Speed Multiplication of Unsigned Binary Numbers using Radix-4 Algorithm

L.Arul Leo Felix[1],A.Rishikesh[2], S.Asvika[3], G.Pasupathi[4], M.Hariharan[5]

Assistant Professor, Department of Electronics and Communication Engineering, Trichy Engineering College, Sivagnanam Nagar, Konalai, Trichy, Tamil Nadu, India[1]

U.G. Students, Department of Electronics and Communication Engineering,Trichy Engineering College, Sivagnanam Nagar, Konalai, Trichy, Tamil Nadu, India [2,3,4& 5]

ABSTRACT: To speed up the process of applications such as digital filters, artificial neural networks, and other machine learning algorithms, we choose a two-speed, radix-4, SP multiplier. Our multiplier may be a variant of the modified radix-4 Booth multiplier that adds only the nonzero Booth encodings and skips over the zero operations, making the latency dependent of the multiplier value. Two sub-circuits with different critical paths are utilized so that throughput and latency are improved for a subset of multiplier values. The implementation of multiplication operation has been implemented by using an intel cyclone V field-programmable gate array from standard PP and SP multipliers. We show that for bit widths of 32 and 64, our optimizations can result in a 1.42x -3.36x improvement over the standard parallel booth multiplier in the terms of area-time depending on the input set.

KEYWORDS: Booth Multiplier, Field-Programmable Gate Array (FPGA), machine learning (ML), neural networks**.**

I.INTRODUCTION

Very-large-scale integration (VLSI) is that the tactic of making a combinational circuit by combining thousands of transistors hooked on one chip. Multiplication may be a basic mathematic process which is present in many parts of the computer especially in signal processing systems like graphics and computation system. It requires more hardware resources and time interval than addition and subtraction. There is continuous development in VLSI technologies. As the scale of integration keeps rising, gradually cultured signal processing system is being executed on a VLSI chip. These signal processing applications not solitary demand great computation volume but also consume a significant amount of energy. Multipliers play a crucial role in today's digital signal processing and various other applications. With advances in technology, many researchers have tried and try to style multipliers which supply either of the subsequent design targets-high speed, low power consumption, the regularity of layout and hence fewer area or possibly a mixture of them in one multiplier thus making them suitable for various high speed, low power, and , and compact VLSI implementation. The higher speed results in enlarged power consumption, thus, low power architecture is getting to be the choice of the long run. Low power design directly results in prolonged operation time in these portable devices. Multiplication can be done serially or parallel. Theoretically multiplication can be done by repeated addition. Consider multiplication A x B where A is multiplier and B is the multiplicand, if we add B to itself A times the sum are going to be the merchandise of A x B. Practically this process is very slow so never been used. This method involves generating intermediate products then adding them properly taking under consideration the load of every bit while moving from LSB to MSB.

II. RELATED WORK

The work on the optimization of multiplication circuits has been extensive [1], [2], however, the modified Booth algorithm at higher radixes in combination with Wallace or Dadda tree has generally been accepted as the highest performing implementation for general problems [2]–[4]. In digital circuits, multiplication is generally performed in one of three ways: 1) parallel–parallel; 2) serial-parallel (SP); and 3) serial–serial. Using the modified Booth algorithm [5], , we explore an SP two-speed multiplier (TSM) that conditionally adds the nonzero encoded parts of the multiplication and skips over the zero encoded sections. In DSP and ML implementations, reduced precision representations are often used to improve the performance of a design, striving for the smallest possible bit width to achieve a desired computational accuracy. Precision is usually fixed at design time, and hence, any changes in the requirements necessitate that further modification involves the redesign of the implementation. In cases where a smaller bit width would be sufficient, the design runs at a lower efficiency since unnecessary computation is undertaken. To

mitigate this, mixed-precision algorithms attempt to use a lower bit width some portion of time, and a large bit width when necessary. These are normally implemented with two data paths operating at different precisions.
The keyaids of this paper are as tracks.
1) The first serial modified Booth multiplier where the data path is divided into two sub circuits, each operating with a different critical path.
2) Demonstrations of how this multiplier takes advantage of particular bit-patterns to perform less work; this results in reduced latency, increased throughput, and superior area–time performance than conventional multipliers.
3) A model for estimating the performance of the multiplier and evaluation of the utility of the proposed multiplier via an FPGA implementation.

In parallel multipliers number of partial products to be added is that the main constraint that controls the performance of the multiplier to measure back the number of partial products to be added, modified Booth algorithm is one among the leading widespread algorithms. Online arithmetic has been broadly studied for ASIC implementation. Online components were primarily intended to perform calculations in digit serial with most vital digit (MSD) first, important to the power to chain mathematic operators organized for low latency. In serial-serial multipliers both the operands are loaded in bit -serial manner, to scale back data input pads two. Whereas in serial-parallel multiplier loads one operand during a bit-serial fashion and therefore the other is always available in a parallel fashion. for normal add shift algorithm, every multiplier bit gives one multiple of the multiplicand which is added to partial products. A more number of multiplicands are added, if the multiplier is extremely large. during this state the amount of additives to be performed governs the delay of multipliers. The performance will get well, if the amount of additives is minimum. That the parallel multipliers are plentiful options than the serial multiplier. Just in case of parallel multipliers, the complete area is far but that of serial multipliers. Hence the facility consumption is additionally less. it's lesser delay and good noise immunity.

III.EXISTING SYSTEM

Multiplication is a critical primitive that often dictates the performance of large DSP applications. majority of hardware optimizations for ML is focused on reducing the cost of the multiply and accumulate operations. Hence, careful construction of the compute unit, with a focus on multiplication, leads to the largest performance impact. Other work such as has focused on specialized multiplication structures for the Galois field multiplication. Ten different multiplier alternatives are explored and compared to a reference architecture. The different strategies for combining integer and the Galois field multiplication show area savings up to 20% with only a marginal increase in delay and an increase in power consumption of 25%. Recent work in a bit and digit serial multiplication for FPGAs has focused on online arithmetic and efficient mapping of the algorithms to the FPGA architecture. Shi et al analyzed the effect of overclocking radix-2 online arithmetic implementations and quantified the error introduced by timing violations. They found a significant reduction in error for DSP-based applications compared with conventional arithmetic approaches.

Zhao presented a method for achieving arbitrary precision operations utilizing the on-chip block RAMs to store intermediate values. The most comparable work to this multiplier is the parallel-serial, or shift-add, multiplier. As described in the product p is iteratively calculated by examining individual bits of X each cycle and accumulating a scaled. Recent work in a bit and digit serial multiplication for FPGAs has focused on online arithmetic and efficient mapping of the algorithms to the FPGA architecture. Analyzed the effect of overclocking radix-2 online arithmetic implementations and quantified the error introduced by timing violations. They found a significant reduction in error for DSP-based applications compared with conventional arithmetic approaches. Presented a method for achieving arbitrary precision operations utilizing the on-chip block RAMs to store intermediate values. Furthermore, Rashidi proposed a modified retiming serial multiplier for finite impulse response (FIR) digital filters based on ring topologies. The work involved additional logic which allowed for modification of the scheduling of the FIR filter computation, allowing the number of compute cycles to be reduced from 32 to 24.

To further improve the performance of the FIR filter computation, the author proposed a high-speed logarithmic carry look-ahead adder to work in combination with a carry-save adder. While the TSM is suited for ML and applications with high degrees of sparsity, it differs from the previous research in that the multiplier performs standard signed multiplication and can be used in any application. Our contribution is a new control structure for performing multiplication that dynamically avoids unnecessary computation. In mathematics, a finite field or Galois field is a field that contains a finite number of elements. As with any field, a finite field is a set on which the operations of multiplication, addition, subtraction, and division are defined and satisfy certain basic rules. The most common examples of finite fields are given by the integers mod p when p is a prime number. Finite fields are fundamental in several areas of mathematics and computer science, including number theory, algebraic geometry, Galois theory, finite geometry, cryptography and coding theory.

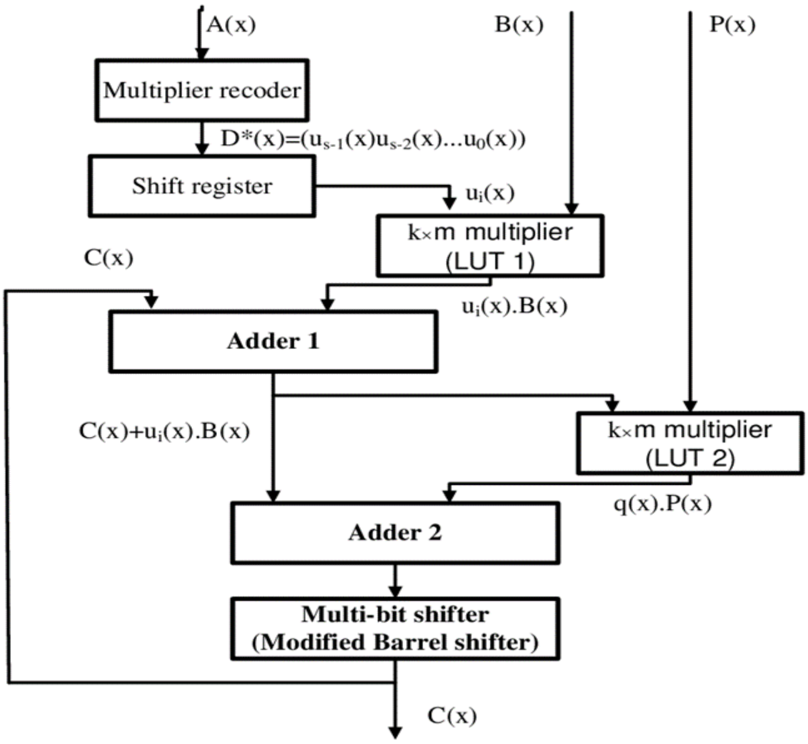


Fig 1 : Block diagram for Galois field multiplication

In digital electronics, a binary decoder is a combinational logic circuit that alters binary data from the n coded inputs to an extreme of 2n unique outputs. In digital circuits, a shift register is a waterfall of flip flops, allotment the same clock, in which the output of the individual flip-flop is linked to the data input of the subsequent flip-flop in the chain, resulting in a circuit that shifts by one place the bit array stored in it, shifting in the data existing at its input and shifting out the last bit in the array, at each changeover of the clock input. A Look-Up Table is a function generator which helps provide solutions to many logical operations. Generally LUT has been executed in the form of a table where equivalent output can be attained for a given input as it is also one of the building blocks of FPGA. An adder is a digital circuit that does the addition of numbers. In many computers and other kinds of processors adders are second hand in the arithmetic logic units or ALU. They are also used in other parts of the processor, where they are used to analyze addresses, table indices, augmentation and decrement operators, and similar operations. A barrel shifter is a digital circuit that can shift a data word by a detailed number of bits without the use of any sequential logic, only pure combinational logic. It is often used to shift and rotate n-bits in modern microprocessors, naturally within a single clock cycle.

IV. PROPOSEDSYSTEM

Booth algorithm gives a procedure for multiplying binary integers in signed 2's complement representation in an efficient way, fewer number of additions/subtractions required. It operates on the very fact that strings of 0's within the multiplier require no addition but just shifting and a string of 1's within the multiplier from bit weight 2^k to weight 2^m are often treated as 2^(k+1) to 2^m. As altogether multiplication schemes, the booth algorithm requires examination of the multiplier bits and shifting of the partial product. Before the shifting, the multiplicand could also be added to the partial product, subtracted from the partial product, or left unchanged. Radix-4 Booth algorithm an extension to the parallel-serial multiplier. This computes x × y where x and y are the n bit two's complement numbers producing a 2n two's complement value in the product p. This paper introduces a dynamic control structure to remove parts of the computation completely during runtime. This is done using a modified serial Booth multiplier, which skips over encoded all-zero or all-one computations independent of location. The multiplier takes all bits of both operands in parallel and is designed to be a primitive block which is easily incorporated into existing DSPs, CPUs, and GPUs. For certain input sets, the multiplier achieves considerable improvements in computational performance. A key element of the multiplier is that sparsity within the input set and the internal binary representation both lead to performance improvements. The multiplier was tested using field-programmable gate array (FPGA) technology, accounting for four different process–voltage– temperature (PVT) corners. TSM which is an extension to the serial Booth multiplication algorithm and implementation. Booth's encoding is employed to lock the method of multiplication. Here we'd like thatthe multiplier is encoded to Booth'sencoding form first then we do multiplication.

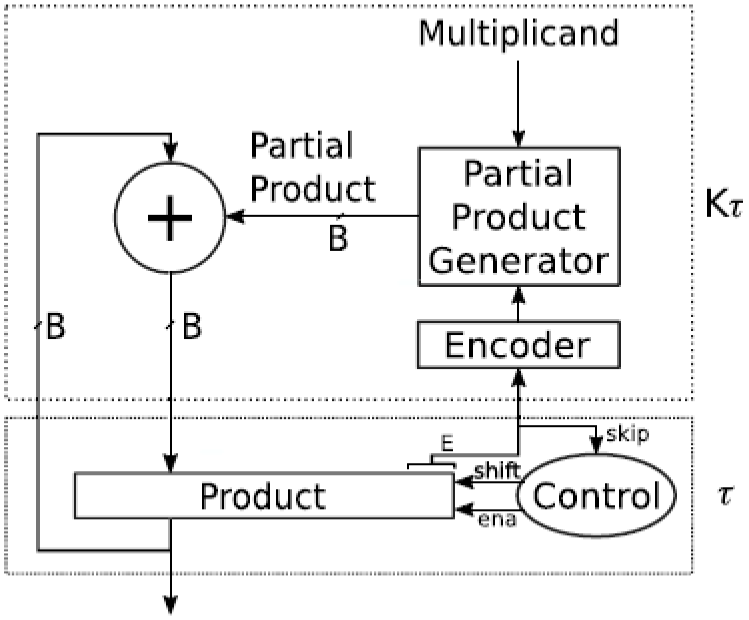


Fig 2: n bit TSM

A digital buffer is an electronic circuit element that's won’t isolate the input from the output, providing either no voltage or a voltage that's the same because of the input voltage. A partial product generator generates a product formed by multiplying the multiplicand by one digit of the multiplier when the multiplier has quite one digit. An Encoder may be a combinational circuit that performs the reverse operation of Decoder. it's a maximum of 2^n input lines and 'n' output lines, hence it encodes the knowledge from 2^n inputs into an n-bit code.

Radix-4 Booth algorithm

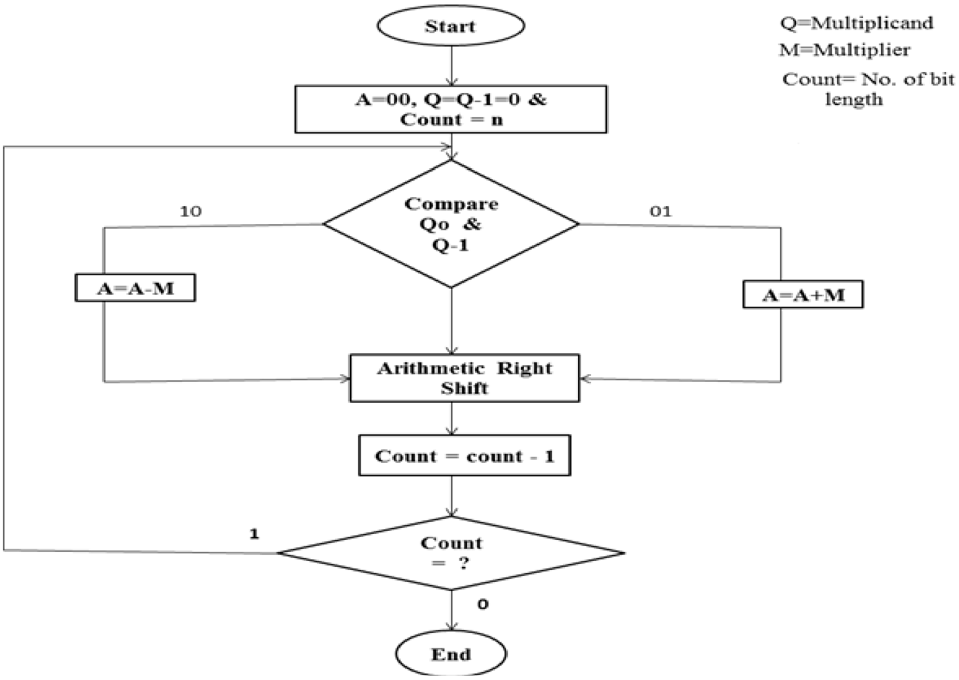


Fig 3 :Flowchart for radix-4 booth algorithm

Radix-4 booth coding

MULTIPLIER BITS	PARTIAL PRODUCTS
000	0
001	1*multiplicand
010	1*multiplicand
011	2*multiplicand
100	-2*multiplicand
101	-1*multiplicand
110	-1*multiplicand
111	0

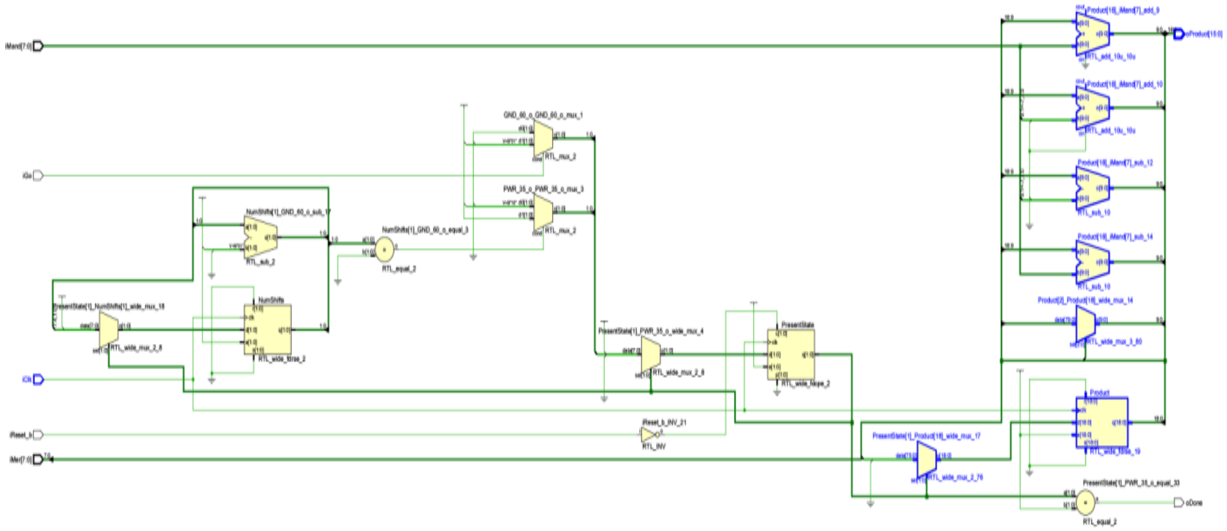
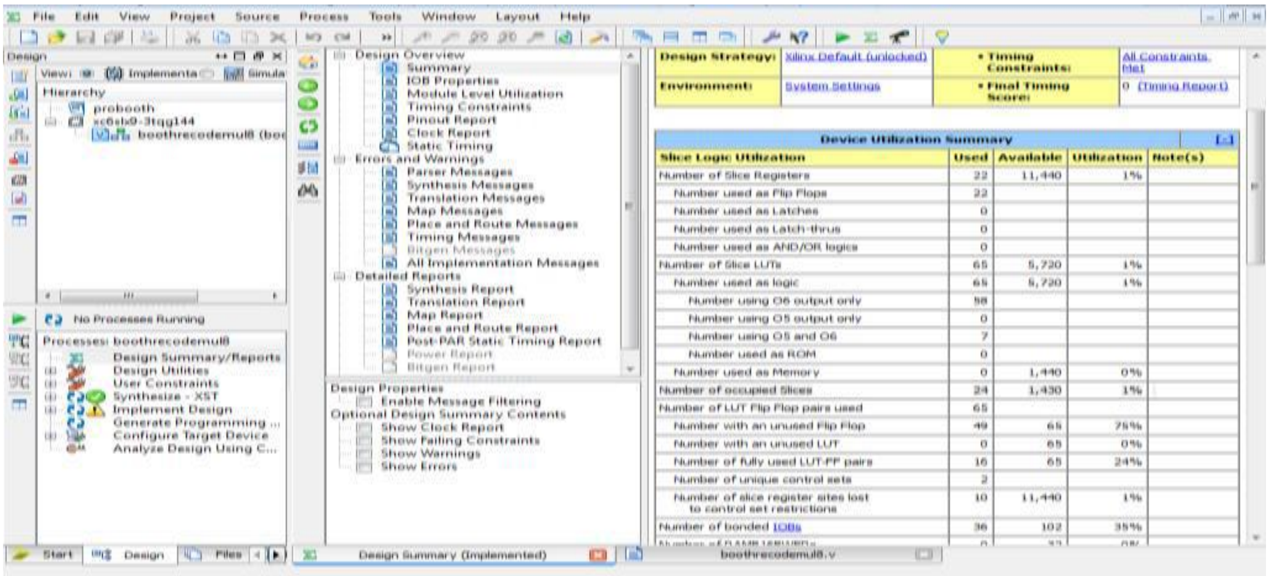


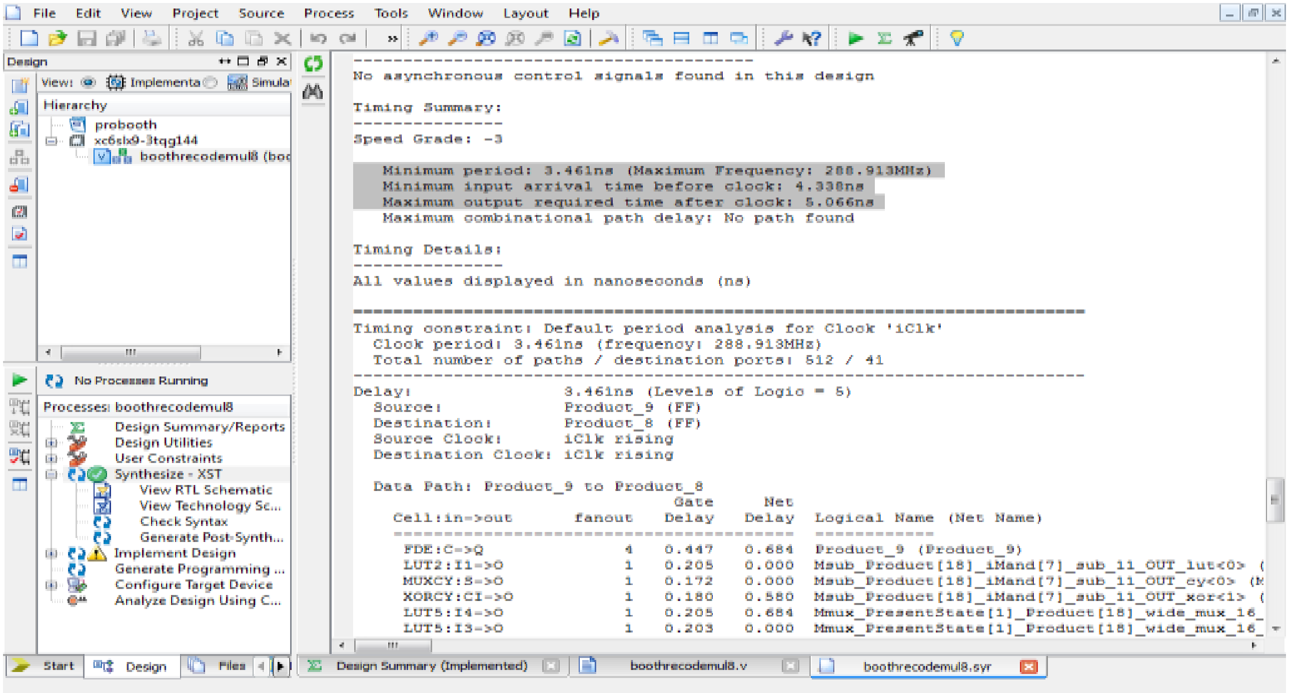
Fig 4:RTL schematic of proposed booth multiplier



Design Strategy:	Xilinx Default (unlocked)	• Timing Constraints:	All Constraints Met
Environment:	System Settings	• Final Timing Score:	0 (Timing Report)

Device Utilization Summary				
Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Registers	22	11,440	1%	
Number used as Flip Flops	22			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	65	5,720	1%	
Number used as logic	65	5,720	1%	
Number using O6 output only	58			
Number using O5 output only	0			
Number using O5 and O6	7			
Number used as ROM	0			
Number used as Memory	0	1,440	0%	
Number of occupied Slices	24	1,430	1%	
Number of LUT Flip Flop pairs used	65			
Number with an unused Flip Flop	49	65	75%	
Number with an unused LUT	0	65	0%	
Number of fully used LUT-FF pairs	16	65	24%	
Number of unique control sets	2			
Number of slice register sites lost to control set restrictions	10	11,440	1%	
Number of bonded IOBs	36	102	35%	

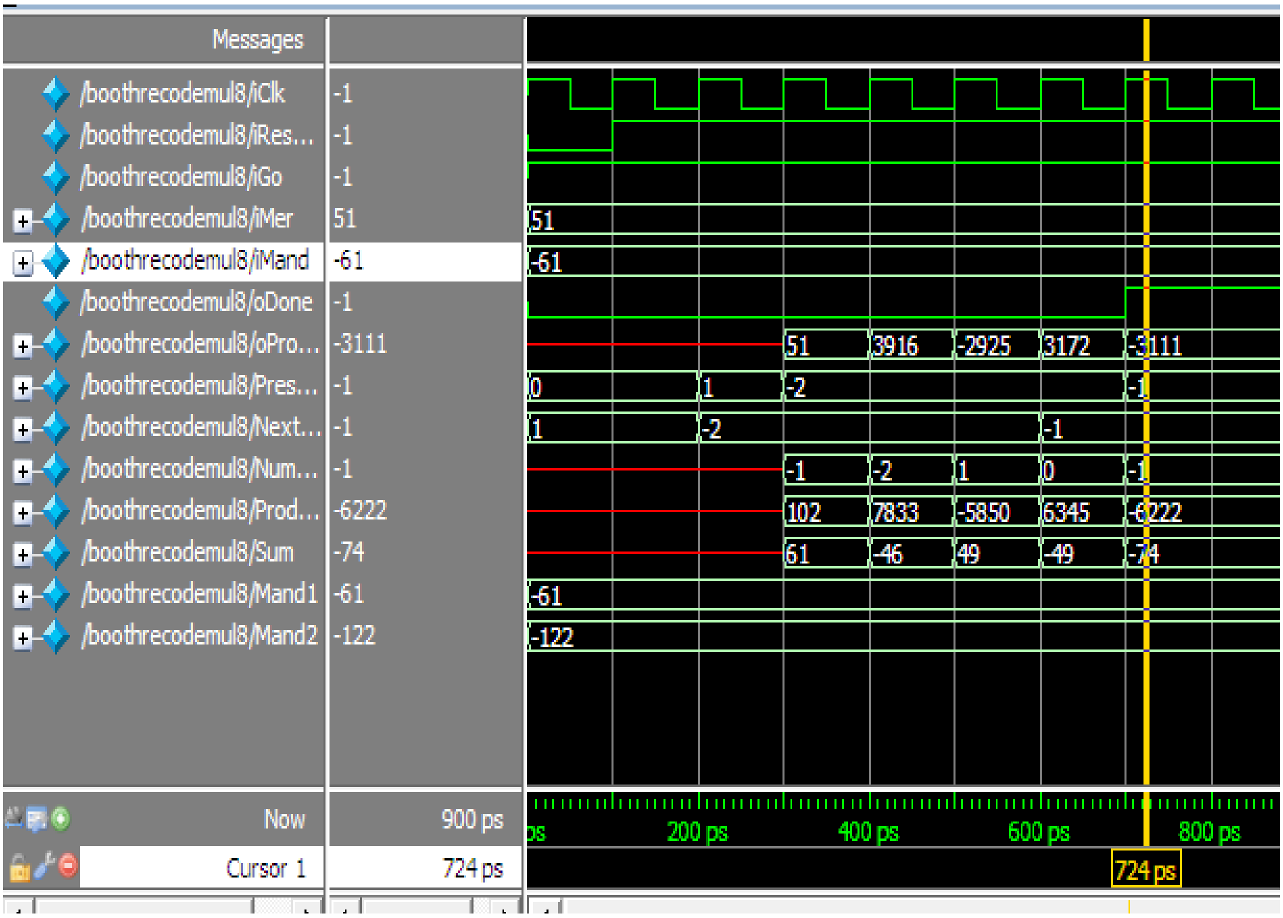
Fig 5: Device Utilization Summary report of proposed booth multiplier



```
---------------------------------------
No asynchronous control signals found in this design

Timing Summary:
---------------
Speed Grade: -3

   Minimum period: 3.461ns (Maximum Frequency: 288.913MHz)
   Minimum input arrival time before clock: 4.338ns
   Maximum output required time after clock: 5.066ns
   Maximum combinational path delay: No path found

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

=========================================================================
Timing constraint: Default period analysis for Clock 'iClk'
  Clock period: 3.461ns (frequency: 288.913MHz)
  Total number of paths / destination ports: 512 / 41
-------------------------------------------------------------------------
Delay:               3.461ns (Levels of Logic = 5)
  Source:            Product_9 (FF)
  Destination:       Product_8 (FF)
  Source Clock:      iClk rising
  Destination Clock: iClk rising

  Data Path: Product_9 to Product_8
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDE:C->Q              4   0.447   0.684  Product_9 (Product_9)
     LUT2:I1->O            1   0.205   0.000  Msub_Product[18]_iMand[7]_sub_11_OUT_lut<0> (
     MUXCY:S->O            1   0.172   0.000  Msub_Product[18]_iMand[7]_sub_11_OUT_cy<0> (M
     XORCY:CI->O           1   0.180   0.580  Msub_Product[18]_iMand[7]_sub_11_OUT_xor<1> (
     LUT5:I4->O            1   0.205   0.684  Mmux_PresentState[1]_Product[18]_wide_mux_16_
     LUT5:I3->O            1   0.203   0.000  Mmux_PresentState[1]_Product[18]_wide_mux_16_
```

Fig 6 :Timing constraint report of proposed booth multiplier

V. SIMULATION RESULTS



Messages	
/boothrecodemul8/iClk	-1
/boothrecodemul8/iRes...	-1
/boothrecodemul8/iGo	-1
/boothrecodemul8/iMer	51
/boothrecodemul8/iMand	-61
/boothrecodemul8/oDone	-1
/boothrecodemul8/oPro...	-3111
/boothrecodemul8/Pres...	-1
/boothrecodemul8/Next...	-1
/boothrecodemul8/Num...	-1
/boothrecodemul8/Prod...	-6222
/boothrecodemul8/Sum	-74
/boothrecodemul8/Mand1	-61
/boothrecodemul8/Mand2	-122
Now	900 ps
Cursor 1	724 ps

Fig 7 : simulation output for proposed Booth multiplier

Table.1 Comparison between existing & proposed system

TYPES	Area	LUT	Delay(ns)	Power(mW)
Existing system	44/1430	104/5720	3.597	651
Proposed system	24/1430	65/5720	3.461	317

From the obtained results, it shows that the proposed booth multiplier offers 46% area reduction, 5% delay reduction, and 52% power reduction than the conventional booth multiplier circuits as shown in Table.1.

VI.CONCLUSION

We presented a TSM, which is divided into two sub-circuits, each operating with a different critical path. In real-time, the performance of this multiplier can be improved solely on the distribution of the bit representation. We illustrated for bit widths of 32 and 64, typical compute sets, such as uniform and Gaussian and neural networks, can expect substantial improvements of 3× and 3.56× using standard learning and sparse techniques, respectively. The cost associated with handling lower bit width representations, such as Gaussian-8 on a 64-bit multiplier is alleviated and shows up to a 3.64× improvement compared to the typical parallel multiplier. Future work will focus on techniques for constructing applications to take full advantage of the two-speed optimization.

REFERENCES

1.A.Rishikesh, S.Asvika, M.Hariharan, G.Pasupathi, L.Arul Leo Felix “A Survey on Serial parallel Multiplication Techniques” in *IJIRCCE Vol. 8, Issue 2, February 2020*

2. O. L. MacSorley, “High-speed arithmetic in binary computers,” *Proc.IRE*, vol. PROC-49, no. 1, pp. 67–91, Jan. 1961.

3. L. P. Rubinfield, “A proof of the modified Booth’s algorithm for multiplication,” *IEEE Trans. Comput.*, vol. C-24, no. 10, pp. 1014–1015, Oct. 1975, DOI: 10.1109/T C.1975.224114.

4.P. Judd, J. Albericio, T. Hetherington, T. M. Aamodt, and A.Moshovos, “Stripes: Bit-serial deep neural network computing,” in *Proc. 49th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2016, pp. 1–12.

5. S. J. Schmidt and D. Boland, “Dynamic bandwidth assignment for efficient dot products,” in *Proc. Int. Conf. Field Program. Log. Appl.*, Sep. 2017, pp. 1–8.