VOL. 13, NO. 24, DECEMBER 2018 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

A CRITICAL SURVEY ON APPROXIMATE COMPUTING WITH CMOS AND MEMRISTORS

S. Muthulakshmi[1], A. Sivasubramanian[1] and S. R. S. Prabaharan[2]
[1]School of Electronics Engineering, Vellore Institute of Technology, Vandalur-Kelambakkam Road, Chennai, India
[2]SRM Institute of Science and Technology, SRM Nagar, Kattankulathur, India
E-Mail:srsprabaharan1611@gmail.com

ABSTRACT

In the nanoscale era, enhancing the performance of digital circuits and systems becomes increasingly cumbersome. As a result, computing becomes increasingly heavy with multimedia processing. Approximate computing is regarded as one of the emerging technologies that could produce less-than-optimal results which are sufficient enough in user's perspective. This is accomplished by minimizing energy consumption as well as hardware area. The slight compromise in output quality is acceptable for inherent error resilient applications. This paper reviews different types of existing CMOS based approximate circuits designed by various functional approximation techniques. Further, this survey focuses on analyzing the existing approximate arithmetic circuits based on various error metrics such as error rate, error distance, mean error distance, normalized error distance, and minimum acceptable accuracy. The impact of approximation on circuit characteristics for instance, delay, power consumption and throughput are also presented. This paper critically reviews the progress made in the context of approximate computing by using CMOS architecture and its impact on replacing with RRAMs (Memristor). The latter is reportedly found to be energy efficient due to its nano-scale dimension. Although memristor based approximation is still in its infancy, it is high time to understand its emerging importance in nanoscale computing in general and approximate computing in particular.

Keywords: approximate computing, approximate arithmetic circuits, nanoscale circuits, memristor, content addressable memory, functional approximation.

1. INTRODUCTION

In todays' computing world, energy efficient techniques are more preferred than performance oriented to handle the huge amount of data. Approximate computing is considered as one of the preferred energy efficient techniques to address the downscaling issues with prevailing CMOS technology [1,12,14] as well as to handle the big data produced by more number of interconnected battery powered devices [2,3,5,6,8,9]. Apart from energy efficiency, minimum hardware area is also expected since most of the handheld devices are operated in battery power. Approximate computing is aimed at producing good enough results yielding acceptable quality rather than being accurate. This means that it trades off between the output quality with performance and energy consumption [1-14, 17-31, 32, 34-39]. Hence, the approximate computing can be applied for applications like (a) inherent error resilient applications that do not produce accurate results rather than producing results that are slightly deviated from to have acceptable user experience [2-6,8,17, 20], (b) when processing huge amount of data produced by 35-70 billions interconnected devices through low power, low cost circuits, they are not expected to produce high precision results [2, 5, 6, 9, 10], (c) the low power real time sensor data which are inaccurate by nature [3, 5, 6, 8].

Extensive research is being carried out to explore the approximate computing at different abstraction level as well as in different granularity [4-8].A fine granular approximation is carried out with individual instruction, memory location, and single data packet whereas coarse granular approximation is applied in block level [4-8]. Apart from this, approximation can be introduced either in software level by skipping some computational modules in algorithms, reducing a bit width, data type conversions like floating point to a fixed point and/or in hardware level either through over scaling technique or by functional approximation technique. In the over scaling based approximation; circuits can produce accurate results under normal operating condition and introduces error in output when the voltage or frequency is increased. While in functional approximation technique, approximation is introduced in architecture level by reducing circuit components which indirectly reduces the data path, to attain performance improvement and power efficiency [4]. This paper critically reviews the approximate arithmetic logic circuits such as adders, multipliers and dividers as they are the basic building blocks of processors [6-8, 21]. Further it is well known that almost 80% of the computer operations are carried out only with addition and subtraction, a bunch of CMOS based approximate adder circuits [12, 13, 21-43] are explored by researchers. Performance of the approximate adders is studied by deducing various performance metrics like error distance, error rate, error distance, mean error distance, acceptable probability and minimum acceptable accuracy[3,5,6,8,12-14, 25,54,55]. Error Rate (ER) is defined as percentage of erroneous results among all possible outputs [24, 25]. Error Distance (ED) can be defined as the difference between the actual and approximated output. Minimum Acceptable Accuracy (MAA) is defined as the threshold value fixed to accept the approximated result [58-59]. Acceptance Probability (AP) is the probability that the accuracy of the proposed adder is more than the minimum acceptable accuracy [58-59].

On the other hand, memristors (RRAM) [60], due to their non-volatile nature and high packing density; they are experimented in crossbar memory architecture [61] to replace the existing memory architectures. Further, because of its CMOS compatibility [62] it is investigated as logic elements as well. With this background, the possibility of memristor based approximate computing is investigated by researchers by designing memristor based associative memory [68-73] and also through memristor augmented approximate arithmetic circuits as well [74,75].

2. CONVENTIONAL ADDERS

Traditionally, digital CMOS circuit design is carried out using the most common adders like Ripple Carry Adder (RCA) and Carry Look Ahead Adder (CLA).

2.1 Ripple carry adder

The conventional n-bit RCA [11,15] is designed by cascading the n-single bit full adders, where a single bit full adder takes 3-bits (A, B, Cin) as input and produces SUM and CARRY as output as shown in Table-1. The carry produced from each single bit full adder is fed into the next successive full adder stage as Carry input (Cin) as shown in Figure-1. Hence, conventional RCA produces output only after adding n-bits i.e. the delay is more and further increases as the number of bits increases.

Table-1. Truth table of single bit full adder.

A	B	C_{in}	SUM	CARRY
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

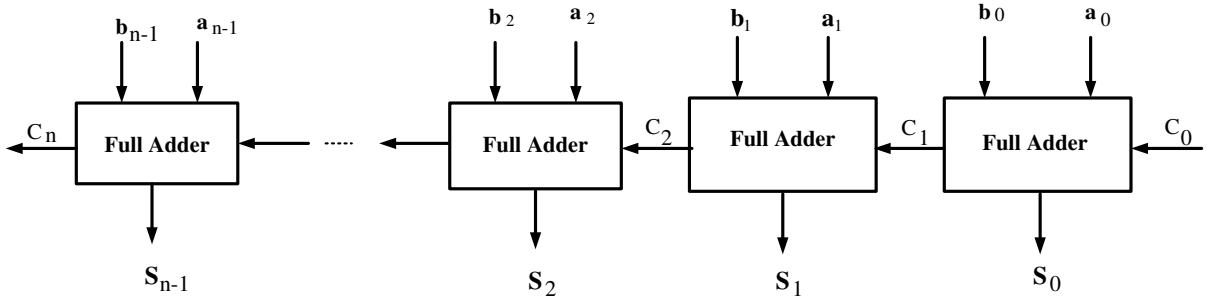


Figure-1. n-bit ripple carry adder where C_0 is the initial input carry(C_{in}), a and b are the two input operands, $C_n \ldots C_1$ are the carry generated through each bit addition, $S_{n-1} \ldots S_0$ are the SUM output of each bit addition.

The main disadvantage of RCA is the Sum output of a particular digit addition cannot be produced without C_{in} from the previous bit. Hence, the critical signal propagation path originates at the LSB and traverse through carry propagation chain to the MSB. This critical path increases as the number of bits increases. As this critical path directly contributes to the performance of the RCA, this increase in delay makes the adder to be slow. Since the generated carry should be propagated from LSB to MSB yet the power consumption is also expected to be more.

2.2 Carry look ahead adder

Another commonly used fast adder compared to the conventional RCA is the Carry Look Ahead (CLA) adder [15] which calculates carry through carry look ahead generator circuit as shown in Fig. 2. Though the delay is less compared to conventional RCA, the complexity of the carry generator circuit increases for large value of 'n', which indirectly increases the required hardware area as well as the power consumption, where 'n' is the total number of input bits.

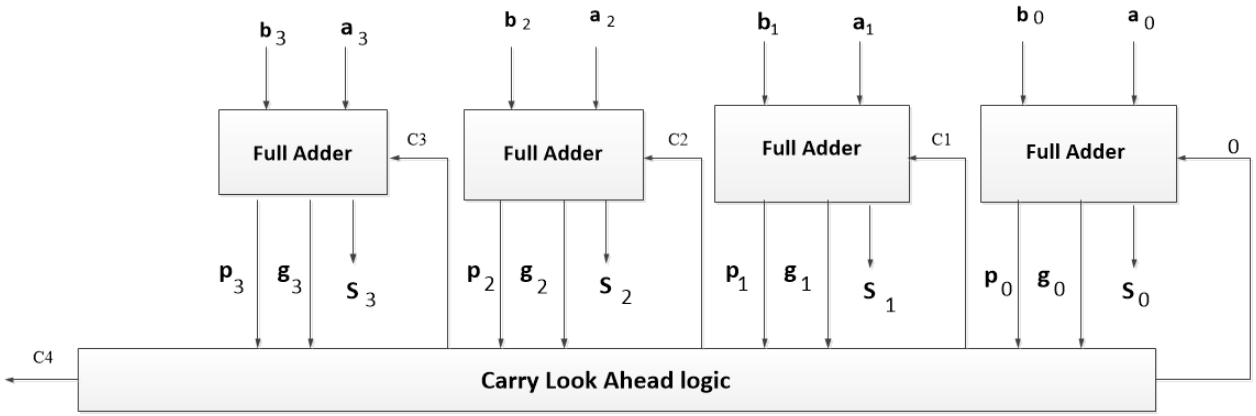


Figure-2. 4 bits carry look ahead adder where p and g are propagate and generate functions respectively: Generate $g_i = a_i b_i$ and propagate $p_i = a_i + b_i$.

The above mentioned conventional adders are mainly designed to produce accurate results causing high energy consumption. To overcome the disadvantages of traditional adder circuits, many approximate adder circuits are proposed and analyzed by researchers [12, 13, 21-43, 56] to improve the performance of the circuit in terms of speed, area, delay and power consumption by compromising the accuracy.

3. APPROXIMATE ARITHMETIC CIRCUITS

Approximate circuits are intended to have less hardware elements therefore minimize the propagation delay, which improves the performance of the circuit in terms of speed and power consumption. These approximate circuits can be designed using many techniques like cell replacement technique, truncation technique, probabilistic logic minimization, gate level pruning, segmentation technique etc., to introduce inexactness in the result. This section discusses the various approximate adders [12, 13, 21-43, 56], multipliers [47-53] and dividers [54-55].

3.1 Approximate adders

3.1.1Probabilistic logic minimization technique

Probabilistic logic minimization technique is a top down, architectural level approach where inexactness is introduced just by inverting the bits of actual truth table for certain input combinations, such that the resulting truth table derives simplified the logical equations [21, 35] as shown in Fig.3. Simplified logical expressions are constructed with less number of digital logic gates, which obviously decreases die area, power consumption and delay.

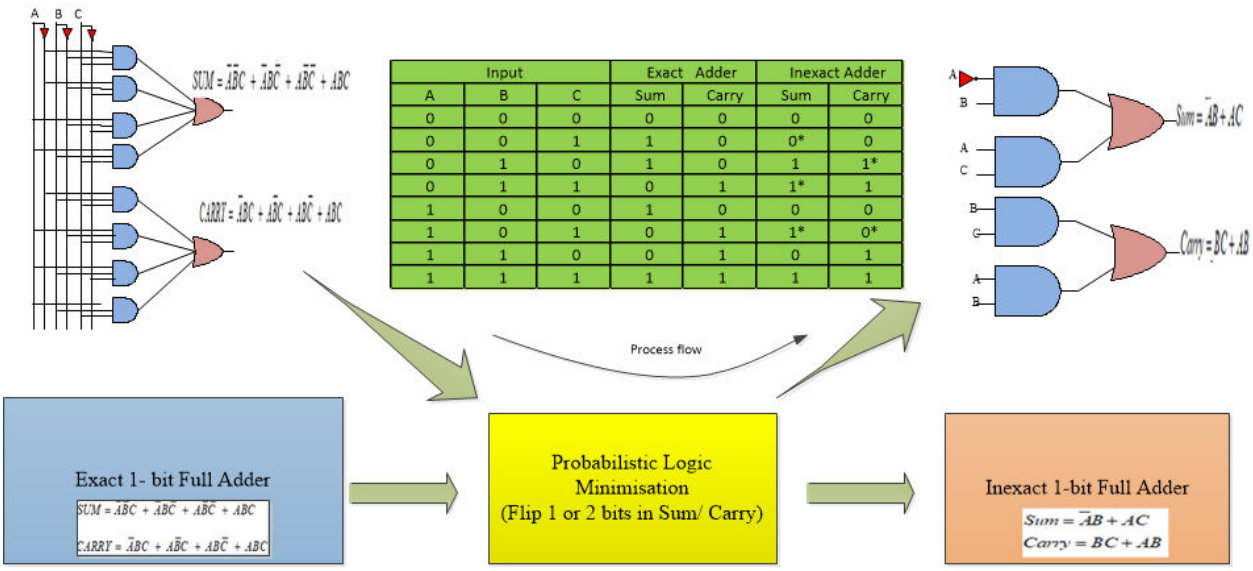


Figure-3. Process flow of designing inexact 1- bit full adder circuit with logic minimization technique by flipping one or two bits (marked with *) in exact adder truth table. Exact adder is designed with 3 inverters, 8 AND gates and 2 OR gates whereas inexact adder uses only one inverter, 4 AND gates and 2 OR gates to produce Sum and Carry.

Two approximate circuits (approximation1 and approximation2) are constructed by Christopher I. Allen. *et al* [24] to generate inexact Carry ($C_{out} = ab$ and

$C_{out} = a + b$) and two more circuits (approximation3 & approximation4) to generate inexact Sum. Approximation 3 has two minterms ($s = \bar{a}\bar{b}c_{in} + a\bar{b}\bar{c}_{in}$) and approximation 4 has three minterms ($s = \bar{a}\bar{b}c + abc + \bar{c}$). Since these approximate adder circuits are designed with less number of minterms, only a few number of logic gates are required to realize the circuits compared with traditional circuits.

An n-bit approximate Ripple Carry Adder is constructed by employing the above mentioned approximation 1 cell in even number of bits and approximation 2 in odd number of bits.

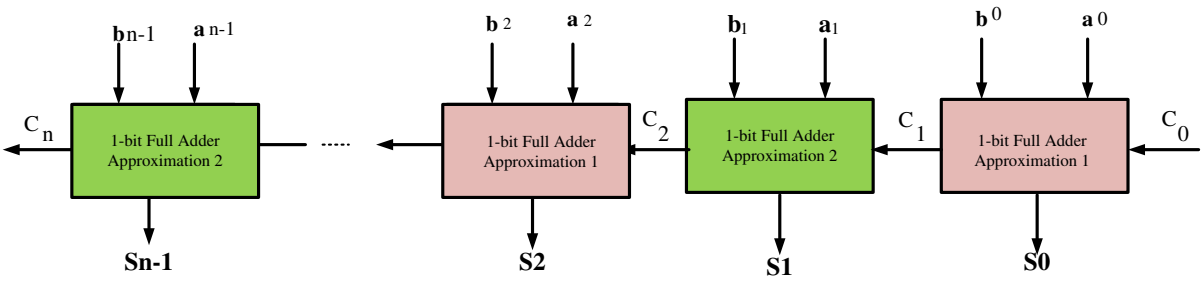


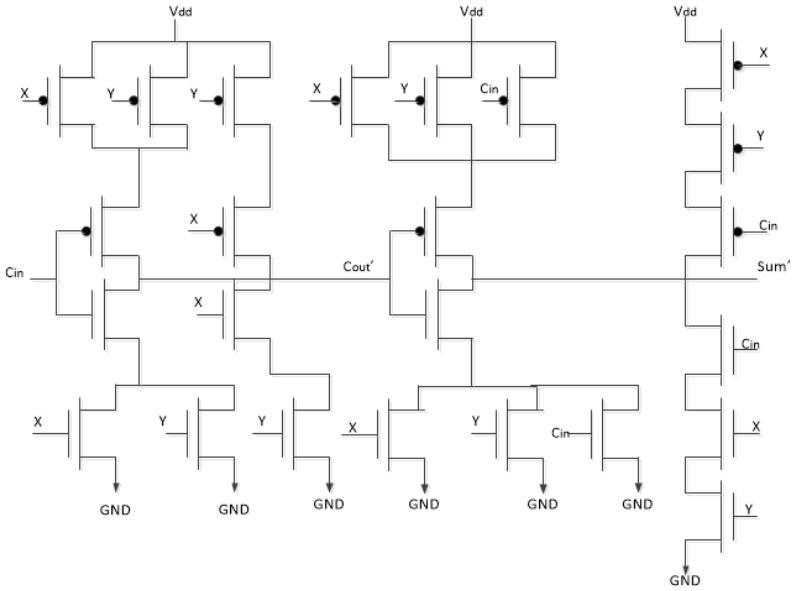
Figure-4. n-bit approximate ripple carry adder designed using 1-bit approximate full adder cells. The full adder cell constructed with approximation1 is used for the addition of even number of bits and Approximation2 is used for the addition of odd number of bits [24].

It is observed that 8-bit approximate RCA designed using the approximated full adder cells has 30% less power dissipation because of the less number of components and the error rate as 25% compared with exact RCA. Further, it is proved that 4-bit approximate RCA constructed using this method has 23% area reduction and 31% power reduction than its exact counterpart.

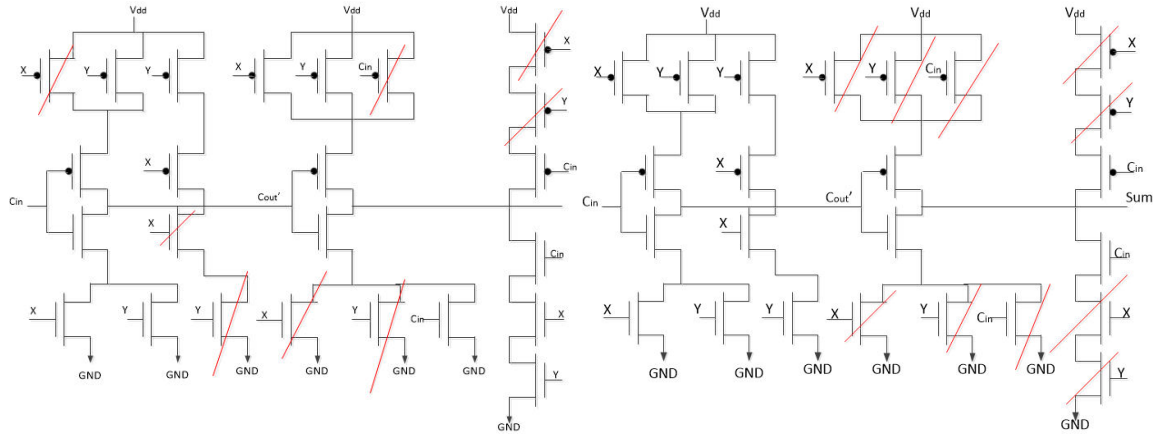
3.1.2 Cell replacement technique

Approximate adder cells are designed by removing one or more transistors from the traditional adder cell. These approximate adder cells are replacing the accurate cells of the traditional adder circuits.

Since the cell replacement technique offers lesser critical path, Gupta et.al proposed five approximate mirror adders (AMA) in [25, 26] by cell replacement technique with less number of transistors, load capacitance and compared their performance with conventional mirror adder (MA) [18, 19]. First two approximate mirror adders (AMA1 and AMA2) are designed by removing one or two transistors from the conventional mirror adder (Figure-5a) without leaving any terminal open or short as shown in Figure-5 (b & c) and the corresponding truth table is derived with minimal error as shown in Table-2.



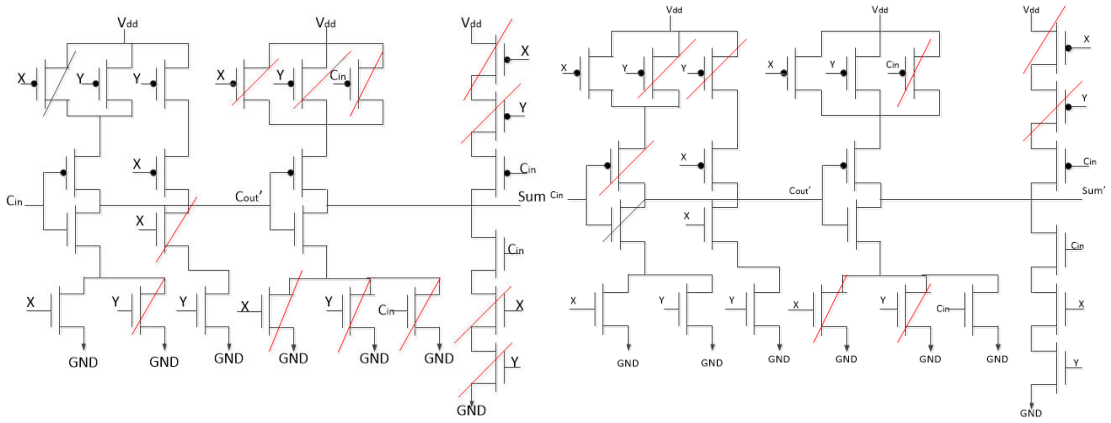
(a)



(b) (c)

Figure-5(a). Conventional mirror adder designed using 24 transistors (b). Approximate Mirror Adder 1 (AMA 1) after removing 8 transistors which are stroked out (shown in red color) without leaving out any terminal open or short from the conventional mirror adder (c). Approximate Mirror Adder 2 (AMA 2) after removing 10 transistors from the conventional mirror adder which are stroked out (shown in red color) without leaving out any terminal open or short.

AMA1 introduces 2 errors in SUM and one error in CARRY and uses 8 transistors lesser than conventional MA which was built using 24 transistors whereas, AMA2 is constructed by simply approximating the SUM through inverting the Carry out signal of conventional MA. This gives out two errors in SUM and exact CARRY. AMA3 is the combination of AMA1 and AMA2 as in Figure-6(a). In AMA4, approximation is introduced in Carry signal generation by inverting input X. Sum is approximated similar to AMA1 as depicted in Figure-6(b).



(a) (b)

Figure-6.(a). Approximate mirror adder 3 (b). Approximate mirror adder 4. Both the circuits are designed by removing the marked transistors without leaving any terminal open or short.

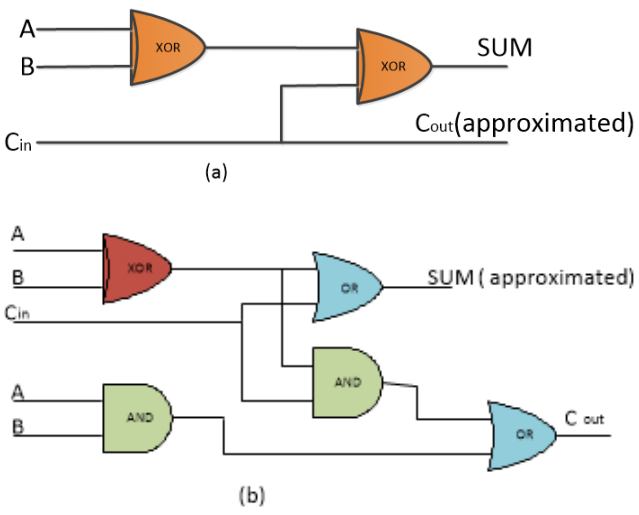
Table-2. Truth table of approximate mirror adder.

X	Y	C_{in}	AMA 1		AMA 2		AMA 3		AMA 4	
			SUM	CARRY	SUM	CARRY	SUM	CARRY	SUM	CARRY
0	0	0	1	0	1 x	0	1	0	0	0
0	0	1	1	0	1	0	1	0	1	0
0	1	0	0 x	1 x	1	0	0 x	1 x	0 x	0
0	1	1	0	1	0	1	0	1	1 x	0 x
1	0	0	0 x	0	1	0	1	0	0 x	1 x
1	0	1	0	1	0	1	0	1	0	1
1	1	0	0	1	0	1	0	1	0	1
1	1	1	1	1	0 x	1	0 x	1	1	1

Note: x indicates the approximated sum/carry with reference to the accurate full adder truth table (refer Table-1).

All these approximated adders are designed in IBM 90 nm technology. Approximate RCA is designed using these approximated cells in lower order bits. Further, approximation is introduced in carry by eliminating the carry propagation of the lower order bits. As these approximate mirror adders uses less number of transistors which obviously reduces the hardware area, load capacitance along with propagation delay and supply voltage. Conventional MA occupies 40.66µm^2 area whereas AMA1, AMA2 and AMA3 occupies 22.56, 23.91 and 13.54µm^2 respectively, which clearly indicates the reduction in hardware area.

Similar to this, Almurib *et al.* designed three inexact adders (InXA1, InXA2, InXA3) in [27] by cell replacement technique as shown in Figure-7. First approximation cell (InXA1) is designed by approximating the carry while generating the accurate Sum. This errored carry is disseminated to the next immediate cell of the multi bit adder which produces approximate output. Second approximate adder cell (InXA2) is designed with accurate carry and approximated sum. Third approximate adder cell (InXA3) is designed similar to InXA1 but errors are introduced only in the first and the last row of the Carry out signal. Refer Table 3for the approximated truth table of all the three inexact adders.



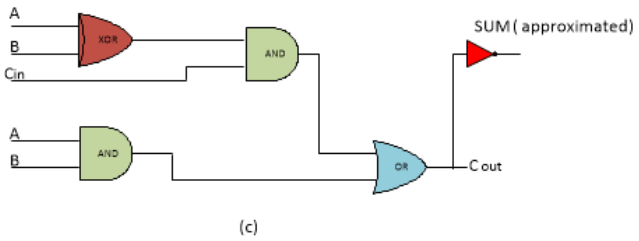


Figure-7. Logical gate diagram of Inexact adders. (a) InXA1 produces exact Sum and approximated Carry (Cout = Cin) (b) InXA2 produces approximated Sum and exact Carry (c) InXA3 produces approximated Sum and exact Carry.

Performance of these approximated cells are compared with approximate mirror adders suggested in [23] in terms of energy, energy delay product, number of circuit components and load capacitance. All the suggested approximated cells bring in only two errors either in sum or carry; therefore, the error rate is only 25%, whereas all the approximate mirror adders have the error rate more than 25%. Also the proposed adder uses fewer number of transistors (6 to 8) for the design, ultimately reduces the circuit complexity while AMAs uses more than 12 transistors. Ripple Carry Adders are designed with newly designed approximated cells. Though all the three inexact adders based RCAs also have almost same error rate of 25%, InXA2 is recommended since it has minimum error distance of 1. Further, the carry generated from LSB is not disseminated to the higher order bits; hence less delay is also reported.

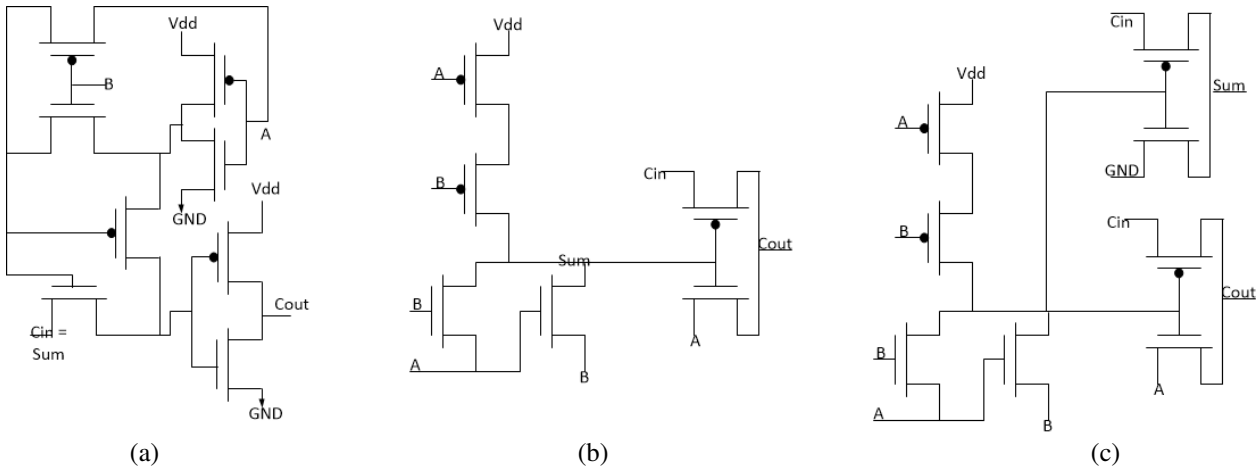
Table-3. Truth table of inexact adders.

A	B	C_{in}	Exact		InXA 1		InXA 2		InXA 3	
			SUM	CARRY	SUM	CARRY	SUM	CARRY	SUM	CARRY
0	0	0	0	0	0	0	0	0	1 x	0
0	0	1	1	0	1	1 x	1	0	1	0
0	1	0	1	0	1	0	1	0	1	0
0	1	1	0	1	0	1	1 x	1	0	1
1	0	0	1	0	1	0	1	0	1	0
1	0	1	0	1	0	1	1 x	1	0	1
1	1	0	0	1	0	0 x	0	1	0	1
1	1	1	1	1	1	1	1	1	0 x	1

Note: x indicates the errored sum/carry.

3.1.3 Truncation technique

Another way of introducing errors into exact circuit is through truncation which helps to achieve less delay, area and power consumption. Zhixi Yang *et al.* constructed three approximate adders (AXA1, AXA2, AXA3) as shown in Figure-8, through XOR/XNOR gates with multiplexers designed using pass transistors [28]. AXA1 is designed with XOR gates (realized by two pass transistors and an inverter) as in Figure-8(a). 8 transistors are used to construct this approximate adder cell that produces accurate Sum and Carry output for four input combinations out of total possible eight combinations, which leads to the error distance of 4.



(a) (b) (c)

Figure-8. Approximate Adders designed using XOR/XNOR gates adapted from [28]. (a) AXA1 produces errored sum and carry for 4 input combinations. (b) AXA 2 generates errored sum for 4 input combinations and accurate carry. (c). AXA 3 gives out exact carry and inexact sum for 2 input combinations.

In the same way, AXA 2 and AXA 3 are designed using 4T-XNOR gates and a pass transistor as shown in Fig. 8 (b & c). AXA 2 and AXA 3 gives out exact carry for all the eight input combinations while generating errored Sum for 4 input combinations in AXA 2 and for 2 input combinations in AXA 3 as shown in Table-4. Hence the calculated error distance is 4 and 2 for AXA2 and AXA 3 respectively. In terms of transistor count, AXA 2 uses only 6 transistors whereas AXA 3 is designed with 8 transistors in order to improve the accuracy of AXA 2.

Table-4. Truth table of approximate adders.

A	B	C_{in}	AXA 1		AXA 2		AXA 3	
			SUM	CARRY	SUM	CARRY	SUM	CARRY
0	0	0	0	0	1	0	0	0
0	0	1	1	0	1	0	1	0
0	1	0	0 x	1x	0 x	0	0 x	0
0	1	1	1 x	0x	0	1	0	1
1	0	0	0 x	1x	0 x	0	0 x	0
1	0	1	1 x	0x	0	1	0	1
1	1	0	0	1	1 x	1	0	1
1	1	1	1	1	1	1	1	1

Note: x indicates the approximated sum/carry with reference to the full adder truth table (refer Table-1).

Performance of all the three proposed approximate adders were compared with accurate adder designed using 10 transistors[29,30] for area, power consumption, delay and power delay product. Based on the comparison the author has concluded that AXA 3 is the power efficient approximation that gives the minimal error distance, while AXA 2 occupies less area as it uses only 6 transistors and AXA 1 has the lowest power delay product along with the shortest delay offers better performance. The main drawback of these approximate adders is the pass transistor which does not allow the output to full voltage swing there by decreasing the noise margin.

3.1.4 Segmentation technique

In full adder circuits, carry propagation is the main cause for huge delay. Hence, lots of efforts are made to reduce the delay of the adder circuits, mainly by cutting short the carry propagation, consequently introducing errors (approximation) in the output. One way of achieving this is by segmenting the total of number bits into smaller number of segments where each segment length must be greater than 2. Addition of the operands in lower order segments are carried out by any approximation logic that reduces the delay.

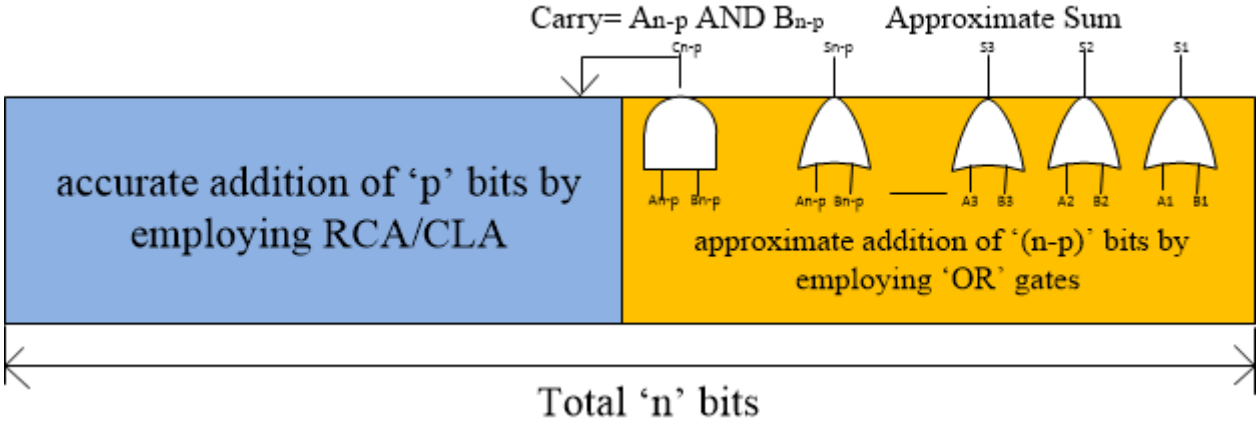


Figure-9. Lower part OR adder designed using segmentation technique. Total 'n' bits are segmented into two blocks: accurate block of 'p' number of higher order bits in which the accurate addition is carried out by conventional adders like RCA, CLA etc. and inaccurate block of 'n-p' lower order bits where the addition is performed with simple OR logic & carry is omitted for all the lower order bits except for the two MSB's of the inaccurate block. Carry is calculated using simple AND gate and the same are fed in as Carry input for the accurate block.

Lower part OR Adder (LOA) proposed in [31] by Mahdiani et.al is such type of adder, where the errors are introduced in LSBs. Total 'N' bits are divided into two parts: precise upper block of 'p' bits and imprecise lower block of '(N-p)' bits as in Figure-9. Accurate adder (by conventional RCA/CLA) is employed for 'p' MSBs whereas simple OR gates are engaged for the addition of '(N-p)' LSBs. Carry generated from all the other LSBs are discarded except from the two MSBs of the lower block. Carry generation from the lower block is taken care by an AND gate when two MSBs of lower block operands are high. Introducing approximation through the use of OR gates in lower order bytes not only reduces the hardware area and delay but also the energy consumption. However, the accuracy of the adder is dependent on the length of the lower block OR adder; more the length of the OR block less the accuracy. Hence LOA is not suitable for the applications which expect high accuracy.

Another approximate adder suggested in improving the performance in terms of speed and energy consumption is the bunch of Error Tolerant Adders [ETAs] in [32-35]. The first version of the ETA is simply called as Error Tolerant Adder (ETA) [32] carries out the addition by segmenting the operands into two fragments namely, higher and lower fragments, both need not be of equal length.

Addition of the higher order bits is performed from right to left by any conventional accurate adders like RCA, CLA by assuming C_{in} =0, since MSBs are the main contributors of accuracy. The bits in the lower fragment is added from left to right by removing the carry propagation with new logic i.e when both the operands have '0' value or different values then simple bit wise addition is performed whereas when both the operands have value '1' then the calculation is stopped and all other output bits right to this place are made high which is depicted in Figure-10.

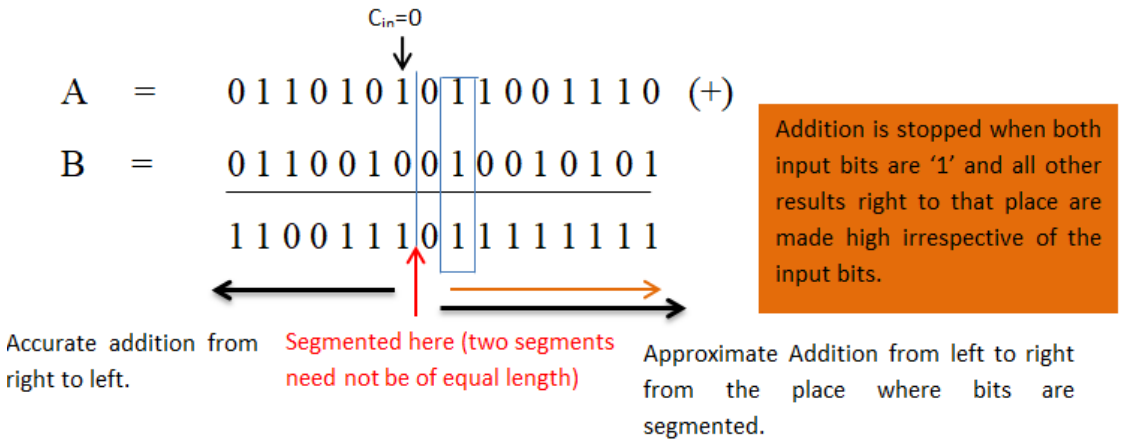


Figure-10. Sample calculation of error tolerant adder. A and B are two input operands that are segmented after 9th bit position. Addition of higher order segment is taken care by conventional adders while the lower order block is done approximate logic. Both the blocks need not be of equal length.

This inaccurate calculation is performed by generating a control signal through modified XOR gate and carry free addition block. The upper part and lower part additions are performed simultaneously starting from the place from where it is separated as shown in Figure-10. This parallel process decreases the overall delay; therefore the improvement in speed and reduction in power consumption is achieved. The overall delay can further be reduced by splitting the total number of lower fragment bits into some random number of blocks and perform the addition operation for the MSBs of each block. If the result is high, the signal is directly passed on to the MSB of the next block by skipping carry propagation to the next immediate bits of the same block. The performance of the ETA is evaluated in terms of power consumption, delay and power delay product by comparing it with the traditional RCA, CLA, Carry Skip Adder (CSK), and Carry Select Adder (CSA) for 32 bits addition. All the adders are designed using CMOS 180nm technology and the simulation is done using HSPICE. From the observation, it is noted that the designed ETA consumes 130µW power whereas RCA, CLA, CSL, CSA consumes 220 µW, 460 µW, 600 µW, 510 µW respectively. In the same way, the constructed ETA takes only 2.29ns delay compared with the above mentioned traditional adders. With respect to transistor count, ETA is relatively as good as RCA, where ETA uses 1006 transistors and RCA uses only 896. From the analysis it is understood that ETA is better than all the conventional adders considered for comparison.

The main drawback of ETA is the accuracy of the result for the addition of two smaller numbers [32]. For instance, when two 16 bit numbers 0000000000001111 and 0000000000001111 are added, ETA produces 0000000000001111 as a result whereas the expected sum is 0000000000011110, which is very much deviating from the result. The newly proposed ETA type II [33]overcome this drawback by dividing the entire carry propagation path into 'n' number of smaller paths to calculate the carry of all the blocks simultaneously instead of completely eliminating the carry propagation. The total 'n' bits are split into 'p' number of blocks; each block has a sum generator and a carry generator, to process 'n/p' bits. The entire carry generator blocks produce Carry out (C_{out}) with C_{in} signal as '0'. This C_{out} is fed into the Sum generator block of the next immediate stage thus producing Sum bits. Since the carry propagation lies between only the carry generator block and the sum generator block, the delay is drastically reduced. For example, a 16 bit adder is divided into 4 blocks, where each block is taking care of the addition of 4 bits as shown in Figure-11. This improves the speed of ETA II. Since long chain carry propagation is removed off, power consumption is significantly reduced. Further, this ETA II has better accuracy than ETA. Nevertheless, this accuracy depends upon the number of bits in each block. Less number of blocks with more number of bits gives accurate output at the cost of increase in delay path. On the other hand, more number of blocks with less number of bits improves the speed of the adder but with less accuracy.

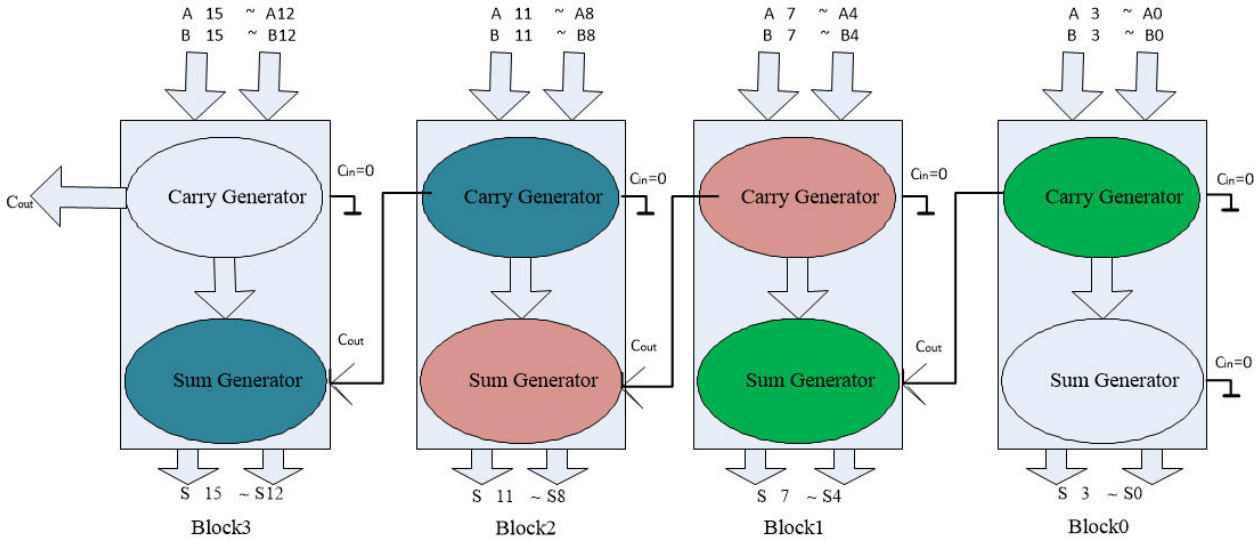


Figure-11. 16- bit error tolerant adder divided into 4 sub adders of 4 bits. Carry generated by the carry generator circuit is fed in as a carry input to the next immediate sum generator circuit which is shown in the same color (critical path). In order to reduce the carry propagation path all the carry generator circuits consider input carry as 0.

Though ETA II improves the performance of the adder in terms of speed and power consumption, it suffers from accuracy degradation for large input operands [34]. This is taken care by cascading first three carry signals to produce carry signals for next higher block, guarantees the accuracy of the adder but with increase in delay of 63.5% which affects the performance of the adder. This slight modification proposed in structure of ETA II is named as ETA II modified (ETA II M).

Both ETA II and ETA II M improve the performance by increasing the speed and reducing the power consumption with better accuracy for even smaller

number of additions. Compared with traditional adders, these ETAs offers less accuracy. Hence ETAs are not suitable for applications which give more importance to accuracy than speed and power consumption.

In this series, one more error tolerant adder, ETA IV is constructed by the same author to further reduce the delay than ETA IIM. The structure of ETA IV is similar to carry select adder with 2:1 multiplexers. Compared to ETA, ETA IV has commendable accuracy for all input ranges. The performance of all the ETAs is compared for hardware area in terms of transistor count, power, delay and power delay product with conventional adders. It is proved that the designed ETAIV provides 26% improved delay performance than ETAIIM with increase in 5% transistor count [35].

Another error tolerant adder designed by Matthew Weber is balancing adder [36], aims to detect and correct the error in carry chain through the balance (BAL) block, to minimize the errors. This BAL block is placed only in the lower part of the adder. ETAIIM does not work well when both the input operands have equal magnitude in opposite polarity. These kinds of killer inputs are managed well by the balancing adder that reduces error rate drastically.

3.1.5 Pruning technique

All the approximate arithmetic circuits discussed so far decides the quantity of inexactness during design time itself and is not subjected to alteration. But the proposed gate level pruning technique [37, 40] is a circuit level approximation technique, designs inaccurate circuit during runtime by pruning the least significant gates from the generated gate level net list of the intended adder circuit. This technique suits larger circuits which exhibits high degree of parallelism. It is not suitable for RCA kind of cascaded architecture because pruning in between cells result in breaking the carry propagation which gives out high error rate.

Another approximate adder proposed at architecture level [41] is a carry cut back adder, which has a chain of simple fixed point addition blocks connected together with multiplexers. Multiplexers take actual carry and speculated carry (from SPEC block) as inputs and a 'cut' signal from PROP block as a control signal to decide the carry propagation which minimizes the effective critical path.

3.1.6 Carry speculation technique

The inaccurate adders discussed so far are designed to meet up certain performance parameters to trade off accuracy of error resilient applications. But as we know, all the error resilient applications do not have the same characteristics, therefore variations in expected performance metrics. So the inaccurate adder constructed for a particular error tolerant application cannot be applied for another application. Hence, approximate adders have to be redesigned depending upon the design parameters which is the biggest disadvantage of approximate circuits. In order to overcome this problem, error detection and correction circuits are included to configure the accuracy of the approximate adder circuit at the cost of additional hardware area. In this way, a variable speculative carry select adder (VLSA)[42] is designed, which produces accurate result in two stages: the first stage produces an Almost Correct Adder output along with a signal to indicate whether the generated output is accurate or not. If it is not accurate, then the error correction circuit corrects the error and produces accurate output after two cycles based on the timing analysis in the second stage.

Since VLSA suffers from huge delay and hardware area overhead, another adder proposed with error correction logic to overcome this drawback is Accuracy configurable adder (ACA) [43] in which accuracy can be configured dynamically during runtime. This feature supports to employ the adder both in accurate and inexact operations. An n-bit adder is divided into (n/k-1) sub adders where each sub adder adds 2k consecutive bits with an overlap of k bits as shown in Fig. 12. All the sub-adders function in parallel which reduces the delay. In each sub-adder, the half most significant sum bits are selected as the partial sum. Figure-13 shows 16 bits ACA divided into 3 sub adders where each sub adder takes 8 consecutive bits as input with an overlap of 4 bits and produces approximate sum.

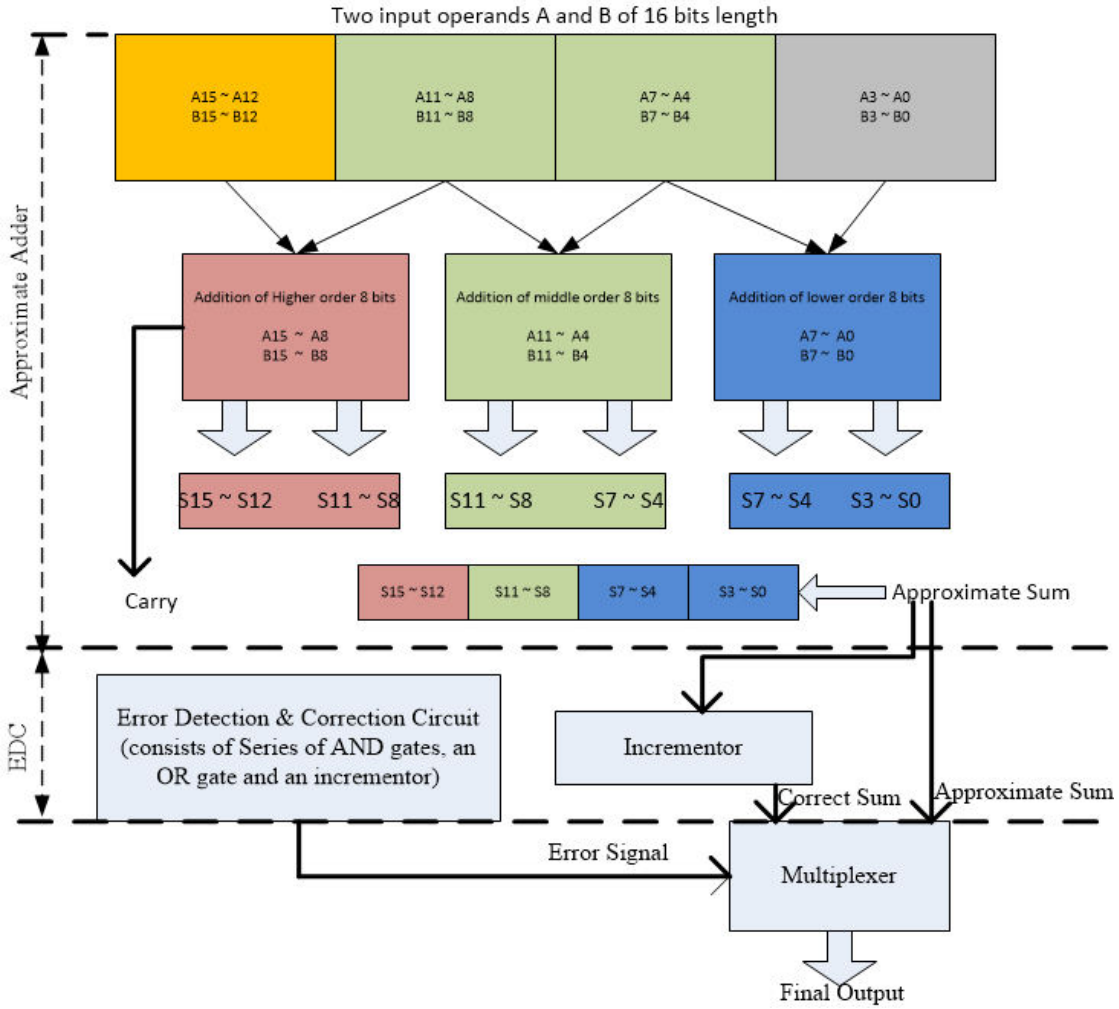


Figure-12.16 bit ACA divided into 3 sub adders where each sub adder takes 8 consecutive bits as input with an overlap of 4 bits and produces approximate sum.

Since the output produced by ACA is approximated sum, an error detection circuit (EDC) constructed with series of AND gates are used to detect the errors and an incrementor is used as a correction circuit. As approximate adder and error detection & correction circuits work in pipelined fashion, the speed of the circuit is more compared with the conventional adders. The ACA provides 26.3% improvement in throughput and 37% reduction in power consumption than CLA. Another speculative adder [44] proposed by Du et.al. is speculative carry select adder (SCSA), which improves the speed and accuracy of the adder by reducing the critical path through carry speculation and error correction but at the cost of additional hardware. SCSA adds two 'n' bit operands by segmenting into 'm' blocks sub adders where each block adds 'n/m' bits. Each block is employed with two adders, namely adder0, adder1. Adder 0 adds two 'n/m' bits addition with C_{in} =0 whereas adder 1 adds with C_{in}=1.Either one of the adder output is selected by multiplexer as a speculative sum bit based on the C_{out} of the previous bit. Figure-13 explains the addition of two 8 bit operands which are basically divided into two blocks. Each block takes care of the addition of 4 bits. Each block has two adders namely, adder 0 and adder 1. The Adder 0 calculates the sum by assuming C_{in} as 0 whereas the Adder 1 calculates sum with C_{in} as 1. Either one of the adder output is directed to output based on the Cout signal of the previous block.

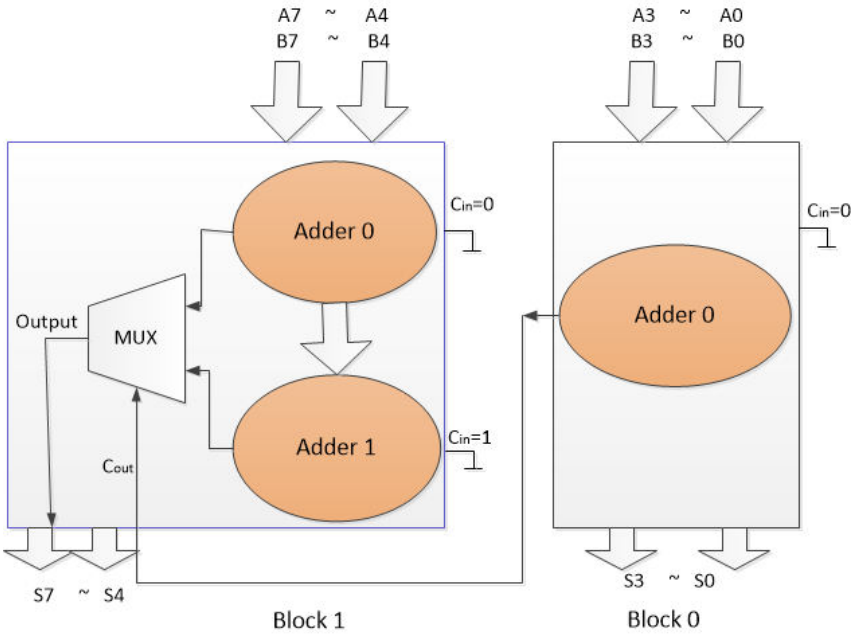


Figure-13. 8 bits SCSA divided into 2 blocks where each block has two adders: adder0, adder1; Adder 0 produces sum assuming C_{in} as 0 whereas adder1 produces output with C_{in} as 1. Either one of the output is selected by the multiplexer based on the C_{out} signal of the previous block.

Another adder works in the principle of speculation is carry skip adder which segments 'n' bits adder into 'm' sub blocks of adders and carry generators. Carry generators together with multiplexers helps to speculate the carry to reduce the critical carry propagation path. Another approximate adder designed in this category is generic accuracy configurable adder (GeAR) in [42]. It is an open source accuracy configurable adder model with error correction logic. Addition of two N-bits are carried out by dividing the operands into 'k' number of 'L' bits sub adders in parallel with limited carry propagation. Hence, reduced delay with faster response. Table-5 summarize the circuit characteristics of the approximate adders of 16 bits [47].

Table-5. Comparison of CMOS-approximate adders based on circuit characteristics.

Approximate adders-CMOS	**Delay**(ps)	**Power**(μW)	**Area**(μm^2)
LOA	223.76	109.95	69.25
ETA I	223.76	143.62	138.35
ETA II	223.76	192.63	164.79
ACA	223.76	260.86	177.28

3.2 Approximate multipliers

Like adders, multipliers also play an important role in building digital signal processing applications. Hence performance of the multiplier has significant role on deciding the performance of the application. Therefore, approximate multipliers can be a good choice for error tolerant applications to improve the performance with less circuit complexity. In general, multiplication is carried out in three steps: partial product generation, partial product accumulation, partial product addition. Various approximate multipliers [48-54] are proposed by introducing approximation in either one of these three steps as depicted in Figure-14.

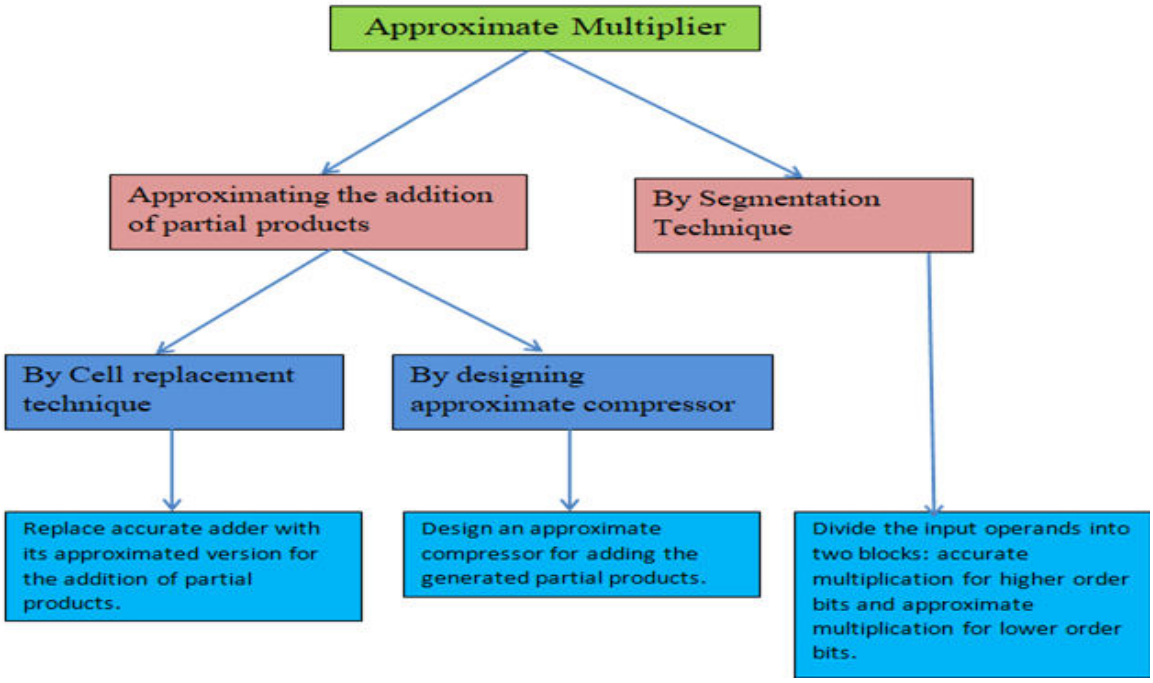
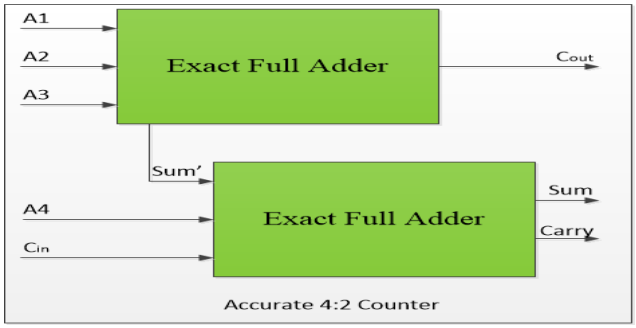


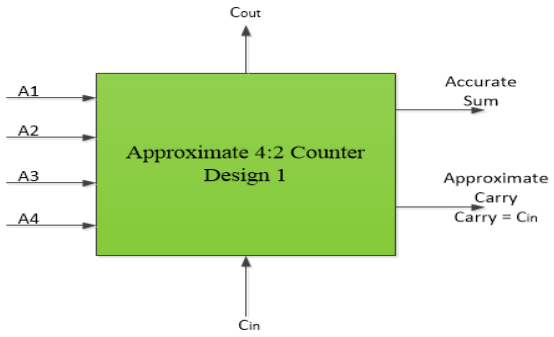
Figure-14. Various ways of designing approximate multiplier using different techniques.

3.2.1 Approximation in partial product accumulation using compressors/counters

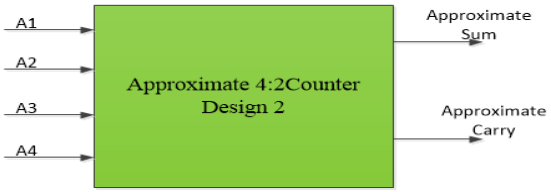
Momeni *et al,* proposed two types of approximate 4-2 counters / compressors [48] for approximate multiplication.



(a)



(b)



(c)

Figure-15. 4:2 Counters (a) Accurate (b) Approximate counter 1 generates approximate carry = C_{in} (c) Approximate counter 2 produces approximate sum and approximate carry.

First type of approximate counter is designed by assuming the output carry (C_{out}) equal to the input carry (C_{in}) because, from the exact counter truth table (refer Table-6) it is understood that C_{out} is equal to C_{in} for the 24 combinations of input. This introduces more error in the output i.e. high error rate, which is not acceptable. Hence, another method of counter design is proposed where Sum and intermediate carry signals are simplified to bring out the acceptable output with decrease in error distance and critical path delay. The performance of the counter is further improved by neglecting C_{in} and C_{out}, which minimizes the error rate in turn, decreases the hardware area and the critical path delay of the circuit.

Table-6. Truth table of accurate 4:2 Counter.

Cin	A1	A2	A3	A4	Cout	Sum	Carry
0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0
0	0	0	1	0	0	1	0
0	0	0	1	1	1	0	1
0	0	1	0	0	0	1	0
0	0	1	0	1	1	0	1
0	0	1	1	0	1	0	1
0	0	1	1	1	1	1	1
0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0
0	0	0	1	0	0	1	0
0	0	0	1	1	1	0	1
0	0	1	0	0	0	1	0
0	0	1	0	1	1	0	1
0	0	1	1	0	1	0	1
0	0	1	1	1	1	1	1
0	1	0	0	0	0	1	0
0	1	0	0	1	0	0	1
0	1	0	1	0	0	0	1
0	1	0	1	1	1	1	1
0	1	1	0	0	0	0	1
0	1	1	0	1	1	1	1
0	1	1	1	0	1	1	1
0	1	1	1	1	1	0	1
0	1	0	0	0	0	1	0
0	1	0	0	1	0	0	1
0	1	0	1	0	0	0	1
0	1	0	1	1	1	1	1
0	1	1	0	0	0	0	1
0	1	1	0	1	1	1	1
0	1	1	1	0	1	1	1
0	1	1	1	1	1	0	1

Approximate multiplier is constructed using either one of the approximate counters in partial product accumulation tree. As partial product accumulation is the root cause for huge delay and power consumption, approximation is introduced in this stage to reduce the delay and power consumption in order to enhance the performance of the multiplier. The dadda tree multipliers are designed using the above mentioned approximate counters to carry out the unsigned 8 bit multiplication and the performance metrics like error rate, delay, transistor count, error distance and power consumption was compared with exact dadda tree multiplier. As a result, second approximate counter design is considered to be the best since it reduces 50% of the transistor count and 60% power consumption than exact compressor. Also 35% faster than the first counter design approach.

3.2.2 Approximating the partial product accumulation Of the multiplier by eliminating carry propagation

The approximate multiplier with low power and high performance [49] is constructed by fitting in the high speed, easy approximate adder in the second stage of multiplication i.e. partial product accumulation tree. High speed is achieved by eliminating the carry propagation which gives the way for parallel computation of sum and error signal. This error signal contributes to the accuracy of the multiplier. Approximate adder is devised by input pre processing (IPP) in the concept of interchangeability

of bits. This logic is applied for the input combination of X=0 and Y=1 without changing the other combinations (00, 10, 11), to make the operand X with higher number of 1's and Y operand with higher number of 0's. For example, the added result is 1111 When two inputs X = 1100 and Y = 0011 or X = 1111 and Y = 0000 are added, produces the same result 1111as output. Hence, addition is carried out by preprocessing the inputs for the combination X=0 and Y=1 by interchanging the bits.

Hence, the preprocessed input is generated by the following equations (1) & (2),

$$X_n' = X_n + Y_n \quad (1)$$

$$Y_n' = X_n Y_n \quad (2)$$

Where n is the n^{th} bit and X_n' and Y_n' are the preprocessed input bits.

Table-7. Truth table of the approximate adder.

X_n'	$Y_n'Y_{n-1}'$	C_{n-1}/Y_{n-1}'	S_n	E_n
X_n'	00	0	X_n'	0
X_n'	01	1	1	X_n'
1	10	0	0	0
1	11	1	1	0

A truth table is derived by applying this logic as shown in Table-7 and the approximate adder is designed to produce approximate sum and an error signal which satisfies the equations 3 &4.

$$S_n = (X_n \oplus Y_n) + X_{n-1}Y_{n-1} + Y_n'X_n' \quad (3)$$

$$E_n = (X_n \oplus Y_n)\, X_{n-1}Y_{n-1} \quad (4)$$

The generated sum through this approximate adder is always less than or equal with the exact adder. Hence the error signal generated can be added with approximated sum using an adder function to compensate the error. Therefore the accurate result can be obtained by adding the generated error signal with the approximated adder output.

Accurate adder output = Approximated Sum + Error signal

Since the approximate adder is employed to accumulate the sum it decreases the circuit complexity. When compared with error tolerant multiplier [51], this approximate multiplier consumes 26.8 % less power. It is proved that the delay of the proposed approximate multiplier is reduced by more than 50% compared with Wallace and Dadda multiplier for 64 bit multiplication.

3.2.3 Approximating the adder stage of multiplier

In the same way, accuracy configurable approximate multiplier with error detection and correction is designed by Ashutosh Mehta in [50] as shown in Figure-16.

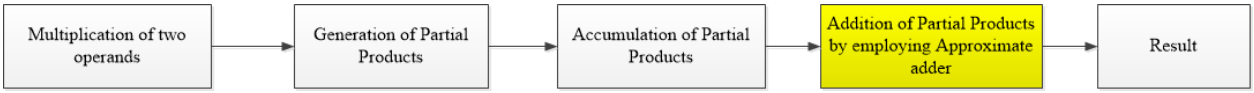


Figure-16. Flow diagram depicts three different stages of multiplication process. Highlighted block shows where the approximation is introduced in the adder stage using any approximate adder designed in section 2.

16 bit multiplication is carried out using this approximate multiplier where approximation is introduced in the adder stage. Accuracy configurable approximate adder is used with error detection and correction circuit, provides the facility to configure the accuracy during run time. Apart from the general parameters like delay and power consumption, the designed multiplier is compared with high accuracy approximate multiplier, inaccurate multiplier proposed by Kulkarni et al.[54], on pass rate for 16 bit multiplier. It is proved that the multiplier designed by this method achieves better pass rate of 98 % for lesser stages compared with other high performance approximate multipliers. Also the designed approximate multiplier performs better than Wallace tree multiplier [52], High Accuracy Approximate Multiplier (HAAM) in terms of propagation delay without error detection and correction circuit. The propagation delay of the proposed multiplier is 26% less than Kulkarni multiplier [54], 17%, and 13% less than Wallace tree and HAAM respectively without error detection and correction circuit. But the main drawback is that the proposed multiplier consumes 66% more power than the Wallace tree multiplier because of additional hardware circuit used in adder stage.

3.2.4 Approximate multiplication by segmentation technique

An Ultra low power high speed multiplier [51] is designed for error tolerant applications using segmentation technique, where input is divided into two parts i.e. higher order bytes (multiplication part) and lower order bytes (non multiplication part), both need not be of equal size and the multiplication is carried out in parallel for both the parts similar to the design process of approximate adder using segmentation technique (refer Figure-11). Conventional multiplication logic is applied for the multiplication part as higher order bytes contributes to accuracy of the final result. Simultaneously, the result is calculated for non multiplication part from left to right. Here all the output bits are made high from the place where any one or both operands are at logic 1. Otherwise logic 0 is produced as output. This logic removes the generation of partial products output which directly contributes to the reduction in delay and power

consumption. It is observed that 75-90% of reduction in power dissipation and greater than 50% of economy in hardware area are achieved in the proposed 12 X 12 ETM compared with traditional multiplier.

Another approximate multiplier proposed [52] by Kartikeya Bhardwaj is Power and area efficient approximate Wallace tree multiplier where the multiplication of b X b bits is carried out in four parts.

a) The accurate 'b/2' LSB's- by traditional multiplication

b) completely inexact next 'b/2 bits' –set as '1' without any calculation

c) exact 'b' MSB bits which is the main contributor of accuracy along with carry from the previous inexact calculation

d) Carry in pre computation logic for the critical column which has maximum number of components in the multiplication calculation. Carry is set as 1 and passed on to the next column if this critical column has number of ones more than 1.

Since the multiplier is introduced for error resilient applications, this carry precomputation is approximated to minimize the required area, i.e. the carry calculation is carried out only through OR gates i.e. OR operation of all the elements present in critical column. The horizontal critical path is diminished to half and Wallace tree reduction is used to reduce the vertical critical path delay also. The proposed multiplier has more than 98% of acceptance probability with 99% minimum acceptable accuracy. Mean accuracy of 99.85 % to 99.965%, reduction in power is up to 41.6%, area reduction is 34.49% and latency for a single cycle implementation of 24% is obtained for the simulation of 4X4, 8X8 and 16X16 bits multiplication.

Srinivasan Narayanamoorthy *et al.* suggested an approximate multiplier [53] by Dynamic Segment Method (DSM) that works by selecting 'm' consecutive bits, where m>=n/2, from total 'n' bits. The 'm' bits can begin from the place where the leading one bit is placed. Two 'm' bits are considered from two 'n' bits operands to perform m X m multiplication which improves not only the accuracy of the result than performing n X n multiplication but also consumes less energy. However, the selection of 'm' consecutive bits may skip the signed bit which obviously decreases the accuracy. Moreover, the DSM increases the hardware components, hence consumes more energy.

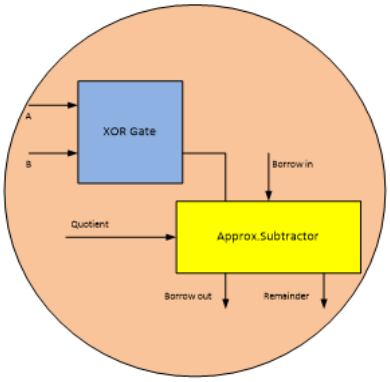
3.2.5 Approximate multiplier using cell replacement technique

Parag Kulkarni et al. [54] designed a higher order inaccurate multiplier using the proposed inaccurate 2x2 basic multiplier as a building block by cell replacement technique. The 2x2 multiplier is designed by voluntarily introducing error for the combination 3*3 in K-map i.e instead of the expected result 1001, it is made as 111. This decreases the complexity of the circuit with error rate of 1/16. The under designed 2x2 multiplier has shorter critical path, lesser switching capacitance and requires only half of the area of the exact multiplier. Further, the proposed multiplier consumes less power for the same frequency of operation. Larger multipliers are constructed using 2X2 inaccurate multiplier. Here inexactness is introduced in the calculation of partial products whereas the adder is an accurate one. The error rate of 2X2 multiplier is 22.22% and increases with increase in bit width. As 2X2 multiplier is the basic unit, error rate can still be reduced by replacing the inexact multiplier with its accurate version. Simulation is carried out for 4x4 bit multiplier and compared the result with accurate multiplier. It was observed that the proposed multiplier drastically reduces the power in the range of 31.8%-45.4% for increase in bit width.

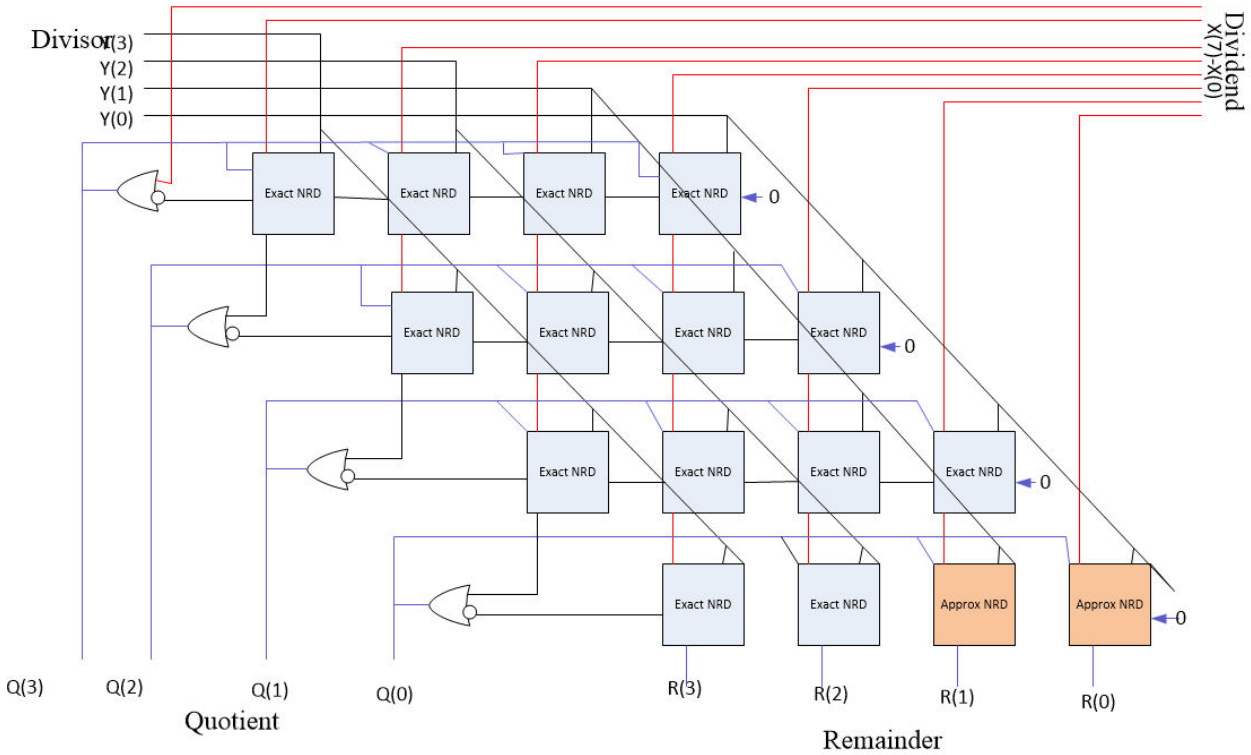
3.3 Approximate non - restoring divider by cell replacement technique

Designing approximate adders and multipliers are explored by lot of researchers in many error tolerant applications. But, subtraction and division is not in the vicinity of the research people. Linbin Chen *et al.* proposed approximate non restoring divider (NRD) using approximate non restoring divider cell [55, 56]. Approximate NRD cell is built with approximate subtractor cell as basic unit. These approximate subtractor cell is derived from the accurate subtractor cell by removing one or two transistors. Here, accurate subtractor cell is constructed with XOR/XNOR for difference, AND & OR gates for borrow. Hence by removing the transistors, three types of approximate version of the subtractor cell namely AXSC1, AXSC2 and AXSC3 are designed. As the accuracy of the subtractor mainly depends on Borrow out signal than Difference, AXSC1 and AXSC3 are designed with error distance of 2 keeping borrow out unchanged. The number of transistors used to design an approximate subtractor cell is 5 to 7 whereas the exact subtractor cell is designed with 10 transistors.

One way of constructing the approximate NRD is by cell replacement technique where the exact subtractor cell is replaced by its approximated version as shown in Figure-17, to reduce the circuit complexity and power consumption. Cell replacement is performed in four ways: Vertical replacement, Horizontal replacement, Square replacement and Triangular replacement.



(a)



(b)

Figure-17(a). Approximate NRD cell. (b). Approximate non restoring divider using cell replacement technique in which exact NRD cells are replaced with approximate NRD cells. Two exact NRD cells are replaced by the approximate NRD cellsin horizontal position.

Vertical replacement is performed by replacing least significant bits of exact NRD cell by approximate cell. More number of exact cells can be replaced by approximate cells to introduce more errors into the circuit, which directly reduces the circuit complexity in turn decreases power consumption. Nevertheless, horizontal replacement is achieved by replacing all the cells of a row by approximate cells which is directly related to the accuracy of the quotient. Square replacement is a combination of both vertical and horizontal replacements. Approximate cells are introduced in place of exact cells in a triangular manner by considering the depth of the approximation.

Another way of designing approximate NRD is by completely removing the exact subtractor cell which is called as truncation scheme, to introduce error in the result. Truncation is also carried out in four ways similar to cell replacement method.

In all the proposed methods, power consumption is decreased as depth of approximation increased. Comparing truncation method, replacement techniques consumes more power. A remainder correction circuit should be employed to make the sign of the remainder reliable with dividend. This additional circuitry increases the number of circuit components hence the power consumption. This disadvantage is eliminated by using the restoring array divider.

4. RRAM (MEMRISTOR) IN APPROXIMATE COMPUTING

The previous chapter discusses approximate computing exhibited through redesigning the hardware circuits like adders, multipliers and dividers using functional approximation method. Though all these approximate arithmetic circuits are aimed to improve the performance by trading off hardware area, energy with accuracy, they are all based on conventional CMOS technology which suffers from high energy consumption. Further, these arithmetic circuits are designed to perform fixed functionality [57] which restricts further improvement in performance. As memristor [61] is identified as plausible replacement for traditional CMOS owing to its efficiency to fulfill the necessity of high performance memory and power efficient logic circuits. Memristor, due to its nonvolatile memory characteristics [62] and its CMOS-compatibility [63], it is experimented in various applications like data storage, logic computing, analog and neuromorphic computing [64-67] etc. Since it is a nano scale memory element, it is in the limelight of the researchers in experimenting approximate computing. Nevertheless only few results are reported by the researchers[68-75].

4.1 Memristor based associative approximate memory using voltage over scaling technique

Boxun Li et al. proposed a programmable RRAM based Approximate Computing Unit (ACU) along with scalable power efficient approximate computing framework in [68]. Each framework consists of RRAM processing elements as basic unit. Each processing element has digital to analog converter, approximate computing units, local memories and analog to digital converter as in Figure-18.

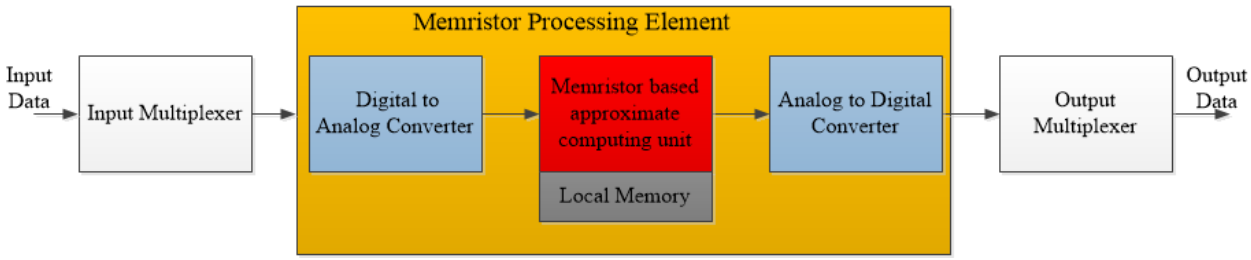


Figure-18. RRAM based approximate computing unit.

RRAM based ACU is three layer architecture acts as a neural approximator, which can be realized by matrix-vector multiplication (similar to cross bar array) and a sigmoid function [69]. Weights of the neural network are configured to the resistance states of the memristor. Since memristance can take only positive values, two cross bar arrays are needed to handle both positive and negative values of the neural network weights with the aid of inverters. RRAM processing element is controlled by two multiplexers in a round robin manner. During the processing stage, input multiplexer sends the input data to RRAM processing element for approximate computation. The output data from ACU will be given out for further processing through output multiplexer. The conversion of input analog data for processing is taken care by ADC and the vice versa by DAC. ACU is programmed in three phases: training phase to customize the approximator according to the training scheme; mapping phase to map the parameters of the first step to the corresponding resistance state of RRAM and tuning phase to tune the RRAM resistance states to target value. Amirali et al. proposed [69] Associative memristive memory for approximate computing in GPUs where Associative Memristive Memory (AMM) is built in look up table enabling computation in memory. AMM consists of TCAM designed with memristor and a Memristive memory block to store the pre computed values which are accessed often. Here approximation is achieved through voltage over scaling method. AMM module finds out the exact matching pattern from TCAM under normal operating voltage. When the degree of applied voltage is varied, it leads to the approximate (inexact match) search operation in TCAM. If the applied voltage is changed from 1V to 800 mV, the matching pattern obtained from Memristive TCAM is with hamming distance of 0, 1 or 2. This approximate matching under voltage overscaling further improves power efficiency at the cost of deviation in the expected output. Further reduction in voltage results further increase in bit error which is not tolerable. Since TCAM is proposed with memristor which is a nano scale device, decreases the required hardware area and power efficiency compared with traditional TCAM constructed using CMOS. Nevertheless, the proposed TCAM structure uses two access transistors, take lead in energy consumption. Hence access transistor free Memristive TCAM is proposed to further improve the energy consumption. However, this is limited to error tolerant applications which can be implemented with tiny associative memory. To overcome this drawback, a resistive configurable associative memory (ReCAM) is proposed in [71] by Mohsen *et al*. is similar to the one proposed by Rahimi *et al.* But here Voltage overscaling is applied only to the selective bit line or to the rows to reduce energy consumption with acceptable quality of output during run time. The adaptive selection of bit line or TCAM rows solely depends on the application type at the cost of energy overhead. Any mismatch in MSB directly affects the quality of output. All the above reports discussed so far employ Voltage overscaling technique to introduce approximation in memristor based associative

memory. Recently, [74-75] reports approximate adder/subtractor circuits designed using memristor by functional approximation through probabilistic logic minimization method. Approximation is introduced by flipping one or two output bits in adder truth table. Since approximate adder circuits are constructed with memristors, reported less hardware die area and delay thus power efficiency.

4.2 Memristor based approximate arithmetic circuits using probabilistic logic minimization technique

Recently, memristor augmented approximate adder [74-75] and subtractor [75] circuits are designed by functional approximation through logic minimization method as explained in section 3.1. In this method, approximation is introduced by flipping one or two output bits (SUM, CARRY, BORROW, DIFFERENCE) in full adder/subtractor truth table to redesign the approximate version of memristor based full adder/full subtractor circuit as shown in Figure-19.

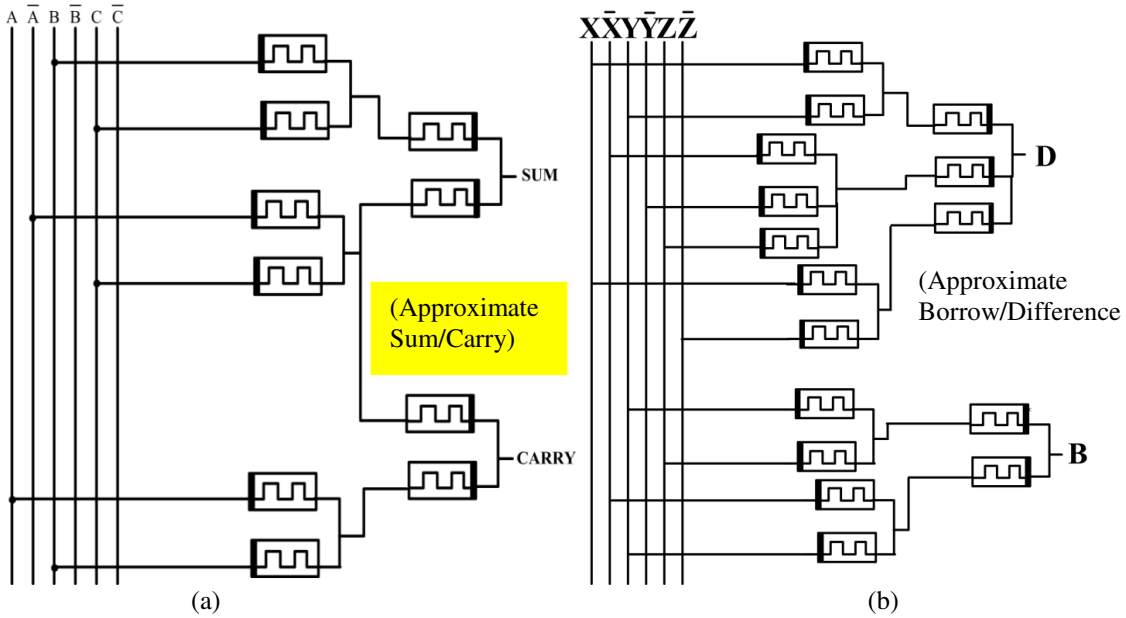


Figure-19. (a). Approximate 1-bit adder using memristors. (b). Approximate 1-bit Subtractor using memristors. Both the circuits are designed using logic minimization technique similar to section 3.1.

An 8- bit ripple carry adder and two versions of the 8-bit ripple borrow subtractors are designed using the proposed approximate single bit cells equipped with memristors. Since the approximate adder and subtractor circuits are constructed with memristors, they reported less hardware die area and delay thus power efficiency [see Table-8]. Table-9 compares the general characteristics of CMOS and Memristor.

Table-8. Delay comparison of memristor based accurate approximate adder and subtractors.

	Accurate [73, 74]			Approximate [73, 74]			
	Delay (ns)	**No. of memristors**	**No.of CMOS transistors**	**Delay** (ns)	**Error rate %**	**No. of memristors**	**No.of CMOS transistors**
Adder (1-bit)	32.56	33	0	0.2316	25	14	0
Adder (4-bits)	Not given	116	18	Not given	-	64	6
Subtractor (1-bit)	16.78	25	0	0.8(APXS1) 0.2(APXS2)	12.5 25	16 19	0 0

Table-9. Comparison on general parameters- Memristors vs Cmos.

Parameters	Memristors	CMOS
Packing density	High	Low
Power Consumption	Low	High
Memory Applications	Yes	Yes
Logic circuits	Yes	Yes
Memory Driven Computing	Yes	No
Possibility for Approximate computing	Yes	Yes
Applications reported so far in approximate computing	Image	Image and speech
Techniques explored for approximate computing	Only voltage over scaling and probability logic minimization.	Many

5. SUMMARY

Existing CMOS based approximate arithmetic circuits and its performance analysis based on error metrics is critically reviewed. Though many versions of approximate adders are designed with various design techniques, carry speculation considered to be better one because accuracy of the adders is decided during the run time. In addition, approximate adders are built with error detection and correction logic helps the user to work both in accurate and approximate mode in this carry speculation technique. Moreover, speed is also high which is attributed to both adder and EDC can work in pipelined fashion which improves throughput as well. As far as multiplier is concerned, approximate multiplier constructed using Dynamic Segment Method (DSM) decides accuracy during run time. But it suffers from high power consumption because of the additional EDC circuits. Furthermore, memristor based approximate circuits employing functional approximation to redesign the hardware circuits and associative memristive memory for approximate computing adapting voltage over scaling is also reviewed.

ACKNOWLEDGEMENT

Authors are indebted to the VIT management for extending simulation facilities to carry out this work.

REFERENCES

[1] International Technology Roadmap for Semiconductors (ITRS), http://www.itrs2.net/reports.html.

[2] Gao M, Wang Q, Nagendra ASK, Qu G. 2017. A Novel Data Format for Approximate Arithmetic Computing. In: IEEE /22nd Asia and South Pacific Design Automation Conference (ASP-DAC).pp. 390-395.

[3] Han J, Orshansky M. 2013. Approximate computing: an emerging paradigm for energy-efficient design. In: ETS'13.

[4] Venkatesan R, Agarwal A, Roy K, Raghunathan A. 2010.Macaco: Modeling and analysis of circuits for approximate computing. In: ICCAD.pp. 667-673.

[5] Venkataramani S, Chakradhar ST, Roy K, Raghunathan A. 2015. Approximate computing and the quest for computing efficiency. In: Proc.52nd Annual Design Automation Conference.

[6] Venkataramani S, Chippa VK, Chakradhar ST. 2013. Quality programmable vector processors for approximate computing. In: Proc. Int. Symp. Microarchitect.pp. 1-12.

[7] Venkataramani S, Sabne A, Kozhikkottu V, Roy K, Raghunathan A. 2012. SALSA: Systematic logic synthesis of approximate circuits. In: Proc. IEEE/ACM Design Autom. Conf.pp.796-801.

[8] Chippa VK, Chakradhar ST, Roy K, Raghunathan A. 2013. Analysis and characterization of inherent application resilience for approximate computing, In: 50th ACM/EDAC/IEEE Design Automation Conference (DAC).pp.1-9.

[9] More Than 30 Billion Devices Will Wirelessly Connect to the Internet of Everything in 2020. press release ABI Research, 9 May 2013; https://www.abiresearch.com/press/more-than-30-billion-devices-will-wirelessly-conne/

[10] T. Danova. 2013. Morgan Stanley: 75 Billion Devices Will Be Connected to the Internet of Things by 2020. Business Insider, 2 Oct. www.businessinsider.com/75-billion-devices-will-be-connected-to-the-internet-by-2020-2013-10.

[11] Lu SL. 2004. Speeding up Processing with Approximation Circuits. IEEE Transactions on Computers. 37: 67-73.

[12] Garg B, Dutt S, Sharma G.K. 2016. Bit-width-aware constant-delay run-time Accuracy Programmable Adder for error-resilient applications. Microelectronics Journal. 50: 1-7.

[13] Panahi A, Sharifi F, Moaiyeria MH, Navi K. 2016. CNFET-based approximate ternary adders for energy-efficient image processing applications. Microprocessors and Microsystems. 47:454-65.

[14] Hegde R, Shanbhag N. 1999. Energy-efficient signal processing via algorithmic noise-tolerance. In: ISLPED.pp. 30-35.

[15] Jiang H, Han J, Lombardi F. 2015. A Comparative Review and Evaluation of Approximate Adders. In: Proc. ACM/IEEE Great Lakes Symposium on VLSI; pp. 343-348.

[16] Xu Q, Kim NS, Mytkowicz T. 2015. Approximate Computing: A Survey. IEEE Des. & Test. 33: 8-22.

[17] Leem L, Cho H, Bau J, Jacobson QA, Mitra S. 2010. ERSA: Error Resilient System Architecture for probabilistic applications. In: DATE.pp. 1560-1565.

[18] Chippa VK, Raghunathan A, Roy K,Chakradhar ST. 2011. Dynamic Effort Scaling: Managing the Quality Efficiency Tradeoff. In: Proc.48th DesignAutomation Conference (DAC'11).pp. 603-608.

[19] Esmaeilzadeh H, Sampson A, Ceze L, Burger D. 2012. Architecture Support for Disciplined Approximate Programming. SIGARCH Comput. Archit. News.40: 301-312.

[20] Krause PK, Polian I. 2011. Adaptive voltage over-scaling for resilient applications. In: Design, Automation & Test in Europe Conference &Exhibition (DATE); 2011. p. 1–6.

[21] Hegde R, Shanbhag NR. 2001. Soft digital signal processing.IEEE Trans. VLSI. 9: 813-823.

[22] Hegde R, Shanbhag NR. 2004. A voltage overscaled low power digital filter IC. IEEE Jour. Solid-State Circuits. 39: 388-391.

[23] Varatkar GV, Shanbhag NR. 2008. Error-resilient motion estimation architecture. IEEE Trans. VLSI.16: 1399-1412.

[24] Christopher IA, Langley D, James CL. 2014. Inexact Computing with Approximate Adder Application. In: NAECON 2014 - IEEE National Aerospace and Electronics Conference. pp.21-28.

[25] Gupta V, Mohapatra D, Park SP, Raghunathan A, Roy K. 2011. IMPACT: Imprecise adders for low-power approximate computing. In: Proc. IEEE/ACM Int. Symp. Low-Power Electron. Design. pp. 409–414.

[26] Gupta V, Mohapatra D, Raghunathan A, Roy K. 2013. Low-power digital signal processing using approximate adders. IEEE Trans. CAD. 32: 124-137.

[27] Almurib HAF, Kumar TN, Lombardi F. 2016. Inexact Designs for Approximate Low Power Addition by Cell Replacement. In: Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE), Mar 14-18; IEEE; pp. 660-65.

[28] Yang Z, Jain A, Liang J, Han J, Lombardi F. 2013. Approximate XOR/XNOR-based adders for inexact computing. IEEE-NANO. 690-693.

[29] Mahmoud HA, Bayoumi MA. 1999. A 10-transistor low-power high-speed full adder cell. In: ISCAS'99. 1. pp. 43-46.

[30] Lin J-F, Hwang Y-T, Sheu M-H, Ho C-C. 2007. A novel high-speed and energy efficient 10-transistor full adder design. IEEE Trans. On Circuits and Systems-I: Regular Papers. 2007; 54.

[31] Mahdiani HR, Ahmadi A, Fakhraie SM, Lucas C. 2010. Bio-Inspired Imprecise Computational Blocks for Efficient VLSI Implementation of Soft-Computing Applications. IEEE Trans. Circuits Syst.-I: Regular Papers. 57:850-62.

[32] Zhu N, Goh WL, Zhang W, Yeo KS, Kong ZH. 2010. Design of Low-Power High-Speed Truncation-Error-Tolerant Adder and Its Application in Digital Signal Processing. IEEE Trans. VLSI Syst. 18: 1225-1229.

[33] Zhu N, Goh WL, Wang G, Yeo KS. 2010. Enhanced low-power high-speed adder for error-tolerant application. In: SOCC. pp. 323-327.

[34] Zhu N, Goh WL, Yeo KS. 2009. An enhanced low-power high-speed adder for error-tolerant application. In: ISIC.pp. 69-72.

[35] Zhu N, Goh WL, Yeo KS. Ultra low-power high-speed flexible Probabilistic Adder for Error-Tolerant Applications. In: SOCC; 2011. p. 393-396.

[36] Weber M, Putic M, Zhang H, Lach J, Huang J. 2013. Balancing Adder for Error Tolerant Applications. In: IEEE International Symposium on Circuits and Systems (ISCAS).pp. 3038-3041.

[37] Schlachter J, Camus V, Palem KV, Enz C. 2017. Design and Applications of Approximate Circuits by Gate-Level Pruning.IEEE transactions on very large scale integration (VLSI) SYSTEMS. 25: 1694-1702.

[38] Lingamneni A, Enz C, Palem K, Piguet C. 2011. Parsimonious circuits for error-tolerant applications through probabilistic logic minimization. Integrated Circuit and System Design. Power and Timing Modeling, Optimization, and Simulation. pp.204-213.

[39] Lingamneni, Enz C, Nagel J L, Palem K, Piguet C. 2011. Energy parsimonious circuit design through probabilistic pruning. In: Proc.Design, Autom. Test Eur. Conf. (DATE). pp. 1–6.

[40] Schlachter J, Camus V, Enz C, Palem K. 2015. Automatic generation of inexact digital circuits by gate-level pruning. In: Proc. IEEE Int. Symp.Circuits Syst. (ISCAS). pp. 173-176.

[41] Camus V, Schlachter J, Enz C. 2016. A low-power *carry cut-back* approximate adder with fixed-point implementation and floating-point precision. In: Proc. 53rd ACM/EDAC/IEEE Design Autom. Conf. (DAC). pp. 1-127.

[42] Verma AK, Brisk P, Ienne P. 2008. Variable Latency Speculative Addition: A New Paradigm for Arithmetic Circuit Design. In: Design, Automation and Test in Europe Conference and Exhibition (DATE). pp. 1250-1255.

[43] Kahng B, Kang S. 2012. Accuracy-configurable adder for approximate arithmetic designs. In: Proc. 49th Annu. Design Autom. Conf. pp.820-825.

[44] Du K, Varman P, Mohanram K. 2012. High performance reliable variable latency carry select addition. In: DATE. pp. 1257-1262.

[45] Shafique M, Hafiz R, Rehman S, El-Harouni W, Henkel J. 2016.INVITED: Cross-Layer Approximate Computing: From Logic to Architectures. In: 53nd ACM/EDAC/IEEE Design Automation Conference (DAC). pp.1-6.

[46] Jothin R, Vasanthanayaki C. 2016. High Performance Significance Approximation Error Tolerance Adder for Image Processing Applications. J Electron Test. 32: 377-383.

[47] Dutt S, Nandi S, Trivedi G. 2016. A comparative survey of approximate adders. In: 26th International Conference Radioelektronika(RADIOELEKTRONIKA). pp. 61-65.

[48] Momeni A, Han J, Montuschi P, Lombardi F. 2015. Design and analysis of approximate compressors for multiplication. IEEE Trans.Comput. 64: 984-994.

[49] Liu C, Han J, Lombardi F. 2014. A low-power, high-performanceapproximate multiplier with configurable partial error recovery. In: Proc.Conf. Design, Autom. Test Eur. Art. No. 95.

[50] Mehta A, Maurya S, Sharief N, Pranay BM, Jandhyala SS, Purini S. 2015. Accuracy-confgurable Approximate Multiplier with Error Detection and Correction. In: TENCON.

[51] Kyaw KY, Goh WL, Yeo KS. 2010. Low-power high-speed multiplier for error-tolerant application. In: EDSSC. pp. 1-4.

[52] Bhardwaj K, Mane PS, Henkel J. 2014. Power- and area-efficient Approximate Wallace Tree Multiplier for error-resilient systems. In: Proc. 15th International Symposium on Quality Electronic Design. IEEE. pp. 263p269.

[53] Narayanamoorthy S, Moghaddam HA, Liu Z, Park T, Kim NS. 2015. Energy-efficient approximate multiplication for digital signal processing and classification applications. IEEE Transactions on VLSI Systems. 23: 1180-1184.

[54] Kulkarni P, Gupta P, Ercegovac M. 2011. Trading accuracy for power with an underdesigned multiplier architecture. In: International Conference on VLSI Design. pp. 346-351.

[55] Chen L, Liu W, Han J, Lombardi F. 2015. Design of approximate unsigned integer non-restoring divider for inexact computing. Proceedings of the 25th Ed. Great Lakes Symp. VLSI, Pittsburgh, PA, USA; 2015 May 20-22; New York, USA,ACM. pp.51-56.

[56] Chen L, Han J, Liu W, Lombardi F. 2016. On the Design of Approximate Restoring Dividers for Error-Tolerant Applications. IEEE Trans. on Computers. 65(8):2522-33.

[57] Parhami. 2010. Computer Arithmetic: Algorithms and Hardware Designs, second ed. Oxford University Press, New York.

[58] Han J, Liu C, Lombardi F. 2015. An Analytical Framework for Evaluating the Error Characteristics of Approximate Adders. IEEE Transactions on Computers. 64:1268-1281.

[59] Liang J, Han J, Lombardi F. 2013. New Metrics for the Reliability of Approximate and Probabilistic Adders. IEEE Transactions on Computers. 62: 1760-1771.

[60] Mittal S. 2016. A survey of techniques for approximate computing. ACM Comput. Surv. 48: 33.

[61] Strukov DB, Snider GS, Stewart DR, Williams RS. 2008. The missing memristor found. Nature. 453: 80-83.

[62] Zidan MA, Fahmy HAH, Hussain MM, Salama K N. 2014. Memristor-basedmemory: the sneak paths problem and solutions. Microelectron. J. 44: 176-183.

[63] Kvatinsky S, Wald N, Satat G, Kolodny A, Weiser UC, Friedman EG. 2012. MRL memristor ratioed logic. In: 13th International Workshop on Cellular Nanoscale Networks and Their Applications; 2012. p. 1–6.

[64] Ranjan RK, Bhuwal N, Raj N, Khateb F. 2017. Single DVCCTA based High frequencyincremental/decremental Memristor Emulator and its Application. AEU-Int. J Electron Commun. 82:177-90.

[65] Babacan Y, Kaçar F. 2017. Memristor emulator with spike-timing-dependent-plasticity. AEU-Int. J. Electron Commun. 73:16-22.

[66] Wang Y, Liao X. 2017. Stability analysis of multimode oscillations in three coupled memristor-based circuits. AEU-Int. J. Electron Commun. 70(12): 1569-79.

[67] Tarkhan M, Maymandi-Nejad M. 2018. Design of memristor based fuzzy processor. AEU-Int J Electron Commun. 84: 331-341.

[68] Li B, Gu P, Shan Y, Wang Y, Chen Y, Yang H. 2015. RRAM based analog approximate computing. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 34:1905-1917.

[69] Ghofrani A, Rahimi A, Lastras-Montano MA, Benini L, Gupta RK, Cheng KT. 2016. Associative Memristive Memory for Approximate Computing in GPUs.IEEE JETCAS, Special Issue On Emerging Memories - Technology, Architecture & Applications. 6(2):222-34.

[70] Rahimi A, Ghofrani A, Cheng KT, Benini L, Gupta RK. 2015. Approximate Associative Memristive Memory for Energy-efficient GPUs. In: Proc. Design, Automation & Test. In Europe Conference & Exhibition. pp. 1497-1502.

[71] Imani M, Rahimi A, Rosing T. 2016. Resistive Configurable Associative Memory for Approximate Computing. In: Proc. Design, Automation & Test in Europe Conference (DATE).pp. 1327-1332.

[72] Li B, Shan Y, Hu M, Wang Y, Chen Y, Yang H. 2013. Memristor based approximated computation. In: Low Power Electronics and Design (ISLPED). pp. 242-247.

[73] Imani M, Kim Y, Rahimi A, Rosing T. 2016. ACAM: ApproximateComputing Based on Adaptive Associative Memory with Online Learning.In: International Symposium on Low Power Electronics and Design (ISLPED).pp. 162-167.

[74] Muthulakshmi S, Dash CS, Prabaharan S R S. 2018. Memristor-Based Approximate Adders for Error Resilient Applications. In: Nanoelectronic Materials and Devices. Lecture Notes in Electrical Engineering. 466: 51-59.

[75] Muthulakshmi S, Dash CS, Prabaharan S R S. 2018. Memristor augmented approximate adder and subtractors for image processing Applications: an approach. AEU-Int J Electron Commun.91; 91-102.