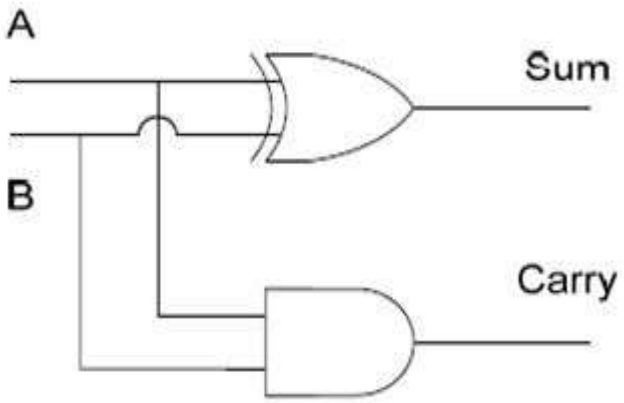

Fig.2.2 Gate level implementation of a HALF ADDER.

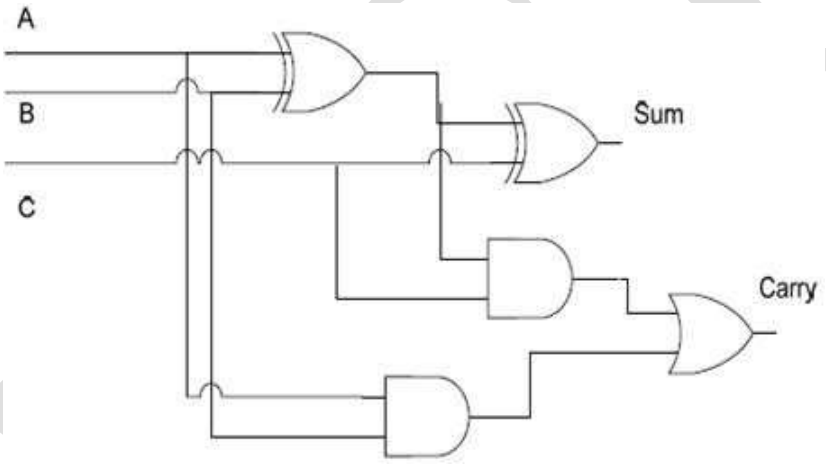

Fig.2.3 Gate level implementation of a FULL ADDER.

3.1 INTRODUCTION FIELD PROGRAMMABLE GATE ARRAYS(FPGA'S)

Before the advent of programmable logic, custom logic circuits were built at the board level using standard components, or at the gate level in expensive application-specific (custom) integrated circuits. The FPGA is an integrated circuit that contains many (64 to over 10,000) identical logic cells that can be viewed as standard components. Each logic cell can independently take on any one of a limited set of personalities. The individual cells are interconnected by a matrix of wires and programmable switches. A

user's design is implemented by specifying the simple logic function for each cell and selectively closing the switches in the interconnect matrix. The array of logic cells and interconnects form a fabric of basic building blocks for logic circuits. Complex designs are created by combining these basic blocks to create the desired circuit.

Field Programmable Gate Arrays (FPGAs) are highly flexible, reprogrammable logic devices that leverage advanced CMOS manufacturing technologies, similar to other industry-leading processors and processor peripherals. Like processors and peripherals, FPGAs are fully user programmable. For FPGAs, the program is called a configuration bit stream, which defines the FPGA's functionality. The bit stream loads into the FPGA at system power-up or upon demand by the system. The process whereby the defining data is loaded or programmed into the FPGA is called configuration. Configuration is designed to be flexible to accommodate different application needs and, wherever possible, to leverage existing system resources to minimize system costs.

Similar to microprocessors, FPGAs optionally load or boot themselves automatically from an external nonvolatile memory device. Alternatively, similar to microprocessor peripherals, Spartan-3 generation FPGAs can be downloaded or programmed by an external “smart agent”, such as a microprocessor, DSP processor, microcontroller, PC, or board tester. In either case, the configuration data path is either serial to minimize pin requirements or byte-wide for maximum performance or for easier interfaces to processors or to byte-wide Flash memory. Similar to both processors and processor peripherals, FPGAs can be reprogrammed, in system, on demand, an unlimited number of times. After configuration, the FPGA configuration bit stream is stored in highly robust CMOS configuration latches (CCLs). Although CCLs are reprogrammable like SRAM memory, CCLs are designed primarily for data integrity, not for performance. The data stored in CCLs is written only during configuration and remains static unless changed by another configuration event

A field-programmable gate array (FPGA**)** is an integrated circuit designed to be configured by the customer or designer after manufacturing hence "field-programmable". The FPGA configuration is generally specified using a hardware description language (HDL), similar to that used for an application-specific integrated circuit (ASIC). FPGAs can be used to implement any logical function that an ASIC could perform. The ability to update the functionality after shipping, partial re-configuration of the portion of the

design and the low non-recurring engineering costs relative to an ASIC design (notwithstanding the generally higher unit cost), offer advantages for many applications.

FPGAs contain programmable logic components called "logic blocks", and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together “somewhat like a one-chip programmable breadboard. Logic blocks can be configured to perform complex combinational functions, or merely simple logic gates like AND,XOR. In most FPGAs, the logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory.

3.2 ARCHITECTURE

The FPGA is an array or island-style FPGA. It consists of an array of logic blocks and routing channels. Two I/O pads fit into the height of one row or the width of one column, as shown Fig 3.1. All the routing channels have the same width (number of wires). Each circuit must be mapped into the smallest square FPGA that can accommodate it. For example, a circuit containing 14 logic blocks and 10 I/O pads would be mapped into an FPGA consisting of a 4x4 array of logic blocks. Note that three of the twenty benchmark circuits used in the FPGA challenge (big key, des, and dsip) are pad-limited in this FPGA architecture.

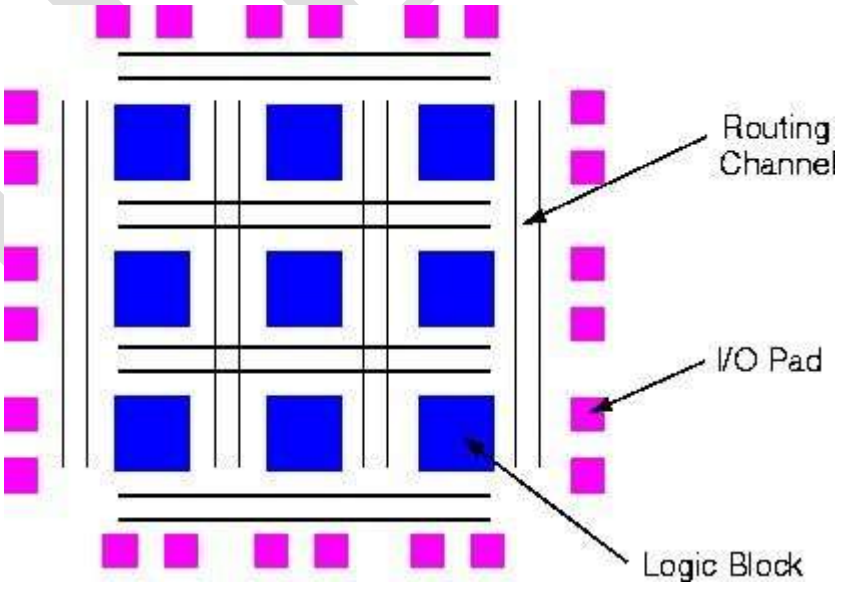

Fig 3.1: FPGA structure

The FPGA logic block consists of a 4-input look-up table (LUT), and a flip flop, as shown in Fig 3.2. There is only one output, which can be either the registered or the unregistered LUT output. The logic block has four inputs for the LUT and a clock input. Since the clock is normally routed via a special-purpose dedicated routing network in commercial FPGAs. That is, you can completely ignore the clock net, since it is assumed to be routed on a special global network.

4. PROPOSED DST FWBM SYSTEM

4.1 INTRODUCTION

DIGITAL multipliers are widely used in arithmetic units of microprocessors, multimedia and digital signal processors. Many algorithms and architectures have been proposed to design high-speed and low-power multipliers [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13]. A normal binary (NB) multiplication by digital circuits includes three steps. In the first step, partial products are generated; in the second step, all partial products are added by a partial product reduction tree until two partial product rows remain. In the third step, the two partial product rows are added by a fast carry propagation adder. Two methods have been used to perform the second step for the partial product reduction. A first method uses four-two compressors, while a second method uses redundant binary (RB) numbers [5], [6]. Both methods allow the partial product reduction tree to be reduced at a rate of 2:1.

Multipliers are widely used in digital signal processing (DSP) techniques, such as discrete cosine transform (DCT) and fast Fourier transformThe need for a simple but accurate fixedwidth multiplier for use in DSP systems has been a topic of discussion for many years.Two of the most popular types of fixed-width multipliers are the Baugh–Wooley (BW) array multiplier and the Booth multiplier.The Booth encoder reduces the number of truncated partial products, and therefore, the accuracy of Booth multipliers is higher than that of BW multipliers

4.2 DATA SCALING TECHNOLOGY IN FWBM

The proposed DST-FWBM consists of a DST circuit and a low-error FWBM, which could be a GPEB FWBM, ACPE FWBM, PACS FWBM, or MLCP FWBM. The DST circuit is implemented using $(2L-1)$ 2-to 1 MUXs with DSb = 1. The FWBM is surrounded by the DST circuit and consists of a Booth encoder, a carry-save adder (CSA)

tree, and a parallel prefix adder. The CSA, which consists of either full adders or half adders, adds the partial products from the MP, TP_{ma}, and the estimated TP_{mi}in all the low-error FWBMs tested. Finally, the high-speed parallel prefix adder calculates the products P_d, and the final results P_q are obtained by using the DST circuit.

4.3ARCHITECTURE

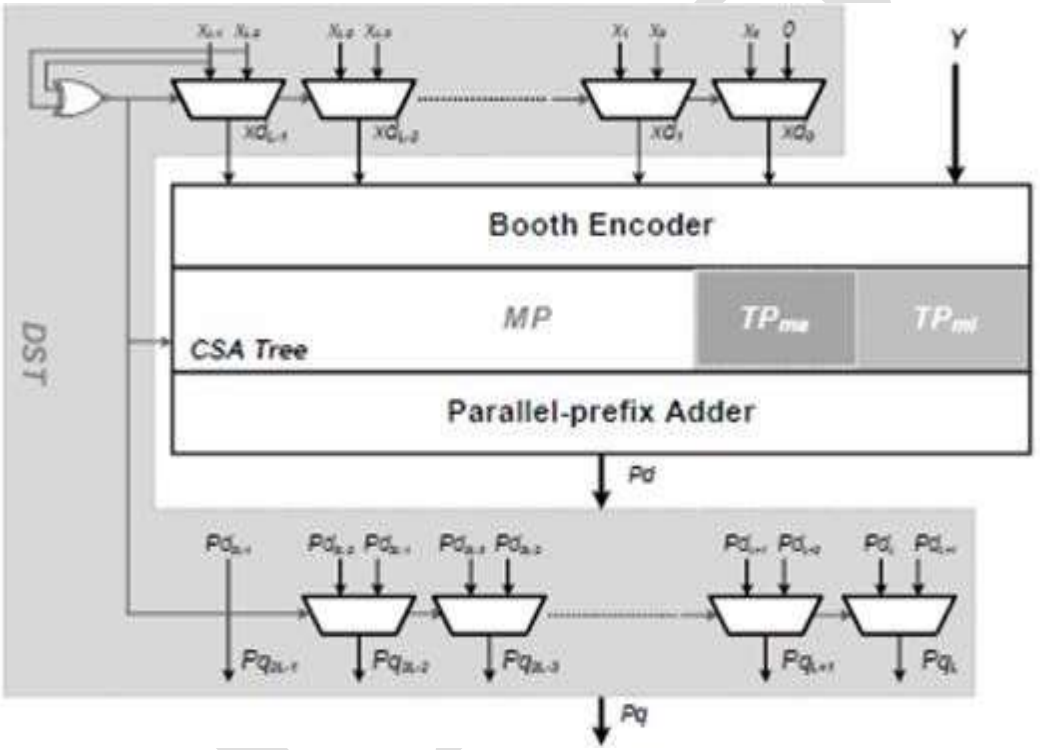

Fig :4.1 Architecture of proposed DST-FWBM with DSb = 1.

In Proposed Architecture there are three stages they are, Booth Encoder, Carry Select Adder Tree(CSA Tree), & Parallel-prefix Adder.In booth encoder we do binary multiplication then carry and sum will produce,carry sends to CSA Tree.In CSA Tree there will be many full adders and half adders then carry is process in CSA Tree then send to parallel-prefix adder. Then parallel-prefix adder produces product Pd, and Pq is the result output.

4.4 BOOTH'S ALGORITHM FLOWCHART

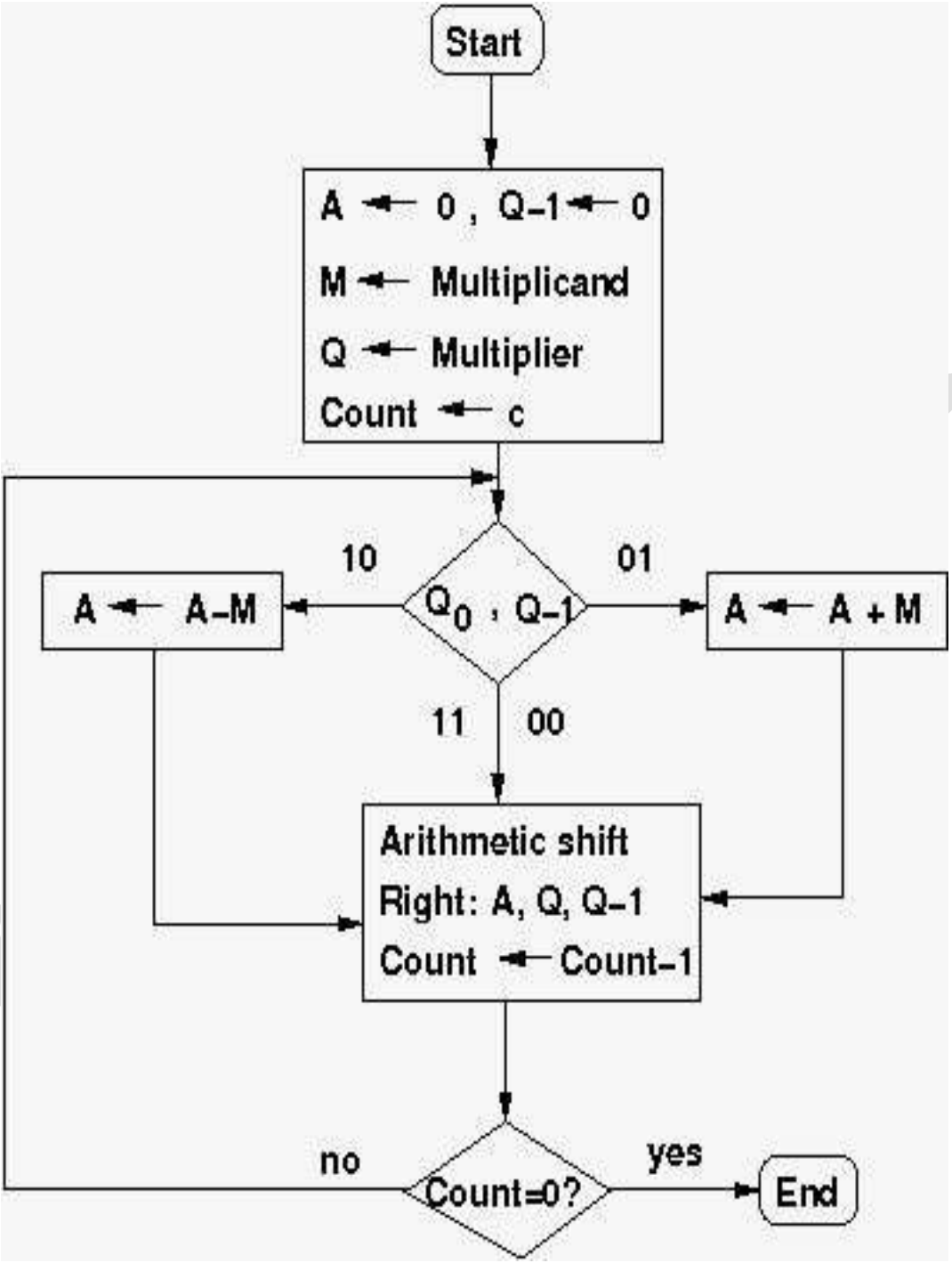

FIG:4.2 Flow Chart of Booth's Algorithm.

4.4.1 EXAMPLE FOR BOOTH'S ALGORITHM

Qn	Qn+1	M= (0111) M' + 1 = (1001) & Operation	AC	Q	Qn+1	SC
1	0	Initial	0000	0001	0	4
		Subtract (M' + 1)	1001			
			1001			
		Perform Arithmetic Right Shift Operations (ashr)	1100	1001	1	3
1	1	Perform Arithmetic Right Shift Operations (ashr)	1110	0100	1	2
0	1	Addition (A + M)	0111			
			0101	0100		
		Perform Arithmetic Right Shift Operations	0010	1010	0	1
0	0	Perform Arithmetic Right Shift Operations	0001	0101	0	0

4.5 ADVANTAGES

- It improves the accuracy of FWBMs
- Delay will be less
- Area will be decrease
- Power consumption is low.

5. SIMULATIONS AND RESULTS

5.1 RTL Schematic of DST in FWBM

- After Synthesis, you need to click on “Synthesized design”, note - this is applicable only when your design is Synthesized!, you can schematic option below “Synthesized design”,
- Synthesized design schematic are technology dependent - The inference of the components includes,
  - LUTs, FFs, Carry-Chain, Muxes,
  - Block RAMs, DSPs,
  - Clocking elements - BUFG, MMCM,
  - IO elements - IBUF, OBUF, …
- After Implementation you can click to open “Implemented design” under which you can schematic option,
- This is similar to Synthesized design on inference but the logic optimized one

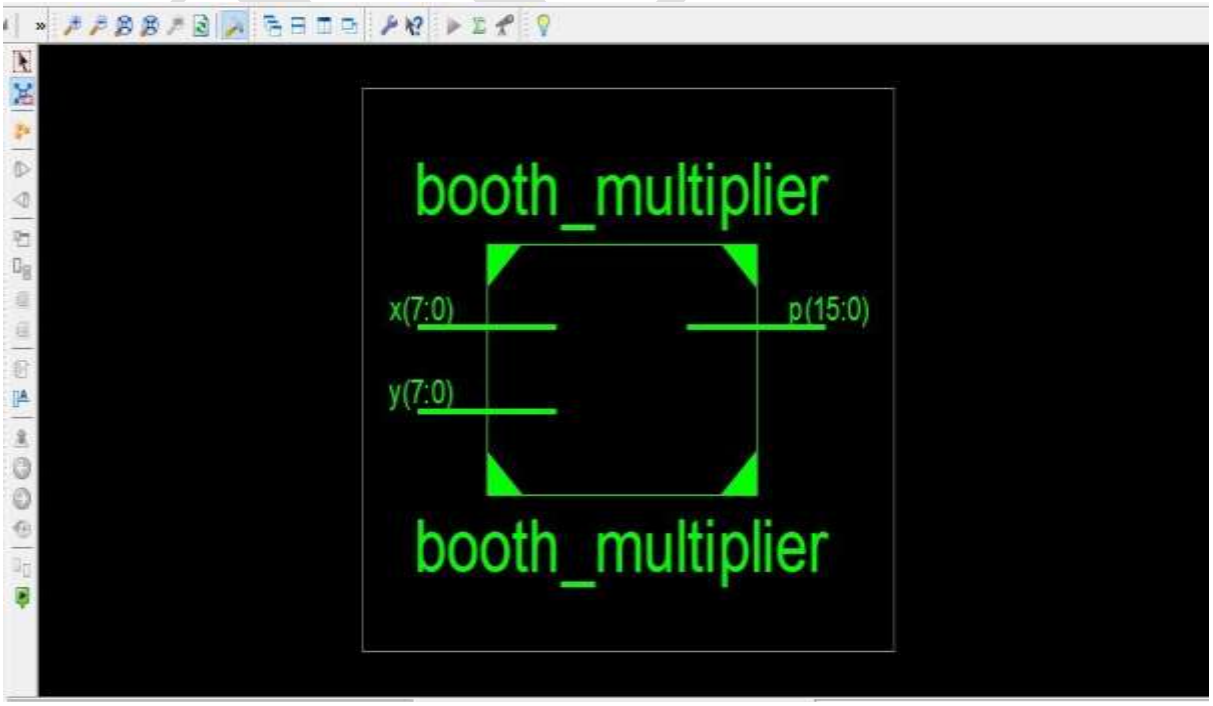

Fig: -5.1 RTL schematic

5.2 Technology Schematic of DST in FWBM

Technology schematic means the equivalent blocks to u r logic in the library. after clicking on the technology schematic u will have LUT (Look up table) all those are well defined in the library after dumping the program into the FPGA kit, all these elements in the technology schematic will be placed and routed. In the proposed system there were 24 look up tables have been used.

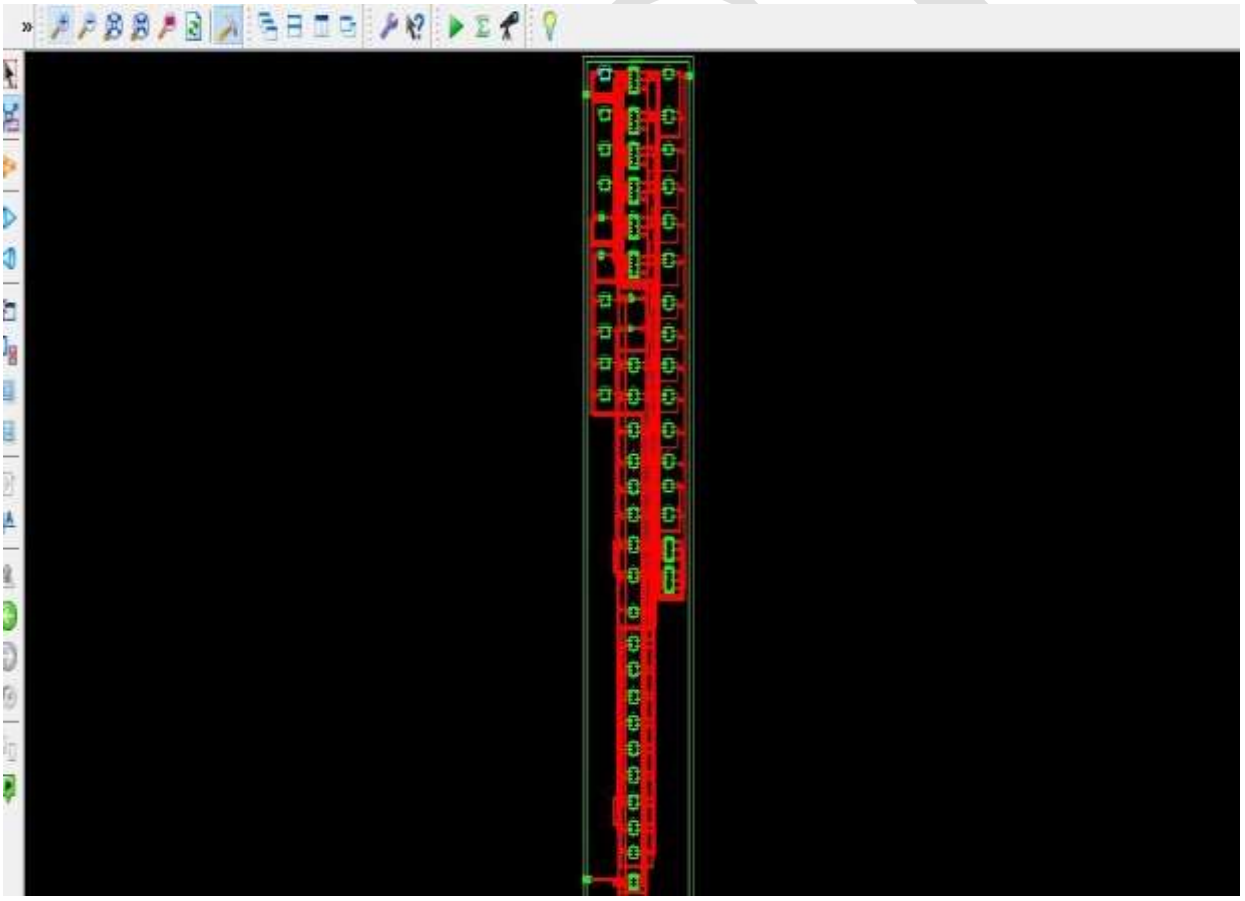

Fig: - 5.2 Technology schematic

5.3 SIMULATION RESULTS

5.3.1 (a) SIMULATION RESULT OF DST FWBM WITH SIGNED VALUES

ISE Simulator is an application that integrates with Xilinx ISE to provide simulation and testing tools. Two kinds of simulation are used for testing a design: functional simulation and timing simulation. Functional simulation is used to make sure that the logic of a design is correct. 53 The inputs were given in unsigned decimal format by changing from binary by changing the radix of the inputs. The inputs were given by applying force constants of given inputs a, b, cin.

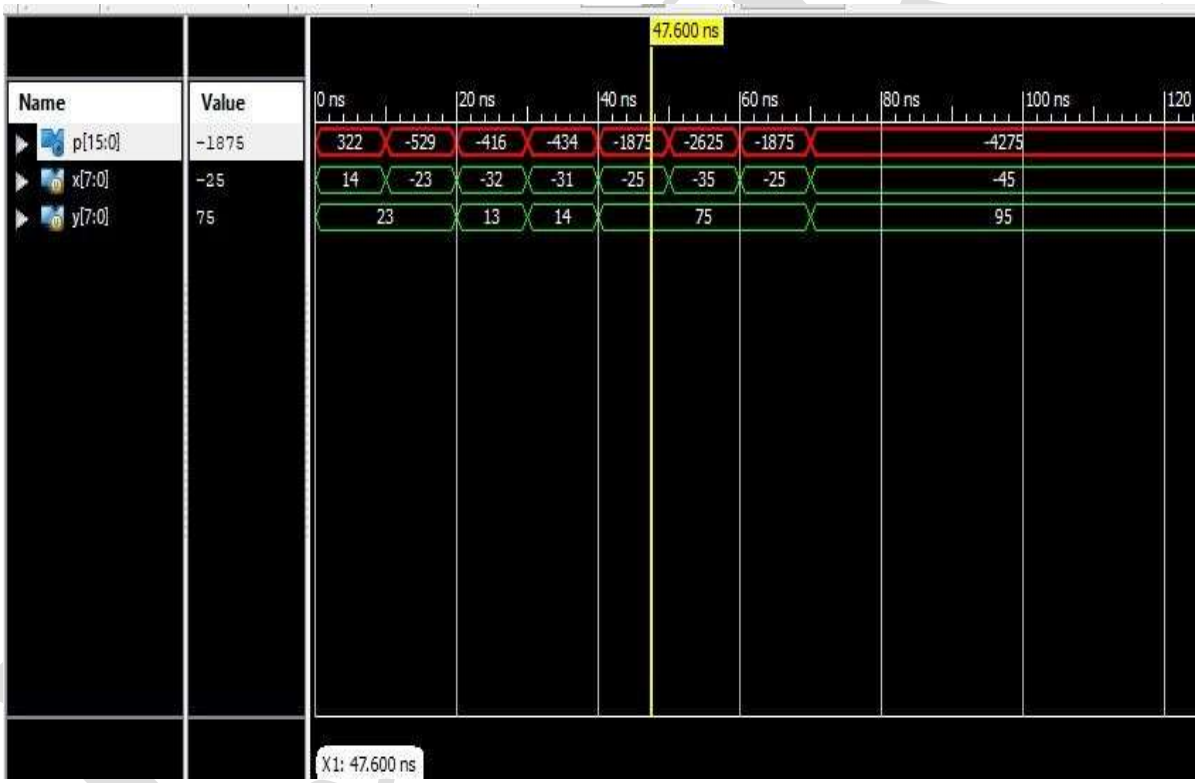

Fig.5.3.1 (a) Simulation result of DST FWBM

It performs better multiplication operation (75*-25= -1875).

5.3.1 (b)

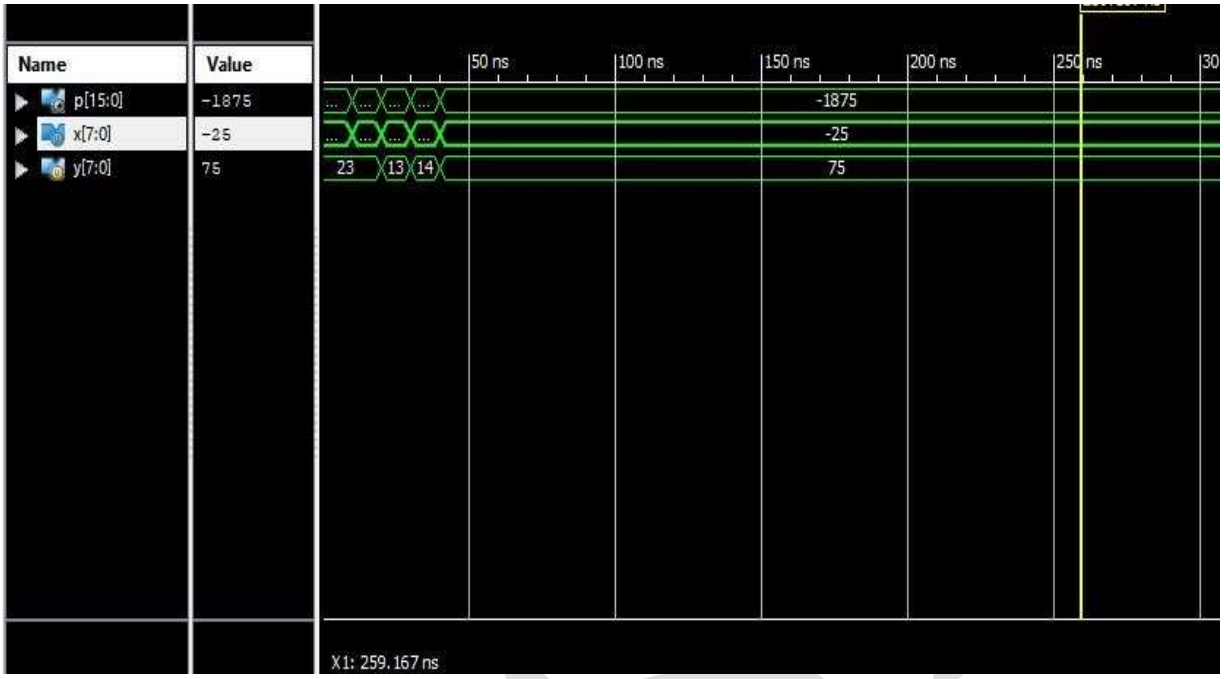

Fig: 5.3.1 (b) Simulation Result

5.3.2 SIMULATION RESULT OF DST FWBM WITH BINARY VALUES

Fig: -5.3.2 Simulation Result of DST FWBM

5.4 SYNTHESIS REPORT

Synthesis in VLSI is the process of converting your code(program) into a circuit. In terms of logic gates, synthesis is the process of translating an abstract design into a properly implemented chip.

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of 4 input LUTs	144	9,312	1%	
Number of occupied Slices	77	4,656	1%	
Number of Slices containing only related logic	77	77	100%	
Number of Slices containing unrelated logic	0	77	0%	
Total Number of 4 input LUTs	144	9,312	1%	
Number of bonded IOBs	32	232	13%	
Average Fanout of Non-Clock Nets	4.15			

Fig.5.4 Synthesis report

- Design summary gives information about Area and Input and Output pins.
- Total no. of Look Up Tables (LUTs) is 9,312. In that 9,312 LUTs are 144 LUTs are used in our proposed system.
- Total no.of IOs are 232. In that 32 IOs are used in our proposed system.
- Total Area is 144.

5.5 TIMING REPORT

```
Destination:       p<15> (PAD)

Data Path: y<3> to p<15>
                                   Gate     Net
  Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
  ----------------------------------------  ------------
   IBUF:I->O             35   1.106   1.226  y_3_IBUF (y_3_IBUF)
   LUT4:I0->O             1   0.612   0.000  q1/k_2_or0000_F (N161)
   MUXF5:I0->O            4   0.278   0.568  q1/k_2_or0000 (q1/k<2>)
   LUT2:I1->O             2   0.612   0.383  q1/Mxor_p_Result1 (sp<1>)
   LUT4:I3->O             3   0.612   0.481  fa2/cy1 (c1<1>)
   LUT3:I2->O             2   0.612   0.532  fa3/cy1 (c1<2>)
   LUT4:I0->O             2   0.612   0.532  fa4/cy1 (c1<3>)
   LUT4:I0->O             2   0.612   0.532  fa5/cy1 (c1<4>)
   LUT3:I0->O             2   0.612   0.532  fa6/h2/Mxor_s_Result1 (ip1<3>)
   LUT3:I0->O             2   0.612   0.532  sa4/cy1 (c2<3>)
   LUT4:I0->O             2   0.612   0.532  sa5/cy1 (c2<4>)
   LUT3:I0->O             2   0.612   0.449  sa6/h2/Mxor_s_Result1 (ip2<3>)
   LUT3:I1->O             2   0.612   0.532  foa4/cy1 (c3<3>)
   LUT4:I0->O             2   0.612   0.410  foa5/cy1 (c3<4>)
   LUT3:I2->O             2   0.612   0.410  foa6/cy1 (c3<5>)
   LUT3:I2->O             2   0.612   0.532  foa7/cy1 (c3<6>)
   LUT4:I0->O             2   0.612   0.449  foa8/cy1 (c3<7>)
   LUT3:I1->O             1   0.612   0.357  foa10/h2/Mxor_s_Result1 (p_15_OBUF)
   OBUF:I->O                  3.169          p_15_OBUF (p<15>)
  ----------------------------------------
  Total                      23.333ns (14.345ns logic, 8.988ns route)
                                      (61.5% logic, 38.5% route)
```

Fig.5.5 Timimg report

Timing paths start and end at clocked elements. Input and Output ports are not sequential elements, and by default Vivado timing analysis does not time paths to or from I/O ports in the design, unless input/output delay constraints are specified.

In this the total delay is 23.333ns.

5.6 POWER ANALYSIS

Device	
Family	Spartan3e
Part	xc3s500e
Package	fg320
Temp Grade	Commercial
Process	Typical
Speed Grade	-5

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	0

Characterization	
PRODUCTION	v1.2,06-23-09

On-Chip	Power (W)	Used	Available	Utilization (%)
Logic	0.000	144	9312	2
Signals	0.000	136	---	---
IOs	0.000	32	232	14
Leakage	0.081			
Total	0.081			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	26.1	82.9	27.1

Supply Summary		Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	1.200	0.026	0.000	0.026
Vccaux	2.500	0.018	0.000	0.018
Vcco25	2.500	0.002	0.000	0.002

	Total	Dynamic	Quiescent
Supply Power (W)	0.081	0.000	0.081

Fig.5.6 Power analysis

- Power analysis uses the VCD (Value Change Dump) file generated by the simulator for the analysis of the switching power and then do the placement and routing.
- Under place and route, click on the option Analyze Power Distribution.
- The design file would be like “design_name”.ncd. In the physical Constraint file, select the file “design_name”.pcf. Finally in the simulation activity add the VCD file “file_name”.vcd.
- Click “OK” and your power report will be generated

Total Power Analysis is “0.081”.

6. CONCLUSION:

In this study, we presented the application of a DST circuit to low-error FWBMs; the proposed DST circuit considerably improved the accuracy of the FWBMs. The accuracy of the proposed DST-FWBM was close to the ideal accuracy value of P-T multipliers and exhibited a justifiably small area cost. Upon evaluation of its system application, the proposed DSTFWBM achieved high accuracy. The DST circuit also helped improve the accuracy of long-width FWBMs. In summary, DST can be used in DSP applications, particularly those that require high accuracy.

This project focuses on optimizing the design of the Fused-Add Multiply (FAM) operator. We propose a structured technique for the direct recoding of the sum of two numbers to its MB form. We explore three alternative designs of the proposed S-MB recoder and compare them to the existing ones. The proposed recoding schemes, when they are incorporated in FAM designs, yield considerable performance improvements in comparison with the most efficient recoding schemes found in literature.

REFERENCES

[1] Y. H. Chen, T. Y. Chang, and C. Y. Li, "High Throughput DA-based DCT with High Accuracy Error-compensated Adder Tree," IEEE Trans. VLSI Syst., vol. 19, no. 4, pp. 709–714, Apr. 2011.

[2] S. N. Tang, J. W. Tsai, and T. Y. Chang, "A 2.4-GS/s FFT Processor for OFDM-Based WPAN Applications," IEEE Trans. Circuits Syst. II, vol. 57, no. 6, pp. 451–455, Jun. 2010.

[3] W. Liu, T. Cao, P. Yin, Y. Zhu, C. Wang, E. E. Swartzlander, and F. Lombardi, "Design and analysis of approximate redundant binary multipliers," IEEE Trans. Comput., vol. 68, no. 6, pp. 804–819, Jun. 2019.

[4] L. D. Van, S. S. Wang, and W. S. Feng, "Design of the lower error fixed-width multiplier and its application," IEEE Trans. Circuits Syst. II, vol. 47, no. 10, pp. 1112–1118, Oct. 2000.

[5] I. C. Wey and C. C. Wang, "Low-error and hardware-efficient fixedwidth multiplier by using the dual-group minor input correction vector to lower input correction vector compensation error," IEEE Trans. VLSI Syst., vol. 20, no. 10, pp. 1923–1928, Oct. 2012.

[6] D. Esposito, A. G. M. Strollo, E. Napoli, D. De Caro, and N. Petra, "Approximate multipliers based on new approximate compressors," IEEE Trans. Circuits Syst. I, vol. 65, no. 12, pp. 4169–4182, Dec. 2018.

[7] S. J. Jou, M. H. Tsai, and Y. L. Tsao, "Low-error reduced-width Booth multipliers for DSP applications," IEEE Trans. Circuits Syst. I, vol. 50, no. 11, pp. 1470–1474, Nov. 2003.

[8] M. A. Song, L. D. Van, and S. Y. Kuo, “Adaptive low-error fixed- width Booth multipliers,” IEICE Trans. Fundamentals, vol. E90-A, no. 6, pp. 1180–1187, Jun. 2007.

[9] Y. H. Chen, T. Y. Chang, and R. Y. Jou, “A statistical error-compensated Booth multiplier and its DCT applications,” in Proc. IEEE Region 10 Conf., 2010, pp. 1146–1149.

[10] J. P. Wang, S. R. Kuang, and S. C. Liang, “High-Accuracy Fixed-Width Modified Booth Multipliers for Lossy Applications,” IEEE Trans. VLSI Syst., vol. 19, no. 1, pp. 52–60, Jan. 2011.

[11] C. Y. Li, Y. H. Chen, T. Y. Chang, and J. N. Chen, “A probabilistic estimation bias circuit for fixed-width Booth multiplier and its DCT applications,” IEEE Trans. Circuits Syst. II, vol. 58, no. 4, pp. 215–219, Apr. 2011.

[12] Y. H. Chen, C. Y. Li, and T. Y. Chang, “Area-Effective and PowerEfficient Fixed-Width Booth Multipliers Using Generalized Probabilistic Estimation Bias,” IEEE J. Emerging Sel. Topics Circuits Syst., vol. 1, no. 3, pp. 277–288, Sep. 2011.

[13] Y. H. Chen and T. Y. Chang, “A High-Accuracy Adaptive ConditionalProbability Estimator for Fixed-width Booth Multipliers,” IEEE Trans. Circuits Syst. I, vol. 59, no. 3, pp. 594–603, Mar. 2012.

[14] W. Q. He, Y. H. Chen, and S. J. Jou, “High-Accuracy Fixed-Width Booth Multipliers Based on Probability and Simulation,” IEEE Trans. Circuits Syst. I, vol. 62, no. 8, pp. 2052–2061, Aug. 2015.

[15] Y. H. Chen, “An Accuracy-Adjustment Fixed-Width Booth Multiplier Based on Multilevel Conditional Probability,” IEEE Trans. VLSI Syst., vol. 23, no. 1, pp. 203–207, Jan. 2015.

[16] Z. Zhang and Y. He, “A Low-Error Energy-Efficient Fixed-Width Booth Multiplier With Sign-Digit-Based Conditional Probability Estimation,” IEEE Trans. Circuits Syst. II, vol. 65, no. 2, pp. 236–240, Feb. 2018.

[17] M. Chakraborty, R. Shaik, and M. H. Lee, “A Block-Floating-PointBased Realization of the Block LMS Algorithm,” IEEE Trans. Circuits Syst. II, vol. 53, no. 9, pp. 812–816, 2006.

[18] B. Parhami, Computer arithmetic: algorithms and hardware designs. Oxford, UK: Oxford University Press, 2000.

[19] S. C. Hsia and S. H. Wang, “Shift-register-based data transposition for cost-effective discrete cosine transform,” IEEE Trans. VLSI Syst., vol. 15, no. 6, pp. 725–728, Jun. 2007.