International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 11 | Nov 2018 www.irjet.net p-ISSN: 2395-0072

Efficient Design of Radix Booth Multiplier

Dr. Turakane S. M[1], Wani Akanksha Sudhakar[2]

[1]Head and Associate professor E&TC Department, Pravara Rural Engineering College Loni
[2]ME E&TC Engg, Pravara Rural Engineering College Loni

---***---

Abstract: - ***In*** *this paper, we describe an optimization for binary radix-16 (modified) Booth recoded multipliers to reduce the maximum height of the partial product columns to n/4 for n = 64-bit unsigned operands This is in contrast to the conventional maximum height of (n + 1)/4. Therefore, a reduction of one unit in the maximum height is to be achieved. This reduction may add flexibility during the design of the pipelined multiplier to meet the design goals; it may allow further optimizations of the partial product array reduction stage in terms of area/delay/power and may allow additional addends to be included in the partial product array without increasing the delay. The method can be extended to Booth recoded radix-8 multipliers, signed multipliers, combined signed/unsigned multipliers, and other values of n.*

Keywords: ***pipelined multiplier, 64 bit unsigned operands, Booth recoded radix-8 multipliers, signed multipliers, combined signed/unsigned multipliers***

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

S. Kuang, J. Wang, and C. Guo introduced that 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 post truncated 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.[1]

B. Parhami explored and analyzed some arithmetic operators' architectures aiming for this type of processor. These arithmetic components are present in almost all processor subsystems from the arithmetic core to the memory addressing, thus its importance to observe which architectures offer the best solutions based on their speed, area and leakage power .The structure of this work is composed as follows: firstly, it is presented the context the most common adders and multipliers designs. Then, the adopted methodology is described with the developed tools to support perform all analysis followed by the obtained results comparing all studied architectures.[2]

III. Problem Statement & Objective

3.1 Statement

1) This is in contrast to the conventional maximum height of $(n + 1)/4$.
2) The Conventional partial product array of multiplier in terms of area/delay/power is more.

3.2 Objectives

1) A reduction of one unit in the maximum height is achieved.
2)This reduction may add flexibility during the design of the pipelined multiplier to meet the design goals, it may allow further optimizations of the partial product array reduction stage in terms of area/delay/power and/or may allow additional addends to be included in the partial product array without increasing the delay.

IV. Proposed Method

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. 2(b) shows the elements of the bit array to be added by the short adder. Fig. 2(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 $n = 64$. Fig. 3 shows the specific elements of the bit array (boxes) to be added by the short carry-propagate addition. In this figure, $p_{i,j}$ corresponds to the bit *j* of partial product *i*, s_0 is the sign bit of partial product 0, $c_0 = \text{NOT}(s_0)$, b_i is the bit for the two's complement of partial product *i*, and z_i is the *i*th bit of the result of the short addition.

The selection of these specific bits to be added is justified by the fact that, in this way, the short addition delay is hidden from the critical path that corresponds to a regular partial product generation (this will be shown in Section IV). We perform the computation in two concurrent parts A and B as indicated in Fig. 3. The elements of the part A 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 y_3 since there is no transfer digit from a previous radix-16 digit;
- Bits 3 to 7 of partial product 16: the recoded digit for partial product 16 can only be 0 or 1, since it is just a transfer digit. Therefore the bits of this partial product are generated by a simple AND operations of the bits of the multiplicand X and bit y_{63} (that generates the transfer from the previous 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 [7]. Fig. 4 shows the implementation of Here the Master will select a Cluster of slaves. The master and Cluster head communicate with each other using request and response protocol. The inter cluster communication is based on TDMA (Time division multiple access).

Once the carry-in is obtained (from part B), the correct result is selected by 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 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. We decided to implement a specific circuit to embed the computation of the least-significant

Part B should be consistent with the regular computation performed for the most significant bits of partial product 15.Fig. 5 shows the computation of part B. We decided to compute part B as a three operand addition with a 3:2 carry save adder and a carry-propagate adder. Two of the operands correspond to the least-significant bits of the partial product 15and the other operand corresponds to the three least-significant bits of partial product 16 (that are easily obtained by an AND operation). We perform the computation of the bits of theradix-16 partial product 15 as the addition of two radix-4partial products. Therefore, we perform two concurrent radix-4recodings and multiple selections. The multiples of the leastsignificantradix-4 digit are {−2, −1, 0, 1, 2}, while the multiples for the most significant radix-4 digit are {−8, −4, 0, 4, 8}(radix-4 digit set {−2, −1, 0,1,2}, but with relative weight of4 with respect to the least-significant recoding). These two radix-4 recordings produce exactly the same digit as a directradix-16 recoding for most of the bit combinations. However, among the 32 5-bit combinations for a full radix-16 digit recoding, there are six not consistent with the two concurrent radix-4 recordings. Specifically: The bit strings 00100 and 11011 are recoded in radix-16to 2 and −2 respectively. However, when performing two parallel radix-4 recordings the resulting digits are(4, −2) and (−4, 2) respectively. That is, the radix-4 recoding performs the computation of 2X (-2X) as 4X-2X(−4X + 2X). To have a consistent computation we modified the radix-4 recorders so that these strings produce-4 digits of the form (0, 2) and (0, −2).

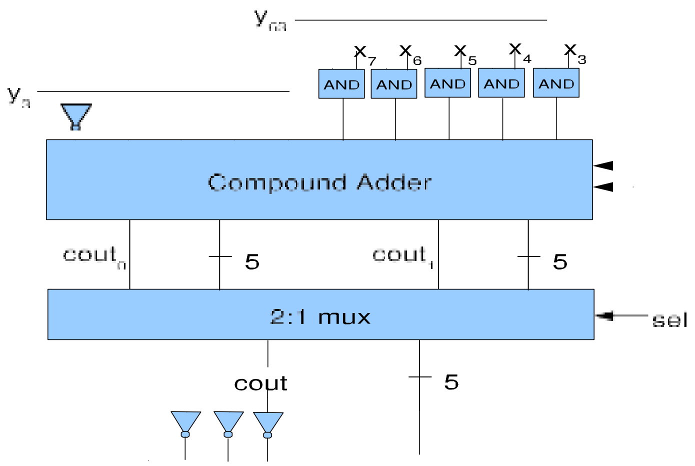


Fig. 4. 1. Speculative addition of part A.

For the set of digits 8, 7,..., 0,... , 7, 8 , the multiples 1X, 2X, 4X, and 8X are easy to compute, since they are obtained by simple logic shifts. The negative versions of these multiples are obtained by bit inversion and addition of a 1 in the corresponding position in the bit array of the partial products. The generations of 3X, 5X, and 7X (odd multiples) requires carry-propagate adders (the negative versions of these multiples are obtained as before). Finally, 6X is obtained by a simple one bit left shift of 3X. Fig. 1 illustrates a possible implementation of the partial product generation. Five bits of the multiplier Y are used to obtain the recoded digit (four bits of one digit and one bit of the previous digit to determine the transfer digit to be added). The resultant digit is obtained as a one-hot code to directly drive a 8 to 1 multiplexer with an implicit zero output (output equal to zero when all the control signals of the multiplexer are zero). The recoding requires the implementation of simple logic equations that are not in the critical path due to the generation in parallel of the odd multiples (carry-propagate addition).

The XOR at the output of the multiplexer is for bit complementation (part of the computation of the two's complement when the multiplier digit is negative). Fig. 2(a) illustrates part of the resultant bit array for n = 64 after the simplification of the sign extension [7].In general, each partial product has n +4 bits including the sign in two's complement representation. The extra four bits are required to host a digit multiplication by up to 8 and a sign bit due to the possible multiplication by negative multiplier digits. 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 con- catenation 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). After the generation of the partial product bit array, the reduction (multioperand addition) from a maximum height of 17 (for n = 64) to 2 is performed.

The methods for multioperand addition are well known, with a common solution consisting of using 3 to 2 bit reduction with full adders (or 3:2 carry-save adders) or 4 to 2 bit reduction with 4:2 carry-save adders .The delay and design effort of this stage are highly dependent on the maximum height of the bit array. It is recognized that reduction arrays of 4:2 carry-save adders may lead to more regular layouts [16]. For instance, with a maximum height of 16, a 17 (for n = 64) to 2 is performed. The methods for multioperand addition are well known, with a common solution consisting of using 3 to 2 bit reduction with full adders (or 3:2 carry-save adders) or 4 to 2 bit reduction with 4:2 carry-save adders.

The delay and design effort of this stage are highly dependent on the maximum height of the bit array. It is recognized that reduction arrays of 4:2 carry-save adders may lead to more regular layouts [16]. For instance, with a maximum height of 16, a total of 3 levels of 4:2 carry-save adders.

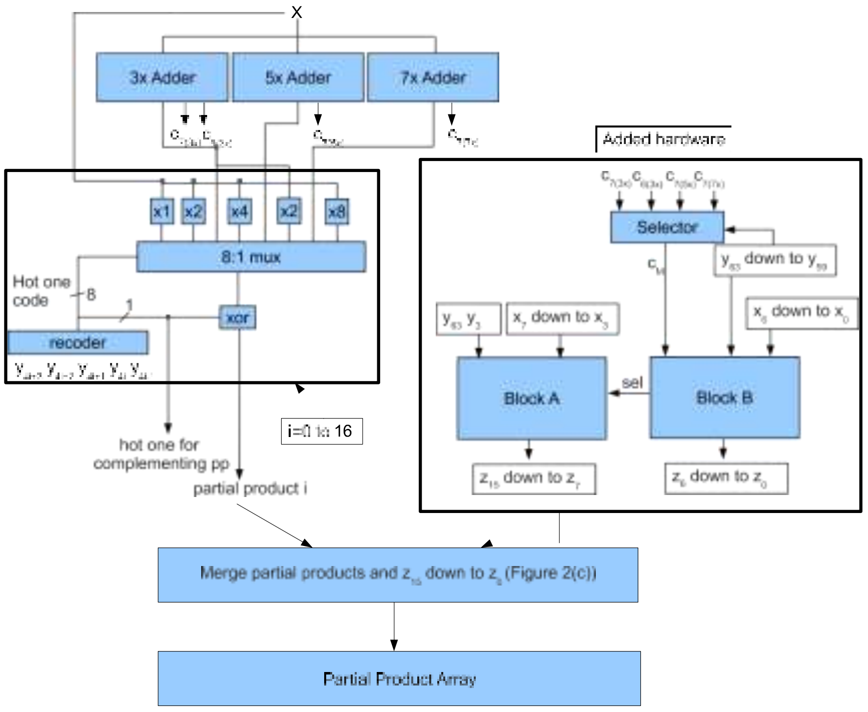


Fig. 4.2 : High level view of the recoding and partial product generation stage including our proposed scheme

V. Required software

Software: Xilinx 14.5 ISE
Programming language: Verilog HDL.

VI. Advantages And Applications

Advantages

1) We have presented a method to reduce by one the maximum height of the partial product array for 64-bitradix-16 Booth recoded magnitude multipliers.2) This reduction may allow more flexibility in the design of the reduction tree of the pipelined multiplier.3) We have shown that this reduction is achieved with no extra delay for *n* ≥ 32 for a cell-based design. The method can be extended to Booth recoded radix-8Multipliers, signed multipliers and combined signed/unsigned Multipliers.4)Radix-8 and radix-16 Booth recoded multipliers are attractive for low power designs, mainly to the lower complexity and depth of the reduction tree, and therefore they might be very popular in this area of power-constrained designs with increasing overheads due to wiring.

Applications

- Design of microprocessors
- Embedded systems.

VII. System Simulation

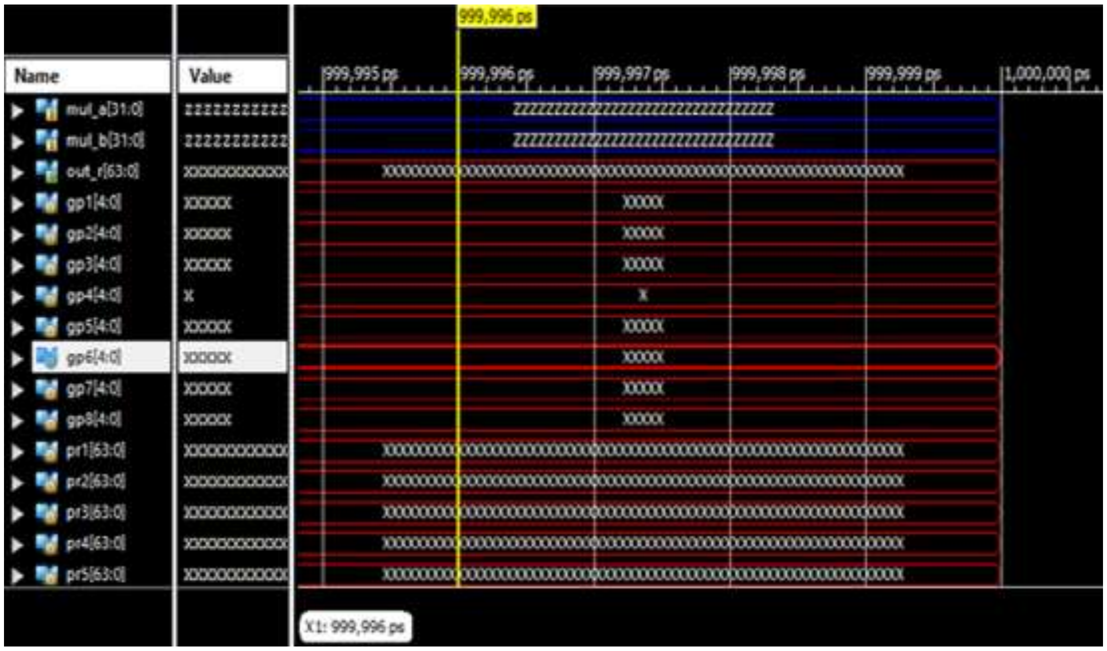
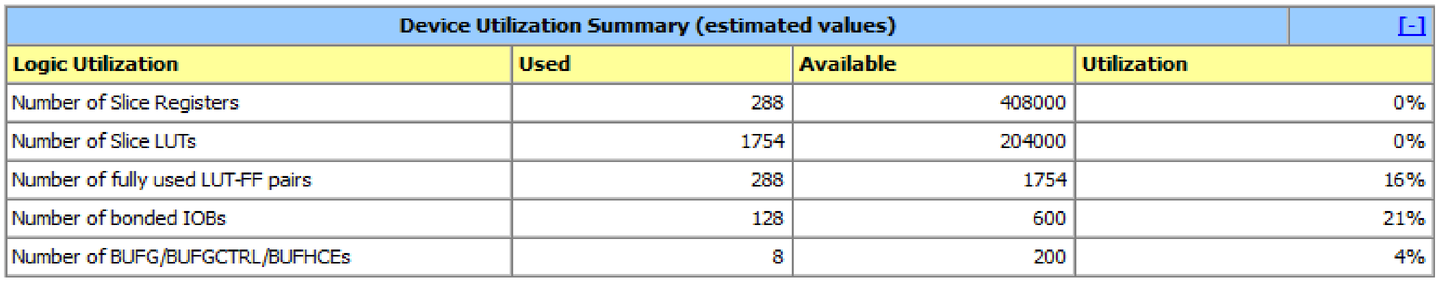


Figure 7.1 : Simulation Result 1



Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	288	408000	0%
Number of Slice LUTs	1754	204000	0%
Number of fully used LUT-FF pairs	288	1754	16%
Number of bonded IOBs	128	600	21%
Number of BUFG/BUFGCTRL/BUFHCEs	8	200	4%

Fig 7.2 Device summary

VIII RESULT AND DISCUSSION

8.1 AREA ANALYSIS

Table 8.1: Performance analysis in terms of area by Tabular Representation

Sr. No	BOOTH MULTIPLIER	NO. OF CELLS	TOTAL AREA
1	RADIX 2	9426	27569
2	RADIX 4	2280	4359
3	RADIX8	14672	32973
4	RADIX16	25703	57091
5	PROPOSED MODEL	1690	3959

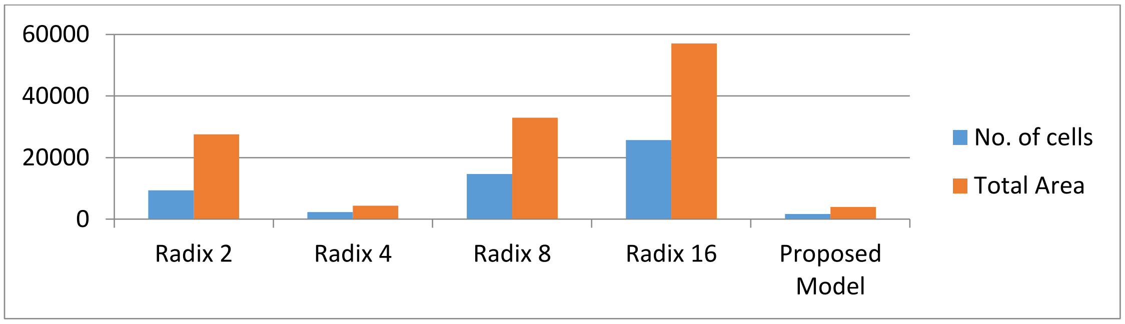


Fig 8.1: Performance analysis in terms of area by Graphical Representation

From the result, we observed that proposed Booth encoding multiplier outperformed the other radix based encoding multiplier and has very less area compared to others. As the no. of cells reduces area reduced to about 85.64% from radix-2 to radix-4 booth encoding multiplier.

8.2 TIME DELAY ANALYSIS

Table 8.2 : Performance analysis in terms of time delay by Tabular Representation

S.NO	BOOTH MULTIPLIER	TIME DELAY(ns)
1	RADIX 2	23.382
2	RADIX 4	14.810
3	RADIX8	26.581
4	RADIX16	27.729
5	PROPOSED MODEL	13.993

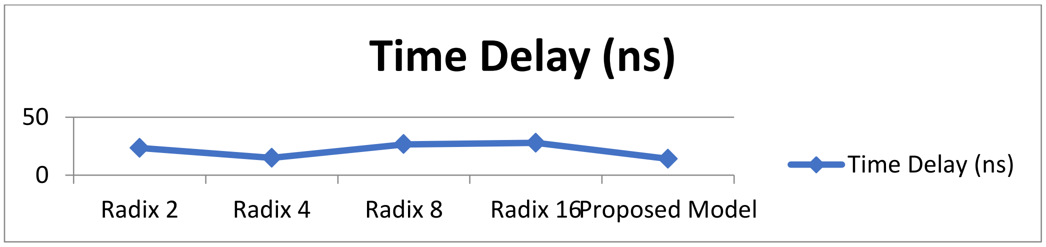


Fig 8.2 : Performance analysis in terms of time delay by Graphical Representation

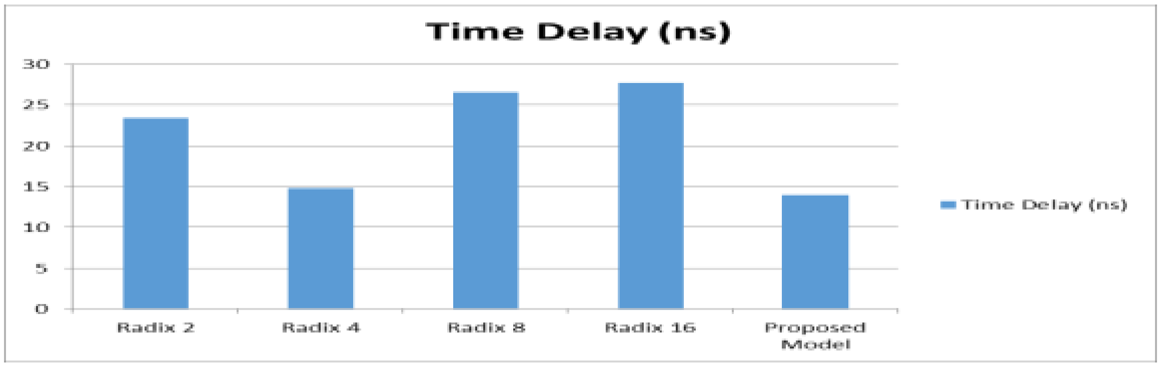


Fig8.3. Comparative Analysis of various multipliers in terms of time delay

The best case given by delay analysis results shows that Proposed Booth encoding multiplier outperformed the other Radix based Booth multiplier and has very less delay.

IX. CONCLUSION

Pipelined large word length digital multipliers are difficult to design under the constraints of core cycle time (for nominal voltage), pipeline depth, power and energy consumption and area. Low level optimizations might be required to meet these constraints.

In this work, we have presented a method to reduce by one the maximum height of the partial product array for 64-bit radix-16 Booth recoded magnitude multipliers. This reduction may allow more flexibility in the design of the reduction tree of the pipelined multiplier. We have shown that this reduction is achieved with no extra delay for $n \geq 32$ for a cell-based design.. Proposed Radix-8 and radix-16 Booth recoded multipliers are attractive for low power designs, mainly to the lower complexity and depth of the reduction tree, and therefore they might be very popular in this era of power-constrained designs with increasing overheads due to wiring.

REFERENCES

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

[2] B. Parhami, Computer Arithmetic Algorithms and Hardware Designs. New York: Oxford Univ. Press, 20 G. W. Bewick, “Fast multiplication: Algorithms and implementation,” Ph.D. dissertation, Stanford Univ., Stanford, CA, 1994.

[3] O. L. MacSorley, “High-speed arithmetic in binary computers,” IRE Proc., vol. 49, pp. 67–91, Jan. 1961
[4]P.Kornerup, Digit-set conversions: Generalizations and applications,IEEE Trans. Comput., vol. 43, no. 5, pp. 622–629, May 1994.

[5] 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.

[6] Y. He, C. H. Chang, J. Gu, and H. A. H. Fahmy, “A novel covalent redundant binary Booth encoder,” in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS‟2005), Kobe, Japan, May 2005, vol. 1, pp. 69V. Kantabutra, “A recursive carry-lookahead/carry-select hybrid adder,” IEEE Trans. Comput., vol. 42, no. 12

[7] 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.

[8] “Binary Multibit Multiplier,” Patent 4 745 570 A, 1986. [9] D. Dobberpuhl et al., “A 200-MHz 64-b dual-issue CMOS microprocessor,” IEEE J. Solid-State Circuits, vol. 27, no. 11, pp. 1555–1567, Nov. 1992.

[10] E. M. Schwarz, R. M. A. III, and L. J. Sigal, "A radix-8 CMOS S/390 multiplier," in Proc. 13th IEEE Symp. Comput. Arithmetic (ARITH), Jul. 1997, pp. 2–9.

[11] J.Clouser etal.,"A600-MHz superscalar floating-point processor," IEEE J. Solid-State Circuits, vol. 34, no. 7, pp. 1026–1029, Jul. 1999.

[12] S. Oberman, "Floating point division and square root algorithms and implementation in the AMD-K7 microprocessor," in Proc. 14th IEEE Symp. Comput. Arithmetic (ARITH), Apr. 1999, pp. 106–115.

[13] R. Senthinathan et al., "A 650-MHz, IA-32 microprocessor with enhanced data streaming for graphics and video," IEEE J. Solid-State Circuits, vol. 34, no. 11, pp. 1454–1465, Nov. 1999.

[14] K. Muhammad et al., "Speed, power, area, latency tradeoffs in adaptive FIR filtering for PRML read channels," IEEE Trans. Very Large Scale Intgr. Syst., vol. 9, no. 1, pp. 42–51, Feb. 2001.

[15] G. Colon-Bonet and P. Winterrowd, "Multiplier evolution: A family of multiplier VLSI implementations," Comput. J., vol. 51, no. 5, pp. 585–594, 2008.

[16] R.Riedlinger etal.,"A32nm,3.1billion transistor, 12 wide issue titanium processor for mission-critical servers," IEEE J. Solid-State Circuits, vol. 47, no. 1, pp. 177–193, Jan. 2012.

[17] B. Cherkauer and E. Friedman, "A hybrid radix-4/radix-8 low power signed multiplier architecture," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol. 44, no. 8, pp. 656–659, Aug. 1997.

[18] D. Lutz, "ARM FPUs:Low latency is low energy," presented at the 22nd IEEE Symposium in Computer Arithmetic, Jun. 2015, [last visited Jul. 1, 2016].[Online].Available: http://arith22.gforge.inria.fr/slides/s1-lutz.pdf

[19] V. G. Oklobdzija, D. Villeger, and S. S. Liu, "A method for speed optimized partial product reduction and generation of fast parallel multipliers using an algorithmic approach," IEEE Trans. Comput., vol. 45, no. 3, pp. 294–306, Mar. 1996.

[20] Synopsys Inc., "Design Compiler," [Online]. Available: http://www. synopsys.com