Industrial Engineering Journal
ISSN: 0970-2555
Volume : 48, Issue 1, No. 1, January 2019

# A LOW POWER HIGH SPEED HYBRID HIGH RADIX ENCODING FOR GENERATING THE PARTIAL PRODUCTS OF A SIGNED MULTIPLIER

G.Aruna kumari[1], K. Gowthami[2], G. Mahesh[3]
*[1]Assistant Professor, [2]Assistant Professor, [3]Assistant Professor, ECE Department, Anantha Lakshmi Institute of Technology and Sciences, Ananthapuramu, Andhra Pradesh, India.*

***ABSTRACT—Inexact register structures a plan elective that adventures the characteristic blunder strength of different applications and produces vitality efficient circuits with little precision misfortune. In this paper, we propose a rough half and half high radix encoding for creating the incomplete items in marked increases that encodes the most significant bits with the exact radix-4 encoding and the least significant bits with an inexact higher radix encoding. The approximations are performed by adjusting the high radix esteems to their closest intensity of two. The proposed strategy can be configured to accomplish the ideal vitality precision tradeoffs. Contrasted and the exact radix-4 multiplier, the proposed multipliers convey up to 56% vitality and 55% region reserve funds, when working at a similar recurrence, while the forced blunder is limited by a Gaussian dissemination with close to zero normal. Additionally, the proposed multipliers are contrasted and cutting edge estimated multipliers, outperforming them by up to 40% in vitality utilization, for comparative blunder esteems. At long last, we exhibit the adaptability of our procedure.***

*Index Terms—Approximate computing, error resiliency, low power, radix encodings, signed multipliers.*

## 1.INTRODUCTION

In modern inserted frameworks and server farms, vitality efficiency is a required plan concern. Taking into account that a lot of use spaces displays an inborn mistake resistance, e.g., computerized sign handling

(DSP), picture preparing, information investigation, and information mining [1], [2], estimated processing shows up as a compelling answer for decrease their capacity dissemination. In estimated figuring, blunder has been seen as a product that can be exchanged for significant gains in expense (e.g., power, vitality, and execution) [3], and therefore, it makes a promising structure worldview focusing on vitality efficient frameworks by very diminishing the power utilization of inalienably mistake strong applications. Specifically, estimated figuring misuses the inborn blunder tolerance of the respective applications and deliberately relaxes the rightness of certain calculations, so as to diminish their capacity utilization as well as quicken their execution. As of late, focusing to exploit its benefits, huge research has been directed in the field of equipmen testimated circuits. The fundamental targets are math units, e.g., adders and multipliers, that are the center parts in many inserted gadgets and equipment quickening agents. Broad research is accounted for in surmised adders [4]–[8], giving significant gains regarding deferral and power dispersal. Be that as it may, look into exercises on the inexact multipliers [9]–[21] is less extensive contrasted and the separate on surmised adders. In inaccurate multipliers, approximations can be connected on the fractional item age [16]–[18], just as the halfway item collection [9]–[11], [13]. Approximations on the fractional item age and approximations on their gathering are synergistic, and can be connected in cooperation so as to accomplish higher power decrease [17], [18], [22]. In spite of the fact that significant research has been directed in the fractional item amassing, inquire about movement on the estimate of the incomplete item age is as yet restricted. At long last, another restriction of the current surmised multipliers is that most of them (see [9], [10], [12], [19], [20]) does not look at marked augmentation. In this paper, focusing on the plan of vague multipliers by applying approximations on the halfway item age, we propose a novel rough crossover high radix encoding. In the proposed method, the most significant bits (MSBs) of the multiplicand are encoded utilizing the radix-4 encoding, though the k least significant bits (LSBs) are encoded utilizing a radix-2 k (with k ≥4). To streamline the expanded multifaceted nature initiated by the proposed half breed encoding, the circuit for creating the halfway items is approximated by modifying in like manner its reality table. Thus, the quantity of the incomplete items diminishes significantly and more straightforward tree designs are utilized for their amassing, decreasing the multiplier's vitality utilization, region, and basic way delay. The significant commitments of this paper are condensed as pursues.

1) We propose and empower the utilization of cross breed high radix encodings for the age of vitality efficient estimated multipliers, surpassing the expanded equipment multifaceted nature of exceptionally high radix encodings.

2) The proposed method can be connected to any multiplier design and is reconfigurable, empowering the client to choose the ideal per application vitality blunder tradeoff.

3) An investigative mistake examination is led, demonstrating that the yield blunder of the proposed procedure is limited and unsurprising. Such a thorough blunder examination prompts exact and a prior error estimation for any information circulation, without the need of tedious reenactments.

4) We demonstrate that the proposed strategy outflanks related best in class inexact marked multipliers interms of equipment and precision, accomplishing up to 40% less vitality dissemination for similar blunder esteems.

All the more specifically, the proposed strategy is connected to a 16×16 piece multiplier and is assessed utilizing mechanical quality devices, i.e., Synopsys Design Compiler, PrimeTime, and Mentor Graphics ModelSim. Contrasted and the precise multiplier, the proposed method conveys up to 55% vitality and territory decrease, for mean relative blunder up to 0.93%.

## 2.LITERATURE SURVEY

**V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan,** "*Analysis and characterization of inherent application resilience for approximate computing*,"[1] Approximate computing is an emerging design paradigm that enables highly efficient hardware and software implementations by exploiting the inherent resilience of applications to in-exactness in their computations. Previous work in this area has demonstrated the potential for significant energy and performance improvements, but largely consists of ad hoc techniques that have been applied to a small number of applications. Taking approximate computing closer to mainstream adoption requires (i) a deeper understanding of inherent application resilience across a broader range of applications (ii) tools that can quantitatively establish the inherent resilience of an application, and (iii) methods to quickly assess the potential of various approximate computing techniques for a given application. We make two key contributions in this direction. Our primary contribution is the analysis and characterization of inherent application resilience present in a suite of 12 widely used applications from the domains of recognition, data mining, and search. Based on this analysis, we present several new insights into the nature of resilience and its relationship to various key application characteristics. To facilitate our analysis, we propose a systematic framework for Application Resilience Characterization (ARC) that (a) partitions an application into resilient and sensitive parts and (b) characterizes the resilient parts using approximation models that abstract a wide range of approximate computing techniques. We believe that the key insights that we present can help shape further research in the area of approximate computing, while automatic resilience characterization frameworks such as ARC can greatly aid designers in the adoption approximate computing.

## 3.PROPOSED WORK

### 1. Related Work

In this section, prior works in the field of approximate multipliers, related to our proposed design, are discussed. As far as the approximate adders are concerned, [4] produces approximate adders by simplifying the logic used in the adder. Gupta et al. [5] design imprecise full adder cells by approximating their logic function and then use them to build approximate adders. Nevertheless, it is not clear how these adders can be used in different tree architectures and how the error scales in the case of multi operand accumulation. Reference [6] proposes an approximate adder that consists of an accurate and an inaccurate part. Reference [7] designs a multiplier in which the LSBs of the additions are approximated by applying bitwise OR to the respective input bits. In [8], a fast approximate adder is produced by limiting the carry propagation, based on a proof that the longest carry chain in an n-bit adder is logn. However, despite the fact that all these techniques demonstrate the benefit of approximate computing, their fixed functionality and low-level design limit further improvements in efficiency. Regarding the approximations in multiplication schemes, Kulkarni et al. [9]

propose an under designed 2×2 inaccurate multiplier block to produce the partial products,and then use it as a building block to design larger multipliers. This technique is characterized by high error and hardware overhead for the error detection and correction due to the small building block. Momeni et al. [10] propose two approximate 4:2 compressors to accumulate the partial products by modifying the respective accurate truth table, and then use them to build approximate multipliers. The negative aspect of this technique is that there is no parameter to adjust the accuracy of the multiplier. Lin and Lin [11] introduce a high accuracy approximate 4 × 4 Wallace tree multiplier by employing a 4:2 approximatecounter that reduces the partial product stages, and then use it to build larger multipliers. However, the delay is large due to the array structure of the multiplier. Kyaw et al. [12] propose a multiplier that divides the operands in two parts: an accurate multiplication-based part that includes the MSBs and an approximate nonmultiplication-basedpart for the LSBs that does not generate partial products. This design delivers significant reductions in power and delay, but only for specific input combinations. Moreover, Liu et al. [13] propose an approximate multiplier with configurable error recovery that uses inaccurate fast adders for the partial product additions. Although this multiplier delivers low power, the error imposed cannot be predicted, as it depends on the carry propagation. Narayanamoorthy et al. [14] statically split the operands in three m-bit segments and perform the multiplication utilizing the segment that contains the most significant nonzero bit. However, this approach exhibits small scalability, as m should be at least half the operand bit-width in order to keep the accuracy in acceptable limits. Reference [15] extended this idea to enable dynamic range multiplications. The dynamic partition technique requires extra components for the signed multiplications, adding significant hardware overhead. Recently, Zendegani et al. [21] proposed a multiplier that rounds the input operands into the nearest exponent of two. Finally, [23] replaces the floating-point operations with fixed-point ones, and by applying the proposed stochastic rounding, achieves good accuracy results in training deep neural networks while delivering high energy savings by limiting the data precision representation. The modified Booth encoding is commonly used in signed multipliers [16]–[18], [24],[25]. Although these techniques perform fast multiplications, the number of the partial products is not reduced in most cases, in contrast with our design. Zervakis et al. [16] introduce the partial product perforation technique, where they omit the generation of some partial products based on the modified Booth encoding. Jiang et al. [17] propose an approximate radix-8 booth multiplier that uses an approximate adder for producing±3 A, and combine this idea with the truncation method. Recently, Liu et al. [18] designed approximate modified Booth encoders by Another significant aspect of this paper is that the error imposed depends only on the configuration parameter k, and as a result, it can be calculated without the need for exhaustive simulations. Consequently, a precise estimation of the output quality can be extracted for the application's inputs, giving the flexibility to target the maximum energy reduction for a specific error bound.

## 2. Approximate Hybrid High Radix Multipliers

High radix encodings offer partial products reduction, and as a result, their accumulation requires smaller trees, leadingto energy, area, and/or delay savings. However, high radix encodingsrequirecomplexencodingandpartial productgeneration circuits, negating thus the benefits of the partial products reduction. In this section, the proposed hybrid high radix encodingand the performedapproximationsfor simplifying its circuit complexity are presented. In the proposed technique, the multiplicand B is encoded using the approximate high radix encoding, generating ˜ B, and the approximate multiplication A·˜ B is performed.Finally, its adaptation on inexact 16-bit hardware multipliers is described, and a qualitative analysis is conducted, targeting to estimate the potential area gains.

### A. Hybrid High Radix Encoding

In the proposed hybrid high radix encoding, B is divided in two groups: the MSB part of n−k bits and the LSB part of k bits. The configuration parameter, k ≥ 4, is an even number, namely, k = 2m: m ∈ Z, with m ≥ 2. The MSB part is encoded using the radix-4 (modified Booth) encoding, while the LSB part is encoded with the high radix-$2^K$ encoding

$$B = -b_{n-1}2^{n-1} + \sum_{i=0}^{n-2} b_i 2^i = \sum_{\substack{j=k/2 \\ k\geq 4}}^{n/2-1} y_j^{R4} 4^j + y_0^{R2^k} \qquad (1)$$

where

$$y_j^{R4} = -2b_{2j+1} + b_{2j} + b_{2j-1} \qquad (2)$$

and

$$y_0^{R2^k} = -2^{k-1}b_{k-1} + 2^{k-2}b_{k-2} + \cdots + b_0. \qquad (3)$$

The radix-4 encoding includes (n − k)/2 digits $y^{R4}_j \in \{0,\pm1,\pm2\}$, while $y^{R2K}_0 \in \{0,\pm1,\pm2,\pm3,...,\pm(2k-1-1), -2k-1\}$

corresponds to the radix-2k encoding. Overall, B is encoded with (n−k)/2+1 digits. The above hybrid high radix encoding is characterized by increased logic complexity, due to the high radix values of yR2k 0 that are not power of two, and thus, an approximate version is proposed. However, in order to retain high accuracy, the radix-4 encoding of the MSB is performed accurately. In particular, in the approximate encoding, all the values that are not power of two and the k −4 smallest powers of two as well, are rounded to the nearest of the 4 largest powers of two or 0, so that the sum of all the values of the approximate digit ˆ yR2k 0 is 0. We choose to keep only the four largest powers of two, so that the radix-2k encoding circuit requires only about the double area in comparison with the accurate radix-4 encoder. Therefore, B is approximately encoded as follows:

$$\tilde{B} = \sum_{\substack{j=k/2 \\ k\geq 4}}^{n/2-1} y_j^{R4} 4^j + \hat{y}_0^{R2^k}$$

where

$$y_j^{R4} \in \{0, \pm 1, \pm 2\}$$

and

$$\hat{y}_0^{R2^k} \in \{0, \pm 2^{k-4}, \pm 2^{k-3}, \pm 2^{k-2}, \pm 2^{k-1}\}.$$

TABLE I
ACCURATE RADIX-4 ENCODING TABLE

| Input | | | R4 Digit | Output | | |
|---|---|---|---|---|---|---|
| $b_{2j+1}$ | $b_{2j}$ | $b_{2j-1}$ | $y_j^{R4}$ | $sign_j$ | $\times 2_j$ | $\times 1_j$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 2 | 0 | 1 | 0 |
| 1 | 0 | 0 | -2 | 1 | 1 | 0 |
| 1 | 0 | 1 | -1 | 1 | 0 | 1 |
| 1 | 1 | 0 | -1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 |

TABLE II
APPROXIMATE RADIX-$2^k$ ENCODING TABLE

| $R2^k$ Digit | | Output | | | | |
|---|---|---|---|---|---|---|
| $y_0^{R2^k}$ | $\hat{y}_0^{R2^k}$ | $sign$ | $\times 2^{k-1}$ | $\times 2^{k-2}$ | $\times 2^{k-3}$ | $\times 2^{k-4}$ |
| $[0, 2^{k-5})$ | 0 | 0 | 0 | 0 | 0 | 0 |
| $[2^{k-5}, 2^{k-4}+2^{k-5})$ | $2^{k-4}$ | 0 | 0 | 0 | 0 | 1 |
| $[2^{k-4}+2^{k-5}, 2^{k-3}+2^{k-4})$ | $2^{k-3}$ | 0 | 0 | 0 | 1 | 0 |
| $[2^{k-3}+2^{k-4}, 2^{k-2}+2^{k-3})$ | $2^{k-2}$ | 0 | 0 | 1 | 0 | 0 |
| $[2^{k-2}+2^{k-3}, 2^{k-1})$ | $2^{k-1}$ | 0 | 1 | 0 | 0 | 0 |
| $[-2^{k-1}, -2^{k-2}-2^{k-3})$ | $-2^{k-1}$ | 1 | 1 | 0 | 0 | 0 |
| $[-2^{k-2}-2^{k-3}, -2^{k-3}-2^{k-4})$ | $-2^{k-2}$ | 1 | 0 | 1 | 0 | 0 |
| $[-2^{k-3}-2^{k-4}, -2^{k-4}-2^{k-5})$ | $-2^{k-3}$ | 1 | 0 | 0 | 1 | 0 |
| $[-2^{k-4}-2^{k-5}, -2^{k-5})$ | $-2^{k-4}$ | 1 | 0 | 0 | 0 | 1 |
| $[-2^{k-5}, 0)$ | 0 | 1 | 0 | 0 | 0 | 0 |

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 48, Issue 1, No. 1, January 2019

Table I presents the accurate radix-4 encoding. The output signals $\text{sign}_j$, $\times 1_j$, and $\times 2_j$ define the radix-4 digit $y_j^{R4}$. Their logic equations are the following:

$$\text{sign}_j = b_{2j+1} \quad (7)$$

$$\times 1_j = b_{2j-1} \oplus b_{2j} \quad (8)$$

$$\times 2_j = (b_{2j+1} \oplus b_{2j}) \cdot \overline{(b_{2j-1} \oplus b_{2j})}. \quad (9)$$

Table II presents the approximate radix-$2^k$ encoding. The logic equations of the encoding signals that define the radix-$2^k$ digit $\hat{y}_0^{R2^k}$ are the following:

$$\text{sign} = b_{k-1} \quad (10)$$

$$\times 2^{k-4} = (\bar{b}_{k-2} \cdot \bar{b}_{k-3} \cdot \bar{b}_{k-4} + b_{k-2} \cdot b_{k-3} \cdot b_{k-4}) \cdot (b_{k-4} \oplus b_{k-5}) \quad (11)$$

$$\times 2^{k-3} = \bar{b}_{k-1} \cdot \bar{b}_{k-2} \cdot (\bar{b}_{k-3} \cdot b_{k-4} \cdot b_{k-5} + b_{k-3} \cdot \bar{b}_{k-4}) + b_{k-1} \cdot b_{k-2} \cdot (b_{k-3} \cdot \bar{b}_{k-4} \cdot \bar{b}_{k-5} + \bar{b}_{k-3} \cdot b_{k-4}) \quad (12)$$

$$\times 2^{k-2} = \bar{b}_{k-2} \cdot b_{k-3} \cdot (b_{k-1} + b_{k-4}) + b_{k-2} \cdot \bar{b}_{k-3} \cdot (\bar{b}_{k-1} + \bar{b}_{k-4}) \quad (13)$$

$$\times 2^{k-1} = \bar{b}_{k-1} \cdot b_{k-2} \cdot b_{k-3} + b_{k-1} \cdot \bar{b}_{k-2} \cdot \bar{b}_{k-3}. \quad (14)$$

The effectiveness of the approximate hybrid radix encoding technique is explored with its application to 16-bit signed numbers, for k =6,8,10, namely, the LSBs are encoded using We choose to keep only the four largest powers of two, so that the radix-2k encoding circuit requires only about the double area in comparison with the accurate radix-4 encoder.

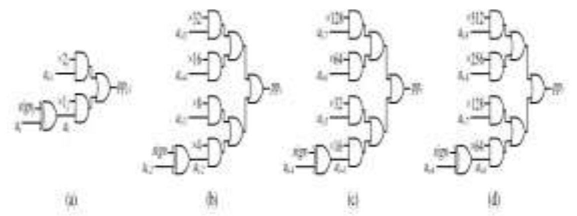


Fig. 1. i-bit partial product generator based on (a) accurate radix-4 encoding and the approximate (b) radix-64, (c) radix-256, and (d) radix-1024 encoding. $a_i$: i-bit of operand A, $\bar{a}_i = a_i \oplus$ sign.

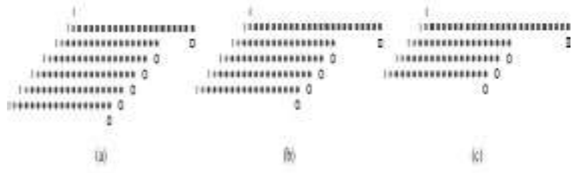


Fig. 2. Partial product tree based on the hybrid encoding of accurate radix-4 and approximate (a) radix-64, (b) radix-256, and (c) radix-1024 encoding. ■: partial product bits from the approximate high radix encoding. •: partial product bits from the accurate radix-4 encoding. ■ and •: inverted MSBs of the partial products. □ and ○: sign factors.

the radix-64,radix-256,and radix-1024encoding,respectively. In the radix-64 encoding, the bits of B are grouped as in

$$\underbrace{b_{15}b_{14}\overbrace{b_{13}}}_{y_7^{R4}}{}^{\!\!y_6^{R4}}b_{12}\underbrace{b_{11}b_{10}\overbrace{b_9}}_{y_5^{R4}}{}^{\!\!y_4^{R4}}b_8\underbrace{b_7b_6\overbrace{b_5b_4b_3b_2b_1b_0}^{y_0^{R64}}}_{y_3^{R4}}. \quad (15)$$

The following values of the digit yR64 0 are rounded to their nearest power of two: ±1, ±3, ±5, ±6, ±7, ±9, ..., ±15, ±17, ..., ±31 are rounded to ±4, ±8, ±16, or ±32, while the smallest powers of two, i.e., ±1 and±2, are rounded to 0 or±4. In radix-1024 encoding, the bits of B are grouped as follows:

$$\underbrace{b_{15}b_{14}\overbrace{b_{13}}}_{y_7^{R4}}{}^{\!\!y_6^{R4}}b_{12}\underbrace{b_{11}b_{10}\overbrace{b_9b_8b_7b_6b_5b_4b_3b_2b_1b_0}^{y_0^{R1024}}}_{y_5^{R4}}. \quad (16)$$

Similarly, the nonpowers of two are rounded to ±64, ±128, ±256, or ±512, and the smallest powers of two (±1, ±2, ±4, ±8, ±16, ±32) are rounded to 0 or ±64. The encoder's

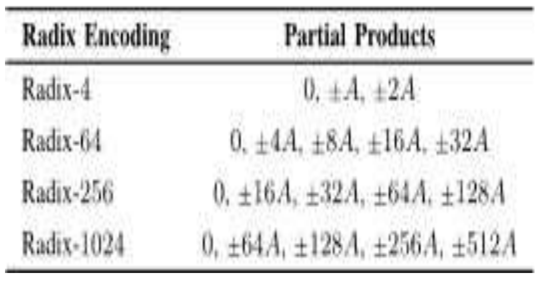
Industrial Engineering Journal
ISSN: 0970-2555
Volume : 48, Issue 1, No. 1, January 2019

inputs are the bits b9,b8,...,b0, the approximate radix-1024 digit is ˆ yR1024 0 ∈{ 0,±64,±128,±256,±512}, and the outputsignals that define ˆ yR1024 0 are sign, ×64, ×128, ×256, ×512.

**B. Partial Product Generation**

In the proposed hybrid encoding, the n−k MSBs of B are encoded with the accurate radix-4 encoding, while the k LSBs are encodedwith an approximateradix-2k encoding.The accurate radix-4 encoder produces the signals defined in (7)–(9), whereas the approximate high radix encoder produces the signals of (10)–(14). Overall, there is a reduction of k/2−1 partial products generated in the multiplication A· ˜ B.

TABLE III
PARTIAL PRODUCTS PER RADIX ENCODING



| Radix Encoding | Partial Products |
|---|---|
| Radix-4 | 0, ±A, ±2A |
| Radix-64 | 0, ±4A, ±8A, ±16A, ±32A |
| Radix-256 | 0, ±16A, ±32A, ±64A, ±128A |
| Radix-1024 | 0, ±64A, ±128A, ±256A, ±512A |

In Fig. 1, four partial product generators are presented, i.e., the circuit of the accurate radix-4 encoding and the ones of the three approximate high radix encodings discussed in Section III-A. The partial products created from each encoding are shown in Table III. In addition, the three hybrid high radix encodings create the partial product trees shown in Fig. 2. The trees also include the encoding's correction term (constant terms and sign factors). The implementation of the partial product accumulation can be chosen by the designer. In this paper, an accurate Wallace tree [26] is used to implement the partial product's sum, whereas the two outputs produced by the Wallace tree are added using a prefix (fast) adder.

Overall, the multiplication circuit consists of stages of operand hybrid radix encoding, partial product generation, partial product accumulation, and final addition. The proposed approximate multipliers are named RAD2k, showing the selected approximate high radix encoding, e.g., RAD64, RAD256, and RAD1024.

**C. Unit Gate Model**

The advantage of the approximate hybrid high radix multipliers is their simple logic, resulting in fast operation and low power performance. Although overhead is added because of the encoding circuits, it is insignificant because of the approximations made. Also, this offset is compensated withthe partial product generators that deliver low area, and the reduction of the number of the partial products.

TABLE IV
UNIT GATES AREA OF THE RADIX ENCODERS AND PARTIAL PRODUCTS GENERATORS

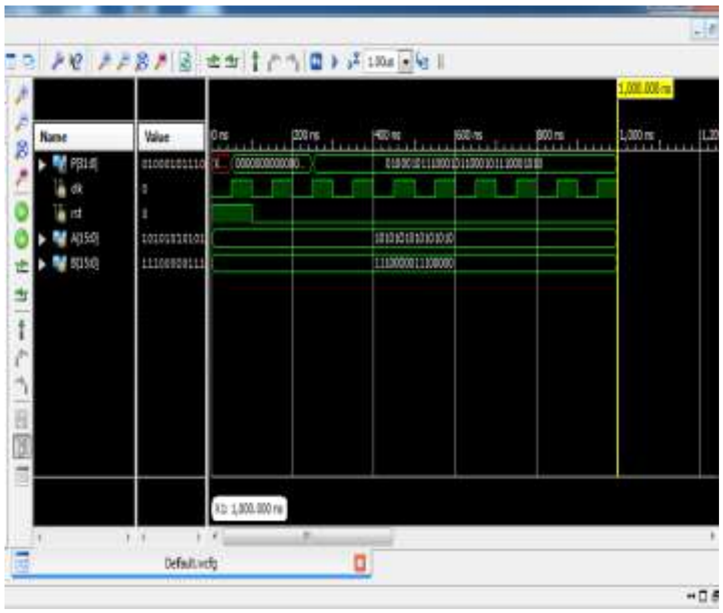
| Circuit | Unit Gates [27] |
|---|---|
| Radix-4 Encoder | 5.5 |
| Radix-$2^k$ Encoder | 41.5 |
| Radix-4 Partial Product Generator | 5 |
| Radix-$2^k$ Partial Product Generator | 9 |

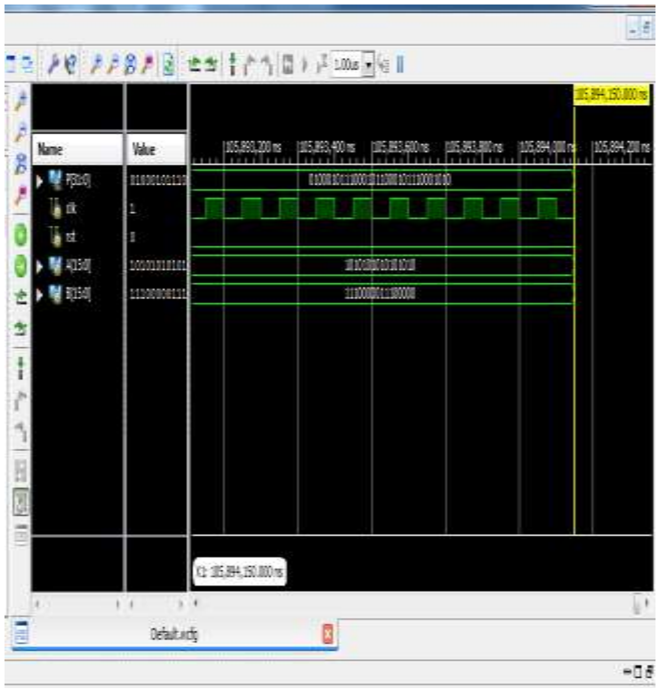
TABLE V
UNIT GATES AREA SAVINGS OF RAD MULTIPLIERS

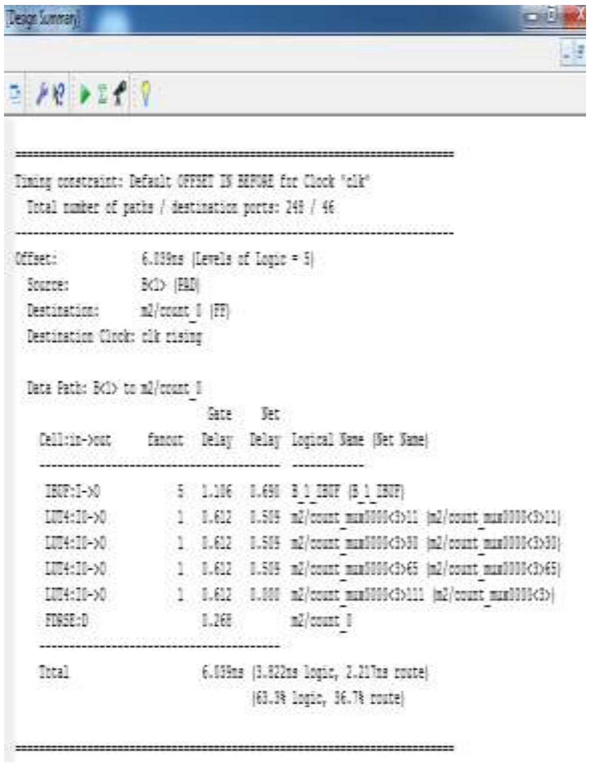
| Multiplier Stage | ACCR4 | RAD64 | RAD256 | RAD1 |
|---|---|---|---|---|
| Radix-4 Encoding | 44 | 25 | 20 | 15 |
| Radix-$2^k$ Encoding | – | 41.5 | 41.5 | 41. |
| Radix-4 PP Gener. | 640 | 400 | 320 | 240 |
| Radix-$2^k$ PP Gener. | – | 180 | 198 | 216 |
| PP Accumulation | 784 | 560 | 448 | 336 |
| Final Addition | 400 | 400 | 400 | 400 |
| **Total Unit Gates** | 1868 | 1606.5 | 1427.5 | 1248 |
| **Reduction %** | – | 14.00 | 23.58 | 33.1 |

In order to give a theoretical evaluation of the proposed multipliers, an area gate model is included. The area evaluation is performed by using the unit gate model of [27]: a XOR-2 gate counts as 2 unit gates, an AND-2 or an OR-2 gate is equal to 1 unit gate, and a NOT gate is equal to 0.5 unit gate. According to this model, Table IV presents the number of unit gates of each circuit used in our multipliers. Overall, the accurate radix-4 n-bit multiplier uses n/2 radix-4encodersandn2/2 radix-4partial product generators to produce the n/2 n-bit partial products. Similarly, the proposed approximate radix-2k, with k ≥4, requires (n−k)/2 radix-4 encoders, 1 radix-2k encoder, n(n − k)/2 radix-4 partial product generators, and n + k − 2 radix-2 k partial product generators.

## 4.RESULTS





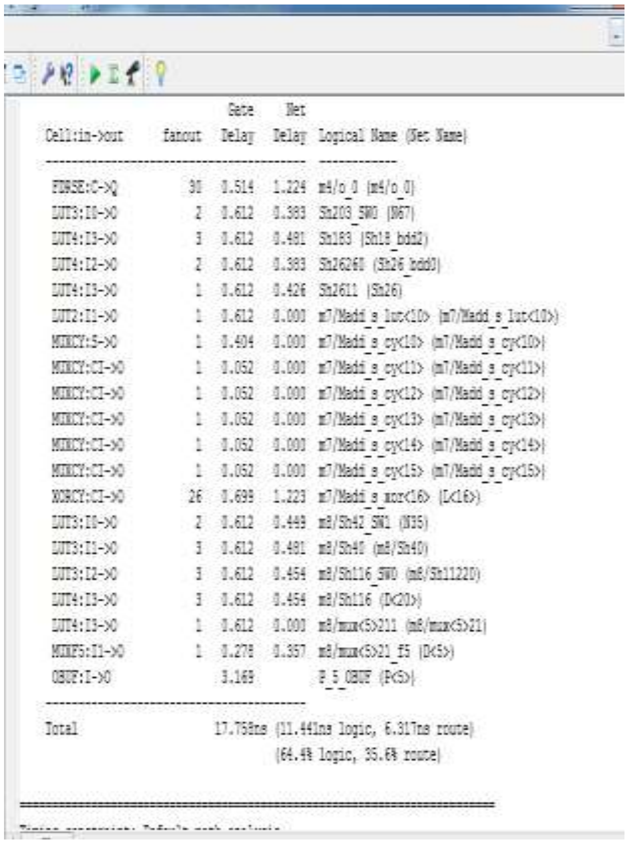


```
Design Summary

================================================================================
Timing constraint: Default OFFSET IN BEFORE for Clock 'clk'
  Total number of paths / destination ports: 249 / 46
--------------------------------------------------------------------------------
Offset:              6.039ns (Levels of Logic = 5)
  Source:            B<1> (PAD)
  Destination:       m2/count_0 (FF)
  Destination Clock: clk rising

  Data Path: B<1> to m2/count_0
                              Gate     Net
    Cell:in->out      fanout  Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O             5   1.106   0.690  B_1_IBUF (B_1_IBUF)
     LUT4:I0->O            1   0.612   0.509  m2/count_mux0000<3>11 (m2/count_mux0000<3>11)
     LUT4:I0->O            1   0.612   0.509  m2/count_mux0000<3>30 (m2/count_mux0000<3>30)
     LUT4:I0->O            1   0.612   0.509  m2/count_mux0000<3>65 (m2/count_mux0000<3>65)
     LUT4:I0->O            1   0.612   0.000  m2/count_mux0000<3>111 (m2/count_mux0000<3>)
     FDRSE:D                   0.268          m2/count_0
    ----------------------------------------
    Total                      6.039ns (3.822ns logic, 2.217ns route)
                                       (63.3% logic, 36.7% route)

================================================================================
```



```
                              Gate     Net
    Cell:in->out      fanout  Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDRSE:C->Q           30   0.514   1.224  m4/o_0 (m4/o_0)
     LUT3:I0->O            2   0.612   0.383  Sh203_SW0 (N67)
     LUT4:I3->O            3   0.612   0.481  Sh183 (Sh18_bdd2)
     LUT4:I2->O            2   0.612   0.383  Sh26260 (Sh26_bdd0)
     LUT4:I3->O            1   0.612   0.426  Sh2611 (Sh26)
     LUT2:I1->O            1   0.612   0.000  m7/Madd_s_lut<10> (m7/Madd_s_lut<10>)
     MUXCY:S->O            1   0.404   0.000  m7/Madd_s_cy<10> (m7/Madd_s_cy<10>)
     MUXCY:CI->O           1   0.052   0.000  m7/Madd_s_cy<11> (m7/Madd_s_cy<11>)
     MUXCY:CI->O           1   0.052   0.000  m7/Madd_s_cy<12> (m7/Madd_s_cy<12>)
     MUXCY:CI->O           1   0.052   0.000  m7/Madd_s_cy<13> (m7/Madd_s_cy<13>)
     MUXCY:CI->O           1   0.052   0.000  m7/Madd_s_cy<14> (m7/Madd_s_cy<14>)
     MUXCY:CI->O           1   0.052   0.000  m7/Madd_s_cy<15> (m7/Madd_s_cy<15>)
     XORCY:CI->O          26   0.699   1.223  m7/Madd_s_xor<16> (L<16>)
     LUT3:I0->O            2   0.612   0.449  m8/Sh42_SW1 (N35)
     LUT3:I1->O            3   0.612   0.481  m8/Sh40 (m8/Sh40)
     LUT3:I2->O            3   0.612   0.454  m8/Sh116_SW0 (m8/Sh11220)
     LUT4:I3->O            3   0.612   0.454  m8/Sh116 (D<20>)
     LUT4:I3->O            1   0.612   0.000  m8/mux<5>211 (m8/mux<5>21)
     MUXF5:I1->O           1   0.278   0.357  m8/mux<5>21_f5 (D<5>)
     OBUF:I->O                 3.169          P_5_OBUF (P<5>)
    ----------------------------------------
    Total                     17.758ns (11.441ns logic, 6.317ns route)
                                       (64.4% logic, 35.6% route)

================================================================================
```

## CONCLUSION

In this paper, we propose an inexact cross breed high radix encoding for creating the halfway results of a marked multiplier. The MSBs of the multiplicand are encoded with the exact radix-4 encoding, while its k LSBs are encoded with an estimated high radix-2k encoding, with k being a configuration parameter that changes the tradeoff among exactness and vitality utilization. The blunder of the proposed procedure pursues a Gaussian conveyance with close to zero normal. Contrasted and best in class estimated multipliers, the proposed ones establish better inexact plan elective, beating them in both vitality utilization and precision. Besides, we demonstrated the efficiency of the proposed multipliers when connected, all things considered, applications. At last, the proposed method is adaptable, conveying higher vitality reserve funds for a similar mistake, as the multiplier's size increments.

## REFERENCES

[1] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Analysis and characterization of inherent application resilience for approximate computing," in Proc. ACM/IEEE Design Autom. Conf., May 2013, pp. 1–9.

[2] S. T. Chakradhar and A. Raghunathan, "Best-effort computing: Re-thinking parallel software and hardware," in Proc. ACM/IEEE Design Autom. Conf., Jun. 2010, pp. 865–870.

[3] A. Lingamneni, C. Enz, K. Palem, and C. Piguet, "Highly energyefficient and quality-tunable inexact FFT accelerators," in Proc. IEEE Custom Integr. Circuits Conf., Sep. 2014, pp. 1–4.

[4] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," IEEE Trans. Comput.Aided Design Integr. Circuits Syst., vol. 32, no. 1, pp. 124–137, Jan. 2013.

[5] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, "IMPACT: IMPrecise adders for low-power approximate computing," in Proc. 17th IEEE/ACM Int. Symp. Low-Power Electron. Design, Aug. 2011, pp. 409–414.

[6] N. Zhu, W. L. Goh, W. Zhang, K. S. Yeo, and Z. H. Kong, "Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 18, no. 8, pp. 1225–1229, Aug. 2010.

[7] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 57, no. 4, pp. 850–862, Apr. 2010.

[8] A. K. Verma, P. Brisk, and P. Ienne, "Variable latency speculative addition: A new paradigm for arithmetic circuit design," in Proc. Design, Autom. Test Eur., Mar. 2008, pp. 1250–1255.

[9] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an under designed multiplier architecture," in Proc. 24th Int. Conf. VLSI Design, Jan. 2011, pp. 346–351.

[10] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design and analysis of approximate compressors for multiplication," IEEE Trans. Comput., vol. 64, no. 4, pp. 984–994, Apr. 2015.