Al-Maaitah K, Tarawneh G, Soltan A, Qiqieh I, Yakovlev A.

Approximate Adder Segmentation Technique and Significance-Driven Error Correction.

In: 27th International Symposium on Power And Timing Modeling, Optimization and Simulation (PATMOS).

25-27 September 2017, Thessaloniki, Greece: IEEE.

Copyright:

© 2017 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

DOI link to article:

<https://doi.org/10.1109/PATMOS.2017.8106986>

Date deposited:

11/12/2017

Approximate Adder Segmentation Technique and Significance-Driven Error Correction

Khaled Al-Maaitah, Ghaith Tarawneh, Ahmed Soltan, Issa Qiqieh, Alex Yakovlev
School of Electrical and Electronic Engineering, Newcastle University, United Kingdom
{k.almaaitah, ghaith.tarawneh, Ahmed.abd-el-aal, I.Qiqieh1, Alex.yakovlev}@newcastle.ac.uk

Abstract—Approximate computing introduces a new era of low-power and high-speed circuit designs. Instead of strict accurate computation, relaxed requirements might increase performance and reduce power consumption with a simplified or inaccurate circuit. One of the recent remarkable research efforts is the accuracy-configurable approximate adder designs, which can gracefully operate in both approximate (inaccurate) and accurate modes. In this paper, a novel technique for segmenting approximate adders was proposed by adding new bit locations that exploit the carry kill signal definition to limit carry propagation at specific locations. Moreover, a light-weight carry-in prediction and error detection techniques were proposed. For error recovery circuit, a significance-driven configurable correction stages were implemented, which imply a fast convergence to exact outputs with a very low magnitude of errors. The proposed design showed improvements of (16%) and (18.6%) for dynamic power and area respectively. Nevertheless, outputs reserved a general high accuracy level, which limited between 99% and 100% for the majority of input space. The proposed design was implemented in an image filter application, which resulted in high PSNR values of (53 and 83 db) for the two premier correction stages, and 100% exact results for the highest accuracy mode.

I. INTRODUCTION

Approximate computing is an emerging design paradigm, which trades off the strict correctness of conventional computations for more performance and energy efficiency of the digital computer system. The main goal of approximate computing is to enhance design parameters metrics such as execution speed and power consumption, meanwhile allowing computing errors to occur within acceptable frequency and magnitudes. As a result, this would lead to introduce high performance and lower power circuits and increased density systems [1]–[5].

Applications from domains such as image and video processing and machine learning can tolerate low magnitude errors in their arithmetic operations. This is due to applications' inherent resilience for approximation errors, which does not yet impact on the end user experience. A few factors might account for the resilience of these applications, such as perceptual limitations of application users, noise and redundancy existence in the real-world input data, in addition to error attenuation characteristics of processing algorithms used in these applications. Thus, despite approximation errors existence, an extensive portion of computations are still able to produce an output of acceptable quality [6], [7]. On the other hand, other domains like biomedical applications might tolerate higher levels of errors and leverage the high speed

and low power approximated image processing circuits, for example, for stimulating human neurones or retina [8].

Essential Arithmetic computing units in digital circuits like Adders have been investigated in the context of approximate computing. Conventional adders have a common problem regarding long carry propagation chains (which is defined as the number of consecutive propagate signals with value (1)). It is known that the glitches caused by carry propagation are considered as a key reason of consuming a large proportion of power [9], [10]. However, considering random uniformly input distribution, these long carry chains are rarely activated and usually much shorter than the full width of the adder [11]. Hence, this results in a kind of motivation to start re-examining the existing adder designs to introduce different approximate versions with higher speed and lower power consumption [12], [13].

Approximate adders divide addition into separated or overlapped smaller width sub-blocks. This, in turn, allows operating in parallel for higher speed and energy efficiency; however, with an existing chance of generating incorrect results [14]. By grouping input bits into blocks, the length of the carry chain can be comparable to the block size with high probability (i.e. lower chance of errors) [6], [11]. Each sub-adder produces a number of resultant bits that contribute to the final summation and makes use of overlapped bits to predict carry propagation. Consequently, approximate adders might be roughly categorised into three techniques: the first one proposed the use of multiple overlapping sub-adders with one resultant bit per sub-adder to the final sum [4], [10], [14]. The second technique divided addition into multiple blocks with overlapping parts. Each block is responsible for generating a range of bits to the final sum [9], [11], [14]–[17]. The proposed design in this paper follows the third technique, in which each sub-adder results in sum bits number equal to its full bit-width as can be found in [11].

In order to handle the requirements of multiple levels of outcomes accuracy, reliable approximate adder designs have been introduced. These designs have an augmented error recovery circuit, which is triggered to correct the result after detecting an erroneous output [4], [6], [11], [15]. However, minimizing the error rate (ratio of incorrect outputs) without significant delay, area and power degradation becomes a common design problem [14]. Remarkably, the configurable-accuracy designs are introduced in [6], [7], [15], [17], where controlled multi-stage of error correction are used to mitigate this challenge and

to get the flexibility to imply different accuracy levels during run-time. Multi-stage error correction mechanism allows the designer to limit the delay of error correction and the ratio of consumed power. This is done by controlling the activation of the number of correction stages (accuracy level), and limiting the error checks number. The more the pipelined stages, the smaller the carry chain length of the design sub-blocks and the more performance achieved [15]. Nevertheless, these designs still show a large area and power overhead, and do not guarantee 100% correctness at the final correction stage.

This paper has the following three contributions which significantly mitigate the design overhead of configurable-accuracy designs:

- 1) A new segmenting technique of sub-adders.
- 2) Light-weight carry-in prediction and error detection techniques, which lead to more scalability for large adder sizes with lower design overhead.
- 3) Significance-driven multi-stage error recovery circuit with fast convergence to exact outputs and 100% accuracy at the final correction stage.

The rest of the paper is organised as follows. Section II presents the motivation of this effort as well as the parts of the proposed design. In section III the experimental results and analysis are provided. Section IV concludes the paper.

II. PROPOSED DESIGN

In general adder carry chain, the carry kill signal shown in (1) would participate with a vital role in limiting the carry propagation and then the critical path delay. Hence, in this effort, we exploited this characteristic in order to divide the adder into smaller sub-adders, and furthermore to make the real carry detection at the same time. This new technique would lead to smaller area and lower power consumption overhead, which is still an emerging design point.

$$Carry_Kill\ Signal_{(j)} = SUM_j[0 + 0] + Carry_{(j-1)}; \quad (1)$$

In this section, three parts of the proposed design have been presented. Part II-A shows the main idea and architecture of adder segmentation. Carry-in prediction technique to each segmented sub-block is introduced in part II-B, and finally, the error detection process and the significant-driven structure of the correction stages are placed at the third part II-C.

A. Segmenting Technique

The proposed design is based on dividing the adder into smaller sub-adders. Then, one bit location is added after each sub-adder to limit the long carry chain as depicted in the general form in Fig. 1. Hence, for an N bit adder, the number of segments should be M where $(M = N/K)$; K is the sub-adder bit width $(K=L-1)$. The new bit locations will be used for both segmentation and holding the real carry-out of each segment. Furthermore, the value of the real-carry at each bit location will be used in the process of error detection and correction of each sub-adder which resulted erroneous SUM value.

Fig. 1: Proposed segmentation technique using carry kill bit locations.

B. Carry prediction Technique

Carry-in prediction for each sub-adder is shown in Fig. 2. The predicted carry of sub-adder (i) would be equal to the generate signal (G) of the most significant bits of previous sub-adder (i-1). A similar technique was used before as in the lower-part-OR adder [18], but with completely different approximate adder architecture. In our design, as the carry-in of the first sub-adder is truncated to '0', the carry-in bit is generated as follows:

$$Carry_in_{(i)} = G_{i-1}^{MSB} = A_{(i-1)}^{MSB} \& B_{(i-1)}^{MSB} \quad (2)$$

An example of (32-bit) approximate adder uses the proposed segmenting technique is shown in Fig. 3, and the following points summarise its main parts:

- Number of sub-adders equals to $M = N/K = 32/8 = 4$.
- The length of each segment (sub-adder) is increased by one additional bit location in order to limit the carry propagation and hold the real carry-out of the adder segment. Hence, L equals to $(K+1) = 9$ bits.
- The carry-in of each sub-adder is predicted to be equal to the AND_{ing} result of the most significant input bits of the previous sub-adder, except the first sub-adder that has been truncated carry-in = 0,
- Each bit in sub-adder participates in one SUM bit in the final approximated SUM output value. However, the carry kill bit location value is not considered as a SUM bit and will be discarded.
- The final carry kill bit location at sub-adder(4) is considered as the final carry-out of the whole adder.
- The length of the sub-adders can be configurable at the design time depending on the application requirements.

C. Error Detection and Correction

Real Carry of each sub-adder would be handled by the added Carry Kill bit location. Fig. 4 shows that for error detection at each prediction circuit, one XOR gate is used, and the error signal will be high if both predicted and real carry (in the carry kill bit location) are not equal as presented in (3). This means that error will signal high when there is

Fig. 2: Proposed Carry-in prediction technique for each sub-adder.

Fig. 3: Example of 32-bit approximate adder uses the proposed carry kill bit locations segmenting technique.

a carry propagation from bits less than the used MSBs for prediction. This case will happen just once when the predicted carry-in is equal to '0' and the real carry-out of previous adder is equal to '1'. Table I, presents the inputs combinations and the probability of error when there is a carry propagation.

$$Error_{(i)} = G_{MSB(i-1)} \wedge Carry\ Kill\ bit\ value_{(i-1)}(3)$$

Fig. 4: Proposed error detection technique augmented with each carry-in prediction circuit.

Error correction stages use incrementors that have the same width of sub-adders output sum; they are organised depending on the significance (priority) of the errors as presented in Figure 5 (i.e. the most significant segment will be corrected first to reach high convergence with the correct result with small delay and power). On the other hand, the least significant segmented part will be corrected last and only in the full correct mode. This structure of arranging the correction stages is similar to what exist in [7]; However, the proposed design second version denoted by (Proposed_Accurate) will include a new error correction technique modification, where the carry-out of each correction stage would not be overlooked in case its value was high. The high value of correction stage carry-out has to be propagated to correct the successive sub-adder output sum. This modification should guarantee the full accuracy of outputs at the final correction stage.

From Fig. 5, the following points can be noticed:

- 1) The Approximated adder gives the approximated SUM at each stage.
- 2) (S0) is always correct as it uses truncated (not predicted) carry-in = '0'.
- 3) The correction stage (incrementor) will result the accurate SUM part (coloured green).

Accuracy mode circuit is used to specify which number of

Fig. 5: Significance-driven fast convergence structure of multi error correction stages.

incrementors needs to be activated for correction when the error signal is high. Consequently, the power consumption would be controlled by turning off unused correction stages. The proposed error detection and correction mechanism has lower overhead, for instance, in the (32-bit) adder example in Fig. 3, it is obviously shown that four segmented sub-adders will need to make just three error checks (as the first (LSB) sub-adder is always correct), in opposite to six error checks have to take place for the same adder bit length in ACA [15]. In addition to the use of one incrementor circuit to correct the erroneous 8-bit sum, then, three incrementors would be needed for the whole correction process. For full accuracy, the new carry-out of each correction stage will not be overlooked and has to be checked for propagation, hence, this would indicate if the successive correction stages output sums also need for correction. Algorithm 1 describes the whole parts of the proposed design while using the (Proposed_Accurate) version (correction stages carry-out considered), which guarantees 100% accuracy at the final correction stage.

Algorithm 1 Proposed Design algorithm

```

1: procedure :FULL ACCURACY PROPOSED DESIGN
2: For Adder of Length (N), Begin
3: Input K; \\Number of Bits in each Sub-adder.
4: Integer i = 2;
5: Integer J = 0;
6: assign L = K + '1' ; \\The total length of each Sub-adder after adding one bit
   location to K.
7: assign M = N/K; \\The total number of segmented Sub-adders.
8: Predicted Carry-in(i) = GMSB(i-1); \\Predicted Carry-in of current sub-adder =
   Generate signal of MSB(s) of previous adder.
9: Sum(sub-adder[1]) = A [K:0] + B [K:0] + (0); \\Carry-in to the first sub-adder
   is truncated to '0'.
10: Error Detection and correction
11: for (J=0; J = (M-1) ; J=J+1) do
12:   \\Error Detection at the current Sub-adder.
13:   if (GMSB (i-1) != Bit Value [K+1](i-1)) then
14:     Error(i) = True;
15:   \\Error Correction for current Sub-adder Sum with error.
16:   Corrected Sum(i) = Approximated Sum(i) + ('1');
17:   \\Final corrected Sum value of current Sub-adder.
18:   Final Sum(i) = Corrected Sum(i);
19:
20:   \\Checking the carry-out of current correction stage.
21:   if (Carry-out of correction stage(i) = ('1')) then
22:     \\Carry propagation to the successive Sub-adder
23:     \\Sum value ( it could be Approximated or
24:     \\pre-corrected)
25:     Corrected Sum(i+1) = Sum(i+1) + ('1');
26:   \\Final corrected Sum of successive Sub-adder.
27:   Final Sum(i+1) = Corrected Sum(i+1);
28: end if
29:
30: \\End of correction stage at sub-adder(i)
31: else
32:   The case when there is no error at sub-adder(i)
33:     Error(i) = False;
34:     Final Sum(i) = Approximated Sum(i);
35:     i=i+1; \\Move to the next adder error check.
36: end if =0

```

TABLE I: One-bit inputs proposed design probability of carry prediction plus error detection and correction.

A	B	Predicted Carry	Real Carry	Error	SUM Correction
0	0	0	0	NO	NO
0	1	0	1	YES	SUM + 1
1	0	0	0	NO	NO
1	1	1	1	NO	NO

III. RESULTS AND DISCUSSION

This section has five parts; part III-A describes the experiment setup methodology, part III-B has the main designs parameters such as power, area and delay evaluation of 32-bit adder example. The error analysis results comparisons are presented in part III-C. Part III-D has the PSNR result values of implementing the proposed design in an image filter application. Finally, the design implementation in adders with large bit widths is presented in part III-E.

A. Experimental Setup

Verilog was used to build (32-bit) different adder designs with their different correction stages. Testbenches were used to test the functionality of each design with different accuracy modes. For the part of comparison, Modelsim was used for error analysis simulations, which based on Monte Carlo method for generating random input values for one million iterations (ten thousand iterations were used to simplify presenting the distribution values). Synopsys Design compiler exploited UMC (Faraday) 90nm technology to synthesize and evaluate the design parameters such as delay, power and area values.

B. Design Parameters Evaluation

In order to make hardware evaluation, the proposed design is compared to the design effort in ACA [15]. The proposed design has two versions, where the first version was applied without considering the carry-out of correction stages, and the second version considers the carry-out of each active correction stage regarding the selected accuracy mode. For simplifying, the design version considering the correction stage carry-out was denoted as (Proposed_Accurate).

From Fig. 6, it can be shown that the proposed design behaves better than the Accuracy Configurable Adder (ACA) [15] in terms of design parameters such as Dynamic Power Fig. 6 (b) and Area Fig. 6 (c). These enhancements referred to not using any overlapped (redundant) parts of the addend inputs, besides the light design weight of error detection circuit. As a result, the proposed design introduces smaller area, and then lower power consumption. For Delay values in Fig. 6 (a), the proposed design has larger values with limited range compared to the ACA design, this due to the use of the carry prediction technique with AND gates, and the increased length of each sub-adder with one bit location. In addition to (8-bit) length incrementors used for correction (instead of 4 bits length as in ACA) that would consume more execution time. However, this proposed design version (i.e. not Proposed_Accurate), shows more stability regarding delay values through all correction stages. This, in turn, presents the independence characteristic of each sub-adder in which

TABLE II: Average Reduction Ratio Values of the proposed design compared to ACA design for all correction stages.

Parameter	Proposed Design vs ACA
Dynamic Power	16%
Leakage Power	17.2%
Area	18.6%
Delay	-16%

the critical path delay is the same for all segmented blocks; in contrast to ACA that depends on kind of memory for the middle carry of the previous sub-adder. On the other hand, in the case of the Proposed_Accurate design version (correction stages carry-out in concern), it reaches the full accuracy in the highest correction mode. Nevertheless, it continues to behave better in terms of power consumption and area when compared to ACA for all stages.

Regarding the analysis of the reduction ratios of the proposed design compared to the ACA design. Although the negative ratios of the delay values, other ratios show that the proposed design has remarkable positive reductions values of (17%) and (20%) for power and area respectively. Table II provides the average values of reduction ratios resulted from the proposed design. obviously, significant improvements are introduced in terms of power (dynamic and leakage) and area values for all stages of the design. On the other hand, although a very small degradation of the delay happened for the proposed design, it still shows higher speed when compared to conventional adder like Ripple Carry Adder.

C. Error Analysis

Approximate designs error characteristics drive a great attention in previous efforts such as in [19], [20]. However, in this paper, the error analysis was made to show the relative error distance (RED) distribution of each design, which simply measures how far the significance of error of the resulted outputs of the proposed design versions (Proposed and Proposed_Accurate) and the ACA [15] adder design when compared to the exact outputs from a conventional correct adder. Despite the simplicity of the this measurement, it would show the effect of the proposed design stages regarding the final quality of the outputs.

The following equation shows the arithmetic expression of the RED value.

$$RED = \frac{|\text{Correct output} - \text{Approximated output}|}{\text{Correct output}} \quad (4)$$

For clarifying, an example when the RED value equals '0', then the approximated output value is correct, and there is no difference between it and the conventional adder exact value. However, if RED value equals '0.01', then the approximated output value is not fully correct, and there is a difference between its value and the exact value by the percentage of 1% (i.e. it has 99% of accuracy).

Fig. 7 (a) presents the case of designs without any correction stages, it can be shown that the proposed design has an acceptable range of outputs with no errors (more than 40% of the tested space), and approximately 55% with a very limited magnitude of error (50% with 99% and 5% with 98% of accuracy). The last 5% of the tested inputs space of the proposed design behaves the same like ACA which lie on

Fig. 6: ACA vs proposed design in the case of 32-bit adder: (a) delay, (b) dynamic power, (c) leakage power, (d) area.

Fig. 7: The probability distribution of REDs in 32-bit proposed adder: (a) no correction stages, (b) one stage, (c) two stages, (d) three stages.

different RED values. However, it has more outputs number with smaller error magnitude.

Fig. 7 (b) shows the error analysis of designs with one correction stage. It can be noticed that our design versions (Proposed and Proposed_Accurate) have more stability in terms of RED values in which they started to be limited strictly between (RED = 0 , 57%) and (RED = 0.01 , 43%) values, in contrast to ACA which still shows different values of RED.

In the case of two correction stages in Fig. 7 (c), the proposed design two versions and ACA have improved the ratio of the fully correct output values; however, our design versions still show more general acceptable results as they limited between 100% and 99% of accuracy, in contrast to ACA that still owns scattered values of RED. Finally, at Fig. 7 (d), the case of three (full) active correction stages (worst case of accuracy level) is presented. it is obviously shown that the behaviour of the proposed design versions show much better accurate results compared to ACA design, especially, the guarantees of 100% correct results in the Proposed_Accurate version. The improved error detection mechanism of the Proposed_Accurate design version by considering the carry propagation of the correction stages, shows the best result in the case of three correction stages; However, although the degradation of its delay value, it still has higher speed than conventional adder like Ripple Carry Adder (RCA) and has much better values of design parameters such as power and delay when compared ACA [15].

D. Implementation Test

For implementation testing, Gaussian blur image filter was used to check the actual behaviour of the proposed design during multiple correction stages. Matlab was used to design the filter which involves the convolution of image kernel described by a Gaussian function, with the pixels of the image.

The new values of a given pixel are calculated by multiplying each kernel value by the corresponding input image pixel values, then all the obtained values are added, and the result will be the value of the current pixel that is overlapped with the centre of the kernel [21]. The Proposed_Accurate version adder of (20-bits) width was implemented. The peak signal to noise ratio (PSNR) is used to measure quality of the output images after applying Gaussian blur filter. The PSNR results in Fig. 8 confirm the advantage of this design version. It shows high PSNR magnitude values, especially starting from stage one of correction with more than 53 db. Moreover, stage two of correction made a well noticed jump and reached a very high value of PSNR=83.6 db. Remarkably, when the implemented design operates at the full accurate mode, it guarantees the same accuracy as the original picture (accurate computations). On the other hand, although the appearance of low PSNR value of the proposed design without correction stages, it might be considered as an attractive adder design for some application like the Biomedical applications, which generally interested in high speed, very low power and acceptable outputs quality.

Fig. 8: Gaussian blur Image Filter Test.

Fig. 9: ACA vs proposed design when increasing the size of the adder: (a) delay, (b) dynamic power, (c) leakage power, (d) area.

E. Large Bit Width Adders Evaluation

For design scalability checking, a further hardware evaluation was implemented for different adder designs (ACA and Proposed versions) with larger bit widths (64-bits, 128-bits and 256-bits). Values from each design full correction stage architecture (i.e. using Three correction stages) are shown in Fig. 9. It can be noticed that the proposed design versions (Proposed and Proposed_Accurate) keep the reduction ratio values in terms of dynamic power Fig. 9 (b), leakage power Fig. 9 (c), and area in Fig. 9 (d). Furthermore, these values start to increase as the length of the adder becomes larger. Fig. 9 (a) apparently shows that the delay degradation of the proposed design (the version without considering the correction stages carry-out) compared to ACA becomes smaller as larger bit-width adders are in use. These results show that the percentage of the large adder designs reduction values can clearly confirm the scalability advantage of the proposed designs. As a result, it can be concluded that the proposed design versions would be adaptive for using in very large bit width adders with acceptable overhead.

IV. CONCLUSION

In this paper, a novel segmentation technique has been proposed for designing configurable-accuracy approximate adder with low power and area requirements. The concept of Carry propagation kill signal was used to introduce a new bit location that can be exploited for both dividing conventional adder into a number of sub-blocks, and holding the real carry of each sub-adder. This new architecture of segmented sub-adders was augmented with light weight carry-in prediction and error detection circuits. For error correction, a significance-driven multi-stage structure was used, while considering the carry-out of each active stage. Thus, this would guarantee full accuracy at the final correction stage. The proposed design presented average reduction ratios of (16%), (17.2%) and (18.6%) for dynamic power, leakage power and area respectively. For error analysis, the results showed fast convergence to exact results at premier correction stages, and the increased stability of output accuracy levels between (99% and 100%) through all correction stages. The proposed design results were confirmed by a real-time implementation using image filter application with high PSNR results, and 100% similar outputs as original filter when the proposed design uses full correction stages. Future work will include using this design in complete circuit

(including control and memory aspects), and other applications like multipliers and DSP or Biomedical applications that pay huge concern for low power and acceptable quality outputs designs.

REFERENCES

- [1] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *ETS*, 2013, pp. 1–6.
- [2] S. Venkataramani *et al.*, "Approximate computing and the quest for computing efficiency," in *DAC*, 2015, pp. 120:1–120:6.
- [3] H. Jiang *et al.*, "A comparative review and evaluation of approximate adders," in *Great Lakes Symposium on VLSI*, 2015, pp. 343–348.
- [4] A. K. Verma *et al.*, "Variable latency speculative addition: A new paradigm for arithmetic circuit design," in *DATE*, 2008, pp. 1250–1255.
- [5] L. Sekanina and Z. Vasicsek, "Approximate circuit design by means of evolvable hardware," in *ICES*, 2013, pp. 21–28.
- [6] M. Shafique *et al.*, "A low latency generic accuracy configurable adder," in *DAC*, 2015, pp. 86:1–86:6.
- [7] V. Benara and S. Purini, "Accurus: A fast convergence technique for accuracy configurable approximate adder circuits," in *ISVLSI*, 2016, pp. 577–582.
- [8] W. Al-Atabany *et al.*, "A processing platform for optoelectronic/optogenetic retinal prosthesis," *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 3, pp. 781–791, 2013.
- [9] N. Zhu *et al.*, "An enhanced low-power high-speed adder for error-tolerant application," in *ISIC*, 2009, pp. 69–72.
- [10] A. A. D. Barrio *et al.*, "Applying speculation techniques to implement functional units," in *ICCD*, 2008, pp. 74–80.
- [11] K. Du *et al.*, "High performance reliable variable latency carry select addition," in *DATE*, 2012, pp. 1257–1262.
- [12] D. Esposito *et al.*, "Variable latency speculative han-carlson adder," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 5, pp. 1353–1361, 2015.
- [13] O. Akbari *et al.*, "Rap-cla: A reconfigurable approximate carry look-ahead adder," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. PP, no. 99, pp. 1–1, 2016.
- [14] G. Liu *et al.*, "Casa: Correlation-aware speculative adders," in *ISLPED*, 2014, pp. 189–194.
- [15] A. B. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs," in *DAC*, 2012, pp. 820–825.
- [16] I. C. Lin *et al.*, "High-performance low-power carry speculative addition with variable latency," *IEEE Transactions on VLSI Systems*, vol. 23, no. 9, pp. 1591–1603, 2015.
- [17] R. Ye *et al.*, "On reconfiguration-oriented approximate adder design and its application," in *ICCAD*, 2013, pp. 48–54.
- [18] H. R. Mahdiani *et al.*, "Bio-inspired imprecise computational blocks for efficient vlsi implementation of soft-computing applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 4, pp. 850–862, 2010.
- [19] C. Liu *et al.*, "An analytical framework for evaluating the error characteristics of approximate adders," *IEEE Transactions on Computers*, vol. 64, no. 5, pp. 1268–1281, 2015.
- [20] S. Mazahir *et al.*, "Probabilistic error modeling for approximate adders," *IEEE Transactions on Computers*, vol. 66, no. 3, pp. 515–530, 2017.
- [21] I. Qiqieh *et al.*, "Energy-efficient approximate multiplier design using bit significance-driven logic compression," in *DATE*, 2017, pp. 7–12.