IOSR Journal of Business and Management (IOSR-JBM)
e-ISSN: 2278-487X, p-ISSN: 2319-7668
PP 43-50
www.iosrjournals.org

A Survey on A High Performance Approximate Adder And Two High Performance Approximate Multipliers

Sumanta Chakraborty1,Siddhartha Chatterjee2
[1](*Computer Science & Engineering, University of Calcutta, India)*
[2](*Department of Computer Application, DSMS Group of Institutions, India)*

Abstract: Approximate computing can be performed where exact computing is not required and the applications are resilient to errors (applications will not crush due to approximation). Human perception level is very limited while interpreting an image, an audio or a video. This allows some applications, especially digital signal processing (DSP) applications to produce approximate output instead of exact output. The reason behind incorporating approximation in the applications to reduce circuit complexities, which leads to the reduction of power consumption, delay, etc. without degrading the performances. In this paper we review one novel approximate adder and two low-power approximate multipliers applicable to high-performance DSP applications. One multiplier for small input pro duces réductions in delay and power upto 20% and 69%, respectively, when implemented on a 28nm CMOS process. Another multiplier produces reductions in delay and power upto 9.8% and 10.74%, respectively, with an error rate from 0.2% to 13.76%.
Keywords : accuracy, adder, approximate, high performance, multiplier.

I. Introduction

Adders, multipliers are extensively studied in the field of approximate computing. Several methodologies for designing and modeling approximate adders have been developed by many researchers. At the same time number of research works on multipliers is still less. A multiplier usually consists of three stages: partial product generation, partial product accumulation and a carry propagation adder (CPA) at the final stage. Lu et al. [1] consider using approximate adders to generate the radix-8 Booth encoding 3x with error reduction. According to Kulkarni et al. [2], approximate partial products are computed using inaccurate 2 × 2 multiplier blocks. Then approximate speculative adders can be used at the final stage addition in a multiplier [3]. Multipliers are widely used in digital signal processing applications. In this paper we will review one approximate multiplier that utilizes a newly-designed approximate adder that limits its carry propagation to the nearest neighbors for fast partial product accumulation.

This paper is organized as follows. In section II we present a detailed review on Liu et al.'s approximate adder its architecture, performances in terms of reduction in delay and power consumption and reduction in error. In the first two subsections of section III we first present very brief introduction of Wallace multiplier, Kyaw et al.'s inaccurate multiplier; then in the last two subsections we present detailed reviews on Lin et al.'s inaccurate 4-bit Wallace multiplier and Liu et al.'s approximate multiplier their architectures and performances.

II. Liu Et Al.'S Approximate Adder

2.1. Architecture

In this subsection we review a new approximate adder proposed by Liu et al., [4] which operates on a set of preprocessed inputs. The input pre-processing (IPP) is based on the interchangeability 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 of i) and ii) produce the same result. In this process, the two input bits A_iB_i = 01 are equivalent to A_iB_i = 10 (with i being the bit index). They have used a rule for the IPP is to switch A_i and B_i if A_i= 0 and B_i = 1 (for any i), while keeping the other combinations (i.e., A_iB_i = 00,10 and 11) unchanged. If $A_{ip,}$ B_{ip} are the pre-processed inputs, the IPP functions are given by (1) and (2):

$$A_{ip} = A_i + B_i \quad (1)$$

$$B_i = A_i B_i \quad (2)$$

(1) and (2) compute the propagate and generate signals used in a parallel adder like the carry look-ahead (CLA). The logical functions of Table I is given by:

$$S_i = B_{(i-1)p} + B_{ip}{}^{C}A_{ip} \quad (3)$$

$$E_i = B_{ip}{}^{C} B_{(i-1)p} A_{ip} \quad (4)$$

Hence $B_{ip}{}^{C}$ is the complement of B_{ip}. Now substituting A_{ip} and B_{ip} in (3) and (4) from (1) and (2), we get

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

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

Say, a 6-bit adder with two inputs given by A = 001111 and B = 000110. The correct (exact) sum S is 010101; however, the approximate adder produces the sum $S' = 001101$ and an error E = 001000. So, it can be said that:

$S = S' + E$ (7)

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.

Table I. Truth Table of the Approximate Adder Cell

$B_{ip}B_{(i-1)p}$	00	01	10	11
A_{ip}	A_{ip}	A_{ip}	1	1
$C_{i-1}/B_{(i-1)p}$	0	1	0	1
S_i	A_{ip}	1	0	1
Ei	0	A_{ip}	0	0

2.2 Performances

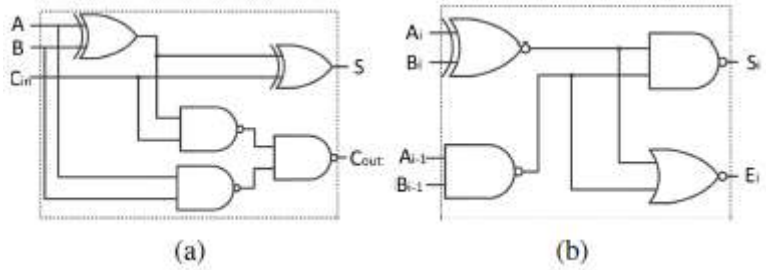

Fig.1. (a) An exact full adder and (b) the approximate adder cell

According to Liu et al. and based on the linear model as described in [10], the delays of a full adder (as shown in Fig. 1(a)) and the approximate adder cell adder (as shown in Fig. 1(b)) are derived to be approximately $3\tau_g$ and $2\tau_g$, respectively, where 'τ_g' is an approximate gate delay.

III. Approximate Multipliers

3.1. Wallace Multiplier

The Wallace multiplier [5] is based on the Wallace tree which is an efficient multiplication algorithm. The major advantage of Wallace is that stage reduction becomes possible by using half-adders and full-adders. In Wallace multiplier, the speeds achievable appear to be greater by a factor of at least four than those obtained in conventional units. Multiplication and division times would be reduced to approximate parity with the time required for, e.g., floating point addition. Fig. 2 shows a 4×4 Wallace multiplier dot-notations. (as in [7]).

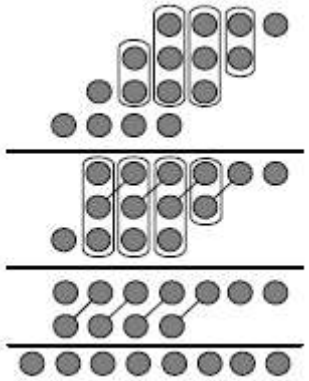

Fig.2. A 4×4 Wallace multiplier dot-notation

3.2. Kyaw et al.'s Inaccurate Multiplier

Kyaw et al. [6] redesigned the multiplier into two different parts – an accurate part (multiplication part) and inaccurate part (non-multiplication part). First, the input operands are split into two parts: a multiplication part that includes a number of higher order bits and a non-multiplication part that is made up of the remaining

lower order bits. However, the length of each part may not be equal. In their multiplier the multiplication process begins at the point where the bits split and move simultaneously towards the two opposite directions till all bits are taken care of. For the higher order bits of the input operands that fall into the multiplication part, the operation is conducted as per in normal multiplication operation, from right to left (LSB to MSB). They showed that by eliminating the partial products and the carry propagation path in the non-multiplication part (LSBs) and performing the multiplication of the MSBs simultaneously, the overall delay time is greatly reduced and so is the power consumption. These multipliers are widely used in application specific data paths in multimedia and wireless communication applications where some degree of saturation error within the dynamic range of interest is tolerable.

3.3. Lin et al.'s Inaccurate 4-bit Wallace Multiplier

3.3.1. Architecture

Lin et al. [7] used a 2:1 MUX to replace a XOR gate in 4:2 counter and that led to shorter delay. The layers of Wallace multiplier have been reduced by an inaccurate 4:2 counter, and so the delay and the power consumption of Wallace multiplier have also been reduced. In Fig. 3 X1 to X4 are the inputs. Sum and Carry are the outputs. Error occurs when all four summands are '1' and the output 111_2 reduces to 10_2. In Fig. 4 an inaccurate 4×4 Wallace multiplier is built by using this inaccurate 4:2 counter. Hence in the design proposed by Lin et al. an ordinary Wallace multiplier reduced the adding stages from three stages to two stages. But their inaccurate 4×4 Wallace multiplier reduced the adding stages from four stages to two stages by using an inaccurate 4:2 counter. They used an inaccurate 4:2 counter to give the sum of a partial product. The probability of partial product to be '1' is 1/4. So, the error of the inaccurate 4:2 counter occurs with a probability of $(1/4)^4$ = 1/256 which is significantly low.

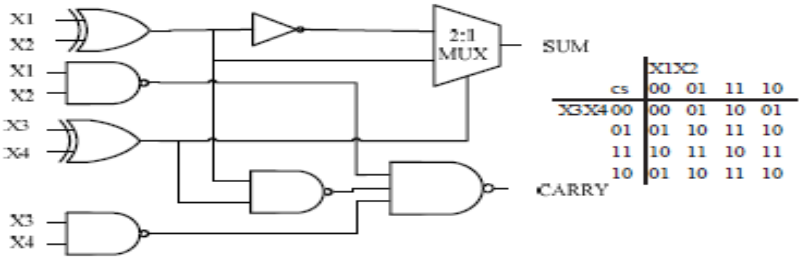

Fig.3. The architecture of Lin et al.'s 4:2 counter

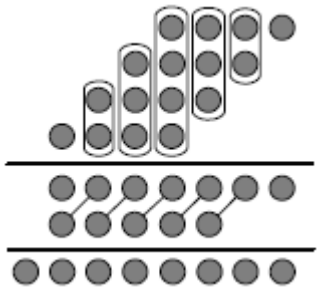

Fig.4. A 4×4 Wallace multiplier dot-notation with 4:2 counter

Larger multipliers can be built by using inaccurate 4×4 Wallace multipliers. To build a 32×32 multiplier, the 32×32 multiplication is decomposed into three additions of 16×16 multiplication results. Each 16×16 multiplication is decomposed to three additions of 8×8 multiplication results. Finally, each 8×8 multiplication is decomposed to three additions of 4×4 multiplication results. To further reduce the delay of the multiplier, they separate the adder of the final stage into two sub sum generators (shown in Fig. 5). The first sum generator is a normal adder, and the second sum generator uses a carry predictor to reduce the error rate. The signal arrival time in the oval lags behind that on the left side. So, the carry predictor only considers the signal value on the left side of the gray circle to reduce the multiplication delay. In the carry predictor, error occurs when S2 ~ S5 + C1 ~ C4 produces a carry bit and S6 ~ S8 + C5 ~ C7 produces 1. They formulate the probability of having erroneous result as follows:

Error rate = $(1/2^{cl}) \times ((2^k - 1)/2^{k+1})$ (8)

Hence in (8) 'cl' denotes the bit-width of the carry predictor 'k' is the bit-width of the first sum generator minus 'cl'. They use this architecture in the final summation to prevent the pass rate from dropping too fast.

3.3.2. Error Detection and Error Correction

Lin et al. enhance the error detection and error correction in their proposed multiplier. For a 4×4 inaccurate Wallace multiplier, error occurs when all the multiplier bits and multiplicand bits are 1. A 4×4 accurate multiplier gives the product 11100001_2 but a 4×4 inaccurate Wallace multiplier gives the product 11010001_2. Hence the differences are the values of the fifth bit and the sixth bit. This error is corrected if the fifth bit is forced to be 0 and the sixth bit is forced to be 1. They implement error detection with an AND gate and error correction with an OR gate and a NOR gate, as shown in Fig. 6 (as in [7]). Their 4×4 inaccurate Wallace multiplier can generate accurate result with error detection and correction (EDC) circuits. Fig. 7 shows the architecture of a 4×4 inaccurate Wallace multiplier with EDC (as in [7]). Their proposed multiplier can generate results according to the accuracy demanded by the applications. When an application needs low accuracy, their multiplier reduces the power consumption by switching to an approximation mode.

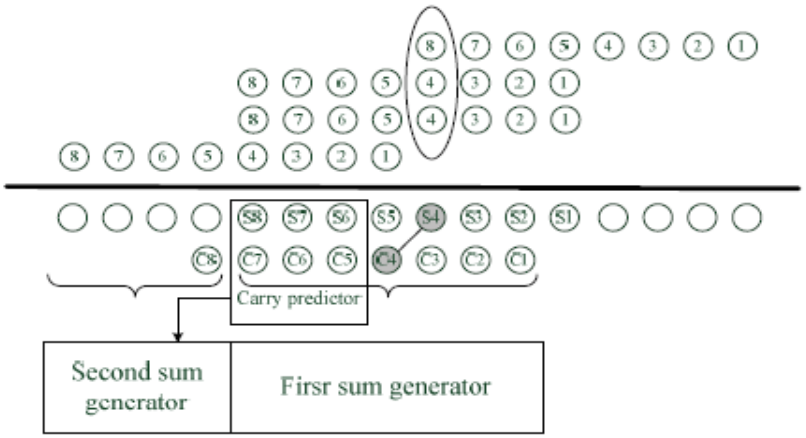

Fig.5. The summation architecture of building 8-bit multiplier

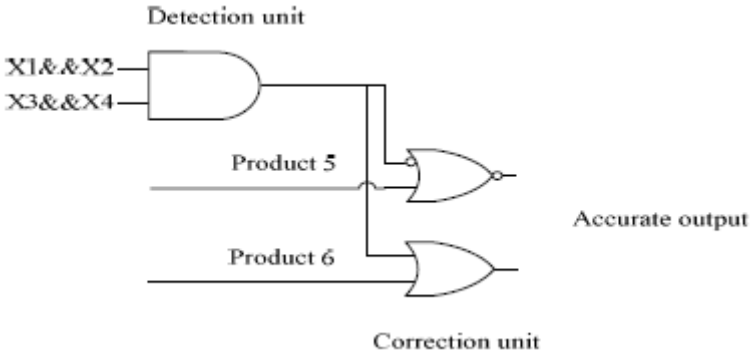

Fig.6. An error detection and correction (EDC) unit

3.3.3. Performances

Lin et al. implement the circuits in Verilog and synthesize them to gate-level netlists using the Synopsis Design Compiler with a standard TSMC 0.18m CMOS cell-library. Then they use the Synopsis Design Compiler to the delay, area and power consumption the circuits. Table II presents a comparison (as in [7]) of the Lin et al.'s 4:2 counter [7] with the 4:2 counters proposed by [8] and [9]. In Table II, row 2 shows the delay, row 3 shows the area of the proposed 4:2 counter. Their proposed 4:2 counter has the minimum delay and minimum area out of the three 4:2 counters. Row 4 and row 5 show the delay and the power of the 4×4 inaccurate Wallace multiplier (IWM) built out of these 4:2 counters. Table II shows that the 4×4 inaccurate Wallace multiplier built on the proposed 4:2 counter has shorter delay and lower power consumption.

In Fig. 8 we present the graphical diagram as shown by Lin et al. [7] for comparing delay of Lin et al.'s 4×4 inaccurate Wallace multiplier (IWM), Wallace multiplier and Kulkarni multiplier in different bit-widths. In Fig. 9 we present the graphical diagram as shown by Lin et al. [7] for comparing power consumption of Lin et al.'s 4×4 inaccurate Wallace multiplier, Wallace multiplier and Kulkarni multiplier in different bit-widths. In Fig. 10 we present the graphical diagram as shown by Lin et al. [7] for comparing power consumption of Lin et al.'s 4×4 inaccurate Wallace multiplier and Lin et al.'s 4×4 inaccurate Wallace multiplier with EDC in different bit-widths.

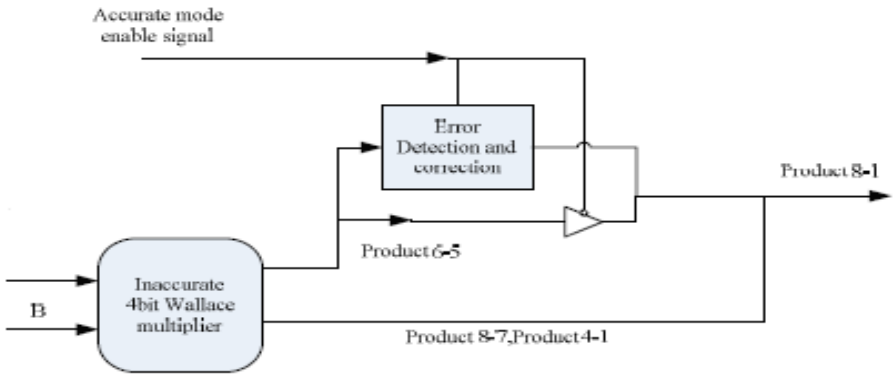

Fig.7. A 4×4 inaccurate Wallace multiplier with EDC

Table II. A Comparison of 4:2 Counters

	4:2 counter [8]	4:2 counter [9]	Lin et al.'s 4:2 counter
Delay (ns)	0.86	0.57	0.53
Area	136.38	143.03	129.73
Delay of 4×4 IWM (ns)	2.24	2.05	1.99
Power of 4×4 IWM (μW)	232.54	236.48	230.91

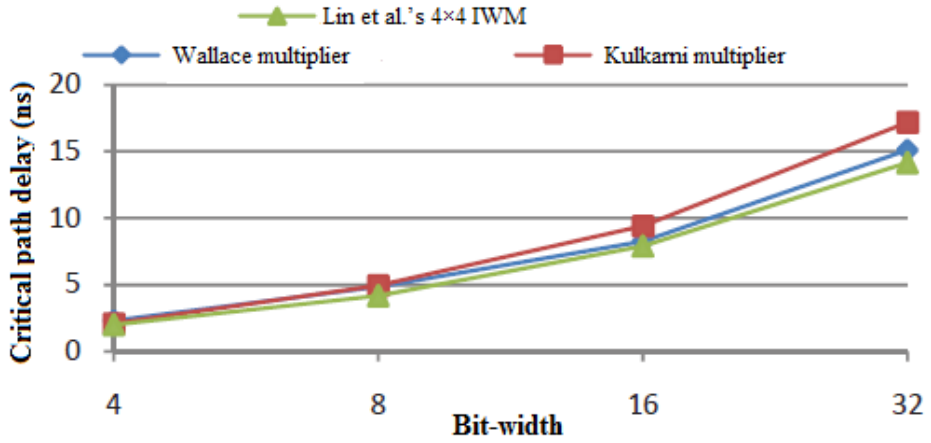

Fig.8. The delay comparison of Lin et al.'s 4×4 IWM, Wallace multiplier and Kulkarni multiplier

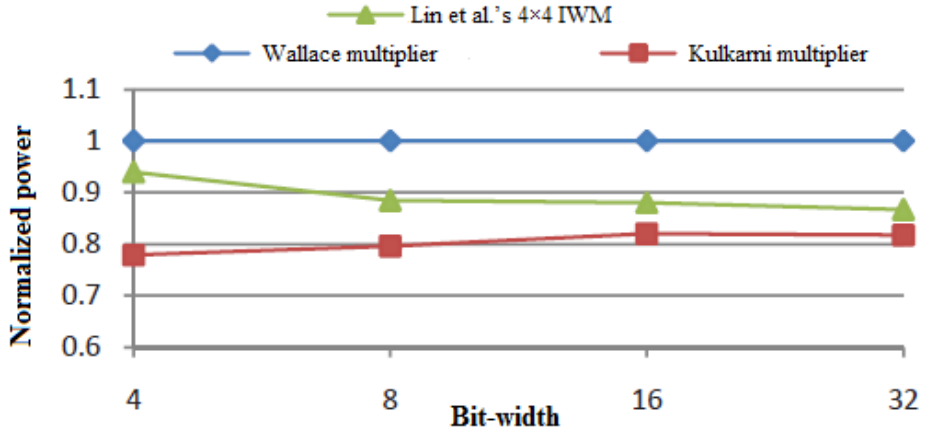

Fig.9. The power comparison of Lin et al.'s 4×4 IWM, Wallace multiplier and Kulkarni multiplier

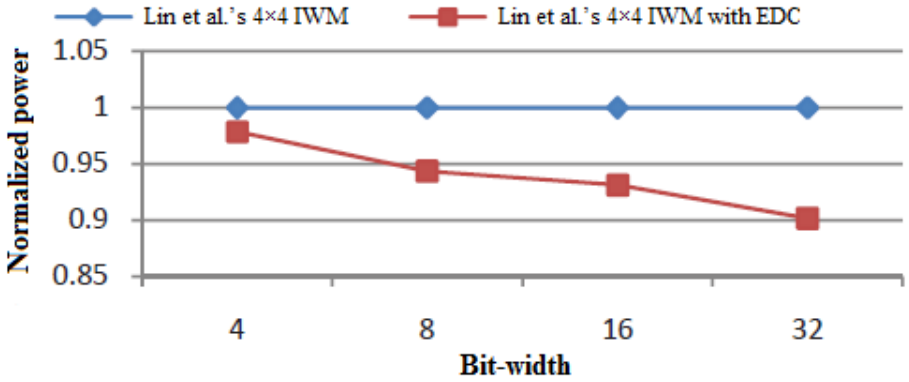

Fig.10. The power comparison of Lin et al.'s 4×4 IWM and Lin et al.'s 4×4 IWM with EDC

Table III shows (as in [7]) the area overhead and power overhead of Lin et al.'s 4×4 IWM with EDC.

Table III. The Overhead with EDC

Bit-width	Area overhead (%)	Power overhead (%)
4	3.81	4.24
8	7.13	6.64
16	5.40	5.77
32	3.86	4.04

3.4. Liu et al.'s Approximate Multiplier

3.4.1. Architecture

Liu et al. [4] proposed an approximate multiplier in which 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 obtain a product with a better accuracy. A significant feature of their proposed approximate multiplier is the simplicity to use approximate adders in the partial product accumulation. Liu et al.'s approximate multiplier utilizes the error signal. 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.

They apply (7) to the sum of every single approximate adder in the tree and, therefore, an error reduction circuit is applied to the final multiplication result rather than to the output of each adder. Two steps are required to reduce errors: i) error accumulation and ii) error recovery by the addition of the accumulated errors to the adder tree output using an accurate adder shown in Fig. 11 (as in [4]).

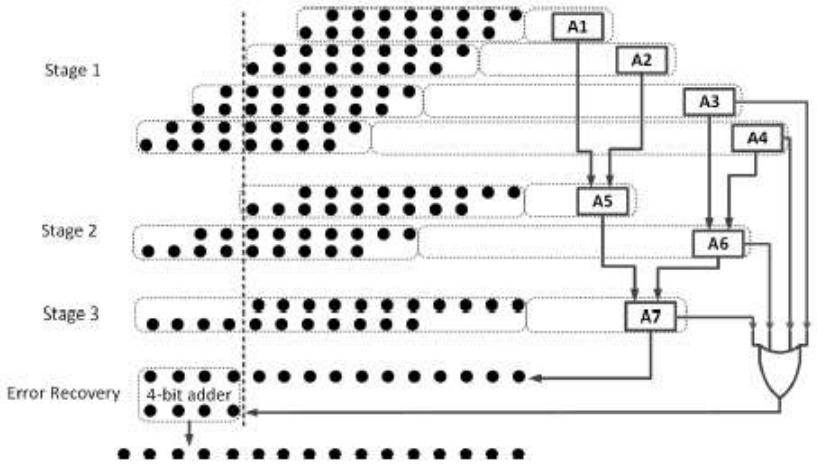

Fig.11. An approximate multiplier with OR-gate based partial error recovery using 4 MSBs of the error vector.

In error accumulation, Liu et al. consider that the error signals can be summed up using accurate adders and thus, the accumulated error can fully compensate the inaccurate product; however to reduce complexity, an approximate error accumulation is introduced. Liu et al. observe that the error vector of each approximate adder tends to have more 0's than 1's. Therefore, the probability that the error vectors have an error bit '1' at the same

position, is quite small. Hence, an OR gate is used to approximately compute the sum of the errors for a single bit. If 'm' error vectors E1, E2, …, Em have to be accumulated , then the sum of these vectors is obtained as:

$E_i = E1_i$ OR $E2_i$ OR … OR Em_i (9)

In error recovery, to reduce error Liu et al. add an accumulated error vector to the adder tree output using a conventional adder (e.g. a carry look-ahead adder). However, only several (e.g. k) MSBs of the error signals are used to compensate the outputs for further reducing the overall complexity. Liu et al. select the number of MSBs according to the extent that errors must be compensated. For example in an 8_8 adder tree, there are a total of 7 error vectors, generated by the 7 approximate adders in the tree. However, not all the bits in the 7 vectors need to be added, because the MSBs of some vectors are less significant than the least significant bits of the 'k' MSBs. In Fig. 10, 4 MSBs (i.e. the 11-14th bits) are considered for error recovery and as a result, 4 error vectors are considered (i.e. the error vectors of adders A3, A4, A6 and A7). Hence the error vectors of the other three adders are less significant than the 11th bit, so they are not considered. The accumulated error E is obtained using (8) and then, the final result is found by adding E to S using a fast accurate adder.

3.4.2. Performances

Since the approximate adder cell is simpler than a full adder, the approximate multiplier has no additional area overhead to achieve the shorter delay. For the 2×2 approximate multiplier in [2] only the partial product generation layer is simplified and the height of the partial product tree is only decreased by 1, so the delay reduction is quite limited. Liu et al.'s approximate multiplier can reduce the delay of the partial product accumulation tree by nearly 60%, which scales with the size of the multiplier. Liu et al. implement 16×16 approximate and Wallace multipliers in VHDL using the Xilinx Spantan3E XC3S500E FPGA. The critical path delays of Liu et al.'s approximate multiplier and the exact Wallace multiplier are 13.990ns and 21.999ns, respectively, thus achieving a reduction of 36.4%. The input data for simulating power consumption are given by the multiplication of two images. The node activity rates are extracted by performing post-place and route simulation running at the maximum frequency of the Wallace multiplier. Based on the activity rates, the Xilinx XPower Analyzer is used to obtain the power consumption. The quiescent power of Liu et al.'s approximate multiplier is slightly smaller than the Wallace multiplier. However, the approximate multiplier saves 44.3% of the dynamic power compared to the Wallace multiplier. Overall, Liu et al.'s approximate multiplier achieves a reduction of 26.8% in total power consumption.

IV. Conclusion

In this paper we review one approximate adder and two approximate multipliers. We have shown these are comparatively improved than the popular multipliers in terms of reduction in power overhead, area overhead, delay and error. But still Liu et al.'s multiplier and Lin et al.'s multiplier have significant amount of errors, especially for large inputs. The approximate adder and multipliers that we review in this paper, can be used in several image and video processing applications.

Acknowledgements

We would like to thank the organizing committee of DSMS Group of Institutions, Durgapur, West Bengal for giving us a very good opportunity and encouragement to present this paper.

References

Journal Papers:

[1]. [1] S.-L. Lu, "Speeding up processing with approximation circuits," *Computer, vol. 37, no. 3,* pp. 67–73, 2004.

[2]. [2] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in *24th IEEE Intl. Conf.* on *VLSI Design,* 2011, pp. 346–351.

[3]. [3] J. Huang, J. Lach, and G. Robins, "A methodology for energy-quality tradeoff using imprecise hardware," in *DAC 2012*, pp. 504–509.

[4]. [4] C. Liu, J. Han and F. Lombardi, "A Low-Power, High-Performance Approximate Multiplier with Configurable Partial Error Recovery", in *IEEE Design, Automation and Test in Europe Conference and Exhibition (DATE) , 2014.*

[5]. [5] C. S. Wallace, "A Suggestion for a Fast Multiplier", in *IEEE Transaction on Electronic Computers*, pp. 14-17, 1964.

[6]. [6] K. Y. Kyaw, W. L. Goh and K.S. Yeo, "Low-Power High-Speed Multiplier For Error-Tolerant Application", in *Electron Devices and Solid-States Circuits (EDSSC),* pp. 1-4, 2010.

[7]. [7] C. H. Lin, I.C. Lin, "High Accuracy Approximate Multiplier with Error Correction", in *IEEE, International Conference on Computer Design,* 2013.

[8]. [8] B. J. Phillips, D. R. Kelly and B. W. Ng, "Estimating adders for a low density parity check decoder", *F. T. Luk, Ed., vol. 6313, no. 1. SPIE,* 2006.

[9]. [9] J. Ma, K. Man, T. Krilavicius, S. Guan and T. Jeong, "Implementation of High Performance Multipliers Based on Approximate Compressor Design", in *International Conference on Electrical and Control Technologies (ECT),* 2011.

Book:

[10]. [10] N. H. Weste and H. David, "*CMOS VLSI Design - A Circuits and Systems Perspective*" (Boston, Massachusetts, *3rd ed. Pearson, Addison- Wesley*, 2005).