ENERGY-EFFICIENT VLSI IMPLEMENTATION OF MULTIPLIERS WITH DOUBLE LSB OPERANDS

K. Jyosna[1], P.M. Naseer Hussain[2] , Dr.S.Siddeswara Reddy[3]

[1]PG scholar, Dept of ECE, VITS, Proddatur, Kadapa, AP, India

[2] AssistantProfessor, Dept of ECE, VITS,Proddatur, Kadapa, AP, India

[3] Associate Professor & HOD , Dept of ECE, VITS, Proddatur, Kadapa, AP, India

Abstract: Redundant Binary Partial Product Generator technique are used to reduce by one row the maximum height of the partial product array generated by a radix16 Modified Booth Encoded multiplier, without any raise in the delay of the partial product creation Block. In this paper, we describe an optimization for binary radix-4 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 achieved. This Arithmetic multipliers increase the performance of ALU and Processors. We evaluate the proposed approach by comparison with Normal Booth Multiplier. Logic synthesis showed its efficiency in terms of area, delay and power. Simulation results show that the proposed Multiplier based designs significantly improve the area, delay and power consumption when the word length of each operand in the multiplier is 64 & n-bits. The proposed architecture of this paper analysis the delay and area using Xilinx 14.2.

Keywords: Modified Booth Encoding, Radix-16, Pipeline, Multiplier, Enhanced, Carry Select Adder, Binary Excess Converter.

1. Introduction

Multiplier is one of the basic hardware block used for many digital and high performance systems such as FIR filters, digital signal processors and microprocessors etc. Many high speed low power multiplication algorithms and architectures have been proposed. Advances in technology have permitted many researchers to design multipliers which offer both high-speed and regularity of layout, thereby making them suitable for VLSI implementation. Digital signal processing requires efficient multiplication operations with the highest possible speed without compromising the power budget. In general, basic multiplication algorithm can be divided into three following steps.1) partial product (pp) generation, 2) partial product reduction and 3) final carry propagated addition [1-2]. In the first step, a set of partial product rows is generated where each one is the result of the product of one bit of the multiplier by multiplicand. For example, if we consider the multiplication X x Y with both X and Y on n bits and of the form xn_1 . . . x0 and yn_1 . . . y0, then the ith row is, in general, a proper left shifting of yi x X, i.e., either a string of all zeros when yi = 0, or the multiplicand X itself when yi = 1. In this case, the number of PP rows generated during the first phase is clearly n [1-4]. Recoding of binary numbers was first hinted at by Booth [5] four decades ago.

MacSorley[6] proposed a modification of Booth's algorithm a decade after. Modified booth encoding (MBE) [6] is a technique that has been introduced to reduce the no of pp rows with a maximum height of [n/2] +1 rows. More specifically, Two's complement multiplier [7] using radix-4 MBE generates a pp array with a maximum height of [n/2] rows without any increase of delay, each row of the pp array follows the one of the following possible values: all zeros, +X, +2X [8].This pp reduction may increases the speed of the multiplier. During pp reduction phase, all pp rows are reduced by using compression tree [9-10].Since the intermediate addition values is not important, the outcome of this phase is a result represented in redundant carry save form, i.e., as two rows, which allows for much faster implementations. The final (carry-propagated) addition has the task of adding these two rows and of presenting the final result in a non-redundant form, i.e., as a single row. In this work, we introduce an idea to reduce the pp array with a maximum height of [n/3] rows by using radix-8 booth recoding process. A similar study aimed at the reduction of the maximum height to [n/3] but using a different approach has recently presented interesting results in [11]. Thus, in the following, we will evaluate and compare the proposed approach with the technique in [7].

2. LITERATURE SURVEY

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. Complex 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.

3. EXISTING METHOD

The Booth algorithm has been used to improve the sign correction issues of signed number multiplication; however, the original Booth algorithm does not reduce the number of PPs. A Modified Booth Encoding (MBE) method (also known as the radix-4 Booth algorithm) has been further proposed. It reduces the number of PP rows by half. The complexity of the parallel multiplier is

reduced significantly by applying MBE. The power consumption and the delay of the entire multiplier are also reduced. Let A = aN−1aN−2 · · · a2a1a0 be the multiplicand and B = bN−1bN−2 · · · b2b1b0 be the multiplier. The multiplier bits are encoded; so they are grouped in sets of three adjacent bits. The two side bits overlap with neighboring groups, except the first multiplier bit group. As per the encoded results from A, the Booth decoders select -2A, - A, 0, A, or 2A to generate the PP rows. 2A is obtained by a simple 1-bit left shift of the multiplicand. The negation operation is achieved by inverting each bit of A and adding 1 at its least significant bit (LSB) position. This is referred to as the correction term in this work. Therefore, the PP of each line can be easily generated by either shifting or inverting the multiplicand bits. The circuit diagram of the MBE scheme is shown in Fig. 2. Table 1 shows the K-Map of a conventional MBE. Therefore, the output of the Booth encoder ppij is given as follows:

$$ppij=(b2i \oplus b2i-1)(b2i+1 \oplus aj) + (b2i \oplus b2i-1)(b2i+1 \oplus b2i)(b2i+1 \oplus aj-1) \quad (1)$$

The correction term for the negation operation is as follows:

$$Ei = b2i+1b2i + b2i+1b2i-1 \quad (2)$$

As per Eq. (2), the correction term (i.e., Ei) of the negation operation is almost equal to the MSB of the multiplier except when b2i+1b2ib2i−1 = 111. Ei can be further simplified by reconsidering this entry in the MBE truth table. In [36], it is observed that all the entries in the 6th column of Table 1 can be changed to 1 to achieve a simplified E0 i along with a slight increase in complexity of a pp0 ij as follows:

$$pp0\ ij=(b2i \oplus b2i-1)(b2i+1 \oplus aj) + (b2i \oplus b2i-1)(b2i+1 \oplus b2i)(b2i+1 \oplus aj-1) + b2i+1b2ib2i-1 \quad (3)$$

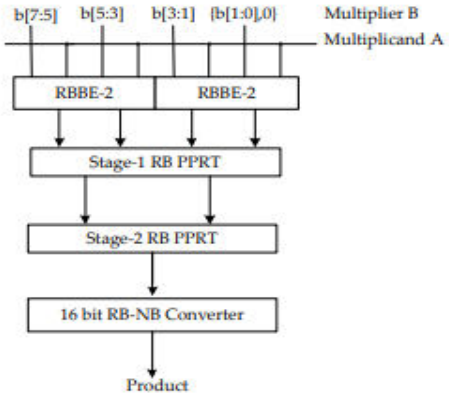

Fig. 1. Overall structure of an 8-bit RB multiplier

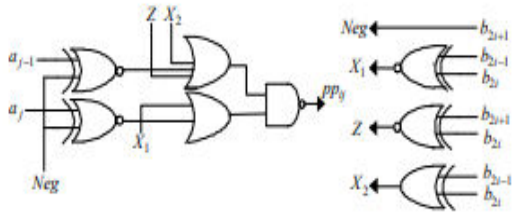

Fig. 2. MBE scheme: encoder and decoder

4. PROPOSED METHOD

The K-map of the radix-4 approximate modified Booth encoder (R4AMBE6), i.e., appij6−1, with 6 errors in the Kmap is shown in Table 1, where 0 denotes an entry in which a '1' is replaced by a '0' and 1 denotes a '0' entry that has been replaced by a '1'. Only 6 entries are modified to simplify the Booth encoding. This approximate design relies on the property that the truth table is as symmetrical as possible for a design with the least complexity. Therefore, three modifications change a '1' to a '0' and three modifications change a '0' to a '1' in the K-map. The output of R4AMBE6 is given as follows:

appij6−1 = (b2i + b2i−1)(b2i+1 ⊕ ai) (17)

Ei = (b2i+1b2i) + (b2i+1b2i−1)

Compared with the exact MBE, R4AMBE6 can significantly reduce both the complexity and the critical path delay of Booth encoding. The error rate, denoted by Pbe, is given by: Pbe = 6/32 = 18.75% (19) The gate level structure of R4AMBE6 is shown in Fig. 5. The conventional design of MBE (Fig. 2) consists of four XNOR-2 gates, one XOR-2 gate, one OR-3 gate, one OR2 gate and one NAND-2 gate. The R4AMBE6 design only requires one XOR-2 gate, one AND-2 gate and one OR-2 gate

The approximate Radix-4 with the new modified Booth Encoding (R4ANMBE6), i.e., app 0 ij6−1 , with 6 errors in the K-map is shown in Table 1. In this approximate design, there are more entries changed from ’0’ to ’1’ than those changed from ’1’ to ’0’. Therefore, the approximate results produced by R4ANMB6 will be usually larger than its exactcounterpart.

TABLE 1: K-Map of R4AMBE6

$a_j a_{j-1}$ \ $b_{2i+1}b_{2i}b_{2i-1}$	000	001	011	010	110	111	101	100
00	0	0	0	0	1	(1)	1	(0)
01	0	0	(0)	0	1	(1)	1	0
11	0	1	1	1	0	0	0	0
10	0	1	(1)	1	0	0	0	(0)

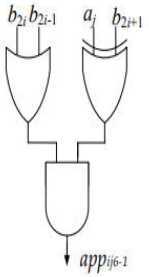

Fig. 3. The gate-level circuit of the proposed R4AMBE6

From Table 1, the approximate pp 0 ij is derived as follows:

app 0 ij6−1 = b2i+1 ⊕ aj + b2ib2i−1 (20) E

0 i = b2i+1

This design further reduces the complexity of the correction term (i.e., Ei). Its error rate is the same as R4AMBE6: The gate level circuit of R4ANMBE6 is shown in Fig. 4. The R4AMBE6 design only requires one XOR-2 gate, one AND-2 gate and one OR-2 gate, which has the same complexity as R4AMBE6

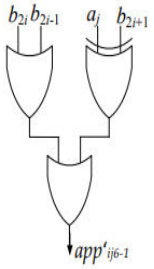

Fig. 4. The gate-level circuit of the proposed R4AMBE6

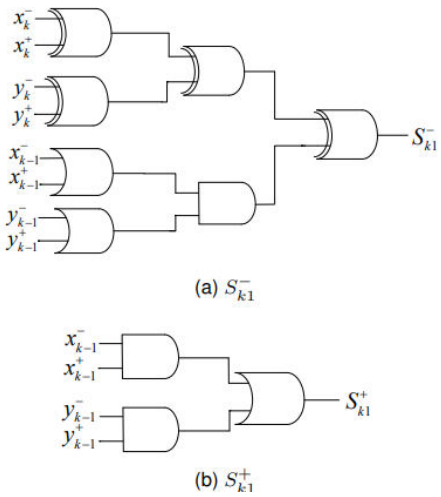

Fig. 5. The gate level circuit of ARBC-1

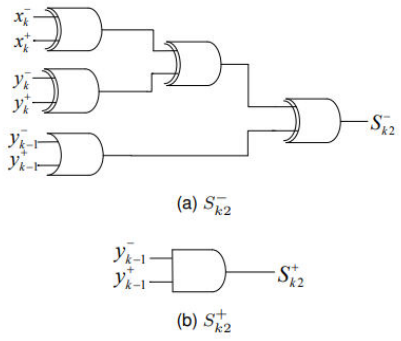

Fig. 6. The gate level circuit of ARBC-2

Efficient designs must ensure that the error between the approximate RB compressor and its exact counterpart remains as small as possible. The final results of compression are the same when (x − k ,x + k) is equal to either (1, 0) or (0, 1). So, when the result of the approximate RB compressor is (x − k ,x + k) = (1, 0) rather than the exact compression result (x − k ,x + k) = (0, 1), the result is still correct. Therefore, the following four types of compression results are equivalent: (0, 0) = (0, 0), (0, 1) = (1, 0), (1, 0) = (0, 1) and (1, 1) = (1, 1).

The Proposed accurate RB-NB Converter

As the approximate Booth encoders and approximate RB compressors generate results that are generally larger than the exact results, the biased approximate results can be compensated using ARNC with smaller values. The principle of compensation is to use an approximate adder that produces results that are smaller than its exact results. Therefore, the complexity of the RB-NB converter can be reduced, while the overall accuracy of the approximate RB multipliers is also increased. The truth table of a possible approximate RB-NB converter is given by Table 6, a simple NOR gate is used in the approximate RB-NB digit converter as follows: S 0 k = S − k + S + k (29)In this section, the approximate RB multipliers are designed as follows. The proposed approximate Booth encoders, i.e., R4AMBE6 and R4ANMBE6, are used to generate approximate PPs. Approximate RB compressors, i.e., ARBC-1 and ARBC-2, are used for RB PP reduction, which can reduce the delay for compression and significantly improve speed performance when the operand size is a power of 2. The approximate RB-NB converter neither (made of NOR gates) is used to convert the RB digit to the NB digit. An approximation factor p (p=1, 2, ..., 2N) that has been proposed is used. This is defined as the number of least significant PP columns that are generated by theapproximate Booth encoders.

TABLE 2: The Truth Table of RB-NB Conversion

RB Digit (S_k^+, S_k^-)	Carry-in(C_k)	ENB (S_k)	ANB1 (S'_{k1})	ANB2 (S'_{k2})	ANB3 (S'_{k3})
(0,0)	0	1	1	1	1
(0,0)	1	0	1	1	1
(0,1)	0	0	0	1	0
(0,1)	1	1	0	1	0
(1,0)	0	0	0	1	0
(1,0)	1	1	0	1	0
(1,1)	0	1	0	0	1
(1,1)	1	0	0	0	1

As p column PPs are already approximate, the approximate PPs can be accumulated with an approximate RB 4:2 compressor to further improve speed and reduce power consumption. For the same reason, the p least significant RB digits are also converted by the approximate RB-NB converter to calculate the final product. Four approximate RB multipliers are proposed. They use the exact regular PP array when p ≤ (N − 4) (as detailed in [36]), and the approximate regular PP array when p > (N − 4) where the bit pairs (E2, 0) and (E3, 1) of Fig. 4 can be ignored in the approximate design of the RB Booth multipliers; however they all use the proposed approximate RB-NB converter. For the 2N-p most significant PP columns, the exact design is used for the final results. The four RB multipliers are different in the p PP columns as follows:

1) The first approximate RB multiplier (R4ARBM1) uses R4AMBE6 to generate the p least significant PP columns and ARBC-1 to perform the approximate PP accumulation.
2) The second approximate RB multiplier (R4ARBM2) uses R4AMBE6 to generate the p least significant PP columns and ARBC-2 for the corresponding approximate PP accumulation.
3) The third approximate RB multiplier (R4ARBM3) uses R4ANMBE6 to generate the p least significant PP columns and ARBC-1 to perform the approximate PP accumulation.
4) The fourth approximate RB multiplier (R4ARBM4) uses R4ANMBE6 to generate the p least significant PP columns and ARBC-2 to perform the approximate PP accumulation. As the error can be controlled by the approximation factor p, a reasonable accuracy can be achieved for different applications.

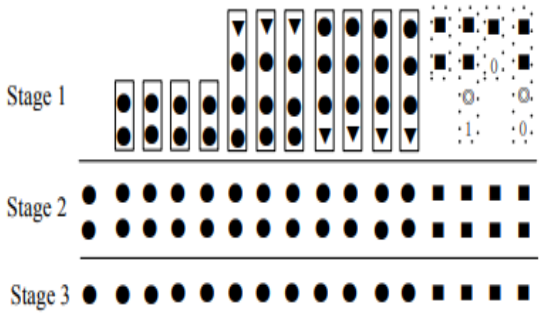

Fig. 9. The dot diagram of the proposed 8-bit approximate RB multiplier

Fig. 9 shows an approximate 8-bit RB multiplier with p=4 using an approximate Booth encoder, an approximate RB compressor, an approximate RB-NB converter, and an exact regular PP. A box with a solid line denotes the use of an exact RB compressor, and a box with a dotted line denotes an approximate RB 4:2 compressor. The exact PP is represented by ● , the modified PP after logic simplification is represented by ▼ , while the approximate PP term is represented by represents Ei .

5. SIMLUATION RESULTS

The developed project is simulated and verified their functionality. Once the functional verification is done, the RTL model is taken to the synthesis process using the Xilinx ISE tool. In synthesis process, the RTL model will be converted to the gate level netlist mapped to a specific technology library. Here in this Spartan 3E family, many different devices were available in the Xilinx ISE tool. In order to synthesis this design the device named as “XC3S500E” has been chosen and the package as “FG320” with the device speed such as “-5”. This design is synthesized and its results were analyzed as follows:

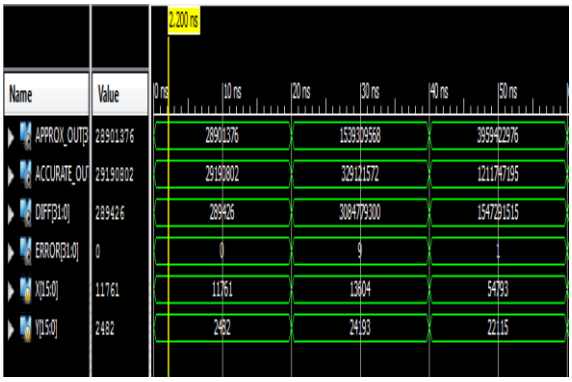

Fig 7: SIMULATION RESULT

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	157	17600	0%
Number of fully used LUT-FF pairs	0	157	0%
Number of bonded IOBs	64	100	64%
Number of DSP48E1s	2	80	2%

Fig 8: PROPOSED DESIGN SUMMARY

```
                                   Gate     Net
Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
----------------------------------------  -----------
 IBUF:I->O            10   0.000   0.592  Y_5_IBUF (Y_5_IBUF)
 LUT6:I0->O            1   0.043   0.343  L2/Mmux_out23 (L2/Mmux_out22)
 LUT3:I1->O            1   0.043   0.428  L2/Mmux_out25_SW0 (N10)
 LUT6:I3->O           27   0.043   0.395  L2/Mmux_out25 (out4y<1>)
 LUT6:I5->O            7   0.043   0.578  T2/GND_3_o_GND_3_o_sub_15_OUT<1>11 (T2/GND_3_o_GND_3_o
 LUT6:I0->O            1   0.043   0.000  T2/Mmux_out2<2>_51 (T2/Mmux_out1<3>_51)
 MUXF7:I1->O           1   0.172   0.000  T2/Mmux_out1<3>_4_f7 (T2/Mmux_out1<3>_4_f7)
 MUXF8:I0->O           2   0.123   0.284  T2/Mmux_out1<3>_2_f8 (YAT<6>)
 DSP48E1:A3->P1        2   2.823   0.433  A1/Maddsub_mult (out1<1>)
 LUT3:I0->O            8   0.043   0.582  Sh11 (Sh1)
 LUT6:I0->O            1   0.043   0.279  Sh811 (Sh81)
 OBUF:I->O                 0.000          out_17_OBUF (out<17>)
----------------------------------------
Total                      7.334ns (3.419ns logic, 3.915ns route)
                                   (46.6% logic, 53.4% route)
```

Fig 9: PROPOSED TIMING REPORT

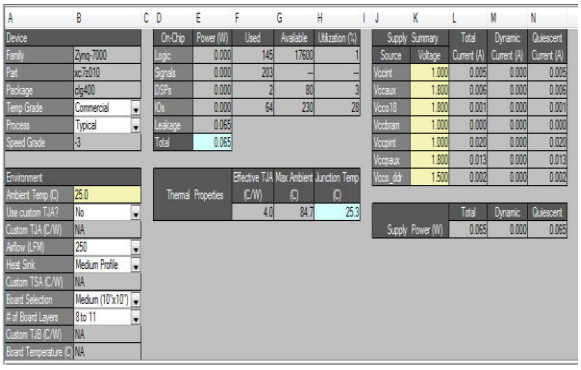

Fig 10: POWER SUMMARY

Table 3: COMPARISION TABLE

	EXISTING	PROPOSED
LUTS	195	157
TIME DELAY	25.418ns	7.334ns
POWER CONSUMPTION	0.114w	0.065w

5. CONCLUSION: The multiplier using the proposed algorithm achieves better power-delay products 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-4 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 ≥ 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.

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] H. Jiang, J. Han and F. Lombardi, "A comparative review and evaluation of approximate adders," in Proc. 25th

IEEE/ACM Great Lakes Symposium on VLSI, 2015, pp. 343-348.
[7] S.-L. Lu, Speeding up processing with approximation circuits, Computer, vol. 37, no. 3, pp. 67-73, 2004.
[8] 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. VLSI Syst., vol. 18, no. 8, pp.1225-1229, 2010.
[9] K. Du, P. Varian and K. Mohanram, "High-performance reliable variable latency carry select addition," in Proc. Design, Automation and Test in Europe Conference and Exhibition (DATE), 2012, pp. 1257-1262.
[10] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie and C. Lucas, "Bioinspired imprecise computational blocks for efficient vlsi implementation of soft-computing applications," IEEE Trans. Circuits Syst.: Part I Regular Papers, vol. 57, no. 4, pp. 850-862, 2010.
[11] M. Imani, D. Peroni, and T. Rosing, "CFPU: Configurable floating point multiplier for energy-efficient computing," in Proc. 54th Annual Design Automation Conference (DAC), 2017, pp. 18-22.
[12] J. Mitchell, "Computer multiplication and division using binary logarithms," IRE Trans. Electronic Computers, vol. EC-11, no. 4, 512-517, 1962.
[13] J. Low and C. Jong, "Unifiedmitchell-based approximation for efficient logarithmic conversion circuit," IEEE Trans. Computers, vol. 64, no. 6, pp. 1783-1797, 2015. [14] M. Sullivan and E. E. Swartzlander, Jr., "Truncated error correction for flexible approximate multiplication," in Proc. Conf. Record the 46th Asilomar Conf. Signals, Systems and Computers, 2012, pp. 355-359. [15] K. Y. Kyaw, W. L. Goh and K. S. Yeo, "Low-power high-speed multiplier for error-tolerant application," in Proc. Electron Devices and Solid-State Circuits, 2010, pp. 1-4.
[16] S. Hashemi, R. Bahar and S. Reda, "DRUM: A dynamic range unbiased multiplier for approximate applications," in Proc. IEEE/ACM International Conference on Computer Design, 2015, pp. 418-425.
[17] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in Proc. 24th Int. Conf. VLSI Design, 2011, pp. 346-351.
[18] K. Bhardwaj, P. Mane and Jorg Henkel, "Power-and area-efficient ¨ approximate wallace tree multiplier for error-resilient systems," in Proc. 15th IEEE Int. Symp. Quality Electronic Design, 2014, pp. 263-269.
[19] H. Jiang, J. Han, F. Qiao, F. Lombardi, "Approximate radix-8 Booth multipliers for low-power and high performance operation," IEEE Trans. Computers, vol. 65, pp. 2638-2644, Aug. 2016.
[20] V. Leon, G. Zervakis, D. Soudris and K. Pekmestzi, "Approximate hybrid high radix encoding for energy-efficient inexact multipliers," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, no. 3, pp. 421-430, 2018