An Area-Efficient Consolidated Configurable Error Correction for Approximate Hardware Accelerators

Sana Mazahir
School of Electrical Engineering & Computer Science, National University of Sciences and Technology
Islamabad, Pakistan
sana.mazahir@seecs.nust.edu.pk

Osman Hasan
School of Electrical Engineering & Computer Science, National University of Sciences and Technology
Islamabad, Pakistan
osman.hasan@seecs.nust.edu.pk

Rehan Hafiz
Electrical Engineering Department, Information Technology University
Lahore, Pakistan
rehan.hafiz@itu.edu.pk

Muhammad Shafique
Chair for Embedded Systems
Karlsruhe Institute of Technology, Germany
muhammad.shafique@kit.edu

Jörg Henkel
Chair for Embedded Systems
Karlsruhe Institute of Technology, Germany
henkel@kit.edu

ABSTRACT

Approximate adders are widely being advocated for developing hardware accelerators to perform complex arithmetic operations. Most of the state-of-the-art accuracy configurable approximate adders utilize some integrated Error Detection and Correction (EDC) circuitry. Consequently, the accumulated area overhead due to the EDC (integrated within individual adders) is significant. In this paper, we propose a low-cost Consolidated Error Correction (CEC) unit, that essentially corrects the accumulated error at the accelerator output. The proposed CEC is based on a mathematical model of approximation error. We integrate our CEC unit in approximate hardware accelerators deployed in different applications to demonstrate its area savings and speed enhancement compared to state-of-the-art.

1. INTRODUCTION AND RELATED WORK

Approximate computing [1] is an emerging trend in hardware design that aims to exploit the intrinsic error tolerance in many applications to gain benefits in terms of speed, power and silicon area. In computationally intensive applications like data mining, image/video processing, machine learning, approximation errors may be adequately tolerable [1–3]. This tolerance can be attributed to many factors, like the presence of noise, perceptual limitations of the user and inherent redundancy in the data.

Adders and multipliers are the most foundational components in any circuit involving arithmetic operations. Hence the design of efficient and fast approximate adders has gained great interest. Several functionally approximate designs for adders have been proposed [4–9]. *In many practical applications, circuits with many approximate adders, connected in cascade, are used to form complex data paths.*

In order to add flexibility in obtaining various configurations according to the application requirements, authors in [4, 5, 7, 8] have proposed to integrate error detection and correction (EDC) units with the approximate adders. In [8], the variable latency speculative addition (VLSA) is proposed in which an implementation similar to the carry look-ahead adder is employed for error recovery, which is achieved at the cost of increased latency of the adder. Thus, when error is detected, one extra clock cycle is required to produce the corrected sum. In the accuracy configurable adder (ACA-II), proposed in [4], incrementor units, with every sub-adder, are used to compute the correct sum, that requires an additional clock cycle. Recently, in the generic accuracy configurable (GeAr) [5], error correction is implemented by changing the inputs to the sub-adders and re-evaluating the partial sums. In the gracefully degrading adder (GDA) in [7], multiplexers are provided to selectively turn on/off the carry prediction so that different levels of accuracy can be achieved.

It is noteworthy that in all the four cases described above, *a generic implementation of error correction/recovery unit is implemented.* This choice inevitably increases both area and latency of the system. Recently, [10] introduced several new research challenges for the approximate computing, including error analysis for approximate circuits and accelerators. For the high-performance adders under consideration in this paper, we found that *the magnitude of error in the approximate adders under consideration can only have certain specific values.* **Novel Contributions:** 1) Based on this observation, in this paper, we propose a novel accuracy configuration unit, in which we exploit the prior knowledge about the magnitude of error to gain improvement in terms of implementation complexity and timing. 2) We first study the distribution of error magnitude in relationship with the occurrence of error in individual sub-adders. 3) Then a simple error recovery scheme is implemented by applying a predetermined correction to the computed approximate sum in case of an error. The functionality of the proposed design is verified by applying it to some most commonly used adder

DAC '16, June 05-09, 2016, Austin, TX, USA

DOI: http://dx.doi.org/10.1145/2897937.2897981

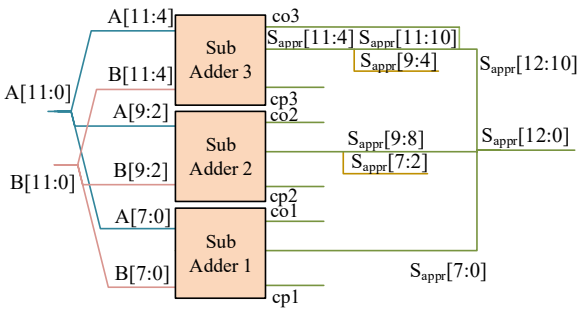

Figure 1: GeAr (12,2,6)

configurations, including ACA-II [4] and GeAr [5]. 4) Since in most practical applications, more than one addition is performed as a part of more complex arithmetic operations, so building upon the same observation that error can have certain specific values, *we propose to integrate the error correction circuitry of individual approximate adders into one consolidated block for error correction unit at the end.* This is demonstrated to significantly reduce the required area and thereby power consumption for providing accuracy configurability. The time delay introduced by the error recovery is also shown to be reduced.

2. AN OVERVIEW OF APPROXIMATE ADDERS

Recent works on approximate adders include the Almost-Correct Adder (ACA-I) [8], Accuracy Configurable Adder (ACA-II) [4], Gracefully Degrading Adder (GDA) [7], Error Tolerant Adder ETA-II [6] and Generic Accuracy Configurable (GeAr) [5] adder. In all of these designs, enhancement in the speed of computation by approximation is achieved by truncating the carry chain. This is done by dividing the input bits among multiple, disjoint or overlapping precise sub-adder units. The basic idea in this type of adders is that for most of the input combinations, the carry chain propagation is shorter than the complete length of the adder. So if this chain is broken, then the output will be erroneous for a very small fraction of the input combinations. Interestingly, all the adders mentioned above, including ACA-I, ACA-II, GDA and ETA-II fall into the design space of the GeAr model [5]. In other words, ACA-I, ACA-II, GDA and ETA-II, are different configurations of GeAr. Therefore, for the rest of the paper, we consider the most recent and advanced GeAr model for approximate adders.

2.1 The GeAr Model

An N-bit GeAr adder is configured by specifying the following parameters:1) Number of sub-adder units k, 2) Number of prediction bits P in each sub-adder, 3) Number of sum bits R from each sub-adder and 4) the bit width $L = R + P$ of each sub-adder.

The GeAr adder with three sub-adders, for 12-bit addition, is shown in Fig. 1. Any GeAr configuration will be specified using the notation GeAr(N, R, P). Libraries for these adders are available online, downloadable at [11].

The output of the GeAr adder is erroneous when following events occur simultaneously:

- All the P prediction bits of a sub-adder are propagating carry-in. In Fig. 1, this event is detected by the cp_i signal in the i^{th} sub-adder, where $cp_i = \bigwedge_{j=0}^{P-1} A_i[j] \oplus B_i[j]$. A_i and B_i are inputs of i^{th} sub-adder.
- The previous less significant bits, that are not included in the input to this i^{th} sub-adder, are generating carry out co_{i-1}.

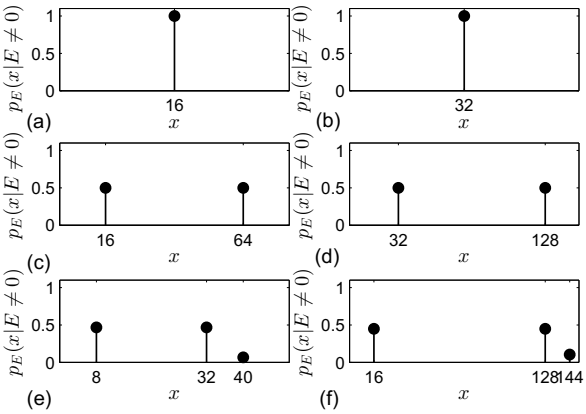

Figure 2: Error distribution in (a) GeAr(6,2,2), (b) GeAr(8,3,2), (c) GeAr(8,2,2), (d) GeAr(9,2,3), (e) GeAr(7,2,1), (f) GeAr(10,3,1)

The error occurs because the carry-out generated by the previous less significant bits is not propagated to the next bit due to the broken carry-chain between the sub-adders.

2.2 Requirement of Accuracy Configurability

The requirement for accuracy can vary from application to application. It may also be required to configure the adder during run-time. So accuracy configuration units have been integrated with many approximate adders [4, 5, 7, 8]. In this paper, we present a novel and more efficient design for the error detection and correction (EDC). The proposed design targets to minimize the overhead due to the accuracy configurability by correcting the error accumulated from multiple additions.

2.3 Trade-offs in Accuracy Configurability

Approximate adders can be equipped with the capability of partial or complete error recovery. Intuitively, in case of complete error correction, we lose the benefits of approximation, as the adder yields the precise sum. However, it is important to note that this happens for a small fraction of input combinations, for which an error is detected. For most inputs, the high-performance adders, like the GeAr adder, are faster than the precise adders. In this paper, our aim is to equip the adders with accuracy configurability with minimal degradation in area and timing.

3. ERROR DISTRIBUTIONS

We have evaluated Probability Mass Functions (PMFs) of approximation error by simulating various configurations of the GeAr model [5]. The random variable E, representing error in the approximate sum, is defined as follows:

$$E = S_{prec} - S_{appr} \quad (1)$$

where S_{prec} is the precise sum and S_{appr} is the output of the approximate adder. The probability of occurrence of error can be found using the model in [5]. Here, we are interested in designing error correction logic. Therefore, we will be studying the distribution of error only when it is non-zero.

Exhaustive simulations are used to evaluate PMFs, i.e., an N bit approximate adder is evaluated for all the 2^{2N} input combinations. Fig. 2 shows the computed PMFs for selected configurations of the approximate adder. It can be observed that 1) in approximate adder with two sub-adders, error can only have one possible value, 2) in approximate adder with three sub-adders, it can have two or three possible values

We propose to leverage upon the observations gained by these distributions to simplify the error correction of approximate adders. *The main idea is that if an error is detected and we know the amount of error, then the logic for error correction can be simplified by customizing it for the particular configuration.*

4. MATHEMATICAL MODELING OF ERROR MAGNITUDE

Since the error in the approximate adders occurs due to the broken carry-chain between successive sub-adder units, *the error is due to the missed addition of carry at a specific bit location.* Hence, in case of error, the approximate sum S_{appr} is always going to be less than the precise sum (assuming unsigned addition) by an amount equal to the weight of that bit location at which the carry bit would have been added. For example, in case of GeAr adder with two sub-adders, the error will always be equal to be 2^{R+P}, as observed in Fig. 2. Hence, when error occurs,

$$S_{prec} = S_{appr} + 2^{R+P} \tag{2}$$

To get the corrected output, we can add 2^{R+P} to the output of the approximate adder when an error is detected.

We developed a mathematical relationship between S_{prec} and S_{appr}, for the approximate adder configurations with three sub-adders. For $R > P$,

$$S_{prec} = \begin{cases} S_{appr} + 2^{R+P}, & \text{if } ED_2 \wedge \overline{ED}_3 = 1 \\ S_{appr} + 2^{2R+P}, & \text{if } \overline{ED}_2 \wedge ED_3 = 1 \\ S_{appr} + 2^{R+P}, & \text{if } ED_2 \wedge ED_3 = 1 \end{cases} \tag{3}$$

where ED_2 and ED_3 represent error detection signals for Sub-Adder 2 and 3, respectively, as shown in Fig. 4, such that $ED_i = cp_i \wedge co_{i-1}$. This is explained in Section 2. The corrections for the first two cases, in (3), can be explained using the same reasoning as in case of two sub-adders. In the third case, since the condition $R \leq P$ implies that all the prediction bits of the Sub-adder 3 completely overlap with the sum bits of the Sub-adder 2, the same carry that was generated at bit location $R+P$ would have also propagated to the bit location $2R+P$ in case of precise addition. Thus the overall sum is deficient by only 2^{R+P}. For configurations with $R > P$, there is partial overlap between the prediction bits of Sub-adder 3 and sum bits of Sub-adder 2, so there is a possibility of a second carry-out generation in the Sub-adder

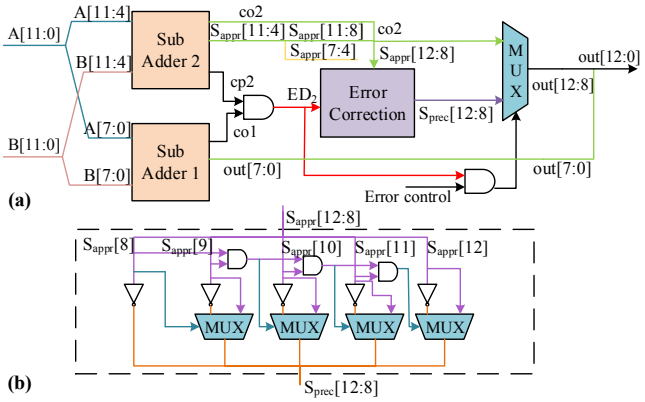

Figure 3: (a) EDC for GeAr(12,4,4), (b) EC unit

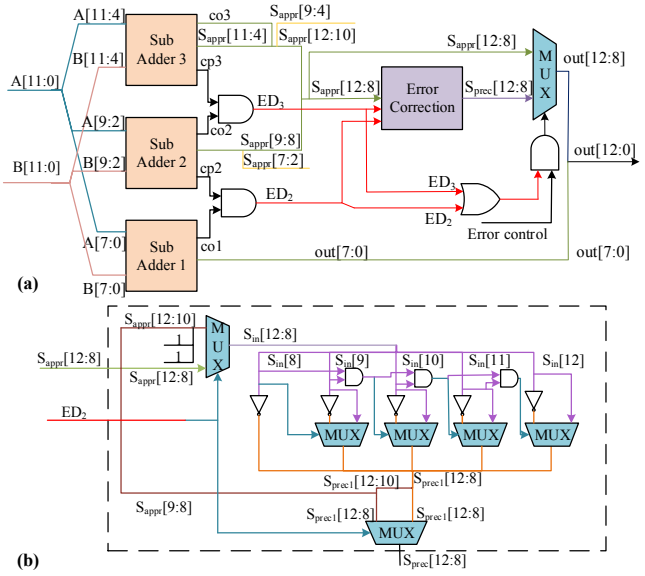

Figure 4: (a) GeAr(12,2,6) with EDC, (b) EC unit

2. Hence, with $R > P$,

$$S_{prec} = \begin{cases} S_{appr} + 2^{R+P}, & \text{if } ED_2 \wedge \overline{ED}_3 = 1 \\ S_{appr} + 2^{2R+P}, & \text{if } \overline{ED}_2 \wedge ED_3 = 1 \\ S_{appr} + 2^{R+P}, & \text{if } ED_2 \wedge ED_3 \wedge Q = 1 \\ S_{appr} + 2^{R+P} + 2^{2R+P}, & \text{if } ED_2 \wedge ED_3 \wedge \overline{Q} = 1 \end{cases} \tag{4}$$

where Q represents the event that the input bits, at locations $[R+P-1 : 2R-1]$, are all propagating carry-in, i.e., $Q = \bigwedge_{i=R+P-1}^{2R-1} A[i] \oplus B[i]$.

These relationships have been verified, through exhaustive simulations, for GeAr with N ranging from 4 to 12.

5. PROPOSED DESIGNS

Based on the above-discussed error model, in this section, we propose a new implementation of EDC for these adders.

5.1 Error Detection

Fig. 3 shows an approximate adder with two sub-adders with the proposed EDC. Error detection requires only an AND gate. The inputs to this AND gate are the carry-out

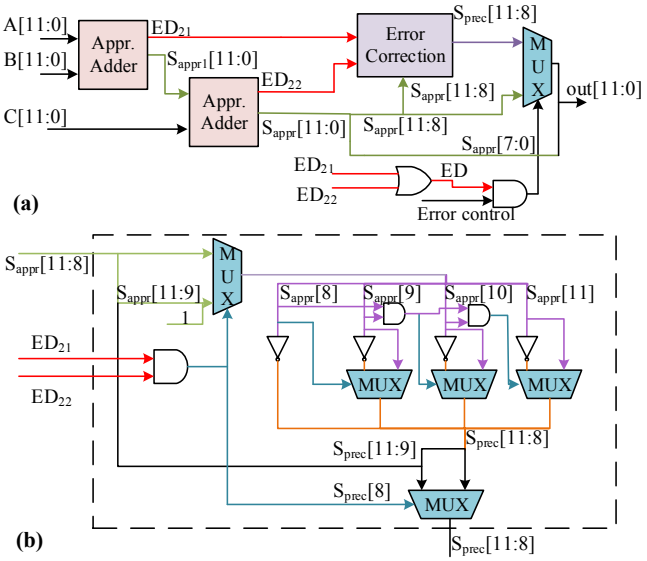

Figure 5: (a) EDC for two approximate adders connected in cascade (b) CEC-2

from Sub-adder 1, $co1$, and carry propagation signal from the Sub-adder 2, $cp2$. Hence, $ED_2 = co1 \wedge cp2$. Since in case of error, $cp2$ will be one, which means that the lower P bits in Sub-adder 2, that are also the higher P bits in the Sub-adder 1, are all propagating. This means that the carry-out $co1$ is actually generated by the lower R bits in the Sub-adder 1. So we use the carry-out from the addition of lower R bits. The advantage of using this carry-out is that the error detection signal will be valid before the addition in the sub-adder is completed. So the error correction circuit can operate in parallel with the addition in the Sub-adders. It was verified by synthesizing the design on FPGA that the critical path remains unchanged with error detection.

5.2 Error Correction for Approximate Adders

According to (2), S_{appr} can be corrected by adding 2^{R+P} to it. Since the number to be added to $S_{appr}[N:R+P]$ is a binary 1, at least significant bit (LSB), followed by binary zeros, so the logic circuit, shown in Fig. 3, has been optimized to perform this particular addition. Starting from the LSB, every bit of $S_{appr}[N:R+P]$ up till the occurrence of first zero is toggled, while the rest of the bits are passed to output without any change. The hardware implementation is shown in Fig. 3(b). After the addition in the approximate adder is completed, only three additional gate delays will be required to get the corrected sum. So the complete EDC can be implemented as a combinational circuit, instead of the sequential logic used in [4,5,8]. Hence by exploiting prior knowledge of the error magnitude, it is possible to avoid the increase in latency of the circuit.

The EDC for an approximate adder with three sub-adders is shown in Fig. 4. According to (3), if ED_2 is 1, then the required correction is 2^{R+P}. Otherwise if only ED_3 is 1, the required correction is 2^{2R+P}. For this configuration, the logic for error correction is modified by using multiplexers at the input and output of the error correction (EC) unit. If 2^{R+P} is required to be added, then the logic circuit is used in the same way as in case of GeAr with two sub-adders, depicted in Fig. 3. But if 2^{2R+P} is the required correction, then the bits at positions $[2R+P-1:R+P]$ are passed to the output unchanged. The intermediary AND gates are bypassed by setting the relevant bits of the input to error correction unit, i.e., $S_{appr}[2R+P-1:R+P]$, to 1. Since the detection signal is valid before the sum bits in Sub-adders 2 and 3 are evaluated, most part of the error correction logic operates in parallel with the addition in the Sub-adders.

5.3 Error Correction for Approximate Hardware Accelerators

In most practical applications, many adders are connected as hardware accelerators that implement more complex arithmetic data paths. In Fig. 5(a), two approximate adders are connected to perform two additions in succession. Using simple modifications of the proposed EC circuit, as shown in Fig. 5(b), a consolidated error correction (CEC) circuit can be implemented for a combination of two approximate adders, instead of implementing it separately for every adder. The design in Fig. 5 will be referred to as CEC-2. This reduces the area-on-chip of the circuit. In the circuit shown in Fig. 5(b), if error occurs in only one of the two approximate adders, i.e., either ED_{21} or ED_{22} is high, then 2^{R+P} is added to $Sum2$. If both ED_{21} and ED_{22} are 1, then 2^{R+P+1} is added, because now the final sum is deficient by twice the amount for individual sums.

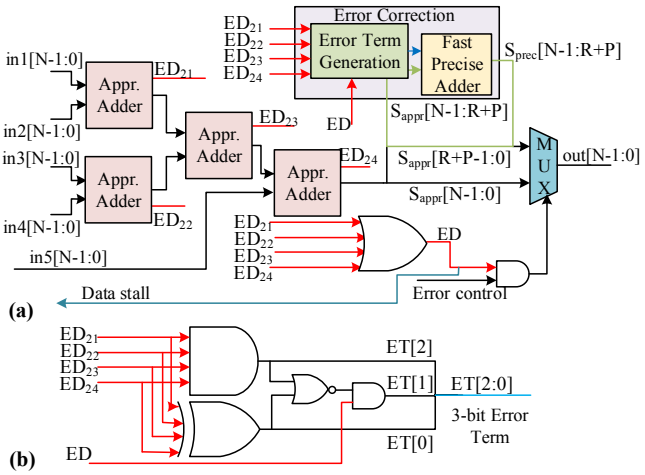

Figure 6: (a) Four Cascaded approximate adders with configurable accuracy(b) CEC-4

For larger number of operands, the accumulated error is calculated as an *error term (ET)* which is added to the output of the accelerator using a $(k-1)R$-bit fast precise adder. The CEC-4 design for an accelerator with four GeAr adders is shown in Fig. 6. The ET is calculated as a function of the four error detection signals from the approximate adders, using the circuit in Fig. 6(b). Building upon the same logic, CEC-8 is designed for a cascade of eight adders, as shown in Fig. 7.

6. RESULTS AND DISCUSSIONS

We have integrated our CEC units in approximate hardware accelerators used in several image processing applications, including Sum of Absolute Differences (SAD), Gaussian Smoothing Filter and K-Means algorithm for image seg-

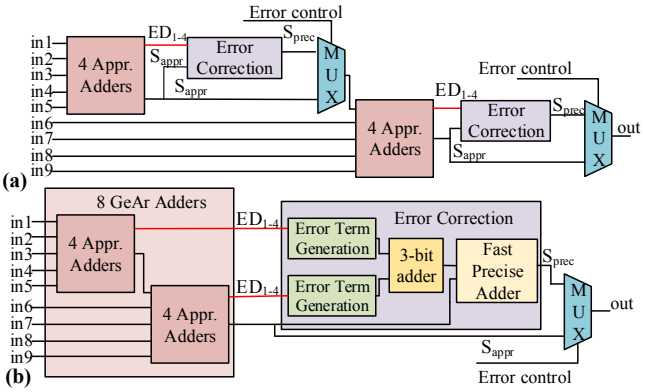

Figure 7: Hardware accelerator with eight approximate adders with (a) CEC-4 in two stages(b) CEC-8

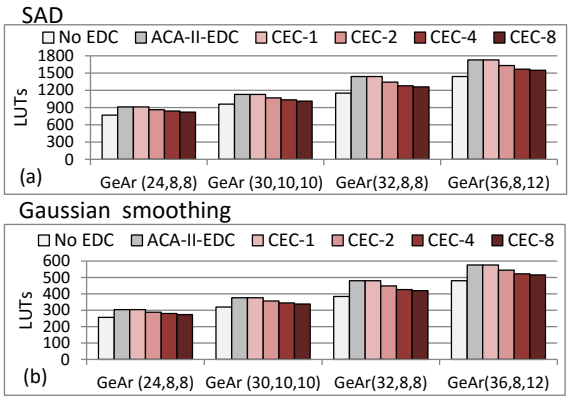

Figure 8: Comparison of area overhead with EDC designs in (a) SAD and (b) Gaussian smoothing

mentation. The proposed designs were implemented in Verilog hardware description language (HDL) and synthesized using Xilinx ISE, for Xilinx Virtex 6 XC6VCX75T FPGA.

6.1 Comparison of CEC-1, CEC-2, CEC-4 and CEC-8 for Combinational Circuits

The proposed CEC units can be integrated with both combinational and sequential circuits. In order to demonstrate the savings in area, we first integrate CEC-1 (EC unit in individual adders), CEC-2 (Figs. 5 and 4), CEC-4 (Fig. 6) and CEC-8 (Fig. 7) in combinational hardware accelerators used in Gaussian smoothing (using 3x3 kernel) and SAD (using 5x5 kernel) applications. The multiplications in Gaussian smoothing and subtractions in SAD are implemented using precise components. GeAr adders are used for the additions only. Fig. 8 shows the comparison of required silicon area (LUTs on FPGA). The proposed CECs are also compared with the EDC proposed for ACA-II in [4], referred to as ACA-II-EDC. We observe that as the error correction is consolidated for larger number of adders, the area overhead due to EDC is reduced. Using GeAr(24,8,8), the area overhead is reduced by approximately 30% in CEC-8 as compared to CEC-1 and ACA-II-EDC.

6.2 CEC in Pipelined and Sequential Circuits

Pipelined cascaded adders are used to operate the application at a higher frequency. For SAD using a 5x5 kernel, a five-stage pipeline is used to perform 24 additions and three CEC-8 units are integrated. For Gaussian smoothing using a 3x3 kernel, a 4-stage pipeline with one CEC-8 unit is used.

Table 1 shows the combinational delays involved in the GeAr adder itself and the corresponding CEC-8 units. Table 1 also shows comparison with the GeAr-EDC design. Due to the larger delay of CEC-8, we need to operate the circuit at a reduced frequency as compared to that with GeAr-EDC, proposed in [5]. However, the total time required to complete the additions is smaller. This is because GeAr-EDC requires one extra cycle for error correction in every sub-adder, whereas CEC-8 requires one extra cycle after eight additions. It is noteworthy that the area overhead in this type of architecture is same as in combinational circuits.

Table 2 shows the comparison of speed performance of CEC-8 with the GeAr-EDC. Approximate mode time is the time required for the cascaded addition without any error

Table 1: Combinational delays

Adders	Delay without EDC (ns)	Delay of CEC-8 (ns)	Delay with GeAr-EDC (ns)
GeAr(24,8,8)	1.527	1.896	1.625
GeAr(30,10,10)	1.595	1.966	1.693
GeAr(32,8,8)	1.527	1.896	1.625
GeAr(36,8,12)	1.602	1.896	1.7
GeAr(40,12,4)	1.527	2.036	1.625

recovery scheme. Best time represents the case when there is no error in any sub-adder in the presence of error recovery. Average and worst times represent cases when half and all of the sub-adders' outputs are erroneous, respectively. The smaller amount of time required in CEC-8 as compared to that in GeAr-EDC is due to the fact that in CEC-8, smaller number of cycles are required as compared to GeAr-EDC. It is evident from Fig. 9 that the average time consumed by the cascaded additions is smaller in CEC-8 as compared to GeAr-EDC. However, since the GeAr-EDC is a time-shared architecture, its area overhead is almost negligible.

In the K-Means algorithm, the number of operands of addition change in every iteration. For hardware implementation of the K-Means algorithm, we have deployed our CEC-8 in a finite state machine (FSM). In this circuit, one addition, using the GeAr adder, is performed per cycle. The error correction is performed after every eight cycles/additions using CEC-8. GeAr-EDC is also deployed in the same application for performance comparison. We found that the total time required for additions is much smaller, as shown in the results given in Table 2.

6.3 Discussions

We can conclude following from our studies on speed and silicon area required in various EDC designs:

- Area overhead due to EDC is reduced as error correction is consolidated for larger number of additions.
- CEC-8 is faster that GeAr-EDC and ACA-II-EDC.
- In terms of silicon area, GeAr-EDC is more efficient than ACA-II-EDC and CEC designs.

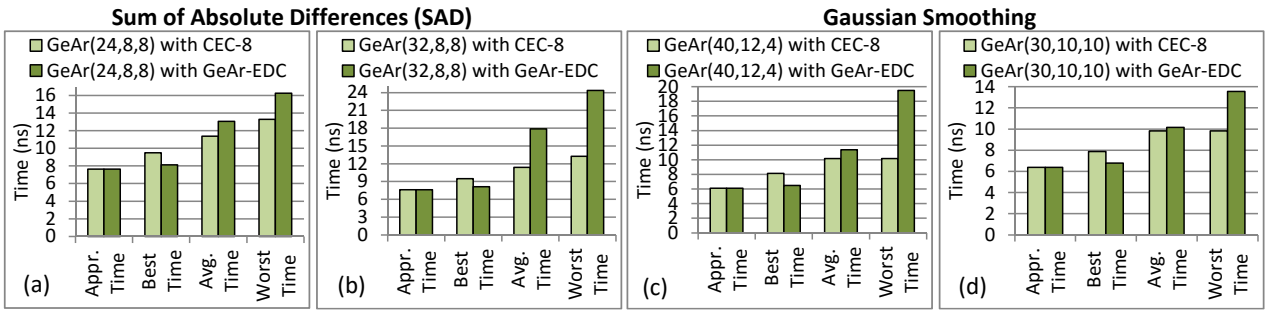

Figure 9: Comparison of time delay in (a) SAD using GeAr(24,8,8), (b) SAD using GeAr(32,8,8), (c) Gaussian smoothing using GeAr(30,10,10) and (d) Gaussian smoothing using GeAr(40,12,4)

Table 2: Summary of speed performance

Adder	Pr[Error] (per adder)	Appr. Time (ns)	Error Correction Result Timings (ns)		
			Best Time	Avg. Time	Worst Time
SAD using 5x5 kernel					
GeAr(32,8,8) with CEC-8	0.003890991	7.635	9.48	11.376	13.272
GeAr(32,8,8) with GeAr-EDC	0.003890991	7.635	8.125	17.875	24.375
Gaussian smoothing using 3x3 kernel					
GeAr(40,12,4) with CEC-8	0.061512231	6.108	8.144	10.18	10.18
GeAr(40,12,4) with GeAr-EDC	0.061512231	6.108	6.5	11.375	19.5
K-Means for Image Segmentation, using an FSM					
GeAr(24,8,8) with CEC-8	0.00194550	12.216	15.168	17.064	17.064
GeAr(24,8,8) with GeAr-EDC	0.00194550	12.216	13	19.5	26.002
GeAr(32,8,8) with CEC-8	0.003890991	12.216	15.168	17.064	17.064
GeAr(32,8,8) with GeAr-EDC	0.003890991	12.216	13.001	26.00	39.001

Area and time overheads due to accuracy configurability differ for different architectures (combinational, pipelined and FSMs). So application requirements must be taken into account to appropriately trade among latency, speed, silicon area and power consumption. The proposed CEC is found to be flexible enough to be integrated in all three types of architectures.

7. CONCLUSION

In this paper, we have presented efficient implementations of EDC for accuracy configurable approximate adders and hardware accelerators. Accuracy configuration for adders in cascade has been presented, in which correction is applied to error accumulated from several additions. The proposed implementations are found to be efficient in terms of timing and required area-on-chip as compared to state-of-the-art designs.

8. REFERENCES

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

[2] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Analysis and characterization of inherent application resilience for approximate computing," in *Proc. IEEE/EDAC/ACM 50th Des. Autom. Conf.*, 2013, p. 113.

[3] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," *IEEE Trans. Comput.-Aided Des. Integ. Circuits Syst.*, vol. 32, no. 1, pp. 124–137, 2013.

[4] A. B. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs," in *Proc. IEEE/EDAC/ACM 49th Des. Autom. Conf.*, 2012, pp. 820–825.

[5] M. Shafique, W. Ahmad, R. Hafiz, and J. Henkel, "A low latency generic accuracy configurable adder," in *Proc. IEEE/ACM/DAC 52nd Des. Autom. Conf.*, 2015, p. 86.

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

[7] R. Ye, T. Wang, F. Yuan, R. Kumar, and Q. Xu, "On reconfiguration-oriented approximate adder design and its application," in *Proc. Int. Conf. Comput.-Aided Des.*, 2013, pp. 48–54.

[8] A. K. Verma, P. Brisk, and P. Ienne, "Variable latency speculative addition: A new paradigm for arithmetic circuit design," in *Proc. Des., Autom. Test Eur. Conf. Exhib.*, 2008, pp. 1250–1255.

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

[10] M. Shafique, R. Hafiz, S. Rehman, W. El-Harouni, and J. Henkel, "Cross-Layer Approximate Computing: From Logic to Architectures," in *Proc. 53rd IEEE/EDAC/ACM Des. Autom. Conf.*, 2016.

[11] "CES ApproxAdderLib," http://sourceforge.net/projects/approxadderlib/, [Online; accessed 2015].