United International Journal of Engineering and Sciences
(UIJES – A Peer-Reviewed Journal); ISSN:2582-5887 | Impact Factor:8.075(SJIF)
Volume 5 | Special Issue 1 | 2025 Edition
National Level Conference on "Advanced Trends in Engineering Science & Technology" – Organized by RKCE

Glitch-Optimized Circuits Blocks For Low Power High Performance Booth Multipliers

Dr.P.Vamsi Krishna[1]
Associate Professor[1], [2345]UG scholar
B. Jyothi Priya[2], M.Akhila[3], I.Sindhura[4], G.Krupavani[5], K.Anusha[6]
Department of Electronics and Communication Engineering
R K College of Engineering
Vijayawada, India

Abstract:
Redundant Binary Partial Product Generator strategy are utilized to diminish by one line the greatest stature of the incomplete item cluster created by a radix-16 Modified Booth Encoded multiplier, with no raise in the deferral of the fractional item creation Block. In this task, we portray a streamlining for parallel radix-16 (adjusted) Booth recoded multipliers to decrease the most extreme tallness of the halfway item sections to [n/4] for n = 64-bit unsigned operands. This is as opposed to the traditional greatest tallness of [(n + 1)/4]. Accordingly, a decrease of one unit in the most extreme stature is accomplished. This Arithmetic multiplier increment the presentation of ALU and Processors. We assess the proposed approach by correlation with Normal Booth Multiplier. Rationale amalgamation showed its productivity as far as region, deferral and force. Task will be created utilizing Verilog. Xilinx ISE instrument is utilized to play out the Simulation and Synthesis.

Keywords: Modified Booth Encoded multiplier, Booth recoded multiplier, Normal Booth Multiplier.

I.INTRODUCTION

Binary multipliers are a widely used building block element in the design of microprocessors and embedded systems, and therefore, they are an important target for implementation optimization [1]–[6]. Current implementations of binary multiplication follow the steps of [7]: 1) recoding of the multiplier in digits in a certain number system; 2) digit multiplication of each digit by the multiplicand, resulting in a certain number of partial products; 3) reduction of the array to two operands using multioperand addition techniques; and 4) carry-propagate addition of the two operands the final result is a key issue, since it determines the number of partial products. The usual recoding process recodes -r digits (by just making groups of m operand into a signed-digit operand with digits in a minimally redundant digit set [7], [8]. The generation of each of these odd multiplies a two term addition or subtraction, yielding a total of carry-propagate additions. However, the advantage of the high radix is that the partial product is further reduced. For instance, for radix-16and n-bit operands, about n/4 partial products are generated. Although less popular than radix-4, there industrial instances of radix-8 [10]–[16]. and radix-16 multipliers [17] in microprocessors implementations. The choice of these radices is related to area/delay/power of pipelined multipliers (or fused multiplier address in the case of a Intel Itanium microprocessor [17]), for balancing delay between stages and/or reduce the number of pipelining flip-flops. Today highly energy-delay optimized, while partial product reductions trees suffer the increasingly serious problems related a complex wiring and glitching due to unbalanced signal. Optimal pipelining in fact, is a key in current and future multiplier (or multiplier-add) units: 1) the of the pipelined unit is very important, even for throughput-oriented applications, as it impacts the energy of the whole core [19]; and 2) the placement of the pipelining flip-flops should

at the same time minimize total power, due to the number of flip-flops required and the signal propagation paths. Two's complement radix-4 Booth multipliers, thus leaving open the research and extension to higher radices and unsigned multiplications unsigned integer arithmetic or mantissa times mantissa in a floating-point unit). For a radix higher than 4, it is necessary to generate the odd multiples (usually with adders), resulting in the of the time slacks necessary to "hide"the simplified three bit assimilation. Unsigned multiplication produces a positive carry out during recoding (this depends to one additional row, increasing the maximum height of the partial product array by one row, not just in one but in several columns. For all these reasons, we need to extend techniques in [1] and [2].In this work, we present a technique that allows partial product arrays of maximum height of _n/m_ (with the goal of not increasing the delay of the partial product generation stage), for r > 4 and unsigned multipliers. Since for the standard unsigned multiplier the maximum height is _(n + 1)/m_, the proposed method allows a reduction of one row when n is a multiple of m. Our technique is general, but its impact (reduction of one unsigned multiplier implemented a synthesis tool and a standard-cell library. We use radix-16 since it is the most complex case, row without increasing the critical path of the partial product generation stage) depends on the specific timing of the different for all practical values of r and n and different implementation technologies. Thus, we concentrate on an specific instance:64-bit radix-16 Booth recoded among the practical values of the radix, for the design of our scheme. The unsigned multiplier is also more complex for the design our scheme than the signed multiplier. We use 64 bits, since it is a representative large word length. The method proposed can be adapted easily to other instances (signed, combined Unsigned/signed, radix-8 recoding, different values of n).

II. LITERATURE SURVEY

Modified booth multipliers with a regular partial product array by S. Kuang, J. Wang, and C. Guo.The conventional modified Booth encoding (MBE) generates an irregular partial product array because of the extra partial product bit at the least significant bit position of each partial product row. In this brief, a simple approach is proposed to generate a regular partial product array with fewer partial product rows and negligible overhead, thereby lowering the complexity of partial product reduction and reducing the area, delay, and power of MBE multipliers. The proposed approach can also be utilized to regularize the partial product array of posttruncated MBE multipliers. Implementation results demonstrate that the proposed MBE multipliers with a regular partial product array really achieve significant improvement in area, delay, and power consumption when compared with conventional MBE multipliers.

Reducing the computation time in (short bit-width) twos complement multipliers by F. Lamberti et alTwo's complement multipliers are important for a wide range of applications. In this paper, we present a technique to reduce by one row the maximum height of the partial product array generated by a radix-4 Modified Booth Encoded multiplier, without any increase in the delay of the partial product generation stage. This reduction may allow for a faster compression of the partial product array and regular layouts. This technique is of particular interest in all multiplier designs, but especially in short bit-width two's complement multipliers for high-performance embedded cores. The proposed method is general and can be extended to higher radix encodings, as well as to any size square and m \times n rectangular multipliers. We evaluated the proposed approach by comparison

with some other possible solutions; the results based on a rough theoretical analysis and on logic synthesis showed its efficiency in terms of both area and delay.

Design of fixed-width multipliers with linear compensation function by N. Petra et alThis paper focuses on fixed-width multipliers with linear compensation function by investigating in detail the effect of coefficients quantization. New fixed-width multiplier topologies, with different accuracy versus hardware complexity trade-off, are obtained by varying the quantization scheme. Two topologies are in particular selected as the most effective ones. The first one is based on a uniform coefficient quantization, while the second topology uses a nonuniform quantization scheme. The novel fixed-width multiplier topologies exhibit better accuracy with respect to previous solutions, close to the theoretical lower bound.

An optimized modified booth recoder for efficient design of the add-multiply operator by K. Tsoumanis et alComplex arithmetic operations are widely used in Digital Signal Processing (DSP) applications. In this work, we focus on optimizing the design of the fused Add-Multiply (FAM) operator for increasing performance. We investigate techniques to implement the direct recoding of the sum of two numbers in its Modified Booth (MB) form. We introduce a structured and efficient recoding technique and explore three different schemes by incorporating them in FAM designs. Comparing them with the FAM designs which use existing recoding schemes, the proposed technique yields considerable reductions in terms of critical delay, hardware complexity and power consumption of the FAM unit.

High speed speculative multipliers based on speculative carry-save tree by A. Cilardo et alThis paper proposes a novel approach to build integer multiplication circuits based on speculation, a technique which performs a faster-but occasionally wrong-operation resorting to a multi-cycle error correction circuit only in the rare case of error. The proposed speculative multiplier uses a novel speculative carry-save reduction tree using three steps: partial products recoding, partial products partitioning, speculative compression. The speculative tree uses speculative (m:2) counters, with m > 3, that are faster than a conventional tree using full-adders and half-adders. A technique to automatically choose the suitable speculative counters, taking into accounts both error probability and delay, is also presented in the paper. The speculative tree is completed with a fast speculative carry-propagate adder and an error correction circuit. We have synthesized speculative multipliers for several operand lengths using the UMC 65 nm library. Comparisons with conventional multipliers show that speculation is effective when high speed is required. Speculative multipliers allow reaching a higher speed compared with conventional counterparts and are also quite effective in terms of power dissipation, when a high speed operation is required.

III. EXISTING METHOD

The architecture of the basic radix-16 Booth multiplier is shown in Fig.1. For sake of simplicity, but without loss of generality, we consider unsigned operands with n = 64. Let us denote with X the multiplicand operand with bit components xi (i = 0 to n − 1, with the least-significant bit, LSB, at position 0) and with Y the multiplier operand and bit components yi. The first step is the recoding of the multiplier operand: groups of four bits with relative values in the set {0, 1,……,14, 15} are recoded to digits in the set {−8,−7,.. …..,0,……..,7, 8}. This recoding is done with the help of a transfer digit ti and an interim digit wi. The recoded digit zi is the sum of the interim and transfer digits zi = wi + ti. That i

$0 \le vi < 8 : ti+1 = 0\ wi = vi\ wi \in [0, 7]$
$8 \le vi \le 15 : ti+1 = 1\ wi = -(16 - vi)$
$wi \in [-8, -1].$

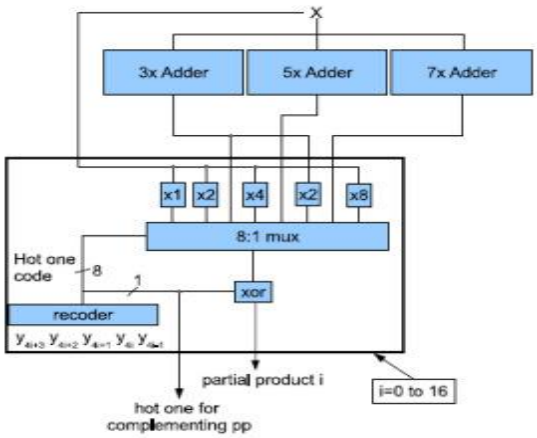


Fig. 3.1: Partial product generation.

After the generation of the partial product bit array, the reduction (multi operand addition) from a maximum height of 17 (for n = 64) to 2 is performed. A maximum height of 17 leads to different approaches that may increase the delay and/or require to use arrays of 3:2 carry-save adders interconnected to minimize delay [20]. After the reduction to two operands, a carry propagate addition is performed. This addition may take advantage of the specific signal arrival times from the partial product reduction step.

3.1 Booth Multiplier

Booth's Algorithm is a smart move for multiplying signed numbers. It initiate with the ability to both add and subtract there are multiple ways to compute a product . Booth's algorithm is a multiplication algorithm that utilizes two's complement notation of signed binary numbers for multiplication . Earlier multiplication was in general implemented via sequence of addition then subtraction, and then shift operations. Multiplication can be well thought-out as a series of repeated additions. The number which is to be added is known as the multiplicand, and the number of times it is added is known as the multiplier, and the result we get is the multiplication result.

After Each step of addition a partial product is generated. When the operands are integers, the product in general is twice the length of operands in order to protect the information content. This repetitive addition method that is recommended by the arithmetic definition is slow as it is always replaced by an algorithm that makes use of positional depiction. We can decompose multipliers into two parts. The first part is committed to the generation of partial products, and the second part collects and then adds them. The fundamental multiplication principle is twofold i.e. evaluation of partial products and gathering of the shifted partial products. It is performed by the consecutive additions of the columns of the shifted partial product matrix.

Signed multiplication is a vigilant process. Through unsigned multiplication there is no need to take the sign of the number into consideration. Even though in signed multiplication the same procedure cannot be applied for the reason that the signed number is in a 2's compliment form which would give in an inaccurate result if multiplied in an analogous manner to unsigned multiplication

.Thus here Booth's algorithm comes in. Booth's algorithm conserves the sign of the end result. While doing multiplication, strings of 0s in the multiplier call for only shifting. While doing multiplication, strings of 1s in the multiplier need an operation only at each end. We require to add or subtract merely at positions in the multiplier where there is a switch from 0 to 1 or from 1 to 0.

VI. PROPOSED SYSTEM

4.1 IMPLEMENTATION OF MULTIIER

To reduce the maximum height of the partial product bit array we perform a short carry propagate addition in parallel to the regular partial product generation. This short addition reduces the maximum height by one row and it is faster than the regular partial product generation. Fig. 4.1(b) shows the elements of the bit array to be added by the short adder. Fig. 4.1(c) shows the resulting partial product bit array after the short addition. Comparing both figures, we observe that the maximum height is reduced from 17 to 16 for n = 64.

Since the partial products are left-shifted four bit positions with respect to each other, a costly sign extension would be necessary. However, the sign extension is simplified by concatenation of some bits to each partial product (S is the sign bit of the partial product and C is S complemented): CSSS for the first partial product and 111C for the rest of partial products (except the partial product at the bottom that is non- negative since the corresponding multiplier digit is 0 or 1). The bits denoted by b in Fig. 4.1 corresponds to the logic 1 that is added for the two's complement for negative partial products.

We perform the computation in two concurrent parts A and B as indicated in Fig.4.2. The elements of the partA are generated faster than the elements of part B. Specifically the elements of part A are obtained from the sign of the first partial product: this is directly obtained from bit y3 since there is no transfer digit from a previous radix-16 digit; Therefore, we decided to implement part A as a speculative addition, by computing two results, a result with carry-in = 0 and a result with carry-in = 1. This can be computed efficiently with a compound adder.

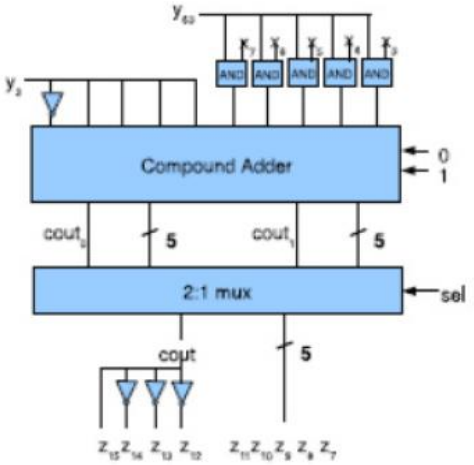


Fig. 4.2: Speculative addition of part A.

Fig. 4.2 shows the implementation of part A. The compound adder determines speculatively the two possible results. Once the carry-in is obtained (from part B), the correct result is selected by a multiplexer. Note that the compound adder is of only five bits, since the propagation of the carry through the most significant three ones is straightforward. The computation of part B is more

complicated. The main issue is that we need the 7 least-significant bits of partial product 15. Of course waiting for the generation of partial product 15 is not an option since we want to hide the short addition delay out of the critical path.

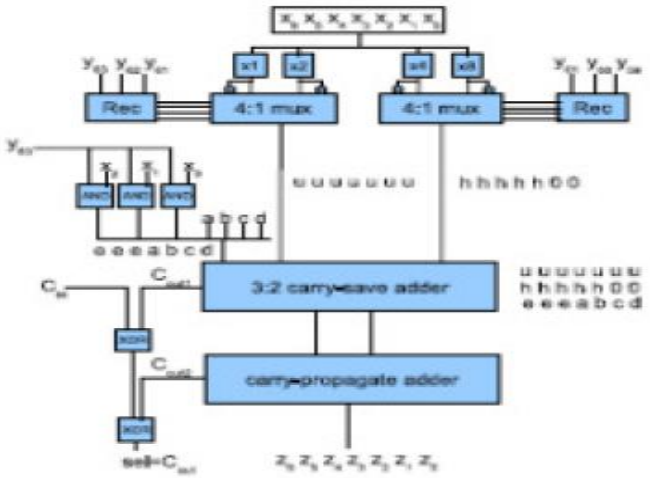


Fig. 4.3: Computation of part B.

Fig. 4.4 shows the recoding and partial product generation stage including the high level view of the hardware scheme proposed.

The way we compute part B may still lead to an inconsistency with the computation of the most significant part of partial product 15. Specifically, when partial product 15 is the result of an odd multiple, a possible carry from the 7 least-significant bits is already incorporated in the most significant part of the partial product. During the computation of part B we should not produce again this carry. This issue is solved as follows. Let us consider first the case of positive odd multiples. Fig. 4.3 shows that the computation of part B may generate two carry outs: the first from the 3:2 carry-save adder (Cout1), and the second from the carry-propagate adder (Cout2). To avoid inconsistencies, we detect the carry propagated to the most significant part of the partial product 15 (we call this CM) and subtract it from the two carries generated in part B.

4.2 EXTENSION

Modified full adder- 1 (FA1):

In this work, the detailed study about Wallace tree multiplier and truncated multipliers is done with both normal full adder and modified full adder-1 and it is shown that modified full adder-1 implementation occupies the less area. In this work, the detailed study about Wallace tree multiplier with conventional full adder, modified full adder-1 and modified full adder- 2 and it is shown that modified full adder3 implementation occupies the less area and less power[7].

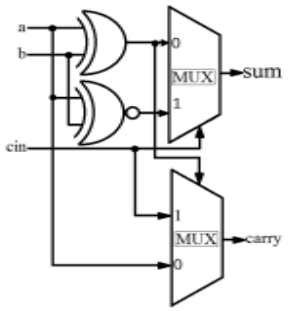


Fig.4 Modified full adder-3 cell[8] (FA3)

A. Ripple Carry Adder (RCA): Ripple Carry Adder is a basic adder circuit which contains individual full adder cells and the carry generated upon addition is propagated between the respective adder cells[4]. The computation of result takes place only after the carry from the previous stage is applied as an input to present stage[4]. Due to these propagation delays, the delay is bound to occur which is a major disadvantage. The architecture of the 32-bit RCA is shown in Fig.5 .

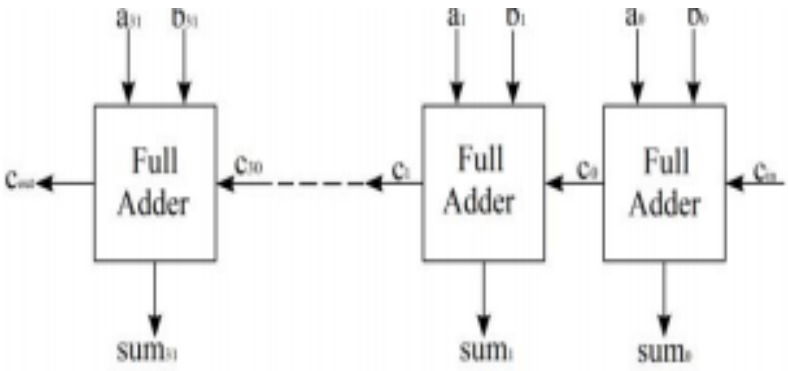
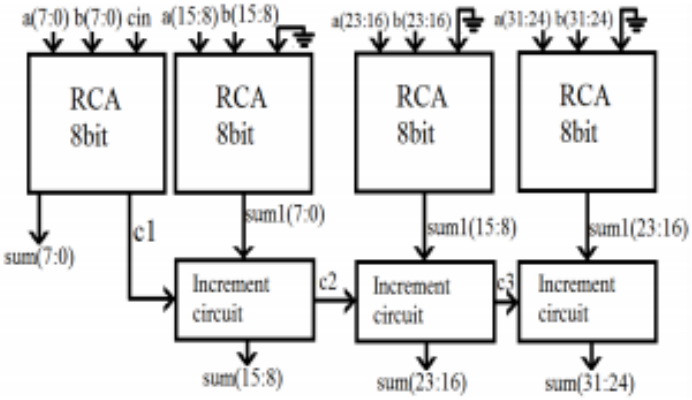


Fig.5 32-bit ripple carry adder

Compute Add Increment (CAI) adder In CAI adder, various RCA blocks are used to compute the results. The first RCA block is given carry-in as input along with addends to get carry(c1) and sum. For the rest RCA blocks, the carry-in is given as logic '0' to get temporary sum (sum1) and temporary carry which are given to increment circuit. Increment circuit consists of half adders which add temporary sum and carries to get the actual sum(sum) and carry (cy). The carry-out of increment circuit is obtained by performing OR operation between carry (cy) and carry-out of the previous stage. As the carry-in for the RCA stages is logic '0' and hence the carry propagation delay decreases. The architecture for the CAI 32bit is given in Fig.



V. SIMULATION RESULTS

5.1 Simulation Result of Multiplier:

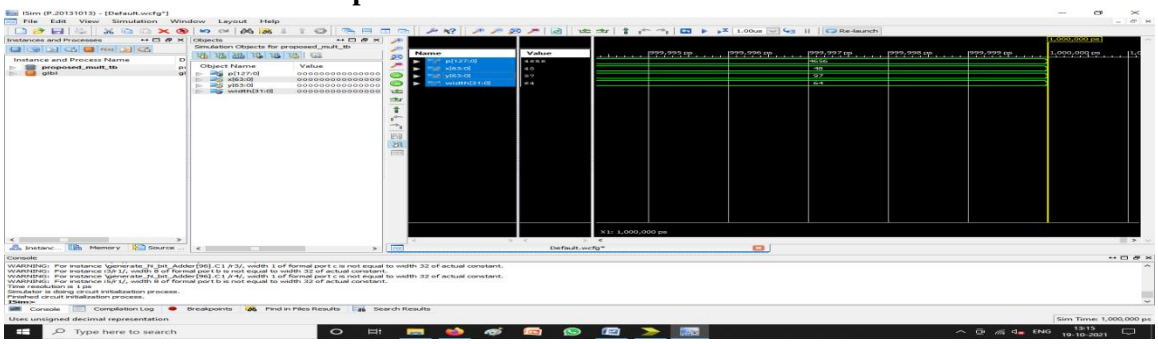


Fig 5.1 Simulation Result of Multiplier

Here we can give the inputs as A=97, B=48, then the final output is 4656.

5.2 Block Diagram of Multiplier:

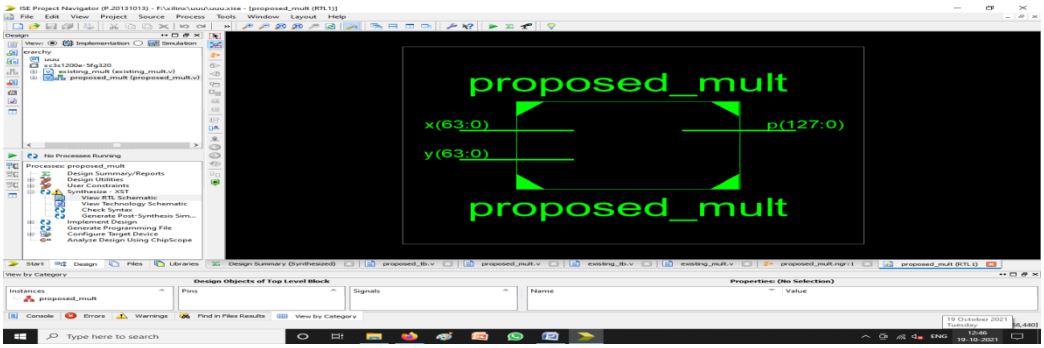


Fig 5.2 Block diagram

5.3 RTL SCHEMATIC:

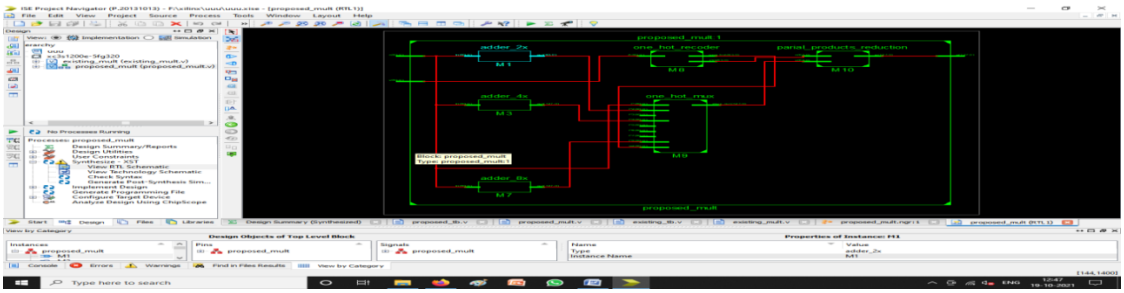


Fig 5.3 Internal diagram of RTL schematic

5.4 Estimation of power:

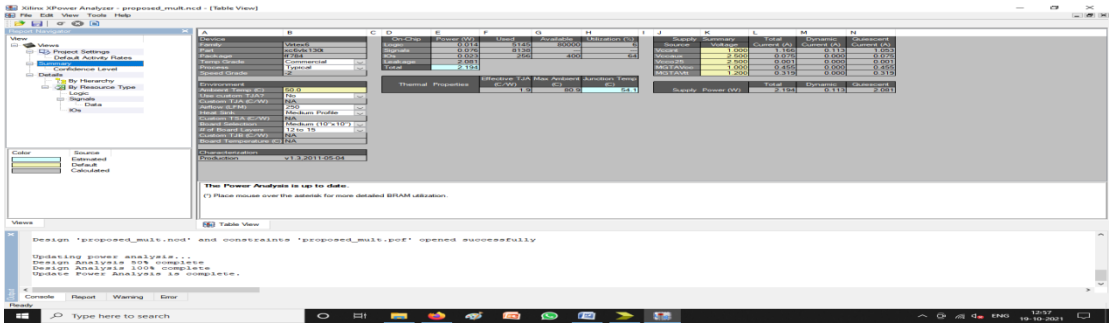


Fig 5.4 Estimation of power

Fig 5.4 Estimates power using approximate multiplier and it is 2.194W power.

5.5 Estimation of Delay:

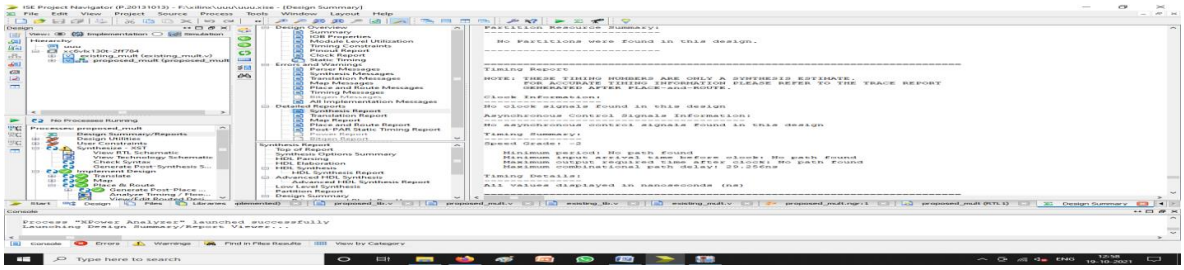


Fig 5.5 Estimation of Delay

Fig 5.5 Estimates delay of approximate multiplier and it is 38.38ns delay.

5.6 Estimation of Area:

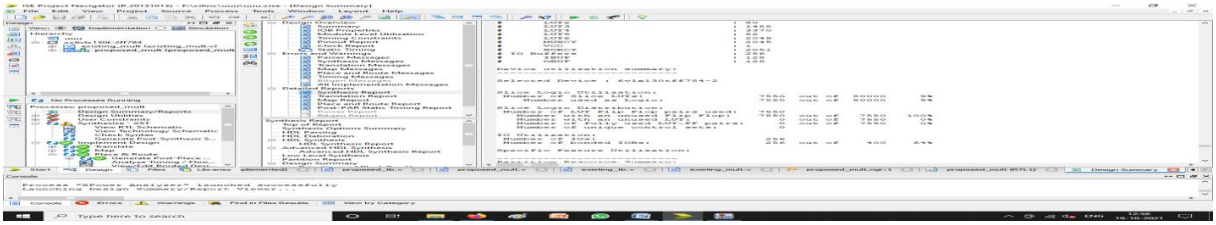


Fig 5.6 Estimation of Area

5.7 Comparison

	Existing system	Proposed System
AREA	**12887**	**7550**

DELAY(ns)	**39.739**	**29.256**
SPEED(Mhz)	**25.16**	**34.18**
POWER	**3.328**	**2.194**

The performance of the proposed system is compared with that of the existing method. We can observe that the designed multiplier is effective and efficient in terms of area-delay tradeoff, delay (speed) and power utilization when compared to the previous one.

VI. CONCLUSION

The multiplier using the proposed algorithm achieves better power-delay analysis than those achieved by conventional Booth multipliers. Here, we have presented a method to reduce by one the maximum height of the partial product array for 64- bit,128-bit radix-16 Booth recoded magnitude multipliers. This reduction may allow more flexibility in the design of the reduction tree of the pipelined multiplier and achieved with no extra delay for $n \geq 32$ for a cell-based design. We believe that the proposed Booth algorithm can be broadly utilized in general processors as well as digital signal processors, mobile application processors, and various arithmetic units that use Booth encoding.

VII. REFERENCES

[1] S. Kuang, J. Wang, and C. Guo, "Modified booth multipliers with a regular partial product array," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 56, no. 5, pp. 404–408, May 2009.

[2] F. Lamberti et al., "Reducing the computation time in (short bit-width) twos complement multipliers," IEEE Trans. Comput., vol. 60, no. 2, pp. 148–156, Feb. 2011.

[3] N. Petra et al., "Design of fixed-width multipliers with linear compensation function," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 58, no. 5, pp. 947– 960, May 2011.

[4] S. Galal et al., "FPU generator for design space exploration," in Proc. 21st IEEE Symp. Comput. Arithmetic (ARITH), Apr. 2013, pp. 25–34.

[5] K. Tsoumanis et al., "An optimized modified booth recoder for efficient design of the add-multiply operator," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 61, no. 4, pp. 1133–1143, Apr. 2014.

[6] A. Cilardo et al., "High speed speculative multipliers based on speculative carry-save tree," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 61, no. 12, pp. 3426– 3435, Dec. 2014.

[7] M. Ercegovac and T. Lang, Digital Arithmetic. Burlington, MA, USA: Morgan Kaufmann, 2004.

[8] S. Vassiliadis, E. Schwarz, and D. Hanrahan, "A general proof for overlapped multiple-bit scanning multiplications," IEEE Trans. Comput., vol. 38, no. 2, pp. 172–183, Feb. 1989.

[9] "Binary Multibit Multiplier," Patent 4 745 570 A, 1986.