Turkish Journal of Computer and Mathematics Education *Vol.12 No.3(2021), 5673-5683*

Research Article

Implementation and Comparison of Radix-8 Booth Multiplier by using 32-bit Parallel prefix adders for High Speed Arithmetic Applications

Barma Venkata RamaLakshmi[a], Fazal Noorbasha[b]

[a]M.Tech. Scholar, Department of ECE, Koneru Lakshmaiah Educational, Foundation, Guntur, Andhra Pradesh, India
[b]Associate Professor, Department of ECE, Koneru Lakshmaiah Educational, Foundation, Guntur, Andhra Pradesh, India

Article History: Received: 10 November 2020; Revised 12 January 2021 Accepted: 27 January 2021; Published online: 5 April 2021

Abstract: This paper presents the implementation and design of Radix-8 booth Multiplier using 32-bit parallel prefix adders. High performance processors have a high demand in the industrial market. For achieving high performance and to enhance the computational speed multiplier plays a key role in performance of digital system. But the major drawback is it consumes more power , area and delay. To enhance the performance and decrease the area consumption and delay there are many algorithms and techniques. In this paper we designed a radix-8 Booth Multiplier using two parallel prefix adders and compared them for best optimized multiplier. The number of parital products generation can be reduced by n/3 by using radix-8 in the multiplier encoding. To further reduce the additions we have used booth recoding mechanism .We have implemented the design using Kogge stone adder and Brent kung adder. We observed that by using parallel prefix adders reduces the delay further more which results in significant increase in speed of the digital systems. The simulation results are carried out on XILINX VIVADO software.

Keywords: Booth multiplier ; parallel prefix adder ; radix-8 ; computational speed

1. Introduction

Now a day's high performance processor has a high demand in the industrial market. For achieving high performance processors arithmetic operations like addition, multiplication, subtraction is invoked in various digital circuits to enhance the computational speed [1]. Multiplier is the slow element in the system, the system overall performance is determined by the performance of a multiplier. The multiplier is also area consuming, and has high power dissipation which effects the overall performance of the system. Multipliers are used in DSP, lossy applications, artificial neural networks, machine learning and many more . To enhance the performance of the multiplier we have many power reduction algorithms and also various multipliers. Booth algorithm takes less time of computation, it takes less area for the design and also power consumption is very less. To reduce the power consumption booth recoding algorithm is used[2-4]. The performance can be achieved by modified model with the help of parallel prefix adders[2]. The entire performance can be increased by using parallel prefix adders. The delay is another factor which can impact on the overall performance of the multiplier. By using parallel prefix adders we can also reduce the time delay in multiplication process. Hence we can achieve a high speed, high performance and low power consuming multiplier.

The multiplication operation has two major steps, those are

a. Partial products generation.

b. Partial products addition.

Multiplier speed can be improved by partial products reduction and improving the speed of summation of partial products.

2. Booth Algorithm

The booth algorithm was first introduced by Andrew Donald Booth in 1950, London. This algorithm was a study of computer architecture. This booth algorithm multiplies two signed binary digits in 2's complement form. This algorithm was introduced while doing research on crystallography. In booth algorithm less number of additions and subtractions can be observed. Booth multiplier is the faster multiplier in doing computations. This booth algorithm is widely used in ASIC products due to its smaller area and high computational speed[7-8].

The major steps in booth algorithm are:

a. Generating the partial products

b. Reducing the partial products

c. Summation of the partial products

In booth algorithm, the generation of partial products depends on recoding mechanism. The process uses booth recoding method based on the partial products that can be created for a set of 0`s and 1`s this is called Booths recoding. The main aim of this algorithm is to generate Partial products efficiently. There will be an rise of partial product[6] which depends on the Radix used for recoding. This recoding mechanism provides less power and area.

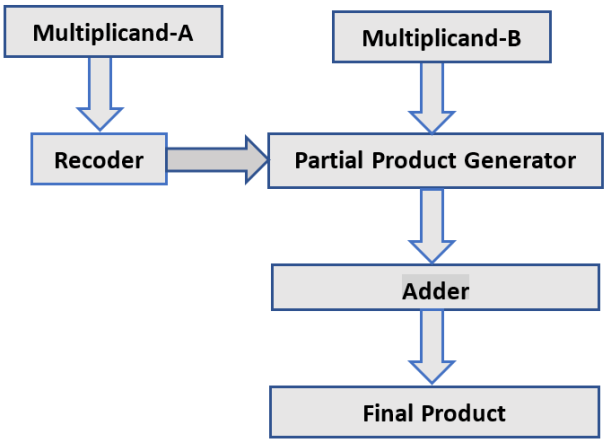


Fig. 1: Block diagram of booth algorithm

Fig. 1 represents the block diagram of booth algorithm. It will multiply 2 signed Binary digits in 2's compliment form.

In this booth algorithm we have four blocks they are working as follows:

i.Encoder:

The digital block encoder performs the opposite working of a decoder. It is having 2^n inputs and n-outputs. Here in the booth algorithm we have given the input of encoder as multiplicand B , the output of encoder has given to the partial product generator. The output of encoder is corresponding to the binary code of input value.

ii. Partial Product Generator:

The main purpose of the partial product generator is to simplify the computational process in multiplication. Here the partial products can be generated for a set of 0`s and 1`s. the partial product reduction helps in improving the speed of summation of partial products.

iii. Adder:

After the generation and decrease of partial products the next step will be summation of the

partial products. We can use carry save adder or carry look ahead adder to do this step. There are many other options available some of them are compressor adders, wallance tree adder, parallel prefix adders like kogge stone adder etc.,. By properly choosing an adder we can reduce the delay parameter.

iv. Final product:

The inputs multiplicand A and multiplicand B can be taken as two signed binary numbers , the final product will be the multiplied 2`s compliment number

Radix-8 booth algorithm:

This algorithm is similar to that of radix 4, but the only Difference is that we consider quartets of bits rather than that of triplets. This algorithm minimizes the number of partial products to n/3 where as in radix 4 it is reduced to n/2. This minimization of Partial products leads to fast computational speed and less power and area. Table1 represents the booth encoding table of Radix 8 booth recoding algorithm.[5]

Multiplier Bits				**Operation on Multiplicand**
A	**B**	**C**	**D**	**X**
0	0	0	0	0X
0	0	0	1	+1X
0	0	1	0	+1X
0	0	1	1	+2X
0	1	0	0	+2X
0	1	0	1	+3X
0	1	1	0	+3X
0	1	1	1	+4X
1	0	0	0	-4X
1	0	0	1	-3X
1	0	1	0	-3X
1	0	1	1	-2X
1	1	0	0	-2X
1	1	0	1	-1X
1	1	1	0	-1X
1	1	1	1	0X

Table 1: Radix 8 Booth algorithm Encoding table

For example , an 8x8 bit radix 8 considering the signed bit as 1 and x as input data of 8-bit ; y as input data of 8-bit and k as the output data , x=11111111 y= 11111111 then k = 1111111111111111.

3. Parallel Prefix Adders

The parallel prefix adders is another form of carry look ahead adder. The main advantage of the parallel prefix adders is that we can avoid the higher delay problem which we can observe in the existing carry adders[13-15]. The prefix adders can be designed in many different ways. Now a days, tree structure kind of adders are used to improve the addition function speed in processors. The parallel prefix adders are also known as Logarithimic delay adders .

The process of addition in parallel prefix adders takes place in 3 stages:

a) Pre-computation stage.

b) Intermediate stage.

c) Final computation stage.

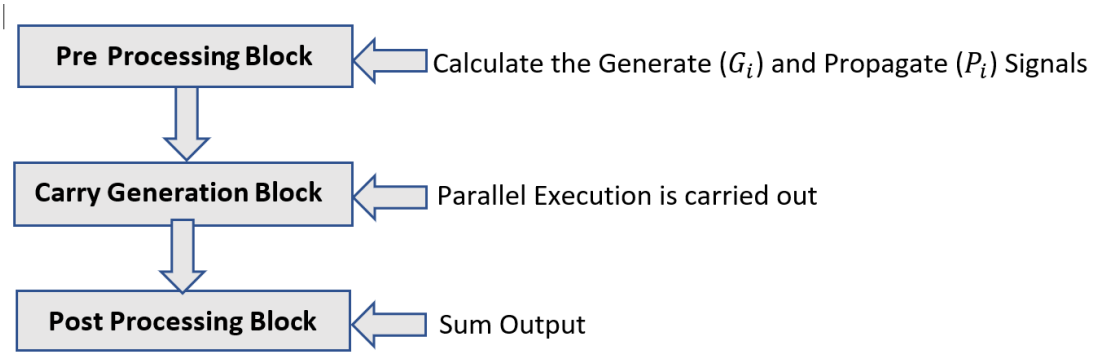


Fig 1: process of parallel prefix adder

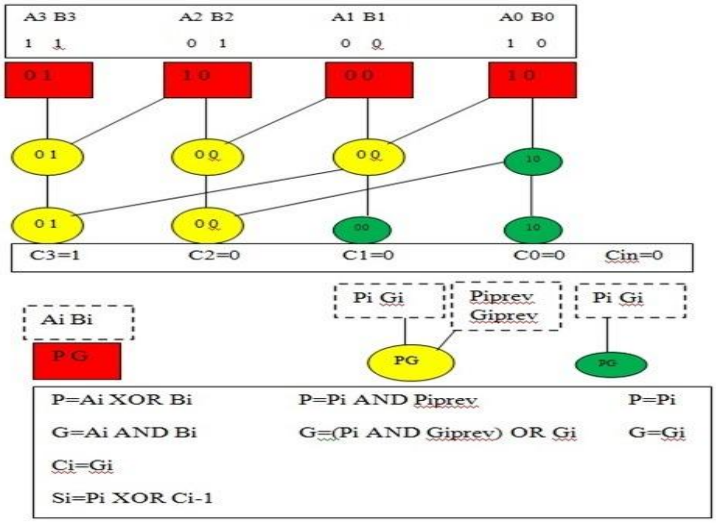


Fig 2: Structure of Parallel prefix adder using Koggestone adder.

- In the precomputation stage, the generate functions and propagate functions are calculated with respect to the inputs given.

The propagate function are given by the equation,

$$P = Ai \text{ XOR } Bi$$

Where the Ai and Bi are the inputs which are composed by XOR logic.

The generate function are given by the equation,

$$G = Ai \text{ AND } Bi$$

Where the Ai and Bi are the inputs which are composed by AND logic.

As the P and G are done in parallel there doesn't exists any increase in area consumption but the delay parameter will be completely depends on the bit length we considered.

- In Intermediate stage the carry propagation and carry generation takes place. As the carry signal uses more than two inputs the increase in delay factor may be observed.

- In final computation stage, it computes the summation of given inputs and generates the output.

4. PROPOSED DESIGN :

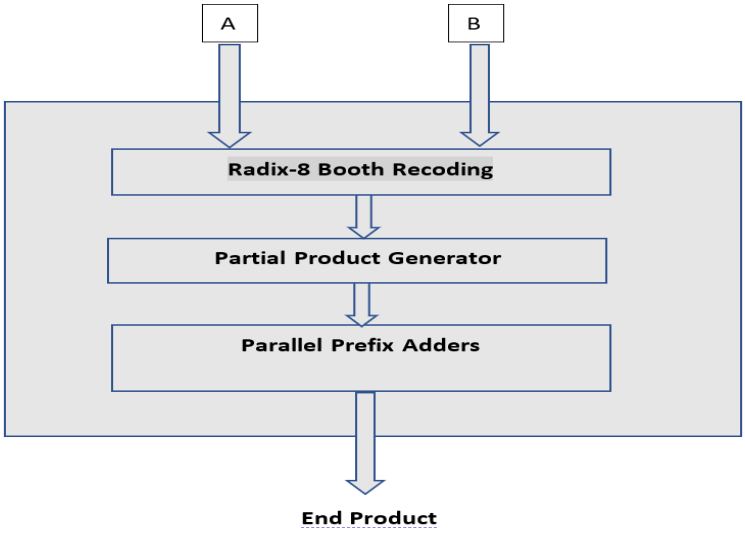
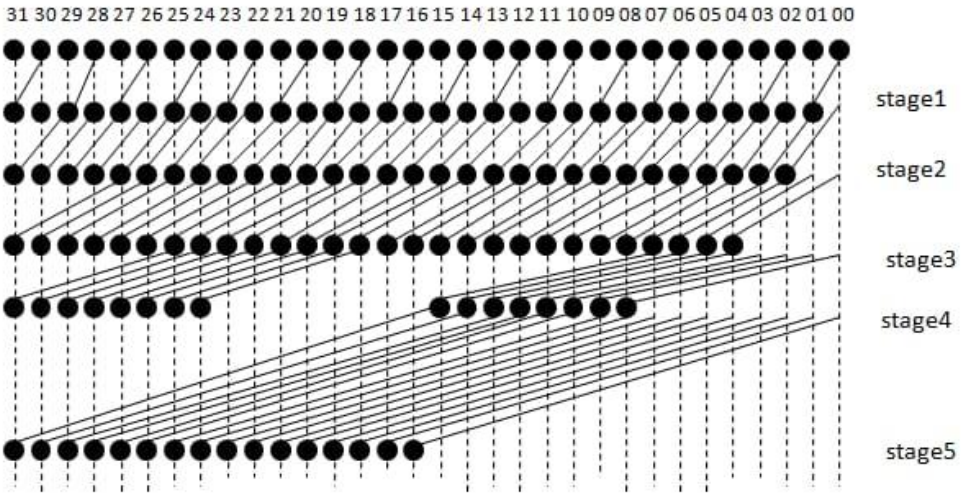


Fig 3: Block diagram of radix-8 booth multiplier using parallel prefix adders.

The fig 3 represents the block diagram of radix-8 booth multiplier using parallel prefix adder.

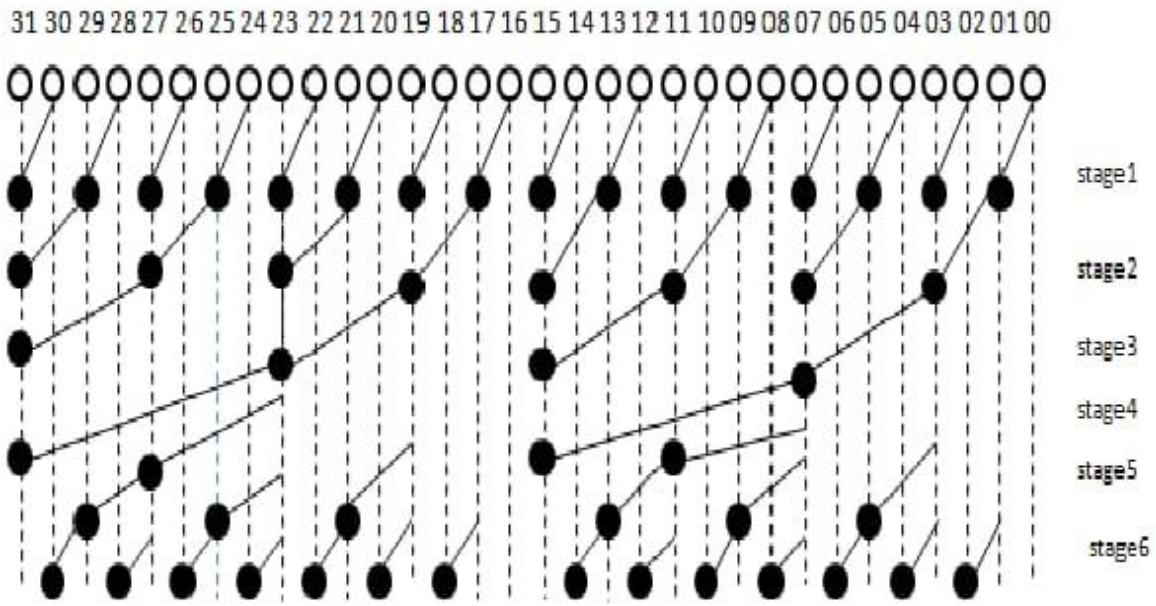
To enhance the performance of the multiplier we need to reduce the number of additions. Booth multiplication is a algorithm that provides fast multiplication by recoding the bits that are multiplied. Where as the normal conventional multiplier uses a large number of partial products. By using the booth algorithm it will decrease the additions required to give the output by using the recoder. So in order to obtain the high performance we have used the booth recoding mechanism and to reduce the number of partial products to n/3, Radix-8 booth multiplier is considered[9-11]. By using parallel prefix adders we can also reduce the time delay in multiplication process. Hence we can achieve a high speed, high performance and low power consuming multiplier[12-14].

KOGGE STONE ADDER:



- The Kogge stone adder is one of the parallel prefix adder.
- It was introduced by peter M.Kogge and Harlond S.Stone in 1973. It is a fast adder design[22].
- the kogge stone adder has best performance in VLSI design implementations.
- It is widely known as parallel prefix adder that performs very fast addition.

BRENT KUNG ADDER:



- Richard P.Brent and H.T.Kung developed Brent Kung Adder in t 1982. It is a parallel prefix adder which gives a minimal number of stages .
- This adder uses a less number of propagating and generate functions than the other adders.
- The increase in performance in Brent-Kung adders is because of its tree structure which also leads to lower power consumption as of fewer stages.

The proposed design is to implement and simulate the Radix-8 booth multiplier using 32-bit Kogge Stone adder and 32-bit Brent kung adder[16-18] and compared the results in terms of Power , Area and Delay. The simulation results are carried out on XILINX VIVADO tool.

5. Experimental Observations:

The experimental observations were carried out on Xilinx Vivado 2016.4 the HDL code was written in Verilog. Below shows the Technology schematic, RTL Schematic and wave forms of Radix-8 booth multiplier using 32-bit Kogge stone adder and Radix-8 Booth multiplier using 32-bit Brent kung adder.

1.Results of Radix-8 Booth Multiplier using Kogge stone adder:

i.Simulation timing diagram:

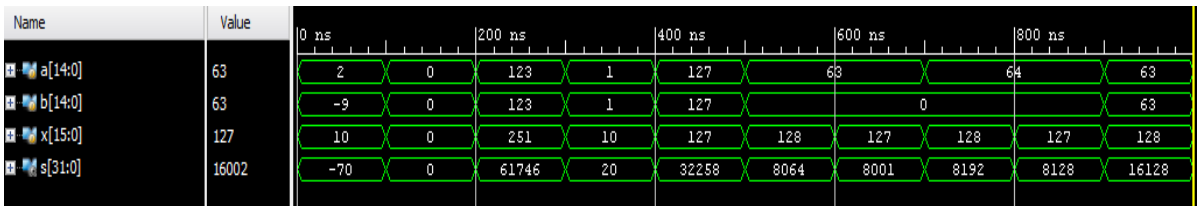


Fig 4: simulation timing diagram of radix-8 booth multiplier using 32-bit Kogee stone adder

The figure 4 represents the simulation result of radix-8 booth multiplier using 32-bit Kogee stone adder. The inputs are { a, b, x} and the output is represented as { s } . If input a and b is given as 1 and 1 then there takes place an addition i.e a+b for the given values it will generate the output as 2 . And assigned the input x value as 10 . The there takes place an multiplication between 2 and 10. hence the end result is finally generated as 20.

ii. RTL Schematic:

The Fig. 5 represents the RTL schematic of radix-8 booth multiplier using 32-bit Kogee stone adder, in which it gives the information regarding logic of the design in the form of symbols like adders , multipliers etc.

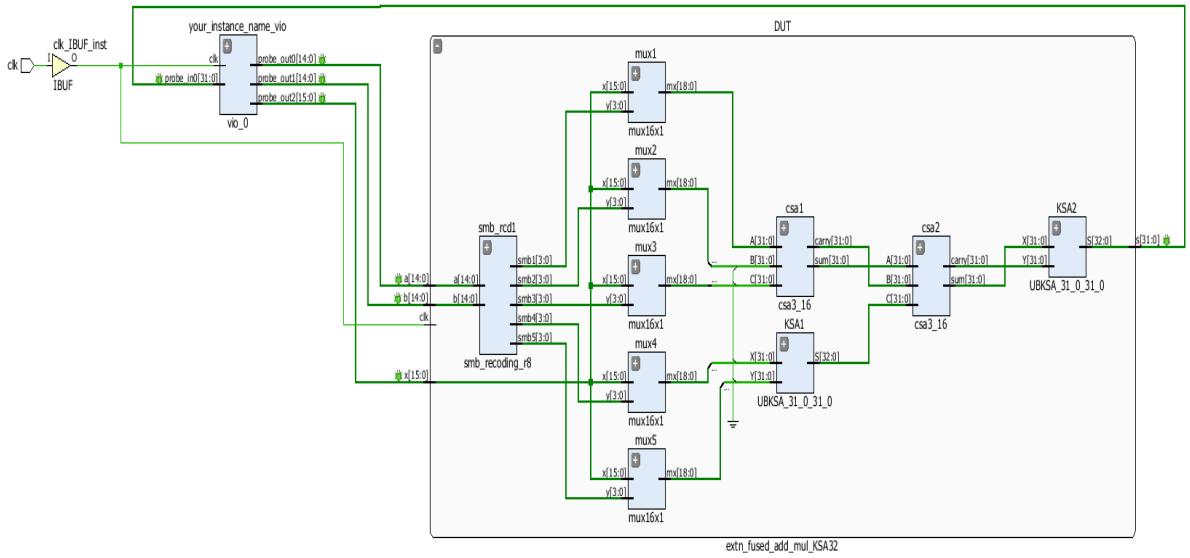


Fig 5: RTL schematic of radix-8 booth multiplier using 32-bit kogge stone adder.

iii. Technology Schematic:

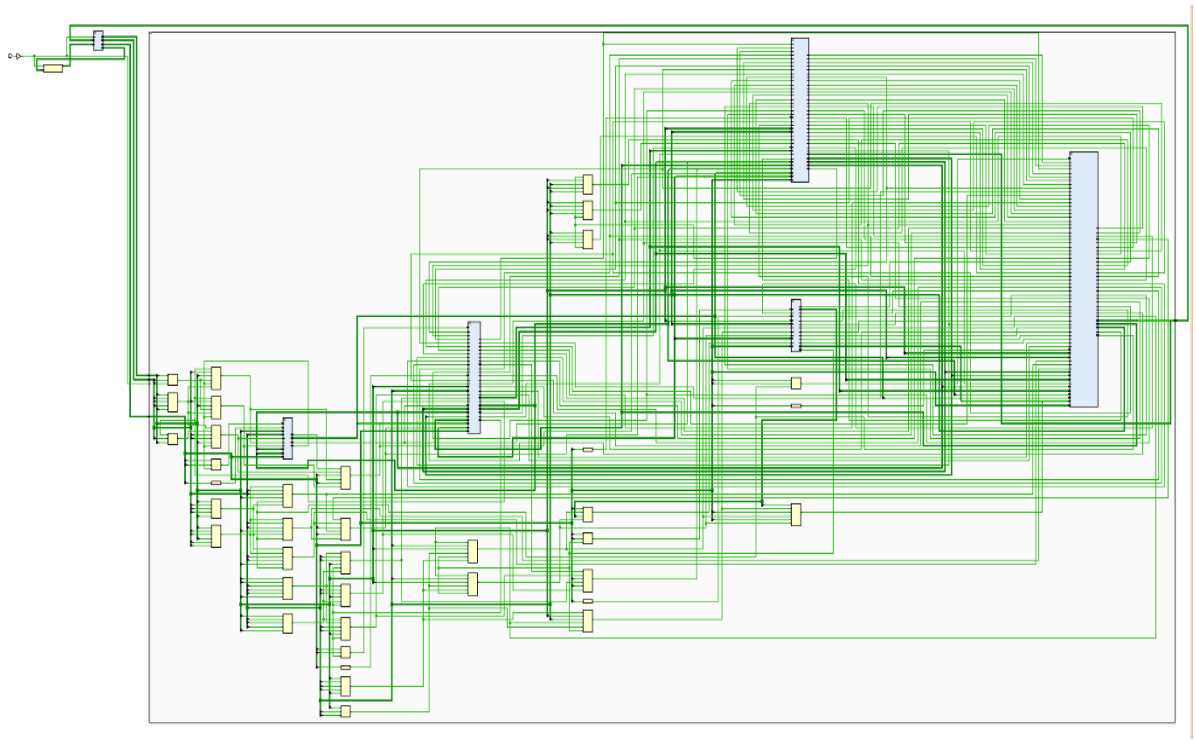


Fig 6: Technology schematic of radix-8 booth multiplier using 32-bit kogge stone adder.

- The Fig.6 represents the Technology schematic of radix-8 booth multiplier using 32-bit Kogee stone adder it gives the information regarding the logic of the design when it is targeted to specific device.

2.Results of Radix-8 Booth Multiplier using 32-bit Brent kung adder:

i.Simulation Timing diagram:

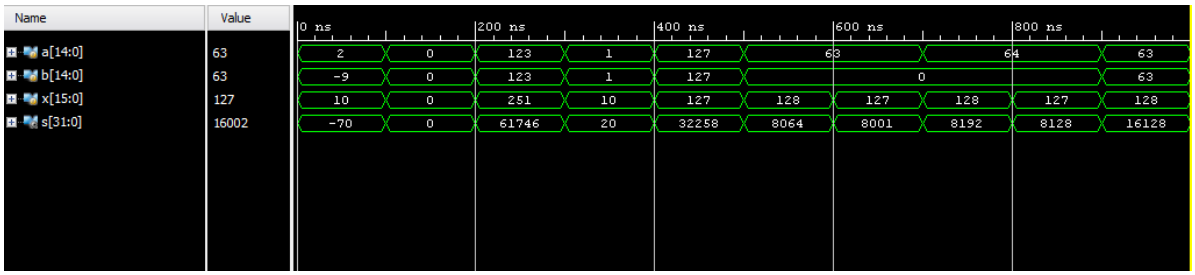


Fig 7: simulation timing diagram of radix-8 booth multiplier using 32-bit Brent Kung adder

The figure 7 represents the simulation result of radix-8 booth multiplier using 32-bit Brent Kung adder. The inputs are { a, b, x} and the output is represented as { s } . If input a and b is given as 2 and -9 then there takes place an addition i.e a+b for the given values it will generate the output as -7 . And assigned the input x value as 10 . The there takes place an multiplication between -7 and 10. hence the end result is finally generated as

-70.

ii. RTL Schematic:

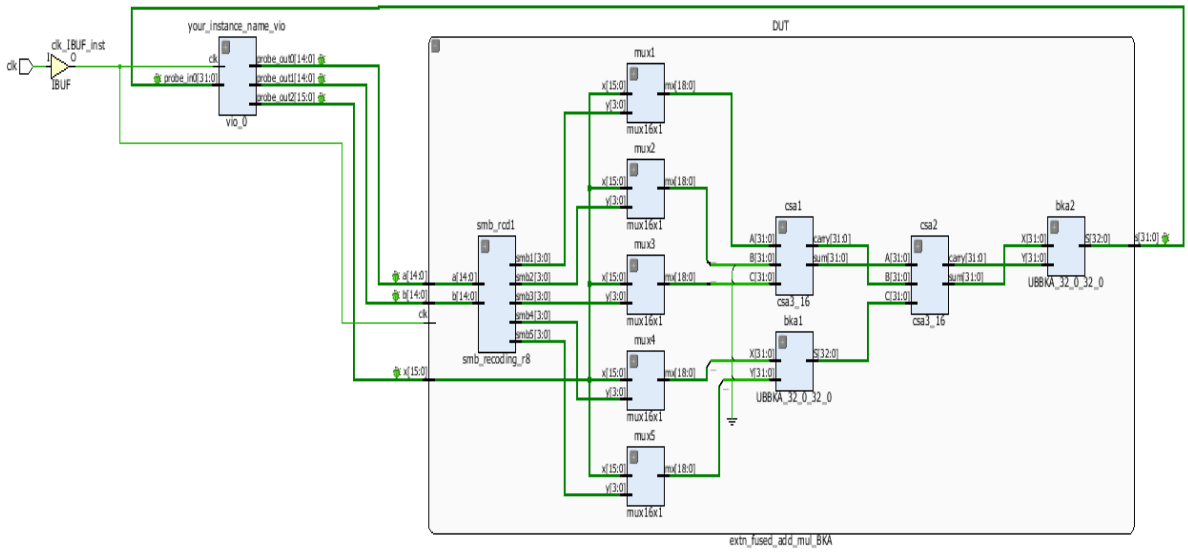


Fig 8 : RTL schematic of radix-8 booth multiplier using 32-bit Brent kung adder

The fig 8 represents the RTL Schematic of Radix-8 Booth Multiplier using 32-bit brent kung adder. The RTL schematic gives the information regarding logic of the design in the form of symbols like adders , multipliers etc.

iii. Technology schematic:

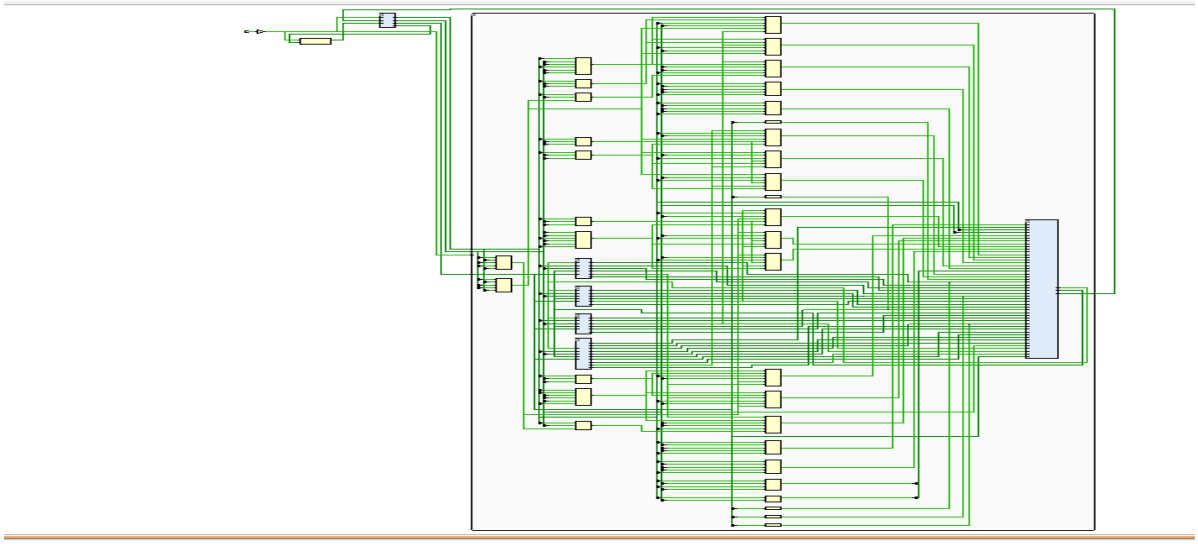


Fig 9: Technology schematic of radix 8 booth multiplier using 32-bit Brent kung adder.

The Fig.9 represents the Technology schematic of radix-8 booth multiplier using 32-bit Brent Kung adder . It gives the information regarding the logic of the design when it is targeted to specific device.

6. Performance Comparison

The device utilization summaries represents the number of LUT, slices, muxes, flip-flops, bonded IOB etc., are utilized and available in the design by which we can have a knowledge of how much area in the chip is utilized by our design. The device utilization summary of Radix-8 booth multiplier using 32-bit Koggestone adder can be observed in Table 2 and Radix-8 booth multiplier using 32-bit Brent kung adder can be observed in Table 3.

Table 2: Device utilization of Radix-8 booth Multiplier using 32-bit Kogge stone adder.

Resource	Utilization	Available	Utilization %
LUT	1239	53200	2.33
LUTRAM	24	17400	0.14
FF	1019	106400	0.96
IO	1	200	0.50
BUFG	2	32	6.25

The table 2 represents the Device utilization of Radix-8 booth Multiplier using 32-bit Kogge stone adder it gives the information about the number of LUT , LUTRAM, FF etc., that are utilized by our Radix-8 booth Multiplier using 32-bit Koggestone adder.

Table 3: Device utilization of Radix-8 booth Multiplier using 32-bit Brent Kung adder.

Resource	Utilization	Available	Utilization %
LUT	1065	53200	2.00
LUTRAM	24	17400	0.14
FF	1019	106400	0.96
IO	1	200	0.50
BUFG	2	32	6.25

The table 3 represents the Device utilization of Radix-8 booth Multiplier using 32-bit Brent kung adder it gives the information about the number of LUT , LUTRAM, FF etc., that are utilized by our Radix-8 booth Multiplier using 32-bit Brent kung adder.

Table 4: LUT, Delay and Power comparison of Radix-8 booth Multiplier using 32-bit KoggeStone adder and Brent Kung adder.

Parameter	Radix-8 booth Multiplier using 32-bit Kogge stone adder [Proposed design-1]	Radix-8 booth Multiplier using 32-bit Brent Kung adder [Proposed design-2]
LUT	1239	1065
DELAY [nS]	15.173	14.124
POWER [mW]	123	123

The table 4 represents comparison oof Radix-8 booth Multiplier using 32-bit Kogge stone adder and Radix-8 booth Multiplier using 32-bit Brent kung adder in terms of LUT, Delay and Power parameters. here, we can clearly observe that the Radix-8 booth multiplier using 32-bit Brent kung adder i.e., Proposed design-2 gives the best optimized results in terms of LUT and delay when compared with the Proposed design-1.

Table 5: Other parameters comparison of proposed design-1 and proposed design-2.

Parameter	Radix-8 booth Multiplier using 32-bit Kogge stone adder [Proposed design-1]	Radix-8 booth Multiplier using 32-bit Brent Kung adder [Proposed design-2]
Setup time [nS]	27.167	27.511
Hold time [nS]	0.087	0.049
Pulse width [nS]	15.250	15.250
Period [nS]	33	33
Clock Frequency [MHz]	30.303	30.303

Table 6: Results comparison with existed work

Parameters	Floating point booth Multiplier [16]	Radix-8 booth Multiplier by 32-bit Kogge stone adder [Proposed design-1]	Radix-8 booth Multiplier by 32-bit Brent Kung adder [Proposed design-2]
LUT	1903	1239	1065
DELAY (nS)	22.763	15.173	14.124
POWER (mW)	160.85	123	123

The Table 6 represents the results comparision with existed work in terms of LUT , power and delay. It clearly shows that our Proposed designs has succeeded to give the best optimized results when compared with the already existed work [16] in terms of LUT , Delay and power.

7. Conclusion:

In this paper, an efficient high speed[23-24] Radix-8 Booth multiplier using 32-bit kogge stone adder and 32-bit Brent kung adder is designed successfully using Xilinx Vivado software. By using Radix-8 booth recoding algorithm we have reduced the number of partial products generation by n/3 and also decreased the addition operations by which we have achieved less area consumption and better performance. The parallel prefix adders helped in reducing the delay which is a great advantage.[19-21] The 32-bit kogge stone adder and 32-bit Brent Kung adder in our proposed design shown significant decrease in the delay and power consumption. The synthesis report shows that among the kogge stone adder and Brent kung adder the radix-8 booth multiplier using 32-bit

Brent kung adder achieved less delay and less area consumption. This work can be extended for higher number of bits.

References

L.P. Rubinfield, “A Proof of the Modified Booth's Algorithm for Multiplication,” IEEE Transaction on computers, vol.39.

“Parallel-prefix structures for binary and modulo {2^n -1, 2^n, 2^n + 1} adders”, September 2011 by Jun Chen.

“Embedded Cryptographic Hardware: Methodologies and Architectures”, august 2012 by Nadia Nedjah, Luiza de Macedo mourelle.

Shivanand pariwar and Raj singh “Efficient Floating Point 32-bit single Precision Multipliers Design using VHDL”, Pilani, Engineering and Technology 2011.

J.A. Hidalgo, V. Moreno-Vergara, O. Oballe, A. Daza, M.J. Martín-Vázquez, A.Gago ,“A Radix-8 multiplier design for specific purpose”@2011.

X. Cui, W. Liu, X. Chen, E. E. Swartzlander, and F. Lombardi, ‘‘A modified partial product generator for redundant binary multipliers,’’ IEEE Trans. Comput., vol. 65, no. 4, pp. 1165–1171, Apr. 2016.

J. M. D. Moss, D. Boland, and H. W. P. Leong, ‘‘A two-speed, radix-4, serial–parallel multiplier,’’ IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 27, no. 4, pp. 769–777, Dec. 2018.

H. Jiang, J. Han, F. Qiao, and F. Lombardi, ‘‘Approximate radix-8 booth multipliers for low-power and high-performance operation,’’ IEEE Trans. Comput., vol. 65, no. 8, pp. 2638–2644, Aug. 2016.

lakshmanan, m. othman, m.a.m. ali, “design and characterization of parallel prefix adders using fpgas,” journal of computers, vol. 5, no. 10, october 2012.

S. Venkatachalam, E. Adams, H. J. Lee, and S.-B. Ko, ‘‘Design and analysis of area and power efficient approximate booth multipliers,’’ IEEE Trans. Comput., vol. 68, no. 11, pp. 1697–1703, Nov. 2019.

L.P. Rubinfield, “A Proof of the Modified Booth's Algorithm for Multiplication,” IEEE Transaction on computers, vol.39.

Y. Choi, "Parallel Prefix Adder Design", Proc. 17th IEEE Symposium on Computer Arithmetic, pp. 90-98, 27th June 2005

Y. Choi, "Parallel Prefix Adder Design", Proc. 17th IEEE Symposium on Computer Arithmetic, pp. 90-98, 27th June 2005.

Basant Kumar Mohanty and Sujit Kumar Patel “Area–Delay–Power Efficient Carry-Select Adder,” IEEE transaction on circuits and systemsII: Express briefs, VOL. NO. 6, JUNE 2014

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

S. Daphni, K. S. Vijula Grace, “A review analysis of parallel prefix adders for better performance in VLSI applications” in Proceedings of 2017 IEEE International Conference on Circuits and Systems (ICCS), 2017. https://doi.org/10.1109/ICCS1.2017.8325971

R. Zlatanovici, S. Kao, and B. Nikolic, “Energy–delay optimization of 64-bit carry-lookahead adders with a 240 ps 90 nm CMOS design example,” IEEE J. Solid-State Circuits, vol. 44, no. 2, pp. 569–583, Feb. 2009.

J. Thomas, R. Pushpangadan, S. Jinesh, “Comparative Study of Performance Vedic Multiplier on The Basis of Adders Used” IEEE International WIE Conference on Electrical and Compter Engineering (WIECON-ECE), 2015. https://doi.org/10.1109/wiecon-ece.2015.7443929

F. Liu, Q. Tan, G. Chen, “Formal proof of prefix adders” Mathematical and Computer Modelling, ELSEVIER, 2010. https://doi.org/10.1016/j.mcm.2010.02.008

M. Macedo, L. Soares, B. Silveira, C. M. Diniz, Eduardo A. C. da Costa, "Exploring the use of parallel prefix adder topologies into approximate adder circuits” in 24th IEEE International Conference on Electronics, Circuits and Systems (ICECS), 2017. https://doi.org/10.1109/icecs.2017.8292078

V. G. Oklobdzija, B. R. Zeydel, H. Q. Dao, S. Mathew, and R. Krishnamurthy, “Comparison of high-performance VLSI adders in the energy-delay space,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 13, no. 6, pp. 754–758, Jun. 2005.

P. M. Kogge & H. S. Stone, “A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations,” IEEE Trans. Computers, Vol. C-22, pp 786-793, 1973.

Shivani Parmar and Kirat Pal Singh," Design of High Speed Hybrid Carry Select Adder", IEEE's 3rd International Advance Computing Conference (IACC) Ghaziabad, ISBN: 978-1- 4673-4527-9,22-23 February 2013.

Yajaun He, Chip-Hong Chang, and Jiangmin Gu, "An area efficient 64- Bit square Root carry-select adder for low power Applications, " in Proc. IEEE International Symposium Circuits and Systems, vol. 4, pp. 4082-4085,2005.