International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 04 | Apr 2020 www.irjet.net p-ISSN: 2395-0072

APPROXIMATE UNSIGNED MULTIPLIER WITH VARIED ERROR RATE

S.MEHATHAB[1], O.HOMA KESAV[2]

[1]PG Student, Dept. of ECE, Annamacharya Institute of Technology and Sciences Kadapa, Andhra Pradesh, India
[2]Assistant Professor, Dept. of ECE, Annamacharya Institute of Technology and Sciences Kadapa, Andhra Pradesh, India

---

ABSTRACT- *In many of today's applications like digital signal processing, image processing etc, multipliers, address play significant role especially approximate circuits. As these circuits improve performance and are energy efficient with loss of some accuracy. In this paper, an approximate multiplier is proposed for high-performance DSP application with improved delay and area. This approximate multiplier limits the carry propagation for fast partial product accumulation, and this can be done with two techniques namely OR gate and proposed approximate adder in configurable error recovery circuit. Compared to Wallace multiplier. These two multipliers achieve better accuracy and 62% reduction delay and 39% reduction in area. This can also be used in a 16*16 design with concept of truncation, applied for two techniques reduced to half of the least significant partial products. Compared with Wallace multiplier these can save 50% to 60% of energy. Compared with existing approximate multipliers, proposed one's show significant advantages in accuracy and low area delay products. Image processing applications, including image sharpening and smoothing, are considered to show the quality of approximate multipliers in error-tolerant applications.*

Index terms: Approximate circuits, adder, multiplier, error recovery, image processing.

1. INTRODUCTION:

Approximate Computing (AC) is a wide spectrum of techniques that relax the accuracy of computation in order to improve performance, energy, and/or another metric of merit. AC exploits the fact that several important applications, like machine learning and multimedia processing, do not require precise results to be useful. For instance, we can use a lower resolution image encoder in applications where high-quality images are not necessary. In a data centre, this may lead to large savings in the amount of required processing, storage and communication band width.

Nowadays, the majority of our computations are being done either on mobile devices or in large data centres (think of cloud computing). Both platforms are sensitive to power consumption. That is, it would be nice if we can extend the operation time of smartphones and other battery powered devices before the next recharge. We know that applying any lossy compression algorithm (JPEG for example) to a raw image will result in an approximate image. Such compression often comes (by design) with little human perceptible loss in image quality. Also, image encoders usually have tuneable algorithmic *knobs*, like compression level, to trade off image size with its quality. Therefore, strict exactness may not be required and an inaccurate result may be sufficient due to the limitation of human perception. As one of the key components in arithmetic circuits, adders are studied for approximate implementation. As the carry propagation chain is usually shorter than the width of an adder, the hypothetical adders use a reduced number of less significant input bits to calculate the sum bits. In [15], approximate 4 × 4 and 8 × 8 bit Wallace multipliers are designed by using a carry-in prediction method. Then, they are used in the design of approximate 16 × 16 Wallace multipliers, referred to as AWTM. The AWTM is configured into four different modes by using a different number of approximate 4 × 4 and 8 × 8 multipliers. These approximate multipliers are designed for unsigned operation. Signed multiplication is usually implemented by using a Booth algorithm. Approximate designs have been proposed for fixed width Booth multipliers.

The proposed multiplier can be configured into two designs by using OR gates and the proposed approximate adders for error reduction, which can be referred as approximate multipliers M1 and M2 respectively. Different levels of error recovery can also be achieved by using a different number of MSBs for error recovery in both multipliers. As per the analysis, the proposed multipliers have significantly shorter critical paths and lower power dissipation than the traditional Wallace multiplier. Functional and circuit simulations are performed to evaluate the performance of the multipliers. Image sharpening and smoothing are considered as approximate multiplication-based DSP applications. Experimental results indicate that the proposed approximate multipliers perform well in these error-tolerant applications. The proposed designs can be used

as effective library cells for the synthesis of approximate circuits.

1.1. The Approximate Adder

The design of a new approximate adder is shown. This adder operates on a set of pre-processed inputs. The input pre-processing (IPP) is based on the commutativity of bits with the same weights in different addends. For example, consider two sets of inputs to a 4-bit adder: i) A = 1010, B = 0101 and ii) A = 1111, B = 0000. Clearly, the additions in i) and ii) produce the same result. In this process, the two input bits Ai Bi = 01 is equal to Ai Bi = 10 (with i being the bit index) due to the commutativity of the corresponding bits in the two operands. The basic rule for the IPP is to switch Ai and Bi if Ai = 0 and Bi = 1 (for any i), while keeping the other combinations (i.e., Ai Bi = 00, 10 and 11) unchanged. By doing so, more 1's are expected in A and more 0's are expected in B. If A'_i B'_i are the ith bits in the pre-processed inputs, the IPP functions are given by:

$$A'_i = A_i + B_i \quad (1)$$

$$B'_i = A_i B_i \quad (2)$$

Equations (1) and (2) compute the propagate and generate signals used in a parallel adder such as the carry lookahead adder (CLA). The proposed adder can process data in parallel by reducing the carry propagation chain. Let A and B denote the two input binary operands of an adder, S be the sum result, and E represent the error vector. A_i , B_i , S_i and E_i are the ith least significant bits of A, B, S and E, respectively. A carry propagation chain starts at the ith bit when $B'_i = 1$, $A'_{i+1} = 1$, $B'_{i+1} = 0$. In an accurate adder, S_{i+1} is 0 and the carry propagates to the higher bit. In the proposed approximate adder, S_{i+1} is set to 1 and an error signal is generated as $E_{i+1} = 1$. This stops the carry signal from propagating to the higher bits. Hence, a carry signal is produced only by the generate signal, i.e., $C_i = 1$ only when $B'_i = 1$, and it only propagates to the next higher bit, i.e., the (i +1)th position.The logical functions are given by

$$S_i = B'_{i-1} + B'_i A'_i, \quad (3)$$

$$E_i = B'_i B'_{i-1} A'_i . \quad (4)$$

By replacing A'_i and B'_i using (1) and (2) respectively, the logic functions with respect to the original inputs are given by

$$S_i = (A_i \oplus B_i) + A_{i-1}B_{i-1}, \quad (5)$$

$$E_i = (A_i \oplus B_i) A_{i-1}Bi_{-1}, \quad (6)$$

where i is the bit index, i.e., i = 0, 1,⋯ , n for an n-bit adder.

Let A−1 = B−1 = 0 when i is 0, thus, $S_0 = A_0 \oplus B_0$ and $E_0 = 0$. Also, $E_i = 0$ when A_{i-1} or Bi−1 is 0. Consider an n-bit adder, the inputs are given by $A = A_{n-1} \cdots A_1A_0$ and $B = B_{n-1} \cdots B_1B_0$, the exact sum is $S = S_{n-1} \cdots S_1S_0$. Then, Si can be computed as $S_i + E_i$ and thus, the exact sum of A and B is given by

$$S = S + E. \quad (7)$$

In (7) '+' means the addition of two binary numbers rather than the 'OR' function. The error E is always non-negative and the approximate sum is always equal to or smaller than the accurate sum. This is an important feature of this adder because an additional adder can be used to add the error to the approximate sum as a compensation step. While this is intuitive in an adder design, it is a particularly useful feature in a multiplier design as only one additional adder is needed to reduce the error in the final product.

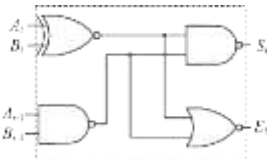


Fig : the approximate adder cell.

1.2. Proposed Approximate Multiplier

An important feature of the proposed approximate multiplier is the simplicity to use approximate adders in the partial product accumulation. The resulting design has a critical path delay that is shorter than a conventional one-bit full adder, because the new n-bit adder can process data in parallel. The approximate adder has a rather high error rate, but the feature of generating both the sum and error signals at the same time reduces errors in the final product. An adder tree is utilized for partial product accumulation; the error signals in the tree are then used to compensate the error in the output to generate a product with a better architecture of the proposed approximate multiplier is shown in Fig. 1. In the proposed design, the simplification of the partial product accumulation stage is accomplished by using an adder tree, in which the number of partial products is reduced by a factor of 2 at each stage of the tree. This adder tree is usually not implemented using accurate multi-bit adders due to the long latency but it is less complex than a conventional adder and has a much shorter critical path delay. The Wallace multiplier or exact multiplier includes full adders, half adders and compressors also, which reduces the critical path with increase in area. When the multipliers of different sizes are considered, the complexity increases so, when compared with proposed design which is simple for various multiplier sizes.

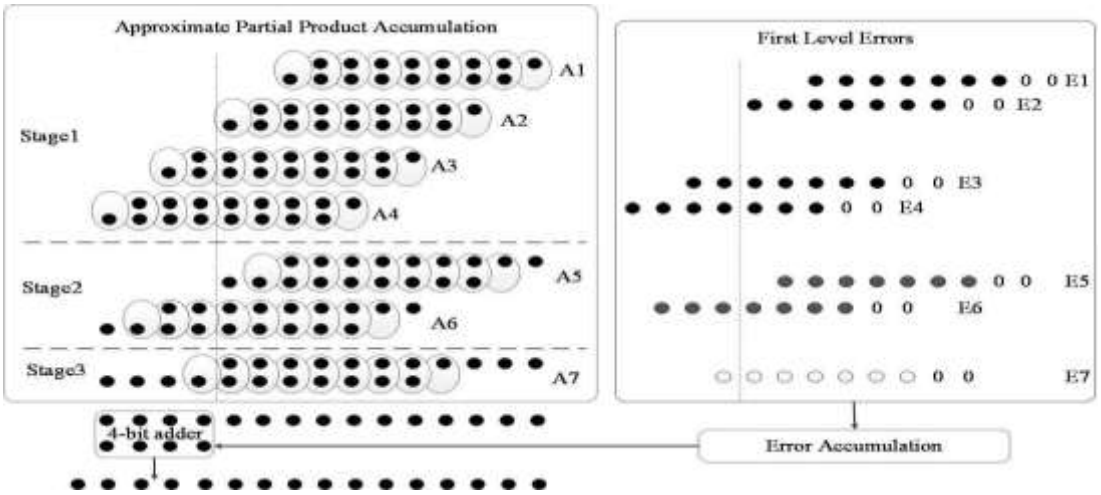


Fig. 1. An approximate multiplier with partial error recovery using 5 MSBs of the error vector. : a partial product, sum or an error bit generated at the first stage;: an error bit generated at the second stage;: an error bit generated at the last stage

2. ERROR REDUCTION

As approximate adder produces sum and error signals. An error reduction circuit is used to improve accuracy by two steps (i) error accumulation (ii) error recovery. Two error accumulation methods are proposed, for approximate multipliers M1 and M2 . Fig. 2 shows the symbols for an OR gate, a full adder and half adder cell and an approximate adder cell used in the error accumulation tree

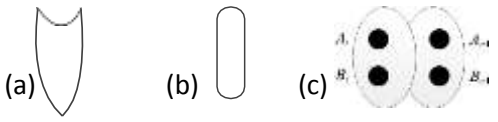


Fig. 2. Symbols for (a) an OR gate, (b) a full adder or a half adder (c) an approximate adder cell

2.1. Error Accumulation for Approximate Multiplier M1:

If the obtained error signals are added with exact adders, it may cause inaccurate results with increased complexity. Consider the observation that the error vector of each approximate adder tends to have more 0's than 1's. The probability that the error vectors have an error bit '1'at the same position, is quite small. Hence, an OR gate can be to approximately compute the sum of the errors for a single bit. If m error vectors (denoted by $E1, E2, ..., Em$) need to be accumulated, then the sum of these vectors is obtained as

$$E_i = E\,1_i\ OR\ E\,2_i\ OR\ ...\ OR\ Em_i. \quad (8)$$

To reduce errors, an accumulated error vector is added to the adder tree output using a conventional CPA (e.g. a carry look- ahead adder) MSBs of the error signals are used to compensate the outputs to further reduce the overall complexity. The number of MSBs is selected according to the extent that errors must be compensated. In the example of Fig. 1, 5 MSBs (i.e. the $(11\text{-}14)^{th}$ bits, no error is generated at the 15^{th} bit position) are considered for error recovery and therefore, 4 error vectors are considered (i.e., the error vectors E 3, E 4, E 6 and E 7). The error vectors of the other three adders are less significant than the 11^{th} bit, so they are not considered. The accumulated error E is obtained using (8); then, the final result is found by adding E to S using a fast accurate CPA. The error accumulation scheme is shown in Fig. 3. As no error is generated at the least significant two bits of each approximate adder Ai $(i$ 1, 2, ... , 7), the least significant two bits of each error vector Ei are not accumulated.

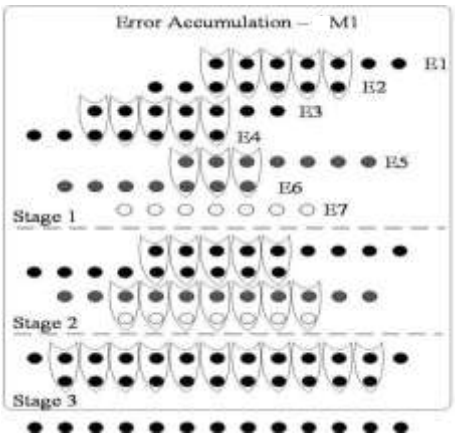


Fig. 3:Error accumulation tree for M1

2.2. Error accumulation for approximate multiplier M2:

The error accumulation scheme for M2 is shown in Fig. 4. To introduce the design of M2, an 8X8 multiplier with two inputs X and Y is considered. For example, consider the first two partial product vectors X_0Y_7, X_0Y_6, ..., X_0Y_0 and X_1Y_7, X_1Y_6, ..., X_1Y_0 accumulated by the first approximate adder (A1 in Fig. 1), where X_i and Y_i are the i^{th} least significant bits of X and Y, respectively. Recall from (6) for the approximate adder, the condition for E_i = 1 is

$$A_{i-1} = B_{i-1} = 1 \text{ and } A_i \neq B_i \quad (9)$$

For the first approximate adder in the partial product accumulation tree, its inputs are $A = X_0Y_7, X_0Y_6, ..., X_0Y_0$ and $B = X_1Y_7, X_1Y_6, ..., X_1Y_0$. Thus, the i^{th} least significant bits for A and B are $A_i = X_0Y_i$ and $B_i = X_1Y_{i-1}$, respectively. If X_0 or X_1 is 0, there will be no error in this approximate adder because either A or B is zero. Therefore, no error occurs unless X_0 X_1 = 11. When X_0 X_1 = 11, A_i and B_i are simplified to Y_i and Y_{i-1}, respectively. Then to calculate E_i ,A_{i-1}, B_{i-1}, A_i and B_i are replaced by Y_{i-1}, Y_{i-2}, Y_i and Y_{i-1}, respectively. For E_i to be 1, Y_i $Y_{i-1}Y_{i-2}$ 011 according to (9). Therefore, an error only occurs when the input has "011" as a bit sequence. Based on this observation, the "distance" between two errors in an approximate multiplier is at least 3 bits. Thus, two adjacent approximate adders in the first stage of the partial product tree cannot have errors at the same column, because the errors in a lower approximate adder are those in the upper adder shifted by 2 bits when both errors exist. The errors in two adjacent approximate adders can then be accurately accumulated by OR gates, e.g., an OR gate can be used to accumulate the two bits in the error vectors E 1 and E 2 in Fig. 1. After applying the OR gates to accumulate E1 and E2 as well as E3 and E4, the four error vectors are compressed into two. For E5, E 6 and E 7, they are generated from the approximate sum of the partial products rather than the partial products. Another interesting feature of the proposed approximate adder is as follows. Assume E_i =1 in (6), then A_{i-1}=B_{i-1}=1 and $A_i \neq B_i$. Since A_{i-1}=B_{i-1}=1, i.e., A_{i-1} B_{i-1} 0, it is easy to show that E_{i-1}= 0.Moreover, as $A_i \neq B_i$,i.e., A_i B_i = 0, then E_{i+1}=0. Thus, once there is an error in one bit, its neighbouring bits are error free, i.e., there are no consecutive error bits in one row. Therefore, there is no carry propagation path longer than two bits when two error vectors are accumulated, and the error vectors are accurately accumulated by the proposed approximate adder.

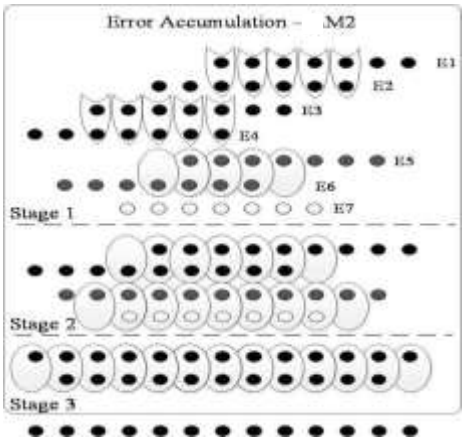


Fig4: Error accumulation tree for M2

Based on the above analysis, E 5 and E 6 can be accumulated by one approximate adder in the first stage . After the first stage of error accumulation, three vectors are generated, and another two approximate adders are used to accumulate these three vectors as well. Hereafter, the proposed *nxn* approximate multiplier with *k*-MSB OR-gate based error reduction is referred to as an *n/k* M1, while an *nxn* approximate multiplier with *k*-MSB approximate adder based error reduction is referred to as an *n/k* M2. The structures of M1 and M2 are shown in Fig. 5.

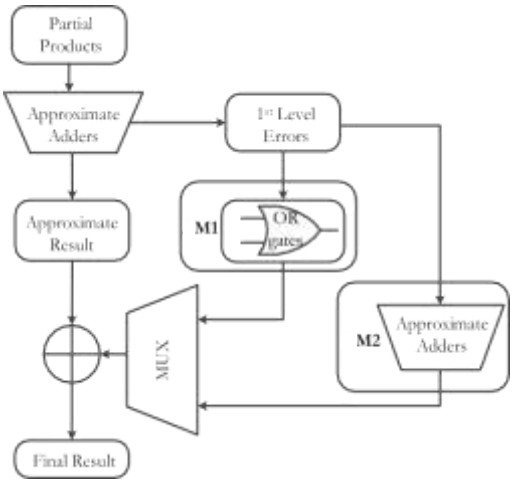


Fig5: Block diagram of approximate multipliers

2.3. 16x16 Approximate Multipliers

In both M1 and M2, all the error vectors are compressed to one vector which is then added back to the approximate output of the partial product. Compared to 8x8 designs, 16x16 multipliers have more error vectors and too much data would be wasted if same strategy is used. In the modified design the error vector generated by approximate adders are compressed to two final error

vectors in a 16x16 multiplier to compress the 8 error vectors and the remaining error vector to another error vector. Truncation can also be applied to the proposed deign if large input operands are used therefore, 16 LSBs of the partial product are truncated in 16x16 M1 and M2 resulting in truncated M1 (TM1) and truncated M2 (TM2).

3. ACCURACY EVALUATION:

To reduce circuit complexity and delay the arithmetic accuracy is sacrificed. The error distance (ED) and mean error distance (MED) are proposed to evaluate the performance of approximate arithmetic circuits. For multipliers, ED is defined to be the arithmetic difference between the accurate product (M) and the approximate product (M^J), i.e.,

$$ED = |M^J - M| \quad (10)$$

MED is the average of EDs for a set of outputs (obtained by applying a set of inputs). A metric applicable for comparing multipliers of different sizes is the normalized MED (NMED), i.e.,

$$NMED = MED/M_{max}, \quad (11)$$

Where M_{max} is the maximum magnitude of the output of an exact multiplier i.e, $(2^n-1)^2$ for an nxn multiplier. The relative error distance (RED) is defined as

$$RED = |M'-M / M| = ED / |M| \quad (12)$$

The error rate (ER) is defined as the percentage of erroneous outputs among all outputs. For evaluating worst case output, maximum error (ME) is defined as the maximum error distance normalized by the maximum output of the exact multiplier.

Accuracy evaluation of the 8x8 multiplier:

The functions if the proposed multipliers are realized using Matlab and an exhaustive simulation is performed for an 8x8 approximate multiplier. Approximate multipliers with both the OR gate and the approximate adder based error reduction, as well as the accurate adder based error reduction, are evaluated. Fig. 6 shows the four metrics (NMED, MRED, ER and ME) in logarithm when using different numbers of MSBs for error reduction. . Let *k* denote the number of MSBs used for error reduction. The values of NMED and MRED of M1 and M2 drop drastically as *k* is increased from 4 to 8 and continues to drop as *k* increases, even though at a slower rate. In terms of ER, the values for the proposed multipliers decrease slowly with an increasing *k* from 4 to 8 and then follow a sharper decline. The MEs for M1 and M2 do not decrease as much as the multiplier with an accurate error accumulation when *k* increases. This occurs because some errors at the higher bit positions are not accurately accumulated by using the OR gates or the proposed approximate adders. The values of NMED, MRED, ER and ME finally drop to zero for the accurate error accumulation when 14 MSBs are used for error reduction. For the same *k*, M2 has a better performance than AM1 in terms of NMED, MRED and ER. For example, if 8 MSBs are used for error reduction, the NMED of M2 is 0.17% while it is 0.30% for M1. Moreover, if 14 MSBs are used for error reduction, M1 has an error rate of 17.6%, while the error rate of M2 can be as low as 5.8%.

These four figures also indicate that the proposed approximate multiplier has a rather high error rate, but the errors are usually very small compared to both the accurate and the largest possible output of the approximate multiplier.

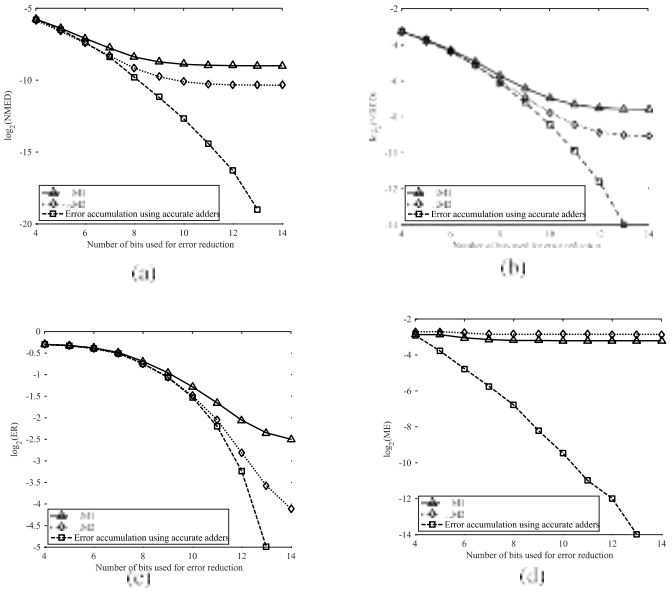


Fig 6: Accuracy comparison of the approximate 8x8 multipliers using approximate and exact error accumulation vs different number of bits for error reduction (a)NMED (b)MRED (c) ER (d) ME

4. DELAY, AREA AND POWER EVALUATION

4.1:Delay estimate

Based on the linear model of the delays of a full adder (Fig. 7(a)) and the approximate adder cell (Fig. 7(b)) are approximately $4\tau_g$ and $3\tau_g$, respectively, where τ_g is an approximate "gate delay". The delay of an XOR (or XNOR) gate is $2\tau_g$ due to its higher complexity compared to an NAND (or NOR gate). For an $n \times n$ approximate multiplier (n is the power of 2), there are $m = \log_2 n$ stages in the partial product accumulation tree. The first stage with 2^m rows of partial products are compressed to 2^{m-1} rows of partial products in the second stage and 2^{m-1} error vectors. These error

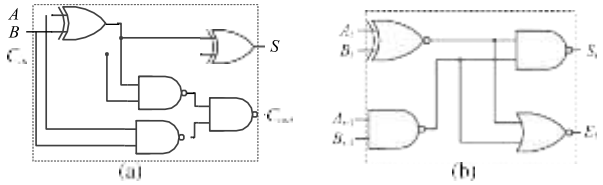


Fig 7: (a) An exact full adder and (b) the approximate adder cell.

vectors are then compressed (i.e., accumulated) using OR gates or approximate adders in a similar tree structure. Since the numbers of rows in the second partial product accumulation stage and the errors generated by the first stage are the same, it takes *m*- 1 stages for both stages to be compressed to 1

n	8	16	32	64	2^l
$D_{M1}(T_g)$	12	16	24	26	$3l+\log_2 l$
$D_{M2}(T_g)$	18	22	28	30	$3l+3\log_2 l$
$D_W(T_g)$	20	26	34	42	=6.5l

Table 2: Estimated delay of the proposed and conventional multipliers

when an *n*-row partial product tree is compressed to 1 row, errors from the $\log_2 n$ stages are also compressed to $\log_2 n$ error vectors, provided that the delays for compressing two partial products and accumulating two error vectors are the same. As the delay of an OR gate is shorter than that of the approximate adder, fewer error vectors remain after $\log_2 n$ stages in M1.The numbers of the remaining error vectors after $\log_2 n$ stages in both M1 and M2 are considered to be approximately $\log_2 n$. Then it takes $\log_2 \log_2 n$ stages to finally compress these $\log_2 n$ error vectors.

$$D_{Mi} = (\log_2 n)\, x3T_g + [\log_2 \log_2 n] x t_i \quad (13)$$

Where $t_i = t_g$ (the delay of an OR gate for M1) for i=1 and $t_i = 3t_g$ (the delay of an approximate adder for M2) for i=2 There are 4 compression stages I an 8x8 wallace tree is approximately given by

$$D_W = 4\,[\log_{1.5} n] t_g \quad (14)$$

Table 2 shows the delay of the partial product accumulation tree in both the proposed and Wallace multipliers. For a 16×16 multiplier, the delay of an exact multiplier tree is nearly 1.5 as large as the delay of the proposed multiplier tree. As the size of the multiplier increases, this factor is approximately 2. As a result, the proposed partial product accumulation design is 29% faster than the optimized Wallace multiplier. In summary, the proposed multiplier can significantly reduce the delay of the partial product accumulation tree, which scales with the size of the multiplier.

4.2: Area estimate: Let the area of a basic gate be α_g, and the area for an XOR (or XNOR) gate be $2\alpha_g$ .Then, the area of a full adder cell is $7\alpha_g$, and the area of the approximate adder cell is $5\alpha_g$. If the error signal E_i is not required, the circuit area for generating a sum S_i is $4\alpha_g$, i.e., an NOR gate is not needed.

As the number of partial product rows is reduced by 1 by using an (*n*-1)-bit approximate adder, (*n*-1)(*n*-1) bit approximate adders are required to compress the *n* partial product rows to one row. Also, (*n*-1) error vectors are generated, because each approximate adder produces an error vector. The number of OR gates (or approximate adders) used for error accumulation is determined by the number of MSBs used for error reduction (i.e., *k*). Thus, the area of the proposed partial product accumulation scheme is estimated to be

$$A_{Mi} = (n-1)^2 \times 4\alpha_g + \alpha_i, \quad (15)$$

Where, αi is the area of the error generation and accumulation circuit in Mi (i=1 or 2).In an nxn Wallace multiplier a full adder compresses 3 partial products to 2 i.e, one bit is reduced by using a full adder. Thus (n-2) rows of full adders are used to compress the n partial product rows to two, each row consist of approximately (n-1) full adders. The area of Wallace tree is given by

$$A_W = 7(n-2)(n-1)\alpha_g \quad (16)$$

K	4	6	8	14
$A_{M1}(\alpha_g)$	210	227	250	288
$A_{M2}(\alpha_g)$	218	272	308	389
$A_W(\alpha_g)$	302	302	302	302

Table 3: estimated area of the proposed and conventional 8x8 multipliers.

Table 3 shows the estimated areas of the Wallace tree and the partial product accumulation tree of the proposed multipliers using different numbers of MSBs for error reduction

4.3. Power estimate : The power consumption of a CMOS circuit consists of short-circuit power, leakage power and dynamic power [26]. Compared to the dynamic power, the short-circuit and leakage powers are relatively small and vary with device fabrication. Dynamic power is dissipated for charging or discharging the load capacitance when the output of a CMOS circuit switches. By using a probabilistic power analysis, the average dynamic power of a circuit is given by

$$P_{avg} = f_{clk} . V^2_{dd} \sum_{I=1}^{N} C_L(xi) . \alpha_{0\text{-}1}(xi) \quad (17)$$

where f_{clk} is the operating clock frequency of the circuit, V_{dd} is the supply voltage, N is the number of nodes in the circuit, $C_L\ (x_i)$ is the load capacitance at node x_i, and $\alpha_{0\to1}(x_i)$ is the probability of the logic transition from 0 to 1 at node x_i. $\alpha_{0\to1}(x_i)$ is computed by

$$\alpha_{0\to1}(x_i) = P_s(x_i)P_s(\bar{x}_i), \quad (18)$$

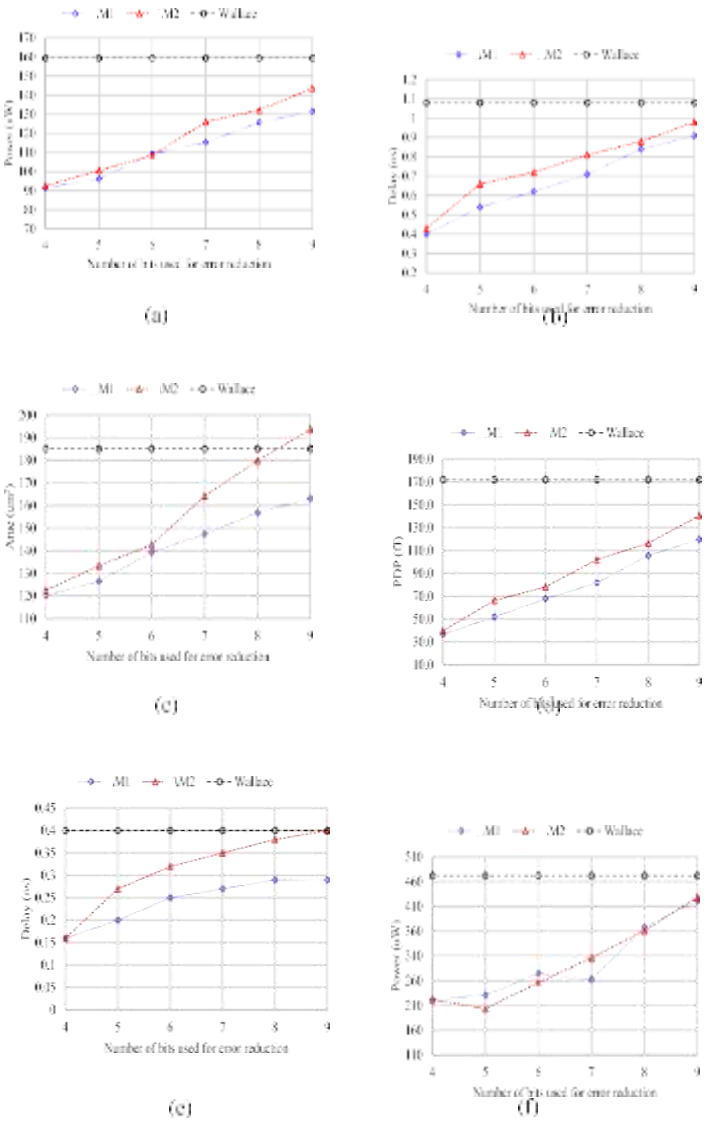
here $P_s\ (x_i)$ is the signal probability at node x_i; it is defined as the probability of a high signal value occurring at x_i.

As the basic components of the Wallace and the proposed multipliers, the full adder and the proposed approximate adder are analyzed using (17). In (17), f_{clk} and V_{dd} are the same for the two components, $C_L\ (x_i)$ depends on the fabrication. Thus, the difference in dynamic power dissipation between these two components is mainly caused by $\alpha_{0\ 1}(x_i)$.

As $P_s\ (S_i) < P_s\ (S)$ and $P_s\ (E_i) < P_s\ (C_{out})$, the dynamic power dissipated at the two outputs of the proposed approximate adder is lower than a full adder. As for the internal nodes, the full adder has one more node than the proposed approximate adder. Thus, the proposed approximate adder consumes lower dynamic power than a full adder. Moreover, the dynamic power consumed by the error vector accumulation circuit is very low due to the low switching activity at E_i. Consequently, the proposed approximate multiplier is more power-efficient than a Wallace multiplier.

5. Simulation Results

8x8 multipliers: M1 has shown advantages in speed and power consumption compared to a Wallace multiplier for FPGA implementations. Designs for 8x8 M1 using 4, 5, … , 9 MSBs for error reduction, 8 8 AM2 using 4, 5, … , 9 MSBs for error reduction, and the 8x8 optimized Wallace multiplier have been implemented in VHDL and synthesized by using the Synopsys Design Compiler (DC) with an industrial 28 nm CMOS process. Simulations are performed at a temperature of 25° C and a supply voltage of 1V. The modules for implementing the multiplier circuits are taken from the 28 nm library as C32_SC_12_CORE_LR_tt28_1.00V_25C The critical path delays of these multipliers are reported by the Synopsys DC tool. The power dissipation is calculated by prime time – PX tool using 10milloin random input combinations with a clock period of 2 ns. The delay, area, power and power delay product (PDP) is shown in fig.8, where the area is optimized to the smallest value for the result in (8a),(8b),(8c),(8d) and the critical path delay is constrained to the smallest value without violation of time in (8e,8f,8g,8h) the obtained power consumption is the total power i.e, dynamic and static powers. As M1 uses simple OR gate based error reduction technique it has shorter delay than M2. M1 and M2 with 4 bit error reduction are faster by 65% and 62% than the Wallace multiplier when optimized for area and 60%when optimised for delay. For the 8 bit error reduction these values are 24%(29%) and 17%(4%) respectively. The power and area of the multipliers show the same values as the delay. For the 8 bit error reduction technique the power savings of M1 and M2 are nearly 20% the area optimized M1 and M2 have small area nearly 23% than the accurate design. The area of M2 is larger when the number of bits is more than eight.



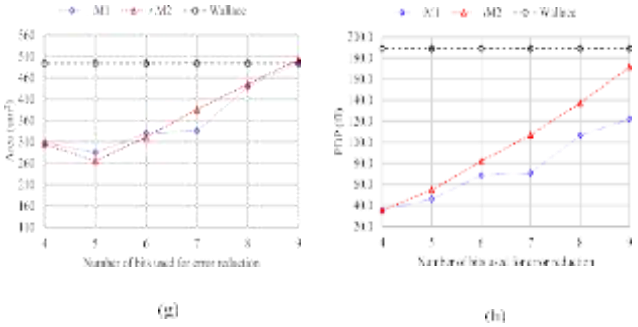


Fig. 8. Delay, power and area comparisons of proposed 8x8 approximate and Wallace multipliers. "Wallace" indicates the accurate 8 x8 Wallace multiplier, and the X-axis is not applicable for it. (a) Delay (optimized for area). (b) Power (optimized for area). (c) Area (optimized for area). (d) PDP (optimized for area). (e) Delay (optimized for delay). (f) Power (optimized for delay). (g) Area (optimized for delay). (h) PDP (optimized for delay)

Fig 8d and 8h show the power delay product of M1 and M2 are smaller than Wallace multiplier by 32 to 78% and 25 to 70% respectively.

6. CONCLUSION

In this paper, we propose a high performance and low area delay approximate partial product accumulation tree for a multiplier using newly proposed approximate adder which cuts the carry propagation and generates sum and error signal. OR gate and approximate order error reduction techniques are used which yields to two different approximate 8 bit multiplier designs as M1 and M2. Further modification is made for 16 bit multiplier designs by truncating 16LSBs of the partial products. Functional analysis has shown that proposed multipliers have small error distance which helps in achieving good accuracy. Simulation results shows that M2 has better accuracy than M1 with more delay and high power consumption with delay, the proposed designs achieve reliable delay and power reductions with better accuracy and performance.

REFERENCES:

[1] *J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in Proc. 18th IEEE Eur. Test Symp., May 2013, pp.1-6.*

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

[3] *N. Zhu, W. L. Goh, and K. S. Yeo, "An enhanced low-power high-speed adder for error-tolerant application," in Proc. 12th Int. Symp. Integr. Circuits, Dec. 2009, pp. 69–72.*

[4] *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.*

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

[6] *A. B. Kahng and S. Kang, "Accuracy-configurable adder for approxi- mate arithmetic designs," in Proc. Design Automat. Conf., Jun. 2012, pp. 820–825.*

[7] *K. Du, P. Varman, and K. Mohanram, "High performance reliable variable latency carry select addition," in Proc. Design, Automat. Test Eur. Conf. Exhib., Mar. 2012, pp. 1257–1262.*

[8] *J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," IEEE Trans. Comput., vol. 62, no. 9, pp. 1760–1771, Jun. 2012.*

[9] *J. Huang, J. Lach, and G. Robins, "A methodology for energy-quality tradeoff using imprecise hardware," in Proc. Design Automat. Conf., Jun. 2012, pp. 504–509.*

[10] *J. Miao, K. He, A. Gerstlauer, and M. Orshansky, "Modeling and synthe- sis of quality-energy optimal approximate adders," in Proc. IEEE/ACM Int. Conf. Comput.-Aided Design, Nov. 2012, pp. 728–735.*

[11] *R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, "MACACO: Modeling and analysis of circuits for approximate computing," in Proc. IEEE/ACM Int. Conf. Comput.-Aided Design, Nov. 2011, pp. 667–673.*

[12] *H. Jiang, C. Liu, L. Liu, F. Lombardi, and J. Han, "A review, classifi- cation, and comparative evaluation of approximate arithmetic circuits," ACM J. Emerg. Technol. Comput. Syst., vol. 13, no. 4, 2017, Art. no. 60.*

[13] *P. Kulkarni, P. Gupta, and M. D. Ercegovac, "Trading accuracy for power in a multiplier architecture," J. Low Power Electron., vol. 7, no. 4, pp. 490–501, 2011.*

[14] *K. Bhardwaj, P. S. Mane, and J. Henkel, "Power- and area-efficient approximate wallace tree multiplier for error-resilient systems," in Proc. 15th Int. Symp. Qual. Electron. Design, Mar. 2014, pp. 263–269.*

[15] *K. Y. Kyaw, W. L. Goh, and K. S. Yeo, "Low-power high-speed mul- tiplier for error-tolerant application," in Proc. IEEE Int. Conf. Electron Devices Solid-State Circuits, Dec. 2010, pp. 1–4.*

[16] *S. Narayanamoorthy, H. A. Moghaddam, Z. Liu, T. Park, and N. S. Kim, "Energy-efficient approximate*

multiplication for digital signal process- ing and classification applications," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 23, no. 6, pp. 1180–1184, Jun. 2015.

[17] *Y.-H. Chen and T.-Y. Chang, "A high-accuracy adaptive conditional- probability estimator for fixed-width booth multipliers," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 59, no. 3, pp. 594– 603, Mar. 2012.*

[18] *B. Shao and P. Li, "Array-based approximate arithmetic computing: A general model and applications to multiplier and squarer design," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 62, no. 4, pp. 1081–1090, Apr. 2015.*

[19] *H. Jiang, J. Han, F. Qiao, and F. Lombardi, "Approximate radix-8 booth multipliers for low-power and high-performance operation," IEEE Trans. Comput., vol. 65, no. 8, pp. 2638–2644, Aug. 2016.*

[20] *K. Nepal, Y. Li, R. I. Bahar, and S. Reda, "ABACUS: A technique for automated behavioral synthesis of approximate computing circuits," in Proc. Design, Automat. Test Eur. Conf. Exhib., Mar. 2014, pp. 1–6.*

[21] *A. Ranjan, A. Raha, S. Venkataramani, K. Roy, and A. Raghunathan, "ASLAN: Synthesis of approximate sequential circuits," in Proc. Design, Automat. Test Eur. Conf. Exhib., Mar. 2014, pp. 1–6.*

[22] *C. Liu, J. Han, and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," in Proc. Design, Automat. Test Eur. Conf. Exhib., Mar. 2014, pp. 1–4.*

[23] *B. Parhami, Computer Arithmetic. London, U.K.: Oxford Univ. Press, 2000.*

[24] *M. A. Breuer, "Intelligible test techniques to support error-tolerance," in Proc. 13th Asian Test Symp., Nov. 2004, pp. 386–393.*

[25] *N. Weste and H. David, CMOS VLSI Design: A Circuits and Systems Perspective, 3rd ed. London, U.K.: Pearson, 2005*

[26] *N. Weste and H. David, CMOS VLSI Design: A Circuits and Systems Perspective, 3rd ed. London, U.K.: Pearson, 2005.*