iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** II **Month of publication:** February 2023

DOI: https://doi.org/10.22214/ijraset.2023.48914

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue II Feb 2023- Available at www.ijraset.com

Implementation of ALU Using Modified Radix-4 Modified Booth Multiplier

Towseef Ul Haq Wani[1], Ravinder Pal Singh[2]
[1]*M. Tech Scholar, Dept. of Electronics and Communications Engg, RIMT University, Mandi Gobindgarh, Punjab,India*
[2]*Tech. Head-Dept. of Research, Innovations & Incubation / DESINNO-CoE, RIMT University, Punjab, India*

***Abstract:** In this project, we propose a novel multiplier hardware design based on a radix - 4 modified booth encoder. The standard modified Booth encoding (MBE) provides an uneven partial product array due to the extra partial product bit at the least significant bit position of each partial product row. In this brief, a simple technique for constructing a regular partial product array with fewer partial product rows and low overhead is provided, reducing the complexity of partial product reduction as well as the space, time, and power of MBE multipliers. A SPST-based adder is examined and designed for the reduction of power in partial product reduction*
***Keywords:** Adders, SPST adder, Xilinx vivado, multiplier, booth encoding, radix 4 booth multiplier, modified booth multiplier.*

I. INTRODUCTION

Since the speed of the multiplication operation dictates the circuit's speed, there is a requirement of quick multipliers in digital signal processing. Systems for digital signal processing contain fast multipliers. Today, especially since the media processing industry has taken off, the speed of the multiply operation is crucial for both general-purpose processors and digital signal processing. In the past, addition, subtraction, and shift operations were typically used to multiply numbers. The multiplicand is the amount to be added, the multiplier is the number of additions, and the product is the outcome. When the operands are interpreted as integers, the product is generally double the length of operands in order to reserve the information content.

Columns of the shifted partial product matrix are added. The correct bit of the "multiplicand" is gated after the "multiplier" has successfully been shifted. The shifted partial product matrix's same column must contain all instances of the multiplicand that are delayed and gated. The product bit for the specific form is then formed by combining them. As a result, multiplication is an operation with several operands. A practical number system would be the representation of numbers in two's complement format, which would allow for the extension of multiplication to both signed and unsigned integers.

An M×N bit multiplication may be thought of as creating N partial product arrays, each with M bits, then adding them together based on their weights. Either a Shift-Add algorithm or approaches for parallel multiplication are used to execute multiplication. M-cycles are needed to multiply M×N bits using the Shift - Add technique. There are several methods for carrying out parallel multiplication. The approach is chosen based on the designs requirements for latency, throughput, area, and complexity. The reduction of the partial products arrays is crucial in parallel multiplication algorithms. Several procedures compact with dropping the partial products arrays, and then Wallace tree or array tree adders are used to lessen the number of logic levels required to perform summation. The finishing two rows are added using a fast Carry Propagate Adder.

Since multiplication is a highly slow process, the performance of any digital system is often evaluated in terms of the multipliers that are employed. It will function at a slower pace and need a lot of hardware components to accomplish this multiplication. Recently, a lot of solutions have been put out to address this issue. With the development of technology, several studies have attempted to create multipliers that are suited for real-time applications by integrating high speed, small size, and low power consumption into a single multiplier. The most crucial factor to take into account in many real-time applications is speed. Mostly, multiplication operation is carried out by first generating the partial product and then adding to it.

Array multiplier is standard due to its edifice. It is based on add and shift algorithm. In parallel multiplication, the number of partial products to be added is the main parameter that determines the performance of the multiplier. With one multiplier bit each partial product is generated by the multiplication of the multiplicand. The partial products are shifted according to their bit order and then it gets added with normal carry propagate adder. For n x n array multiplier, number of adders and gates required are:1.n(n-2) full adders, 2.n half adders.The advantage of array multiplier is that it has minimum complexity and regular structure. Disadvantages are large number of logic gates, so more chip area and it has high power consumption and it is limited to 16-bits. The advantage of an array multiplier is that it has a regular structure and little complexity.

©IJRASET: All Rights are Reserved

The disadvantages include a large number of logic gates, which takes up more chip space, a high power consumption rate, and a 16-bit maximum .Different performance metrics from the multipliers are required by different sorts of applications, including scalability, configurability, high speed, low power consumption, regularity of layout, and minimal area, or a combination of one or more of these properties. The arithmetic and logic operations are performed by tiny computers using the same principles as big ones despite their differences in size and complexity, with the exception of the speed at which hardware units execute algorithms. In addition special techniques are used to improve the speed by performing several operations in parallel.

Utilizing a strategy comparable to manually computing a multiplication is a quicker method to implement multiplication. The complete partial product is produced simultaneously and is arranged in an array. To compute the result, a multioperand addition is used. The two phases that make up most formalised multiplication algorithms are the first step, which creates a partial product, and the second step, which adds the partial product to the previous partial products. The common multiplication method uses add and shift operators. The shift operation generates the partial products and the adder units sum them up. The general scheme for unsigned multiplication in base b is shown in Figure below

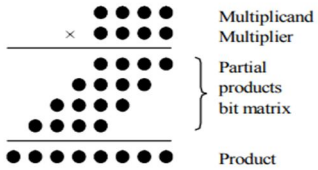


Figure 1: Multiplication Process

Each row or partial product is obtained by multiplying one digit of the multiplier times the multiplicand. The low order digit of a partial product 4 is determined from just one multiplicand digit, but other digits include the effects of the carry from the digits to the right. In binary, partial products are trivial – either a copy of the multiplicand or zero. The sum of the partial products gives the computed result. One of the most significant arithmetic units utilized in computer systems is the multiplier. The construction of an efficient multiplier has been discussed using a number of arithmetic methods. Wallace tree technique is among the most effective approaches. This method may create hardware that executes multiplication operations in parallel and consists of full-adder and half-adder circuits in three phases. The Wallace-based multiplying operation includes three stages:

1) *Stage1:* Multiply (or better expressing, AND) each bit of multiplicand by each bit of multiplier, yielding n2 partial products.
2) *Stage2:* Reduce the number of partial products using the layers of a FA and a HA blocks.
3) *Stage3:* Adding two n-sets resulted from the previous stage to an n-bit adder. It should be noted that the second stage is carried out as follows. As long as there are three or more bits with the same value, add a following layer:

- Three bits of the same value enter into FA and as a result, two bits with different values are produced (one bit with the same value and one bit with a higher value).
- If two bits with the same value remain, put them into an HA.
- If there is just one bit, transfer it to the next layer.

A. Contributions

The principles and architectural designs of a modified booth encoder for radix-4 are covered in this paper. A SPST-based adder in a binary tree addition structure has been used for the reduction of partial products. After carrying out a thorough analysis and trail, SPST with conventional adders is used to maximise the area and delay power. In order to retain efficiency and get optimum parameters, it was later enhanced by the addition of a second adder known as parallel prefix adders to shorten the critical path computation delay by the Ladner Fischer adder. This extended multiplier with SPST and PPA adder is built in a generic ALU as a part of the application, which also includes a reprogrammable clock division circuit. These all designs are coded in verilog HDL and synthesized and simulated using Xilinx Vivado

B. Outline

The rest of this research article is structured as follows. Section II discusses the literature study, Section III introduces the broad vocabulary and fundamental principles of the modified booth encoding scheme. Section IV discusses the implementation of a booth encoding technique utilising an SPST adder. Section V provided an overview of xilinx and verilog HDL. Section VI contains the results and discussion, while Section VII and section VIII contains the findings, future scope, and suggestions.

II. LITERATURE REVIEW

1) Shankar at el. (2021) paper entitled "Design a high performance and low power radix-4 booth multiplier using power reduction techniques" proposed a design that helps to reduce the power by disabling all the available Booth encoders and decoders from additional working. This design is implemented using Cadence 45 nm technology. From the simulation results, it is seen that the , proposed pre-encoded design could be able to reduce the dynamic power and static power by 30% and 41%, respectively, when compared to a traditional radix-4 Booth multiplier (16-bit). Compared to the previous mechanisms, the proposed design has race-free characteristics and consumes less power.
2) Supritha at el. (2020) paper entitled "High speed and efficient ALU using modified booth multiplier" describes about a modified booth multiplier planted in the arithmetic logic unit. The imperative modules of a 1-bit ALU are the module of power and the module of addition. This ALU arrangement has decreased door check and semiconductor count.
3) Suvarna at el. (2016) paper entitled "A Modified Architecture for Radix-4 Booth Multiplier with Adaptive Hold Logic" describes a new multiplier was designed with novel adaptive hold logic (AHL) using Radix-4 Modified Booth Multiplier. By using Radix-4 Modified Booth Encoding (MBE), we can reduce the number of partial products by half. Modified booth multiplier helps to provide higher throughput with low power consumption. This can adjust the AHL circuit to reduce the performance degradation.
4) Nikhare and Ashish Singhadia (2015) paper entitled "A Detailed Review on Architectures for 2-DWT by using Radix-4 Booth Multiplier" describes a review of VLSI architectures for 2-DWT implementation of wavelet transform by using 4 Booth multiplier. This review paper describes implementation of radix-4 Modified Booth Multiplier and this implementation is compared with Radix-2 Booth Multiplier. Modified Booth‟s algorithm employs both addition and subtraction and also treats positive and negative operands uniformly. Parallel MAC is frequently used in digital signal processing and video/graphics applications.
5) Sukhmeet Kaur at el. (2013) paper entitled "Implementation of Modified Booth Algorithm (Radix 4) and its Comparison with Booth Algorithm (Radix-2)" describes implementation of radix-4 Modified Booth Multiplier and this implementation is compared with Radix-2 Booth Multiplier. Modified Booth's algorithm employs both addition and subtraction and also treats positive and negative operands uniformly. No special actions are required for negative numbers.
6) Basha and Badashah (2012) paper entitled "design and implementation of radix-4 based high speed multiplier for alu's using minimal partial products" presents the methods required to implement a high speed and high performance parallel complex number multiplier. The designs are structured using Radix-4 Modified Booth Algorithm and Wallace tree. These two techniques are employed to speed up the multiplication process as their capability to reduce partial products generation and compress partial product term by a ratio of 3:2. Despite that, carry save-adders (CSA) is used to enhance the speed of addition process for the system. The system has been designed efficiently using VHDL codes for 8x8-bit signed numbers and successfully simulated and synthesized using Xilinx.

III. GENERAL TERMINOLOGY

A. Reduction Of Partial Products

In the existing method, using the modified booth encoding scheme partial products are generated in a regular format neglecting the last row bits as in the case of conventional encoding system. For an m bit width of input in the conventional m/2 + 1 number of rows of partial products are generated. And the generation of conventional system was in irregular format. Now it was regularized and the reduction was made to m/2 rows of partial products using radix-4 scheme of modified booth encoding. And then partial products are reduced using half adders and full adders to get the final product output.

The half adder is used to add only two numbers. To overcome this problem, the full adder was developed. The full adder is used to add three 1-bit binary numbers A, B, and carry c.The hardware requirement in terms of full adder (FA) and the length of final adder (FAL) for different size of array multipliers is obtained in the manner given in below Figure.

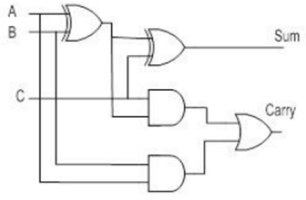


Figure 2: Full adder

The adder known as a "full adder" adds three inputs and generates two outputs. A and B make up the first two inputs, while an input carry marked as C makes up the third. The normal output is denoted as SUM, whereas the output carry is denoted as CARRY.

Inputs			Outputs	
A	B	C_{in}	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

In the above table,

- 'A' and' B' are the input variables. These variables represent the two significant bits which are going to be added.
- 'C_{in}' is the third input which represents the carry. From the previous lower significant position, the carry bit is fetched.
- The 'Sum' and 'Carry' are the output variables that define the output values.
- The eight rows under the input variable designate all possible combinations of 0 and 1 that can occur in these variables.

IV. IMPLEMENTATION

An SPST-based adder is paired with a modified booth encoding method based on radix-4 in order to reduce the power consumption in the multiplier circuit. To achieve parallelism in the instance, a reduction of partial products tree-based structure was also applied. We provide a unique SPST implementation technique for a high-speed low-power multiplier in this study. This multiplier is made using the Spurious Power Suppression Technique (SPST), which is mounted on a modified Booth encoder and managed by a detecting unit that makes use of an AND gate. The SPST adder will stop pointless addition, hence minimising switching power loss.. The proposed MBE can be detailed as follows using a table along with the circuits it can be designed and values are generated. For MB recoding, at least three signals are needed to represent the digit set {−2, −1, 0, 1, 2}. Many different ways have been developed, and Table I shows the encoding scheme proposed in [reference that is assumed to implement the proposed MBE multiplier.

TABLE I
MBE TABLE

b_{2i+1}	b_{2i}	b_{2i-1}	Operation	neg_i	two_i	one_i	p_{ij}
0	0	0	+0	0	0	0	0
0	0	1	+A	0	0	1	a_j
0	1	0	+A	0	0	1	a_j
0	1	1	+2A	0	1	0	a_{j-1}
1	0	0	−2A	1	1	0	$\overline{a_{j-1}}$
1	0	1	−A	1	0	1	$\overline{a_j}$
1	1	0	−A	1	0	1	$\overline{a_j}$
1	1	1	−0	0	0	0	0

The Booth encoder and selector circuits proposed in reference are depicted in Fig. 5(a) and (b), respectively. Based on the recoding scheme and the approach proposed in fig.1, τi0 and ci in Fig. 2 can be derived from the truth table shown in Table II, as follows:

$$\tau_{i0} = one_i \cdot a_0 = \overline{\overline{one_i} + \overline{a_0}} \quad (3)$$

$$c_i = neg_i \cdot (\overline{one_i} + \overline{a_0}) = \overline{\overline{neg_i} + one_i \cdot a_0}. \quad (4)$$

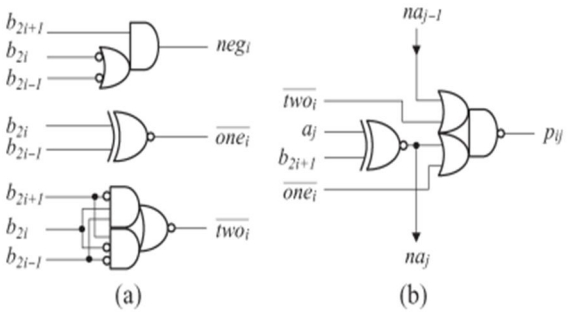


Figure 3: 5 MBE encoder and selector from reference

TABLE II
TRUTH TABLE FOR PARTIAL PRODUCT BITS IN THE PROPOSED PARTIAL PRODUCT ARRAY

b_{2i+1}	b_{2i}	b_{2i-1}	τ_{i0}	c_i	τ_{i1}	d_i	$\tilde{\tau}_{i1}$	e_i
0	0	0	0	0	0	0	1	0
0	0	1	a_0	0	a_1	0	$\overline{a_1}$	a_1
0	1	0	a_0	0	a_1	0	$\overline{a_1}$	a_1
0	1	1	0	0	a_0	0	$\overline{a_0}$	a_0
1	0	0	0	1	a_0	$\overline{a_0}$	$\overline{a_0}$	1
1	0	1	a_0	$\overline{a_0}$	$a_0 \oplus a_1$	$\overline{a_0+a_1}$	$a_0 \odot a_1$	$\overline{a_0 \cdot a_1}$
1	1	0	a_0	$\overline{a_0}$	$a_0 \oplus a_1$	$\overline{a_0+a_1}$	$a_0 \odot a_1$	$\overline{a_0 \cdot a_1}$
1	1	1	0	0	0	0	1	0

According to (3) and (4), τi0 and ci can be produced by one NOR gate and one AOI gate, respectively. Moreover, they are generated no later than other partial product bits. To further remove the additional partial product row PPn/2, we combine the ci for i = n/2 − 1 with the partial product bit pi1 to produce a new partial product bit τi1 and a new carry di. Then, the carry di can be incorporated into the sign extension bits of PP0. However, if τi1 and di are produced by adding ci and pi1, their arrival delays will probably be larger than other partial product bits. Therefore, we directly produce τi1 and di for i = n/2 − 1 from A, B, and the outputs of the Booth encoder (i.e., negi, twoi, and onei), as shown in Table II. The logic expressions of τi1 and di can be written as

$$\tau_{i1} = one_i \cdot \varepsilon + two_i \cdot a_0 = \overline{(\overline{one_i} + \overline{\varepsilon}) \cdot (\overline{two_i} + \overline{a_0})} \quad (5)$$

$$d_i = \overline{(\overline{b_{2i+1}} + a_0)} \cdot \overline{[(b_{2i-1} + a_1) \cdot (b_{2i} + a_1) \cdot (b_{2i} + b_{2i-1})]} \quad (6)$$

Where,

$$\overline{\varepsilon} = \begin{cases} a_1, & \text{if } \overline{a_0 \cdot b_{2i+1}} = 0 \\ \overline{a_1}, & \text{otherwise.} \end{cases} \quad (7)$$

Since the weight of di is 2n, which is equal to the weight of s0 at bit position n, di can be incorporated with the sign extension bits s0s0s0 of PP0. Let α2α1α0 be the new bits after incorporating di into s0s0s0; the relations between them are summarized in Table III.

TABLE III
TRUTH TABLE FOR NEW SIGN EXTENSION BITS

$\overline{s_0}$	s_0	s_0	d_i	α_2	α_1	α_0
1	0	0	0	1	0	0
1	0	0	1	1	0	1
0	1	1	0	0	1	1
0	1	1	1	1	0	0

As can be seen in Table III, the maximal value of s0s0s0 is 100 so that the addition of s0s0s0 and di will never produce an overflow. Therefore, α2α1α0 is enough to represent the sum of s0s0s0 and di. According to Table III, α2, α1, and α0 can be expressed as

$$\alpha_2 = \overline{(s_0 \cdot \overline{d_i})} \quad (8)$$

$$\alpha_1 = s_0 \cdot \overline{d_i} = \overline{\alpha_2} \quad (9)$$

$$\alpha_0 = s_0 \odot \overline{d_i}. \quad (10)$$

The corresponding circuits to generate τi1, di, and α2α1α0 are depicted in Fig. 6(a)–(c), respectively.

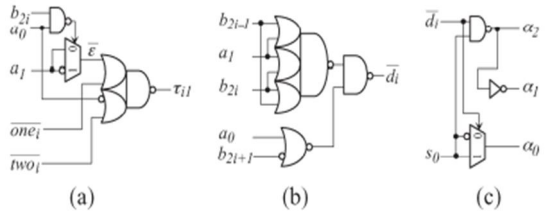


Figure 4: Proposed circuits to generate τi1, di, and α2α1α0 for i = n/2 − 1

The partial product array generated by the proposed approach for the 8 × 8 multiplier is shown in Fig. 4. This regular array is generated by only slightly modifying the original partial product generation circuits and introducing almost no area and delay overhead.

After the computation of partial products as shown in figure four for addition of partial products we are considering a tree based adder structure such that parallelism can be implemented and attain output in an efficient way as shown in below figure.

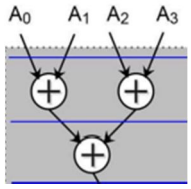


Figure 5: Tree based adder

Among them one of the adder can be selected as a SPST adder such that it will influence the overall design for power reduction

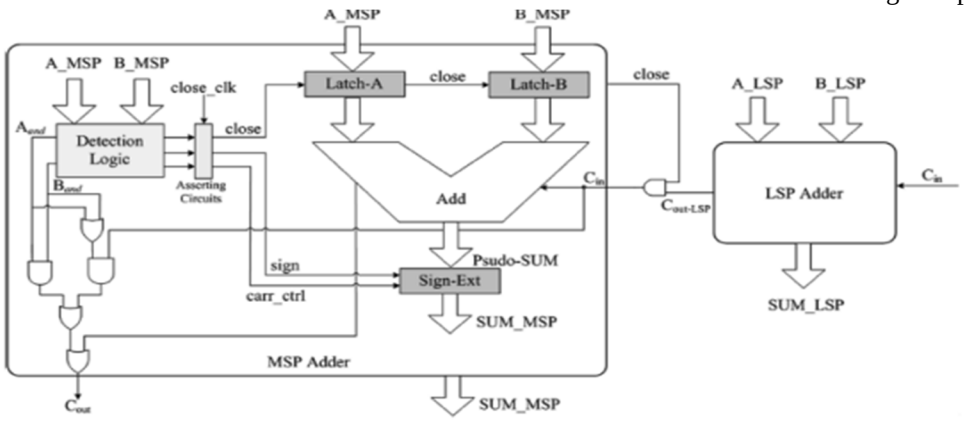


Figure 6: SPST based adder

A. SPST Adder

The above figure shows a 32-bit adder design adopting the proposed SPST. The 32-bit adder is divided into MSP and LSP between the 15th and the 16th bit. Latches implemented by simple AND gates are used to control the input data of the MSP. When the MSP is necessary, the input data of the MSP remain unchanged. However, when the MSP is negligible, the input data of the MSP becomes zero to avoid glitching power consumption. The two operands of the MSP enter the detection logical unit, except the adder, so that the detection logical unit can decide whether to turn off the MSP or not.

The detection logical unit determines whether the input data of the MSP should be latched or not. We use 3 signals for asserting the output of the detection logical unit- CLOSE, CARRY_CTRL, and SIGN. A certain amount of delay is used to assert the close, sign, and carry-ctrl signals. The timing issue of SPST is analysed as follows:

1) When the detection logic unit switches off the MSP, the sign-extension unit immediately compensates for the MSP outputs, thus the time saved by bypassing the computations in the MSP circuits will cancel out the delay brought on by the detection logic unit.

2) When the detection logic unit turns on the MSP: the MSP circuits must wait for the notification of the detection logic unit to turn on the data latches to let the data in. Hence, the delay caused by the detection logic unit will contribute to the delay of the whole combinational circuitry, i.e. the 32-bit adder/subtractor in this design example.

3) When the detection logic unit remains in its decision: no matter whether the last decision is turning on or turning off the MSP, the delay of the detection logic is negligible because the path of the combinational circuitry (i.e. the 32-bit adder/subtractor in this design) remains the same.

V. XILINX AND VERILOG HDL

A. VERILOG

About 1984 marked the beginning of its development as a proprietary hardware modelling language by Gateway Design Automation Inc. It is said that the original language's features were derived from both classic computer languages like C and the then-most-used HDL language, dubbed HiLo. At the time, Verilog was not standardised, and it changed throughout practically every edition that was released between 1984 and 1990.Verilog simulator was first used beginning in 1985 and was extended substantially through 1987. The implementation was the Verilog simulator sold by Gateway. The first major extension was Verilog-XL, which added a few features and implemented the infamous "XL algorithm" which was a very efficient method for doing gate-level simulation

B. HDL

1) HDL is an abbreviation of Hardware Description Language. Any digital system can be represented in a REGISTER TRANSFER LEVEL (RTL) and HDLs are used to describe this RTL.

2) Verilog is one such HDL and it is a general-purpose language –easy to learn and use. Its syntax is similar to C.

3) The idea is to specify how the data flows between registers and how the design processes the data.

4) To define RTL, hierarchical design concepts play a very significant role. Hierarchical design methodology facilitates the digital design flow with several levels of abstraction.

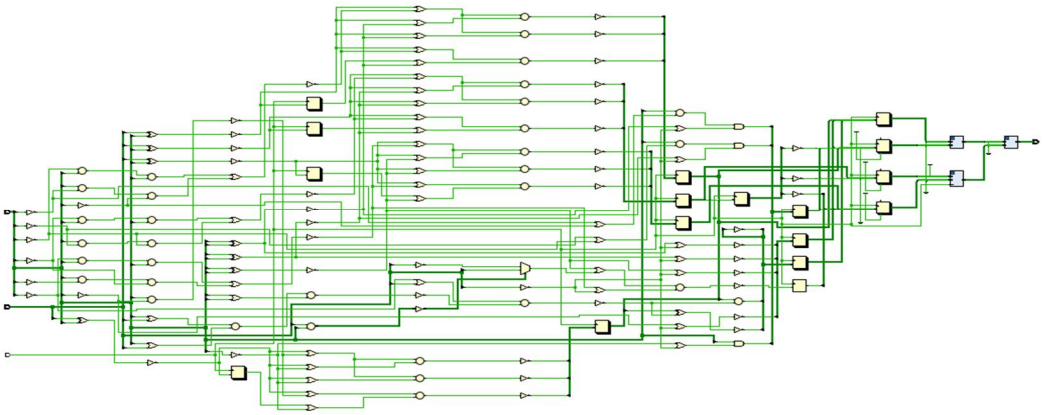
5) Verilog HDL can utilize these levels of abstraction to produce a simplified and efficient representation of the RTL description of any digital design.

6) As an illustration, an HDL may specify the layout of the wires, resistors, and transistors on an integrated circuit (IC) chip, i.e., the switch level, or it might specify the design at a more micro level in terms of logical gates and flip flops in a digital system, i.e., the gate level. All of these levels are supported by Verilog.

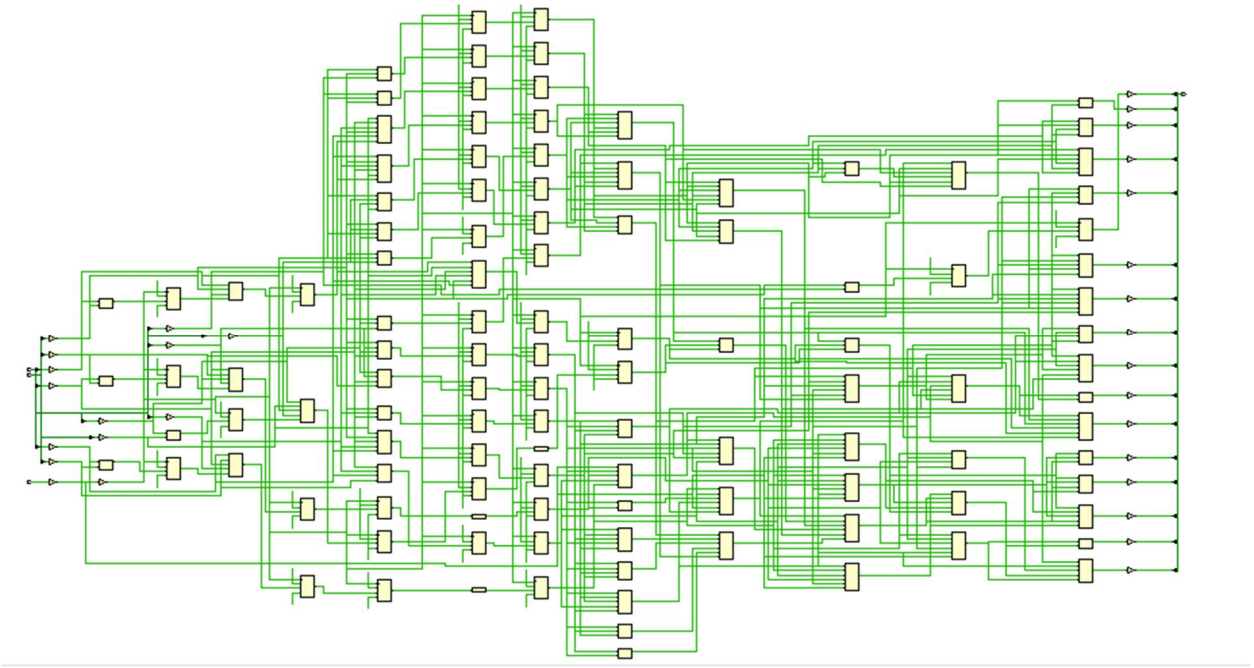
VI. RESULTS

A. Proposed Multiplier

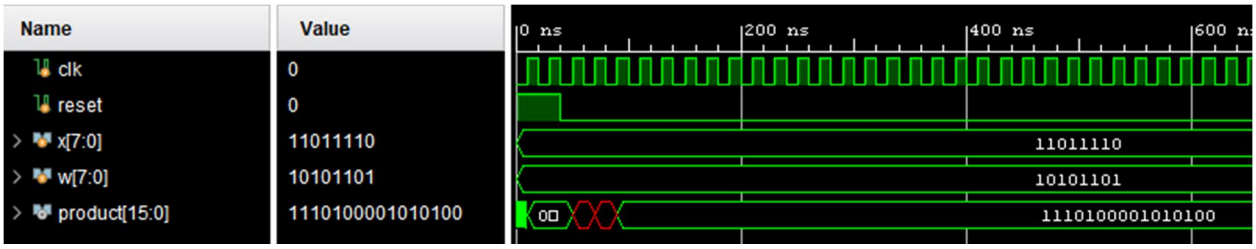
1) RTL Schematic



2) Technology schematic



3) Simulation



4) Area

Resource	Utilization	Available	Utilization %
LUT	51	20800	0.25
FF	73	41600	0.18
IO	30	106	28.30

5) Delay

```
Max Delay Paths
--------------------------------------------------------------------------------
Slack:                    inf
  Source:                 n2/sum_reg[1]/C
                            (rising edge-triggered cell FDRE)
  Destination:            product_reg[12]/D
  Path Group:             (none)
  Path Type:              Max at Slow Process Corner
  Data Path Delay:        4.089ns  (logic 1.247ns (30.496%)  route 2.842ns (69.504%))
  Logic Levels:           6  (FDRE=1 LUT3=1 LUT5=1 LUT6=3)
```

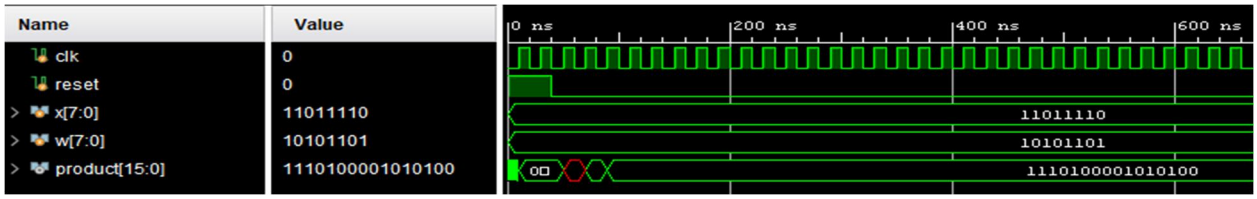
6) Power

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

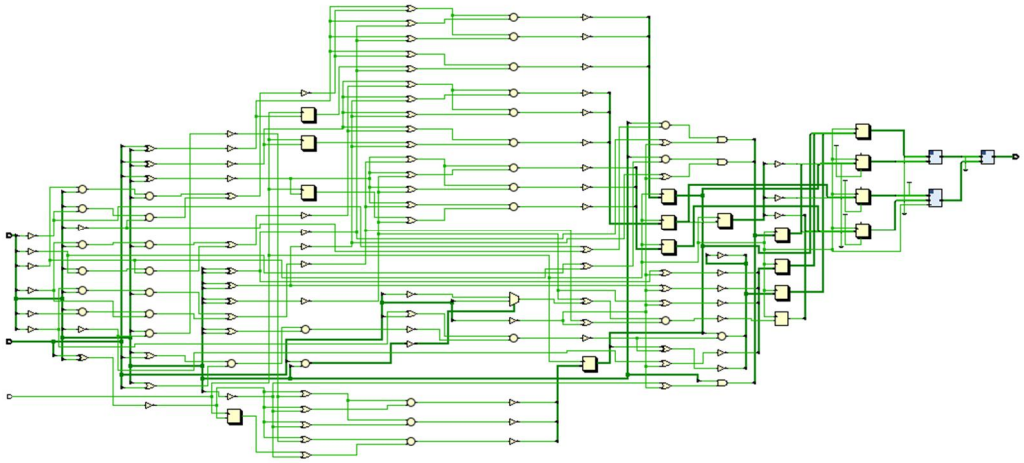
Total On-Chip Power:	**12.56 W (Junction temp exceeded!)**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**87.8°C**

B. Extension Multiplier

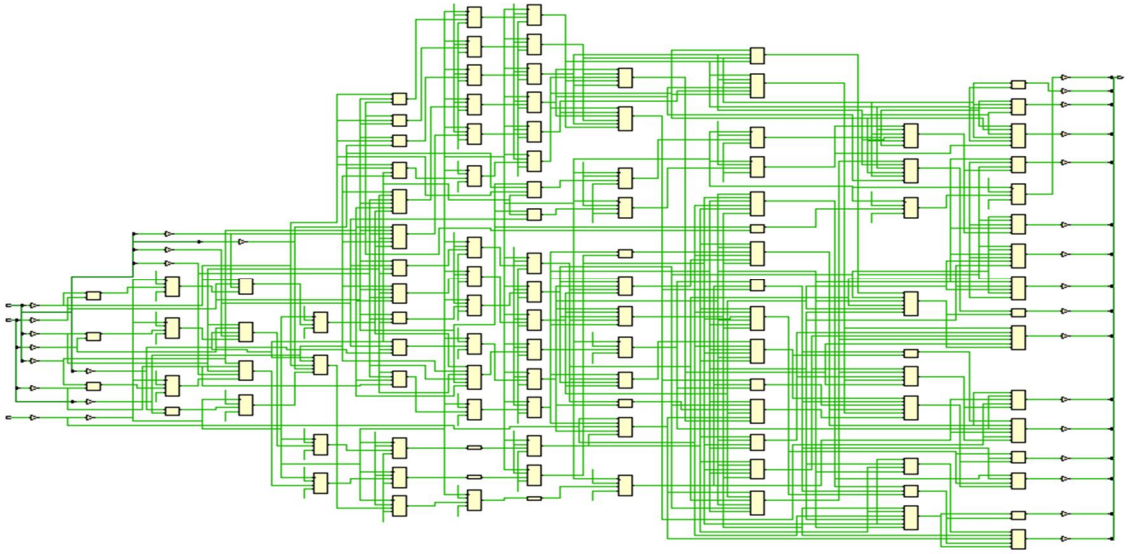
1) Simulation



2) RTL Schematic



3) Technology schematic



4) Area

Resource	Utilization	Available	Utilization %
LUT	50	20800	0.24
FF	73	41600	0.18
IO	30	106	28.30

5) Delay

```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack:                    inf
  Source:                 product_reg[0]/C
                            (rising edge-triggered cell FDRE)
  Destination:            product[0]
                            (output port)
  Path Group:             (none)
  Path Type:              Max at Slow Process Corner
  Data Path Delay:        4.016ns  (logic 3.216ns (80.087%)  route 0.800ns (19.913%))
  Logic Levels:           2  (FDRE=1 OBUF=1)
```

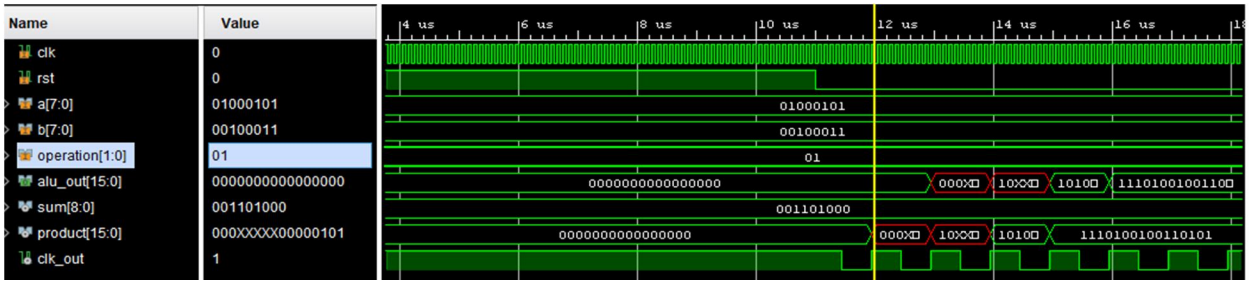
6) Power

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

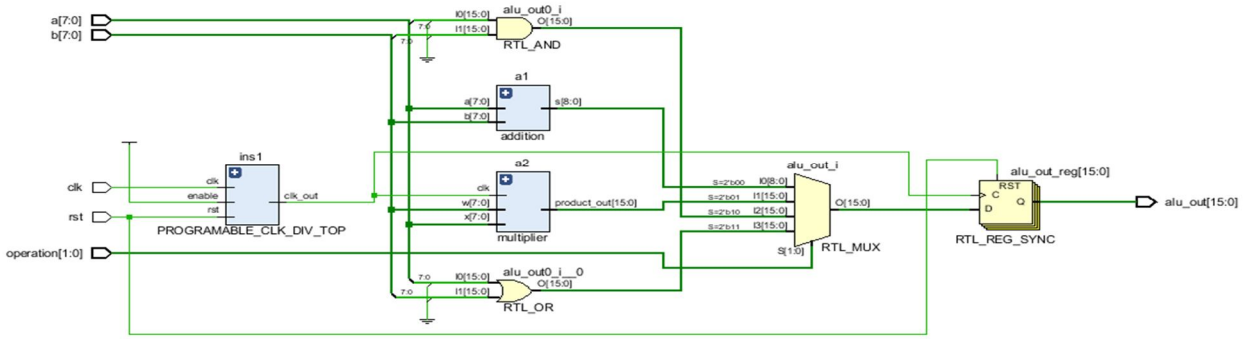
Total On-Chip Power:	**12.179 W (Junction temp exceeded!)**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**85.9°C**

C. Extension Multiplier in ALU

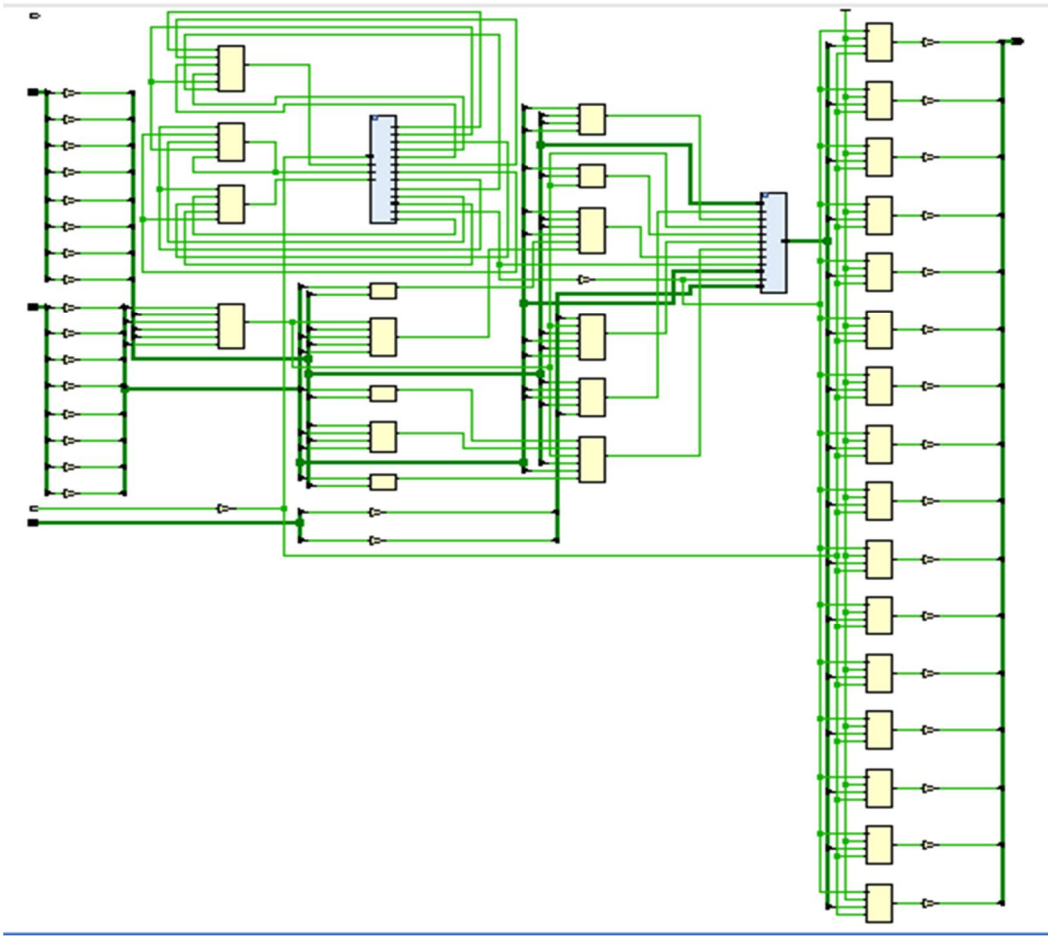
1) Simulation



2) RTL Schematic



3) *Technology Schematic*



4) *Area*

Resource	Utilization	Available	Utilization %
LUT	96	20800	0.46
FF	90	41600	0.22
IO	35	106	33.02

5) *Delay*

```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack:                    inf
  Source:                 alu_out_reg[0]/C
                            (rising edge-triggered cell FDRE)
  Destination:            alu_out[0]
                            (output port)
  Path Group:             (none)
  Path Type:              Max at Slow Process Corner
  Data Path Delay:        4.016ns  (logic 3.216ns (80.087%)  route 0.800ns (19.913%))
  Logic Levels:           2  (FDRE=1 OBUF=1)
```

©IJRASET: All Rights are Reserved

6) Power

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**8.161 W**
Design Power Budget:	**Not Specified**

7) Evaluation table for Area, Delay

	Area	Delay(ns)	Power(W)
Proposed Multiplier	51	4.089	12.56
Extension Multiplier	50	4.016	12.179

VII. CONCLUSION

A radix - 4 modified booth encoder is created in this study. A SPST-based adder in a binary tree structure of addition is used to reduce partial products. SPST with regular adders is been developed after a good trial and analysis, such that area and delay power are optimum. Later, another adder called parallel prefix adders is added to minimise the critical path computation delay by the adder Ladner Fischer adder in order to retain efficiency and acquire optimum parameters. As an application, this extended multiplier with SPST and PPA adder is built in a generic ALU, and a re- programmable clock division circuit is included. All of these designs are written in Verilog HDL and synthesised and simulated with Xilinx Vivado

VIII. FUTURE SCOPE

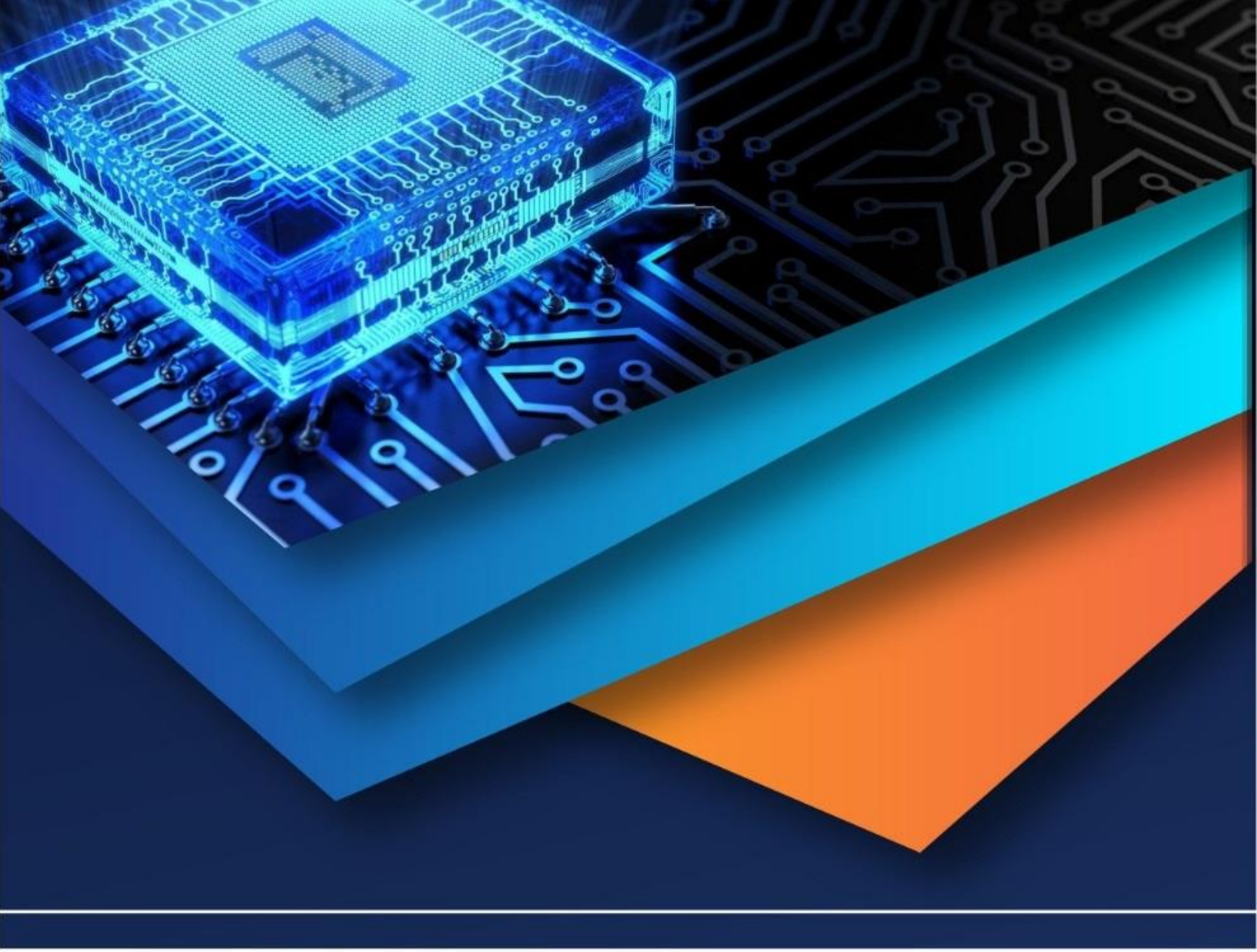
In the future, the proposed multiplier can be enhanced further by simultaneously adding the partial products. By utilising modern technologies in the future, the area and delay can be decreased.

REFERENCES

[1] O. Chen, S. Wang, and Y. W. Wu, “Minimization of switching activities of partial products for designing low-power multipliers,” IEEE Trans.VLSI, vol. 11, no. 3, pp. 418-433, June 2003.
[2] Z. Huang, and M.D. Ercegovac, ”High-performance low-power leftto-right array multiplier design,” IEEE Trans. on Computers, vol.54, no. 3, pp. 272-283, Mar. 2005.
[3] M. C. Wen, S. J. Wang; Y. N. Lin, “Low-power parallel multiplier with column bypassing,” Electronic Letters, vol. 41, no.12, pp. 581- 583, May 2005.
[4] R.Prathiba, P.SandhyaR.Varun,” Design of High Performance and Low Power Multiplier using Modified Booth Encoder”. International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT) – 2016
[5] Shiann-RongKuang, Jiun-Ping Wang, and Cang-Yuan Guo, “Modified Booth Multipliers With a Regular Partial Product Array” , IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS, VOL. 56, NO. 5, MAY 2009
[6] SU MIN CHO, PRAMOD KUMAR MEHER, LUONG TRAN NHAT TRUNG, HYO JIN CHO, AND SANG YOON PARK, “Design of Very High-Speed Pipeline FIR Filter Through Precise Critical Path Analysis”, ieee access, volume 9 2021
[7] W.-C. Yeh and C.-W. Jen, “High-speed booth encoded parallel multiplier design,” IEEE Trans. Comput., vol. 49, no. 7, pp. 692–701, Jul. 2000
[8] J.-Y. Kang and J.-L. Gaudiot, “A simple high-speed multiplier design,” IEEE Trans. Comput., vol. 55, no. 10, pp. 1253–1258, Oct. 2006.
[9] O. Salomon, J.-M. Green, and H. Klar, “General algorithms for a simplified addition of 2’s complement numbers,” IEEE J. Solid-State Circuits, vol. 30, no. 7, pp. 839–844, Jul. 1995.
[10] E. de Angel and E. E. Swartzlander, Jr., “Low power parallel multipliers,” in Workshop VLSI Signal Process. IX, 1996, pp. 199–208.
[11] A. A. Farooqui and V. G. Oklobdzija, “General data–path organization of a MAC unit for VLSI implementation of DSP processors,” in Proc. IEEE Int. Symp. Circuits Syst., 1998, vol. 2, pp. 260–263.
[12] S.-F. Hsiao, M.-R. Jiang, and J.-S. Yeh, “Design of high-speed low-power 3–2 counter and 4–2 compressor for fast multipliers,” Electron. Lett., vol. 34, no. 4, pp. 341–343, Feb. 1998.

[13] Shaik.KalishaBaba ,D.Rajaramesh, "Design and Implementation of Advanced Modified Booth Encoding Multiplier" , International Journal of Engineering Science Invention w ww.ijesi.org Volume 2 Issue 8 ǁ August. 2013 ǁ PP.60-68
[14] Alisha, Tilak raj "Tracing of Delay Estimation in Heterogeneous adders with FPGA" IEEE 2017.
[15] Shubham Sarkar, Sujan Sarkar, JishanMehedi "Comparison of Various adders and their VLSI Implementation" IEEE2018 International Conference on computer communication And Informatics (ICCCI-2018).
[16] K.AnirudhkumarMaurya, K.BalaSindhuri, Y.Ramalakshmanna, N.Udaya Kumar "Design and implementation of 32 bit adder using various Full adders" IEEE (i-PACT2017) International Conference on innovations in power and Advanced computing technology.
[17] L. Qian, C. Wang, W. Liu, F. Lombardi, J. Han, Design and evaluation of an approximate Wallace booth multiplier. In Proceedings of the 2016 IEEE International Symposium on Circuits and Systems (ISCAS), pp 1974–1977, (2016). https://doi.org/10.1109/ISCAS.2016.7538962
[18] N. Ravi, T. Rao, B. Rao, T. Prasad, A new reduced multiplication structure for low power and low area modified booth encoding multiplier. Int. Conf. Model. Optim. Comput. Proced. Eng. 38, 2767–2771 (2012). https://doi.org/10.1016/j.proeng.2012.06.324
[19] S. Ravi, A. Patel, M. Shabaz, P. Chaniyara, H. Kittur. Design of low-power multiplier using ucsla technique. In Artificial Intelligence and Evolutionary Algorithms in Engineering Systems, pp 119– 126, New Delhi. Springer, New York (2015)
[20] A. Saha, R. Pal, A. Naik, D. Pal, Novel cmos multi-bit counter for speed-power optimization in multiplier design. AEU Int. J. Electron. Commun. 95, 189–198 (2018). https://doi.org/10.1016/j.aeue. 2018.08.015
[21] A. K. Sahu, L. Kumre, Low-power less-area bypassing-based multiplier design. In Proceedings of the 2017 International Conference on Inventive Computing and Informatics (ICICI), pp 522–526 (2017). https://doi.org/10.1109/ICICI.2017.8365186
[22] J. Selvakumar, V. C. Bhaskar, Low power and area optimized truncated multiplier architecture. In IET Chennai 3rd International on Sustainable Energy and Intelligent Systems (SEISCON 2012), pp 1–6, (2012). https://doi.org/10.1049/cp.2012.2209

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)