Computers and Electrical Engineering xxx (xxxx) xxx

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

# Energy efficient signed and unsigned radix 16 booth multiplier design

Y Mounica [a], K Naresh Kumar [a], Sreehari Veeramachaneni [b], Noor Mahammad S [c,*]

[a] Department of Electronics and Communication Engineering, Gayatri Vidya Parishad College of Engineering (Autonomous), Visakhapatnam 530 048, India
[b] Department of Electronics and Communication Engineering, Gokaraju Rangaraju Institute of Engineering and Technology Hyderabad, India
[c] Department of Computer Science and Engineering, Indian Institute of Information Technology Design and Manufacturing (IIITDM) Kancheepuram, Chennai, India

ARTICLE INFO

*Keywords:*
Radix-16 booth encoder
Signed multiplier
Unsigned multiplier and energy

ABSTRACT

Low power multiplier unit design is one of Digital Signal Processing (DSP) processors' requirements to meet the growing demands. Though the higher radix Booth multiplier shows the marginal decrease in the power, the generation of hard multiples in the generation unit becomes the bottleneck in implementing the multiplier unit. This paper proposes an energy-efficient radix-16 Booth multiplier design for combined, signed/unsigned numbers. This paper optimizes the partial product generation unit by $(\frac{n}{r} - 1)$ (where *n*represents input bit size, and *r*represents radix). As a result, delay and energy reduce significantly. The proposed 16-bit non-pipelined multiplier energy improved by 27.06%, 4.35%, and 27.79%, and the pipelined multiplier is improved by 25.74%, 31.30%, and 28.42%for signed, unsigned, and combined radix-16 designs respectively.

## 1. Introduction

In the modern era, with technology scaling, growing components integration and power management on the chip have become major challenges. With the continuous advancement in microprocessor design, to achieve an improved energy utilization for high data encryption rates and transmission. The various compact applications design constraints are satisfied to meet the low power consumption. Among all the units, the multiplier unit is a significant one which decides the average performance of a modern-day processor. Since the multiplier plays a vital role in various applications such as signal/image/video/ processing, it has become an important functional block to be optimized to improve performance [1].

In the day-to-day scenario, a low-power consuming multiplier unit has a great demand in industrial applications. However, they have extended latency with more area and consume considerable amounts of power. Therefore, designing a low-power multiplier unit has become a significant challenge in Very Large Scale Integration (VLSI) system design. Since area and speed are the two other contributing parameters to analyze the multiplier unit designs overall performance, optimizing power relative to these factors is another challenge. These parameters are conflicting with each other when we try to optimize any one parameter. Reduction in area, power is a tradeoff with the circuit speed [2].

The multiplication process involves three necessary steps: (1) Partial product generation, (2) Partial product reduction, and (3)

* Corresponding author.
*E-mail address:* noor@iiitdm.ac.in (N. Mahammad S).

https://doi.org/10.1016/j.compeleceng.2020.106892
Received 30 September 2019; Received in revised form 14 October 2020; Accepted 19 October 2020
0045-7906/© 2020 Elsevier Ltd. All rights reserved.

Final addition, which yields the final product. Generally, the partial products are generated either by using *AND* gates or by using the Booth algorithm. The Booth algorithm based partial products will improve the overall multiplier speed. By utilizing a Wallace tree structure or by using the Dadda tree, these partial products are reduced into two output vectors sum and carry. High-speed Carry Lookahead Adders (CLA) or energy-efficient Carry Save Adders (CSA) generate the final product by adding these sum and carry vectors.

Over the past few decades, several optimizations made to reduce the generation unit's power consumption and the adder circuit design. Over 30% − 40%of the research is on reducing the partial products by implementing the different encoder techniques such as Booth encoders. The remaining works have concentrated on the implementations of either low power or high-speed adders and overall circuit-level optimizations.

Modified Booth Encoder of Booth2 or Radix-4 is observed as the most widely used approach in many applications, ensuring improved speed at the cost of increased power dissipation. Extending for the higher radices greater than that of Booth2 further reduces the number of partial products and leads to a marginal decrease in the overall multiplier unit's power. For the higher radices, the area and delay of the generation unit increase, but the number of adders in the reduction stage decreases, leading to a significant reduction in power. Therefore, the energy-efficient design of the multiplier unit using higher radix.

This paper proposes an energy-efficient radix-16 multiplier unit, which reduces the critical path of the generation stage. The devised multiplier can also perform both signed and unsigned operations by using the control signal. The proposed pipeline design adds more flexibility, which increases the overall performance of the multiplier unit.

The organization of the rest of the paper as follows. Section 2 presents the existing works on Booth encoding multipliers, and Section 3 details Booth radix based encoding techniques. Section 4 proposes the energy-efficient radix-16 Booth multiplier. Section 5 details the proposed implementations and results and followed by a conclusion in Section 6.

## 2. Related work

Existing works related to multipliers have mainly targeted reducing either the critical path of the generation unit or the power consumed by the reduction unit. In [3], the authors have employed radix-4 and radix-8 encoding techniques in their design so that the high-speed property of the former and low power property of the later has been utilized has better performance. The design is a non-pipelined one in which the multiplication for the signed numbers alone [4]. designed a circuit which optimizes the computation of hard multiple of 3x generated from the radix-8 encoder. The proposed design reduced the adder's delay required to generate 3x by 20%compared with that of CLA. In [5], the authors have utilized the above proposed optimized radix-8 multiplier architecture to implement the Floating Point Unit (FPU) generator. Though the proposed architecture is energy efficient, the irregular layout of the CSA adder tree in the reduction unit causes power dissipation, which shows an almost 18%impact on the multiplier's performance.

[6] proposed a multiplier unit for MAC (Multiplier Accumulator), which can perform the multiplication operation on both signed and unsigned numbers. The implementation of Radix-4 Booth's encoding technique using an external control signal in this design to achieve the above functionality.

In [7], the authors devised an optimized radix-16 Booth encoding technique for unsigned multiplication to reduce the partial product generation array height from $(n+1/4)$to $n/4$where $n$is the input size. Pipelining the design resulted in low power consumption by the multiplier but at the cost of the increased area in the generation unit. Nannarelli [8]have inherited the above proposed radix-16 implementation to perform the floating-point multiplication operation. The design was able to achieve a significant reduction in the energy required by the multiplier unit.

Pipelining is a practical approach to reduce the worst slack of a combinational circuit while maintaining efficient throughput. As a result, an impact on the energy consumption of the total design unit. Although less popular than the Booth2, higher radices with pipelining do exist in the industrial cites of the microprocessor implementations [9–15].

This paper aims to improve the energy efficiency by extending the technique implemented by authors in [3] for the radix-16 based booth multiplier unit design, which can perform both signed and unsigned multiplication operations depending upon sign signal.

## 3. Background

In this section, different encoding mechanisms of the modified Booth encoding technique are as follows:

### 3.1. Radix-4 booth encoding algorithm

Radix-4 Modified Booth Encoding (MBE) is the most popular approach, which reduces the number of partial products. The most important feature is that it halves the partial product array by inspecting 3 bits of $n$-bit input, making the multiplicand generation easy. The technique encodes the multiplier bits so that every group of 3 bits represents a single value: 0, $\pm 1, or \pm 2$. Therefore, the multiplication on these values may results in one of the following values: {all zeros, $\pm X, \pm 2X$}, where $X$represents the multiplicand. All these values generated using simple shift operations and complementary operations. Extending the algorithm for the higher radices leads to low power dissipation.

### 3.2. Radix-8 booth encoding algorithm

Radix-8 MBE scans 4 bits at an instance of the multiplier input to reduce the partial product array to one-third of the $n$-bit binary

word length. As explained above, all the digit sets {all zeros, $\pm1, \pm2, \pm3, \pm4$} can be obtained by simply shifting and complementing except the generation of the hard multiple $3X$, which is pre-computed by shifting and an adding operation, $3X = 2X + X$.

### 3.3. Radix-16 booth encoding algorithm

Consider binary2's complement form, $n$-bit width multiplicand $X$and multiplier $Y$as inputs,

$$X = -x_{n-1}2^{n-1} + \sum_{i=0}^{n-2} x_i.2^i \tag{1}$$

$$Y = -y_{n-1}2^{n-1} + \sum_{i=0}^{n-2} y_i.2^i \tag{2}$$

The radix-16 Booth encoder groups given input multiplier bits into a set of five-bits $(1 + log_2{}^r)$ (where r=16 for radix-16) where each group is encoded into radix-16 binary digit set.The encoded binary digit set is $\{0, \pm1X, \pm2X, \pm3X, \pm4X, \pm5X, \pm6X, \pm7X, \pm8X\}$. This encoding is done by considering transfer bit ($t_i$) and the intermediate bits ($m_i$) [16]. The encoded digit set ($z_i$) is summation of the previous transfer bit ($t_i$) and the intermediate bits ($m_i$) [7].

$$z_i = m_i + t_i \tag{3}$$

Assuming that the initial transfer digit to be 0 and is padded at LSB of multiplier input (y). Now consider five bits of the multiplier to obtain the encoded digit. Expanding the $n$-bit binary multiplier input $Y$as:

$$\begin{aligned} Y^{'} = -y2^{n-1} + \sum_{i=0}^{n-2} y_i.2^i = {} & 2^{n-4}(-8y_{n-1} + 4y_{n-2} + 2y_{n-3} + y_{n-4} + y_{n-5}) \\ & + 2^{n-8}(-8y_{n-5} + 4y_{n-6} + 2y_{n-7} + y_{n-8} + y_{n-9}) + \\ & \cdots \\ & + 2^4(-8y_3 + 4y_2 + 2y_1 + y_0 + y_{-1}) \end{aligned} \tag{4}$$

Here, $y_i$is the $i$thbit of the multiplier $y$which can be encoded by the formula:
$-8y_{i+3} + 4y_{i+2} + 2y_{i+1} + y_i + y_{i-1}$.
$y_{-1}$represents the initial transfer bit and it is always set to 0 as explained above. The sample grouping of the 16-bit multiplier is depicted below.

$y_{15}y_{14}y_{13}y_{12}$ $y_{11}$ $y_{10}y_9y_8$ $y_7$ $y_6y_5y_4$ $y_3$ $y_2y_1y_00$

After encoding the digit set, the encoder output is given to $8 \times 1$multiplexer to select the encoded digit [7]. Similarly, to handle the unsigned multiplication using the Booth encoding algorithm (if the input bit width is in powers of 2), zeros are padded at the most significant bit position of the multiplier. As a result, the length of the partial product increases by one row, i.e., $(\frac{n}{4} + 1)$. For the 16-bit input multiplier, it results in 5 partial product array, as shown below.

$0\,0\,0\,0\,y_{15}$ $y_{14}y_{13}y_{12}$ $y_{11}$ $y_{10}y_9y_8$ $y_7$ $y_6y_5y_4$ $y_3$ $y_2y_1y_00$

Fig. 1 illustrates the complete flow of partial product generation. According to the encoding digit set, the multiplicand multiplied to form the array of partial products, i.e., $\{0, \pm1X, \pm2X, \pm3X, \pm4X, \pm5X, \pm6X, \pm7X, \pm8X\}$. $1X$is taken from the multiplicand input, whereas

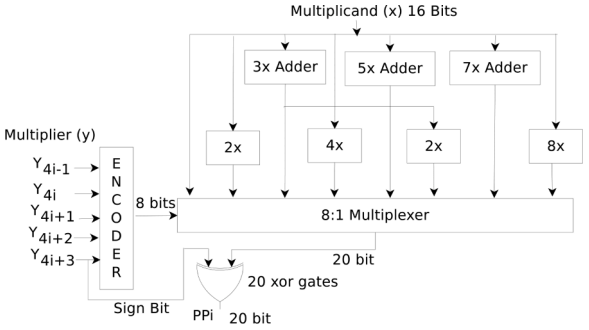


**Fig. 1.** Partial product generation.

$2X, 4X,$ and $8X$can be implemented by left shifting the multiplicand by 1, 2, and 3-bit positions, respectively. There are some hard multiples like $3X, 5X, 7X,$ using two-term energy-efficient Ripple Carry adders (RCA) [7]while encoding. $6X$can be obtained by left shifting the $3X$multiple by one-bit position. Apply negate operation on the encoded values for negative versions. Generate these hard multiples to reduce the delay, and the values are pre-computed along with booth encoding simultaneously.

The bit length of each partial product should be of the size of $(n+4)$(for 16-bit input, the partial product bit length should be of 20 bits). Out of these extra four bits, three bits are to handle the digit multiplication up to 8, whereas the remaining one bit represents the sign bit generated from the multiplication of negative multiplier digits.

Every partial product is left-shifted by 4 bits, and an expensive sign extension would be necessary on the Most Significant Bit (MSB) side of previously generated partial products. However, the sign extension is simplified by concatenating bits of each partial product with the following values. The concatenation of first partial product MSB and $SSSSS$and $111C$to the remaining partial products where $S$is the sign bit, and $C$is complementary of $S$ [7].

### 3.4. Design of partial product reduction unit

In the reduction unit, the array of the partial products needs to be compressed into two output vectors. The partial products of the same weights are added in parallel using the CSA tree, which reduces the tree size to $log_2 n$stages. For instance, in 16-bit signed multiplication, Radix-16 Booth encoder results in 4 partial products. These added partial products used either one stage of 4:2 CSA adder tree or two stages of 3:2 CSA adder tree to produce the two sum and carry vectors as an output reduction unit. In unsigned multiplication, the Radix-16 Booth algorithm encoder added five partial products using one 4:2 CSA and one 3:2 CSA tree or three 3:2 CSA trees [17]. Fig. 2shows the conventional partial product reduction tree structure.

After the reduction stage, a high-speed addition is performed on two output vectors in order to get the final product. Fig. 3 depicts the conventional radix-16 Booth encoding multiplier unit.

## 4. Proposed radix-16 based booth multiplier

Fig. 4 explains the basic flow of the proposed multiplier implementation. As discussed earlier, the computation of hard multiple becomes the bottleneck in the generation unit, which increases the area and delay. Though there is a significant reduction in the number of partial products generated in the generation unit. This paper proposes an approach to decrease the generation unit's delay, thereby increasing the multiplier unit's overall performance.

Use two operands with carry-propagate adders to generate the odd multiples where a significant delay in the multiplier generation unit. To minimize the carry propagation delay in the final stage of the proposed multiplier design uses parallel prefix adders.

These parallel adders add flexibility, thereby decreasing the design's latency. In our design, the Ladner Fischer prefix adder is used, which results in reduced delay [18].

The proposed multiplier design uses a combination of modified booth radix-16 encoder and radix-8 encoder. This technique decreases the delay penalty associated with the generation of hard multiples in the radix-16 encoder. The odd multiples associated with these encoding schemes are pre-computed by using prefix adders [18].

The generated set of the partial products by using the radix-16 encoding by considering the fragment of multiplier input bits which are taken in the form of $m = \left(n - \frac{1}{4}\right)$. The remaining $(n - m)$ bits are used in the radix-8 encoder to generate the remaining partial products. The number of partial products in the proposed 16-bit multiplier is five.

The partial products generated need to be reduced into two output vectors, sum and carry. When the partial products are in the power of 2, to get the better standard layouts, Using a 4:2 carry-save adder tree for adding these vectors. The design makes use of one stage of the 4:2 CSA tree for the first four partial products. The remaining partial product is suppressed in the reduction unit by looking at the MSB bit. To avoid the extra 3:2 adder stage in the reduction unit, which shows a slight decrease in delay and power.

The signed operation may result in either 0000 or 1111, while unsigned operation may result in either 0001 or 0000. The corresponding four operations result in either a 1x or 0. To handle the unsigned multiplication, booth encoding results in multiplication used to handle the unsigned multiplication, and it needs to be considered a peculiar case where the partial product gets valid. This 16-bit multiplicand fragment affects only the MSB part of the output 32-bit vector. In the proposed multiplier design, a 32 bit CLA adder to get the intermediate final product result at its final sum stage.

The multiplicand is added to the MSB part of the intermediate product result using the sign signal; otherwise (in case of all zeros), it just skips and returns the intermediate value as the final product result [19]. Fig. 5 shows the modified reduction tree structure.

As discussed above, Fig. 6 shows the modified final computation stage. With these optimizations, we get an energy-efficient multiplier design when compared to the conventional Radix-16 multiplier designs.

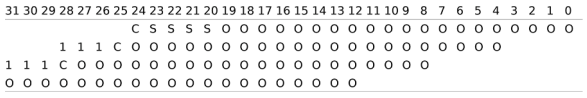


**Fig. 2.** Partial product reduction tree.

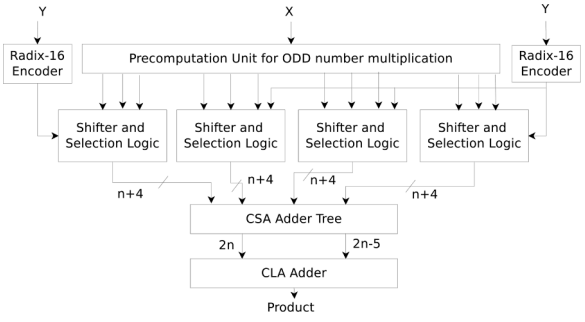


**Fig. 3.** Existing design.

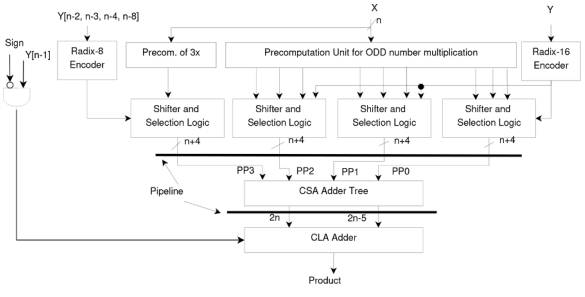


**Fig. 4.** Proposed pipelined design.

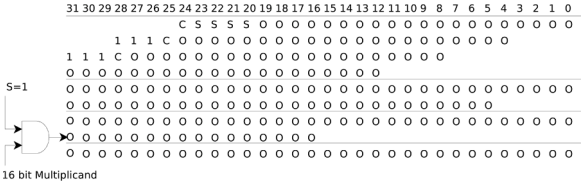


**Fig. 5.** Proposed reduction tree mapping.

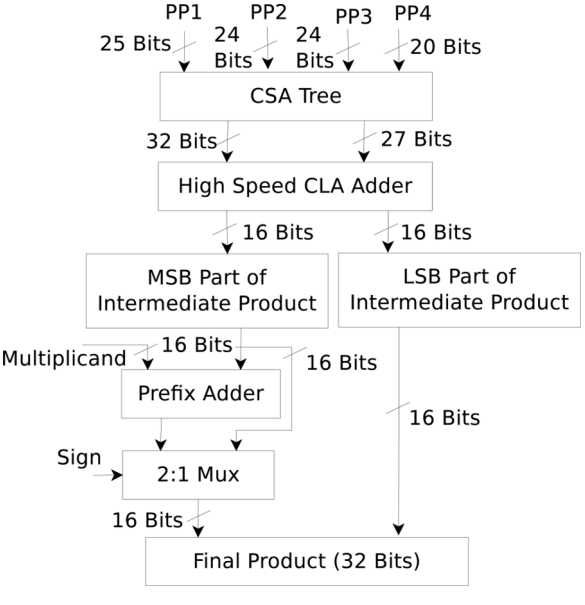


**Fig. 6.** Final sum computation block.

### 4.1. Pipelined structure

As shown in Fig. 4, the proposed multiplier unit is three stages pipelined to increase the throughput of the multiplier unit. Introduced one stage of pipelined registers at the partial product generation unit and introduced another pipelined stage between the partial product reduction stage and the final summation stage.

## 5. Results and discussions

The Radix-16 Booth encoding multiplier designs and the proposed multiplier design are modeled in Verilog HDL (Hardware Description Language) and synthesized using Cadence 6.1 EDA (Electronics Design Automation) tools with a 45nm CMOS (Complementary Metal Oxide Semiconductor) technology standard library.

The generation units of the different formats, such as signed, unsigned, and combined signed/unsigned partial product array, have been implemented by the radix-16 modified Booth encoders' primary method. These reduced partial product arrays using a CSA tree structure, and final addition using a CLA. The combinational circuitry has implemented results, including area, power, delay for these conventional radix-16 multiplier design, and the proposed multiplier for $n = 16$bit, are listed in Table 4, and the tabulated pipelined results in Table 5. The implemented conventional unsigned radix-16 multiplier unit from the extracted primary radix-16 method of [7], and Fig. 3shows the basic block diagram of the signed radix-16 multiplier unit design. Conventional signed radix-16 multiplier unit design and combined signed/unsigned radix-16 multiplier unit designs from Ercegovac and Lang [16].

As observed in the Tables 4 and 5, the different conventional formats of radix-16 multiplier design units have the increased critical path and the increased energy consumption units due to increased computations in the partial product generation unit and increased CSA stage in the reduction unit for the unsigned and combinational signed/unsigned radix-16 multiplication operation.

On the other hand, the two operand pre-computation of hard multiples also increases significantly for the higher radices. For example, radix-16 accompanied three pre-computing adders that include three adders delay to partial product generation unit, which triggers the multiplier unit's worst slack. Table 1 lists the implemented area, power, delay performance of the different types of adders for the 20-bit input word length. Though RCA offers low power but with the extended delay, the Kogge stone adder exhibits the shortest critical path compared to all the other adders. In the proposed design, the goal is to implement the adder, which reduces the delay and can be energy efficient.

Table 2 lists the implemented area, power, delay results of proposed multiplier unit design with different prefix adders, and the proposed architecture with Ladner Fischer shows better performance while comparing with other prefix adder multiplier units.

On the contrary, this paper mainly targets to reduce the critical path of the generation unit. As discussed above, the hard multiples of the generation unit utilize almost three adders delay to generate each partial product, as shown in Fig. 1. The proposed generation unit aims to reduce the multiplier block generation unit's worst slack by embedding the radix-8 encoder, which utilizes the one adder delay to generate the partial product. From the results of Table 3, we can observe that almost 13%, 13%and 18%improved latency over the conventional radix-16 partial product generation unit of signed, unsigned, and combined signed/unsigned multiplier unit at the cost of the increase in area and power dissipation.

For instance, the proposed design for $16 \times 16$multiplier unit is observed the lowest critical path and almost 27.06%, 4.35%, and 27.79%improved energy when compared over signed, unsigned, and combined signed/unsigned radix-16 multiplier units.

### 5.1. 3 Stage pipeline

With this pipelining, the critical path lies in the first stage of multiplier design for the proposed unit and all conventional radix-16 designs. Table 5 shows the comparison of the conventional radix-16 multiplier with the proposed multiplier design. The clock frequency of these designs is 1.1 GHz, and it is observed that there is a significant power reduction due to reduced switching activities in the second stage and improved 25.74%, 21.30%, 28.42%energy efficiency.

## 6. Conclusion

The main objective of this work has been to present an energy-efficient multiplier unit design, which can perform the multiplication of both signed and unsigned binary numbers. Conventional radix-8 and radix-16 Booth's encoders are employed as a base to work. It is achieved by modifying the radix-16 Booth encoder partial product generation unit by embedding the radix-8 Booth encoder. Modest improvements in power and critical path observed over the conventional combined signed/unsigned radix-16 booth's encoder-based multiplier. Comparison results show promising results in terms of improved energy efficiency overall conventional models.

The optimized partial product generation (PPG) unit shows the reduced critical path by dealing with the hard multiples accumulation (like 3, 5, 7) with the prefix adders. Though the generator shows the reduced critical path, it offers increased area and power over existing generation units. However, the pipelined multiplier architecture's overall performance shows reduced area, power, and delay. This proposed and existing designs are simulated and synthesized for a 16-bit input multiplier. These results may vary for the different bit-lengths of the multipliers. Implementing a Booth encoder greater than radix-16 may increase the generation unit complexity due to an increase in a more significant number of hard multiples. Though higher radix reduces the number of partial products, which helps decrease the adder stages in the reduction stage, adding hard multiples increases linearly.

**Table 1**
Comparison of existing adders.

| ADDERS | Area ($nm^2$) | Power ($\mu$W) | Delay (ns) | PDP ($fJ$) |
|---|---|---|---|---|
| RCA | 89 | 8.27 | 0.58 | 4.82 |
| CLA | 128 | 12.74 | 0.45 | 5.83 |
| Kogge Stone [20] | 219 | 23.99 | 0.20 | 4.91 |
| Han-Carlson [21] | 162 | 18.72 | 0.22 | 4.28 |
| Hybrid Han-Carlson [22] | 144 | 17.16 | 0.246 | 4.222 |
| Ladner Fischer [23] | 156 | 18.26 | 0.23 | 4.201 |

**Table 2**
Comparison of proposed multiplier design using existing prefix adders.

| MULTIPLIER UNIT | Area ($nm^2$) | Power ($\mu$W) | Delay ($ns$) | PDP ($fJ$) |
|---|---|---|---|---|
| Kogge Stone [20] | 2383 | 269.54 | 1.24 | 334.23 |
| Han-Carlson [21] | 2160 | 248.18 | 1.23 | 307.50 |
| Hybrid Han-Carlson [22] | 2544 | 397.54 | 1.05 | 419.80 |
| Ladner Fischer [23] | 2118 | 257.4 | 1.09 | 281.6 |

**Table 3**
Comparison of existing and proposed radix-16 generation unit for 16 bit multiplier design.

| GENERATION UNIT | Area ($nm^2$) | Power ($\mu$W) | Delay (ns) | PDP ($fJ$) | % of PDP |
|---|---|---|---|---|---|
| Signed Radix-16 | 1524 | 146.6 | 0.66 | 97.48 | 4.33 |
| Unsigned Radix-16 | 1516 | 154.55 | 0.69 | 107.87 | 13.54 |
| Combined Radix-16 | 1529 | 147.87 | 0.66 | 98.33 | 5.16 |
| Proposed Design | 1659 | 163.04 | 0.57 | 93.26 | == |

**Table 4**
Comparison of existing and proposed multiplier designs.

| Method | Area ($nm^2$) | Power ($\mu W$) | Delay ($ns$) | PDP ($fJ$) | % of PDP |
|---|---|---|---|---|---|
| Signed Radix-16 | 2005 | 230.78 | 1.67 | 386.09 | 27.06 |
| Unsigned-Radix-16 | 2001 | 261.94 | 1.12 | 294.42 | 4.35 |
| Combined Radix-16 | 2249 | 281.17 | 1.38 | 389.99 | 27.79 |
| Proposed Design | 2118 | 257.40 | 1.09 | 281.59 | – |

**Table 5**
Comparison of existing and proposed radix-16 multiplier design with three stage pipeline for 16 bit multiplier.

| Multiplier Unit | Area ($nm^2$) | Power ($\mu$W) | Delay (ns) | PDP ($fJ$) | % of Improv. PDP |
|---|---|---|---|---|---|
| Signed Radix-16 | 3157 | 1637.4 | 1.25 | 206.15 | 25.74 |
| Unsigned Radix-16 | 3063 | 1687.0 | 1.32 | 222.85 | 31.30 |
| Combined Radix-16 | 3283 | 1698.8 | 1.25 | 213.88 | 28.42 |
| Proposed Design | 3031 | 1445.5 | 1.05 | 153.08 | – |

## Authorship statement

All persons who meet authorship criteria are listed as authors, and all authors certify that they have participated sufficiently in the work to take public responsibility for the content, including participation in the concept, design, analysis, writing, or revision of the manuscript. Furthermore, each author certifies that this material or similar material has not been and will not be submitted to or published in any other publication before its appearance in the Jounral of Computer and Electrical Engineering.

## Declaration of Competing Interest

There is NO conflict of interest with any one regarding this work.

## References

[1] Coln-Bonet G, Winterrowd Paul J. Multiplier evolution: A Family of multiplier VLSI implementations. Comput J 2008;51(5):585–94. https://doi.org/10.1093/comjnl/bxm123.
[2] Cilardo A, De Caro D, Petra N, Caserta F, Mazzocca N, Napoli E, et al. High speed speculative multipliers based on speculative carry-save tree. IEEE Trans Circuits Syst I Regul Pap 2014;61(12):3426–35. https://doi.org/10.1109/TCSI.2014.2337231.
[3] Cherkauer BS, Friedman EG. A hybrid radix-4/madix-8 low power signed multiplier architecture. IEEE Trans Circuits Syst II Analog Digital Signal Process 1997;44(8):656–9. https://doi.org/10.1109/82.618039.
[4] Ruiz GA, Granda M. Efficient implementation of 3x for radix-8 encoding. Microelectron J 2008;39(1):152–9. https://doi.org/10.1016/j.mejo.2007.10.006.
[5] Galal S, Shacham O, Brunhaver II JS, Pu J, Vassiliev A, Horowitz M. FPU generator for design space exploration. Proceedings of the IEEE 21st symposium on computer arithmetic. 2013. p. 25–34. https://doi.org/10.1109/ARITH.2013.27.
[6] Farooqui AA, Oklobdzija VG. General data-path organization of a MAC unit for vlsi implementation of DSP processors. Proceedings of the IEEE international symposium on circuits and systems (ISCAS). 2; 1998260–263 vol.2. https://doi.org/10.1109/ISCAS.1998.706891.
[7] Antelo E, Montuschi P, Nannarelli A. Improved 64-bit radix-16 booth multiplier based on partial product array height reduction. IEEE Trans Circuits Syst I Regul Pap 2017;64(2):409–18. https://doi.org/10.1109/TCSI.2016.2561518.
[8] Nannarelli A. A multi-format floating-point multiplier for power-efficient operations. Proceedings of the 30th IEEE international system-on-chip conference (SOCC). 2017. p. 351–6. https://doi.org/10.1109/SOCC.2017.8226076.
[9] Dobberpuhl DW, Witek RT, Allmon R, Anglin R, Bertucci D, Britton S, et al. A 200-mhz 64-b dual-issue cmos microprocessor. IEEE J Solid-State Circuits 1992;27(11):1555–67. https://doi.org/10.1109/4.165336.
[10] Schwarz EM, Averill RM, Sigal LJ. A radix-8 cmos s/390 multiplier. Proceedings of the 13th IEEE sympsoium on computer arithmetic. 1997. p. 2–9. https://doi.org/10.1109/ARITH.1997.614873.
[11] Clouser J, Matson M, Badeau R, Dupcak R, Samudrala S, Allmon R, et al. A 600-mhz superscalar floating-point processor. IEEE J Solid-State Circuits 1999;34(7):1026–9. https://doi.org/10.1109/4.772419.
[12] Oberman SF. Floating point division and square root algorithms and implementation in the AMD-k7/sup tm/ microprocessor. Proceedings 14th IEEE symposium on computer arithmetic (Cat. No.99CB36336). 1999. p. 106–15. https://doi.org/10.1109/ARITH.1999.762835.
[13] Senthinathan R, Fischer S, Rangchi H, Yazdanmehr H. A 650-mhz, ia-32 microprocessor with enhanced data streaming for graphics and video. IEEE J Solid-State Circuits 1999;34(11):1454–65. https://doi.org/10.1109/4.799850.
[14] Muhammad K, Staszewski RB, Balsara PT. Speed, power, area, and latency tradeoffs in adaptive fir filtering for prml read channels. IEEE Trans Very Large Scale Integr VLSI Syst 2001;9(1):42–51. https://doi.org/10.1109/92.920818.
[15] R.J.Riedlinger, R.Bhatia, L.Biro, Bowhill B, Fetzer E, Gronowski P, et al. A 32nm 3.1 billion transistor 12-wide-issue itanium processor for mission-critical servers. Proceedings of the IEEE international solid-state circuits conference. 2011. p. 84–6. https://doi.org/10.1109/ISSCC.2011.5746230.
[16] Ercegovac MD, Lang T. Digital arithmetic. 1st. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.; 2003.ISBN 1558607986, 9781558607989
[17] Oklobdzija VG, Villeger D, Liu SS. A method for speed optimized partial product reduction and generation of fast parallel multipliers using an algorithmic approach. IEEE Trans Comput 1996;45(3):294–306. https://doi.org/10.1109/12.485568.
[18] Vitoroulis K, Al-Khalili AJ. Performance of parallel prefix adders implemented with FPGA technology. Proceedings of the IEEE northeast workshop on circuits and systems. 2007. p. 498–501. https://doi.org/10.1109/NEWCAS.2007.4487969.
[19] Ma Yan, Wang De-li. The design of an architecture-aware 32-bit signed/unsigned multiplier. Proceedings of the 2nd international conference on computer engineering and technology. 7; 2010. https://doi.org/10.1109/ICCET.2010.5485463.
[20] Kogge PM, Stone HS. A parallel algorithm for the efficient solution of a general class of recurrence equations. IEEE Trans Comput 1973;C-22(8):786–93. https://doi.org/10.1109/TC.1973.5009159.
[21] Han T, Carlson DA. Fast area-efficient vlsi adders. Proceedings of the IEEE 8th symposium on computer arithmetic (ARITH). 1987. p. 49–56. https://doi.org/10.1109/ARITH.1987.6158699.
[22] Muthyala Sudhakar S, Chidambaram KP, Swartzlander EE. Hybrid Han-Carlson adder. Proceedings of the IEEE 55th international midwest symposium on circuits and systems (MWSCAS). 2012. p. 818–21. https://doi.org/10.1109/MWSCAS.2012.6292146.
[23] Ladner RE, Fischer MJ. Parallel prefix computation. J ACM 1980;27(4):831–8. https://doi.org/10.1145/322217.322232.

**Y Mounica** is a research intern at the Indian Institute of Information Technology Design and Manufacturing (IIITDM) Kancheepuram. She is pursuing her M. Tech in the Department of Electronics and Communication Engineering, Gayatri Vidya Parishad College of Engineering, Andhra Pradesh, India.

**K Naresh Kumar**, is a faculty member in the Department of Electronics and Communication Engineering, Gayatri Vidya Parishad College of Engineering, Andhra Pradesh, India.

**Sreehari Veeramachaneni** is a faculty member in the Department of Electronics and Communication Engineering at Gokaraju Rangaraju Institute of Engineering and Technology, Hyderabad, India. He received his Ph.D. degree from the International Institute of Information Technology Hyderabad. His areas of research are Arithmetic Circuits, Approximate Computing, Hardware Security, Memory Design, Data Converters, Analog VLSI Design, Low Power VLSI Design, and Computer Architecture.

**Noor Mahammad Sk** is currently working as faculty in the Indian Institute of Information Technology Design and Manufacturing (IIITDM) Kancheepuram, Chennai, India. He received a Ph.D. in Computer Science and Engineering, Indian Institute of Technology Madras. His research interests are reconfigurable computing, computer architecture, Software for VLSI Design, and Network System Design.