International Journal on Applications in Information and Communication Engineering
Volume 9 : Issue 4 : September 2023, pp7 – 14 www.aetsjournal.com ISSN (Online) : 2394 - 6237

DESIGN AND ANALYSIS OF APPROXIMATE COMPRESSORS FOR BALANCED ERROR ACCUMULATION IN MAC OPERATOR

V.GOPALAKRISHNAN , S.DIVYA

Abstract—
In this paper, we present a novel approximate computing scheme suitable for realizing the energy-efficient multiply-accumulate (MAC) processing. .First we design the approximate 4-2 compressors generating errors in the opposite direction while minimizing the computational costs. Based on the probabilistic analysis, positive and negative multipliers are then carefully developed to provide a similar error distance. Simulation results on various practical applications reveal that the proposed MAC processing offers the energy-efficient computing scenario by extending the range of approximate parts. This Design is implemented by Verilog HDL and simulated by Modelsim 6.4 c. The Performance is measured by Xilinx tool Synthesis Process.

IndexTerms—
Approximatecomputing,digitalmultiplier,low-power,energy-efficient,4-2compressor.

I.INTRODUCTION

In applications like multimedia signal processing and data mining which can tolerate error, exact computing units are not always necessary. They can be replaced with their approximate counterparts.

Research on approximate computing for error tolerant applications is on the rise. Adders and multipliers form the key components in these applications. In approximate full adders are proposedat transistor level and they are utilized in digital signal processingapplications. Their proposed full adders are used in accumulation ofpartial products in multipliers.To reduce hardware complexity of multipliers, truncation is widely employed in fixed- width multiplier designs. Then a constant or variable correction term is added to compensate for the quantization error introduced by the truncated part. Approximation techniques in multipliers focus on accumulation of partial products, which is crucial in terms of power consumption. Broken array multiplier is implemented, where the least significant bits of inputs are truncated, while forming partial products to reduce hardware complexity. The proposed multiplier saves few adder circuitsin partial product accumulation. Two designs of approximate 4-2 compressors are presented and used in partial product reduction tree of four variants of 8 × 8Array multiplier. The major drawback of the proposed compressorsis that they give nonzero output for zero valued inputs,which largely affects the mean relative error (MRE) as discussedlater. The approximate design proposed in this brief overcomes the existing drawback. This leads to better precision. In static segment multiplier (SSM) proposed, m-bit segments are derived fromn-bit operands based on leading 1 bit of the operands. Then, m× m multiplication is performed instead of n × n multiplication,where m<n. Partial product perforation (PPP) multiplier omitsk successive partial products starting from jth position, where j ∈ [0,n-1] and k ∈ [1, min(n-j, n-1)] of a n-bit multiplier. In [8], 2 × 2approximate multiplier based on modifying an entry in the Karnaughmap is proposed and used as a building block to construct 4 × 4 and8 × 8 multipliers. In [9], inaccurate counter design has been proposedfor use in power efficient Wallace tree multiplier. A new approximate adder is presented which is utilized for partial product accumulation of the multiplier. For

V.Gopalakrishnan , Assistant Professor, Department of Electronics and Communication Engineering , Sasurie college of Engineering ,Vijayamangalam , Tamil Nadu , India.

S.Divya, PG Scholar , Department of Electronics and Communication Engineering , Sasurie college of Engineering ,Vijayamangalam , Tamil Nadu , India.

16-bit approximate multiplier26% of reduction in power is accomplished compared to exact multiplier. Approximation of 8-bit Wallace tree multiplier due to voltage over-scaling (VOS) is discussed. Lowering supplyvoltage creates paths failing to meet delay constraints leading to error.Previous works on logic complexity reduction focus on straightforwardapplication of approximate adders and compressors to the partial products. In this brief, the partial products are altered tointroduce terms with different probabilities. Probability statistics of the altered partial products are analyzed, which is followed bysystematic approximation. Simplified arithmetic units (half-adder,full-adder, and 4-2 compressor) are proposed for approximation.The arithmetic units are not only reduced in complexity, but careis also taken that error value is maintained low. While systemic approximation helps in achieving better accuracy, reduced logic complexity of approximate arithmetic units consumes less power andarea. The proposed multipliers out performs the existing multiplierdesigns in terms of area, power, and error, and achieves better peaksignal to noise ratio (PSNR) values in image processing application.Multipliers form an important hardware block in the DSP and Embedded applications. Multiplication speed determines processor speed. So high speed multipliers are needed in the processors for many applications. For increase the speed of multiplication different algorithms are used Multiplication is a most commonly used operation in many computing systems. A number (multiplicand) is added to itself a number of times as specified by another number (multiplier) to form a result (product). But the implementation of multiplier takes huge hardware resources and the circuit operates at low speed. Multiplication is one of the fundamental components in DSP and Embedded system.

II. RELATED WORK

We present four dual-quality reconfigurable approximate 4:2 compressors, which provide the ability of switching between the exact and approximate operating modes during the runtime. We developed two types of approximate multipliers having opposite error directions to each other. This compressor designs, more precisely, we only consider the error direction and each approximate multiplier is constructed to minimize its hardware costs without considering the amount of errors at all. The amount of absolute errors from each approximate multiplier is estimated by performing the probabilistic analysis. We properly allocate the ratio between the positive and negative approximate multipliers during the MAC operation.The opposite individual errors offset each other by accumulation process.The resulted error distribution is then narrowed and balanced, accepting more inaccurate computations for each multiplication compared to the prior works utilizing the identical approximate multipliers for realizing MAC operations. Multiplication is a fundamental operation in most signal processing algorithms. Multipliers have large area, long latency and consume considerable power. Therefore low power multiplier design has an important part in low-power VLSI system design. A system is generally determined by the performance of the multiplier because the multiplier is generally the slowest element and more area consuming in the system. Hence optimizing the speed and area of the multiplier is one of the major design issues. However, area and speed are usually conflicting constraints so that improvements in speed results in larger areas. Multiplication is a mathematical operation that include process of adding an integer to itself a specified number of times. A number (multiplicand) is added itself a number of times as specified by another number (multiplier) to form a result(product). Multipliers play an important role in today‟s digital signal processing and various other applications. Multiplier design should offer high speed, low power consumption. Multiplication involves mainly 3 steps

1) Partial product generation
2) Partial product reduction
3) Final addition

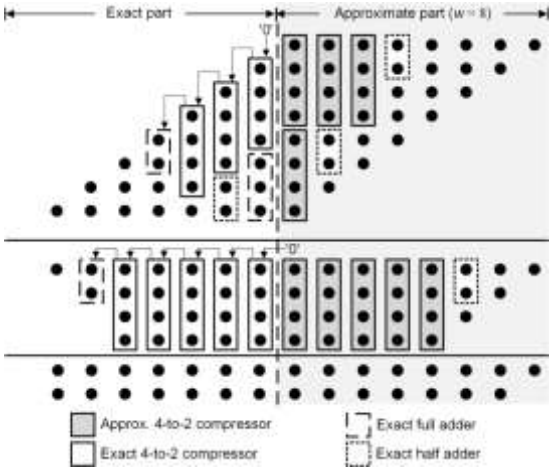

Figure 1 : Array Multiplier Using Proposed Design

Fast arithmetic computation cells including adders and multipliers are the most frequently and widely used circuits in VLSl systems. Microprocessors and digital signal processors rely on the efficient implementation of generic arithmetic logic units and floating point units to execute dedicated algorithms such as convolution and filtering.

In most of these applications, multipliers have been the critical and obligatory component dictating the overall circuit performance when constrained by power consumption and computation speed. . With trends of VLSl technologies towards deep-submicron regime, the most eminent means of achieving power efficacy is by lowering the power supply voltage. Therefore, it is imperative to explore circuit design techniques to achieve high power efficacy of arithmetic circuits at ultra low supply voltages. Fast multipliers are generally composed of three sub- functions: partial product generation, partial product accumulation and carry-propagating addition. In the partial product generation circuit, Booth encodings are often used to reduce the number of partial products. A summation tree, called the Carry Save Adder (CSA) tree, is used in the second function to further reduce the partial products to two. The last function is normally fulfilled by the fast carry propagate adder, such as carry look-ahead adder and carry-skip adder. Early designs of CSA tree use the (3:2) counters, or full adders for the partial product accumulation, in which 3 equally weighted bits were combined to produce two output bits. The (4~2) compressors, due to their ability to form regular interconnected cells structure, are more popularly used nowadays. Higher input compressors such as (5:2), (6:2), etc., have also been studied and increasingly employed in high precision multipliers to achieve greater performance.

The enhanced speed leads to increased power consumption, thus, power saving architectures turn to be the choice of the future. This has given way to the development of novel circuit techniques, with the aim of reducing the power dissipation of multipliers without compromising the speed and performance. A multiplier can be divided into three stages: Partial products generation stage, partial products addition stage, and the final addition stage. In the first stage, the multiplicand and the multiplier are multiplied bit by bit to generate the partial products. The second stage is the most important, as it is the most complicated and determines the speed of the overall multiplier. The 4-2 and 5-2 compressors have been widely employed in the high speed multipliers to lower the latency of the partial product accumulation stage. Owing to its regular interconnection, the 4-2 compressor is ideal for the construction of regularly structured Wallace tree with low complexity. In high-speed designs, the Wallace tree construction method is usually used to add the partial products in a tree-like fashion in order to produce two rows of partial products that can be added in the last stage. The Wallace tree is fast since the critical path delay is proportional to the logarithm of the number of bits in the multiplier. There exist a handful of ways to construct the Wallace Tree. The prominent method considers all the bits in each column at a time and compresses them into two bits (a sum and a carry). The Wallace tree is constructed by considering all the bits in each fours row at a time and compressing them in an appropriate manner. Thus, compressors form the essential requirement of high speed multipliers. The speed, area and power consumption of the multipliers will be in direct proportion to the efficiency of the compressors. Thus, in order to satisfy the requirement of small area low power high

throughput circuitries, this paper provides novel designs of 4:2 compressors with minimum number of transistors.

III. ARRAY MULTIPLIER

The Array multiplier is a hardware multiplier design invented by computer scientist Luigi Array in 1965. It is similar to the Wallace multiplier, but it is slightly faster (for all operand sizes) and requires fewer gates (for all but the smallest operand sizes).

In fact, Array and Wallace multipliers have the same 3 steps:

1. Multiply (logical AND) each bit of one of the arguments, by each bit of the other, yielding n^2 results. Depending on position of the multiplied bits, the wires carry different weights, for example wire of bit carrying result of $a_2 b_3$ is 32.
2. Reduce the number of partial products to two by layers of full and half adders.
3. Group the wires in two numbers, and add them with a conventional adder.

However, unlike Wallace multipliers that reduce as much as possible on each layer, Array Multiplier do as few reductions as possible. Because of this, Array Multiplierhave a less expensive reduction phase, but the numbers may be a few bits longer, thus requiring slightly bigger adders.

To achieve this, the structure of the second step is governed by slightly more complex rules than in the Wallace tree. As in the Wallace tree, a new layer is added if any weight is carried by three or more wires. The reduction rules for the Array tree, however, are as follows:

- Take any three wires with the same weights and input them into a full adder. The result will be an output wire of the same weight and an output wire with a higher weight for each three input wires.
- If there are two wires of the same weight left, and the current number of output wires with that weight is equal to 2 (modulo 3), input them into a half adder. Otherwise, pass them through to the next layer.
- If there is just one wire left, connect it to the next layer.

This step does only as many adds as necessary, so that the number of output weights stays close to a multiple of 3, which is the ideal number of weights when using full adders as 3:2 compressors.

IV. MODULES:

- Partial Product Generation Block
- Propagation Block
- Generation Block
- OR Gate
- Approximate Half Adder
- Approximate Full Adder
- Approximate 4-2 Compressor

V. MODULE EXPLANATIONS:

PARTIAL PRODUCT GENERATION BLOCK

A 8-bit unsigned1 multiplier is used for illustration to describe the proposed method in approximation of multipliers. Consider two 8-bit unsigned input operands The partial product am,n = αm · βn in Fig. 1 is the result of AND operation between the bits of αm and βn.

$$\alpha = \sum_{m=0}^{7} \alpha_m 2^m \text{ and } \beta = \sum_{n=0}^{7} \beta_n 2$$

The compressor designs have been mainly used for realizing the practical multipliers due to their simple and regular structures [17]–[26]. After generating the initial PPs with bit- wise AND operations between the multiplicand A and multiplier X, as reported in [17], the exact p-q compressor reduces the number of PPs from p to q with the assisted carry-in/out values.

For the case of 4-2 compressor, which is widely used in the practical multi-stage multipliers, four PPs along the same i-th bit position denoted as ai , bi , ci , and di , generate two PPs over two columns (yiand yi+1) for the next stage as follows.

yi = (ai ⊕ bi) ⊕ (ci ⊕ di) ⊕ zi, yi+1 = (ai ⊕ bi ⊕ ci ⊕ di)zi + (ai ⊕ bi ⊕ ci ⊕ di)di, zi+1 = (ai ⊕ bi)ci + (ai ⊕ bi)ai, (1) where zi and zi+1 represent the carry-in and carry-out signals of the exact 4-2 compressor at the i-th bit position, respectively [28]. Note that realizing an exact 4-2 compressor basically necessitates two full adders, dominating the overall complexity of multiplier [22]. Hence, for

the pre-defined approximate parts, the recent approximate multipliers typically simplify the compressor designs allowing the inaccurate operations. illustrates the 4-2 compressor-based 8-bit approximate multiplier where w denotes the number of approximate bits. Increasing w obviously results in more energy-efficient multiplier designs, however, the amount of errors also rapidly increases. Hence, recent studies have continuously tried to relax the computing costs of the approximate Fig. 1. The basic concept of 8 × 8 approximate multiplication. compressor while reducing the amount of errors as much as possible [17]–[22]. y˜i+1 = (ai + bi) + (ci + di), (2) where y˜i and y˜i+1 are the approximate results of the simplified compressor. Based on this carry-free concept, Table I summarizes equations for the previous approximate 4-2 compressors. In this work, for the sake of simplicity, we categorize the approximate multipliers into two types, i.e., the cost-aware types and the accuracy-aware ones. By approximating the lower n bits for n × n multiplication, we manually define the cost-aware designs that can save energy consumption by at least 20% and 30% for n = 8 and n = 16, respectively. With these simple criteria, the cost-aware type then includes the designs from [23]–[26], aggressively reducing the hardware complexity of the compressor by allowing a large amount of errors, accordingly saving the energy consumption of the multiplication process. For the same number of approximate bits, i.e., w, on the other hand, the accuracy- aware designs such as [19]–[22] slightly modify the original compressor equations, providing more accurate results than the cost- aware approaches as depicted in Table I. Note that both categories provide different ranges of energy-accuracy trade-offs, hence it is required to properly select the best option depending on the required algorithm-level performance. As the prior works only focus on inaccurate results of each multiplication, however, the generated errors are in general unbalanced, i.e., having a positive or negative direction compared to the exact computation. When we directly apply the prior works to simplify the MAC-oriented operations like image filtering or DNN processing, such unidirectional errors accumulate to high computation errors; severely degrading the algorithm-level performance. In other words, the practical signal processing workloads cannot sufficiently increase the number of approximate bits to get meaningful energy saving with the cost-aware approximation techniques. There are only few works to make the balanced error distributions from the approximate multiplier [19]–[22], however, those accuracy-aware architectures relatively require more logic units, resulting only the marginal improvements. Unlike the previous works, we carefully analyze the direction of errors from the approximate multipliers with simplified compressors. Then, we provide an efficient way to balance the generated errors with larger w values at the accumulation step, leading to more energy-efficient approximate MAC processing.

Proposed Approximate 4-2 Compressor In the proposed work, we first develop the cost-effective approximate 4-2 compressors by considering the error directions, not the amount of errors. Two types of approximate 4-2 compressors are designed to produce inaccurate results in the opposite direction; the positive compressor (PC) and the negative compressor (NC). Starting from the first carry-free compressor approximation from [20], which only produces an error for all one inputs, we modify the truth table to reduce hardware cost. As reported in [25], the hardware complexity and the power consumption are considerably decreased when generating a value of 1 at y˜i regardless of input cases, resulting in large relative errors, especially for zero-valued operands. To guarantee the accuracy for zero-valued inputs with the reduced hardware costs, minimizing the performance degradation in algorithm level at the practical applications [22], the proposed approximate compressors identically produce a value of 1 for y˜i except for all zero inputs as follows. y˜ P i = ˜yN i = ai + bi + ci + di, (3) where y˜P i and y˜N i denote the approximate values of yi in (1) for the proposed PC and NC designs, respectively. the proposed Fig. 2. The modified K-Maps of (a) the approximate signal y˜i , and (b) the approximate signal y˜i+1. approximation of y˜i generates positive errors as we only change the

value of 0 to 1 regardless of the compressor types. Oppositely, the approximation in y˜i+1 is realized with negative modification, inverting the value of 1 of yi+1 in (1) to 0. For the proposed NC architecture, more precisely, we need to invert a sufficient number of values to change the error direction of the compressor, which is originally towards the positive side with the aggressive approximation from the lower-bit position y˜N i in (3). Simultaneously, as detailed in Fig. 2(b), it is required to consider the hardware complexity for realizing the approximate compressor, leading to the approximate equation as follows. y˜N i+1 = aibi + cidi

On the other hand, the approximation of the upper-bit position for the positive compressor is developed by only considering the hardware costs. Hence, we have more freedom for changing the values of the original equation as described in Fig. 2(b), where the approximate equation can be formulated as y˜P i+1 = aibi + bi ci + bidi + cidi . (5) The inversion cases of specific min-terms shown in Fig. 2 may seem to offer two types of approximate compressors having opposite error

directions. However, it is hard to directly conclude that the proposed approaches result in the desired error directions by simply counting the number of inverted patterns due to the different probability for each input configuration [22]. Therefore, it is necessary to develop the probabilistic method to analyze not only the direction but also the magnitude of errors from the proposed works.

VI. PROPOSED APPROXIMATE MAC PROCESSING

1)Interleaving Approximate Multipliers

Interleaving Approximate Multipliers As the hardware complexity of multipliers, in general, dominates the overall computational costs of contemporary parallel MAC operators [20], it is acceptable to re- utilize the pre-designed approximate multipliers for realizing the cost-effective MAC operation without changing the other processing units including the low-cost accurate adder designs. To solve the biased error accumulation caused by the previous approximate multipliers, in this work, we develop a novel interleaving scheme of approximate multipliers with the opposite error directions, generating the balanced error distribution during MAC operations. For the given approximate range w, we first observe E(EP MUL) and E(EN MUL) to determine the blending ratio of two approximate multipliers, which is denoted as ρ = E(EN MUL)/E(EP MUL). During the MAC operations, we then perform ρ multiplications on average with NM architecture after one PM-based multiplication. As the recent MAC- oriented processing usually utilizes parallel processing elements (PEs), it is possible to directly apply the blending ratio to the existing hardware without increasing the Authorized licensed use limited to: Without applying the identical multiplier architecture, note that we can simply introduce the interleaved multiplication sequences associated with PM and NM, which efficiently offset the accumulated errors by the opposite error directions. the proposed interleaving scheme allows more approximate parts without degrading the algorithm-level performance, leading to the energy- efficient MAC processing.

2) Error Analysis of Approximate MAC Operations

To analyze benefits of the proposed designs, we simulated various approximate multipliers based on the convolution operation using fw × fh filter kernels, where the internal MAC operations can be formulated as C l=1 fw i=1 fh j=1 Klij × Ilij . Note that the number of accumulated multiplication results varies with the number of channels in the convolution operations, denoted as C. For the fair evaluation, in addition, we adopted the random inputs for each multiplication Fig. 4. The proposed error-balancing scheme for approximate MAC processing using parallel processing elements. step, which are represented as Klij and Ilij . Applying the approximate multiplication, the MAC result now includes the accumulated error as follows. EMAC = C l=1 fw i=1 fh j=1{AM(Klij, Ilij) − Klij × Ilij}, (7) where the error direction strongly depends on the structure of internal approximate compressors. Using 8×8 multiplications with w =

16, each approximate approach is tested over 106 times by setting C = 64, fw = 3, and fh = 3, where the corresponding box-plot diagram of EMAC is evaluated for denoting the error distribution as depicted in Fig. 5. Note that the accuracy- aware approximate compressors clearly lead to fewer errors, whereas the cost-aware approaches cause much more errors with larger distributions. Even though the energy saving from the cost-aware methods is relatively attractive due to the simplified compressor equations in Table I, the generated imbalance errors cannot be applicable to the practical applications. As the proposed PM and NM designs mainly utilize the proposed PC and NC, respectively, which are relatively close to the cost-aware architectures, we can observe similar unacceptable errors in opposite directions. Note that the NM architecture generates larger absolute errors than the PM design as we expected with the probabilistic analysis described in the Section III-C. If we apply the proposed interleaved computation with the blending ratio of ρ, the opposite errors compensate each other during the MAC operations, successfully generating a balanced and narrowed distribution as shown in Fig. 5. For different approximate MAC operations, we additionally perform the numerical analysis by calculating the normalized mean error distance (NMED) [34], which is defined as follows. NMED = μ(|EMAC|) MAX(C l=1 fw i=1 fh j=1 Klij × Ilij) , (8) where μ(·) and MAX(·) return the mean and maximum values, respectively. Fig. 6 shows the NMED results of signed 16×16 multiplication with w = 14 by changing the channel width C for the 3 ×3 kernels, which are widely used for the practical CNN processing [35], [36]. Regardless of the number of channels, it is interesting that we can observe a quite flat NMED trend for the previous approximate MAC processing. Since the errors are accumulated in one direction, both μ(|EMAC|) and MAX(C l=1 fw i=1 fh j=1 Klij × Ilij) increase simultaneously. the accuracy-aware multipliers, which are associated with small errors, tend to dissipate more power than the cost-aware designs due to their complex compressor architectures. Similar to the other studies [25], the power-delay-product (PDP) metric is used in this work to measure the effectiveness of each multiplier design. Compared to the exact 8 × 8 multiplier, for example, the proposed PM and NM can reduce the PDP values by 56.5% and 60.5%, respectively, by replacing all the exact 4-2 compressors with the approximate ones (w = 16). By mixing two proposed designs with the ratio of ρ, we can expect that the approximate MAC operations consumes 0.0837mW per each multiplication in average, saving the PDP metric by 57.8% compared to the exact version, which is even comparable to the cost- aware designs reducing a large amount of processing energy. If we consider the processing accuracy, on the other hand, the proposed interleaved scheme offers even narrower and balanced error distribution than the accuracy-aware options. In contrast that the prior works only focus on the intra-level approximation, in other words, the proposed approach allows additional inter-level approximation by considering the accumulation property, leading to the energy-efficient MAC processing while preserving the accuracy performance.

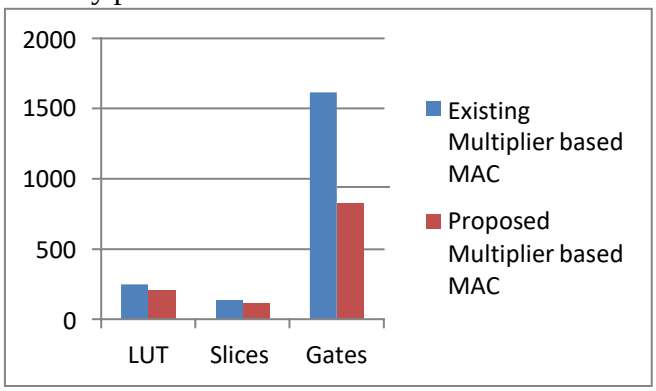

Figure 2 : Area Graph

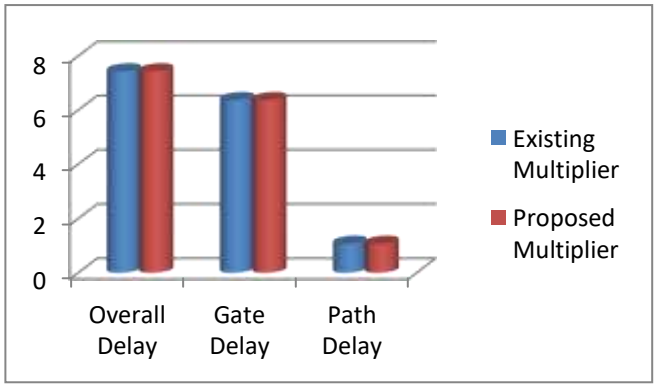

Figure 3 : Delay Graph

VII. CONCLUSION

All approximate multipliers are designed for n = 8. The multipliers are implemented in Verilog and synthesized using deals with the analysis and design of two new approximate 4-2 compressors for utilization in a multiplier. we will design a Efficient Multiplier and MAC Design based on our proposed Multiplier. The proposed approximate compressors are proposed and analyzed for a Array multiplier.

VIII.REFERENCES

1) J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and Probabilistic Adders," IEEE Trans. Computers, vol. 63, no. 9, pp. 1760–1771, Sep. 2013.
2) V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, "IMPACT: IMPrecise adders for low-power approximate computing," in Proc. Int. Symp. Low Power Electron. Design, Aug. 2011, pp. 409–414.
3) S. Cheemalavagu, P. Korkmaz, K. V. Palem, B. E. S. Akgul, and L. N. Chakrapani, "A probabilistic CMOS switch and its realization by exploiting noise," presented at the IFIP Int. Conf. Very Large Scale Integ., Perth, Australia, Oct. 2005.
4) 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. I: Reg. Papers, vol. 57, no. 4, pp. 850–862, Apr. 2010.
5) M. J. Schulte and E. E. Swartzlander Jr., "Truncated multiplication with correction constant," in Proc. Workshop VLSI Signal Process. VI,1993, pp. 388–396.
6) E. J. King and E. E. Swartzlander Jr., "Data dependent truncated scheme for parallel multiplication," in Proc. 31st Asilomar Conf. Signals, Circuits Syst., 1998, pp. 1178–1182.
7) 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.
8) C. Chang, J. Gu, and M. Zhang, "Ultra low-voltage low-power CMOS 4-2 and 5-2 compressors for fast arithmetic circuits," IEEE Trans. Circuits Syst., vol. 51, no. 10, pp. 1985–1997, Oct. 2004.
9) D. Radhakrishnan and A. P. Preethy, "Low-Power CMOS pass logic 4-2 compressor for high-speed multiplication," in Proc. IEEE 43rd Midwest Symp. Circuits Syst., 2000, vol. 3, pp. 1296–1298.
10) Z. Wang, G. A. Jullien, and W. C. Miller, "A new design technique for column compression multipliers," IEEE Trans. Comput., vol. 44, no. 8, pp. 962–970, Aug. 1995.
11) J. Gu and C. H. Chang, "Ultra low-voltage, low-power 4-2 compressor for high speed multiplications," in Proc. 36th IEEE Int. Symp. Circuits Syst., Bangkok, Thailand, May 2003, pp. v-321–v-324.
12) M. Margala and N. G. Durdle, "Low-power low-voltage 4-2 compressors for VLSI Applications," in Proc. IEEE Alessandro Volta Memorial Workshop Low-Power Design, 1999, pp. 84–90.
13) B. Parhami, Computer Arithmetic; Algorithms and Hardware Designs, 2nd ed. London, U.K.: Oxford Univ. Press, 2010.
14) K. Prasad and K. K. Parhi, "Low-power 4-2 and 5-2 compressors," in Proc. 35th Asilomar Conf. Signals, Syst. Comput., 2001, vol. 1, pp. 129–133.
15) M. D. Ercegovac and T. Lang, Digital Arithmetic. Amsterdam, The Netherlands: Elsevier, 2003.