Approximated Ripple Carry Adders Evaluation

Andrei Pochmann Koenich[1], Gabriel Ammes Pinho[1], Paulo Francisco Butzen[2] and Renato Perez Ribas[1]
[1]*Institute of Informatics and* [2] *School of Engineering, UFRGS, Brazil*
{*andrei.koenich, gapinho*}*@inf.ufrgs.br* ,{*paulo.butzen, renato.ribas*}*@ufrgs.br*

***Abstract*—In approximate computing, related circuits are highly relevant to enable the efficiency of specific operations. This work aims to investigate the behavior of binary adder circuits known as ripple carry adder (RCA) in cases where certain breaks in their carry chain occur. Those breaks aim to provide velocity improvement in the operation of binary sums through an approximate adder circuit. In order to evaluate the impact of those breaks in operations involving RCA, some experiments were carried out, taking into account the insertion of 0 or 1 logic values in the broken carry chain position, allowing the analysis of performance gain and functional errors generated. Such an evaluation can be considered a basis for studying other adder architectures.**

***Index Terms*—*Approximate computing*, *ripple carry adder*, *carry chain*, *performance analysis*.**

I. Introduction

Approximate computing comprises a set of techniques applied in a computational system through an imprecise or inexact execution, aiming to obtain performance improvement or energy according to a threshold of accepted errors [1, 2]. Several digital applications of the contemporary world, such as signal processors, sensors, and machine learning, can guarantee an adequated operation even when they are subjected to eventual errors during their operation [3]. This relevant feature, known as error resilience, makes the study of approximate computation quite interesting and worthy in the integrated circuit design domain.

There are different design levels in which it is possible to explore approximate computing, from compilers and high-level programming languages algorithms to lower levels comprising hardware architecture and electronic circuitry [1, 2]. Approximating arithmetic circuits are particularly relevant due to their large applicability in digital system design [4].

This work extensively evaluates approximate RCA, considering different positions of breaking the carry chain. Moreover, the insertion of 0 (cut) and 1 (induction) logic values on these positions result in different behaviors. Furthermore, the relationship between performance and error metrics represents a crucial design choice according to the number of breaks in the carry chain. In this context, another way to make approximations in an RCA is to modify the truth tables of its internal circuits, aiming at energy saving and reduction in the number of electronic components [5, 6].

The structure of this paper consists of reviewing the error metrics related to approximate computation in Section II, and the classical ripple carry adder in Section III. In Section IV, the methodology of the approximate RCA investigation is described. Experimental results are discussed in Section V. Finally, the conclusions are outlined in Section VI.

II. Error Metrics

In approximate computing, error metrics represent mathematical expressions that measure the amount of error in the approximate circuit operation results. Therefore, error metrics can allow a rational and effective control of applied optimization procedures.

The main error metrics considered in the scope of this work are presented below [4, 7]. For each mathematical expression presented, it is assumed that n represents the total number of inputs related to a given digital circuit (that is, the number of bits), and B^n represents all possible combinations of inputs. In contrast, $f(x)$ and $\hat{f}(x)$ correspond to the output values for the original circuit functionality and the approximate circuit behavior, respectively.

The error rate (ER) measures the frequency or probability of observing one or more incorrect output bits in a digital circuit. This error metric is considered the most common, being widely applied in logic and arithmetic circuit analysis. It can be estimated as follows:

$$ER = \frac{1}{2^n} \sum_{\forall x \in \mathbb{B}^n} f(x) \neq \hat{f}(x)$$

The mean absolute error (MAE) consists of the arithmetic mean of all the absolute errors obtained with the operation of an approximate circuit. In turn, the absolute error or error magnitude (EM) corresponds to the modulus of the difference between the values obtained in the same computational operation, with and without the implemented approximation. MAE can be estimated as follows:

$$MAE = \frac{1}{2^n} \sum_{\forall x \in \mathbb{B}^n} |f(x) - \hat{f}(x)|$$

The relative error (RE) is the ratio between the absolute error and the expected output value of a given operation, as follows:

$$RE = \frac{|f(x) - \hat{f}(x)|}{f(x)}$$

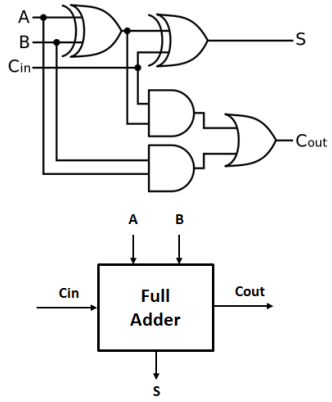
The mean squared error (MSE) is the arithmetic mean of the squares of the absolute errors. With this error metric, it is possible to measure the accuracy of an approximate circuit considering that each absolute error is squared, which prevents the occurrence of errors with high magnitude. The square root

of the metric error (RMSE), in turn, is obtained simply through the square root of the MSE. In dimensional analysis, it causes the error to return to the original unit of measurement. RMSE can be estimated as follows:

$$RMSE = \sqrt{\frac{1}{2^n} \sum_{\forall x \in \mathbb{B}^n} (f(x) - \hat{f}(x))^2}$$

III. Ripple Carry Adder

The RCA architecture comprises a logical chain combination of a certain amount of full adders (FAs), making it possible to perform a sum between two n-bit binary operands [8]. A full adder, whose truth table is shown in Fig. 1, represents the most elementary block and can be implemented, for instance, by two exclusive-OR (XOR) logic gates, two ANDs, and a single OR logic gate.



A	B	Cin	Cout	S
0	0	0	**0**	**0**
0	0	1	**0**	**1**
0	1	0	**0**	**1**
0	1	1	**1**	**0**
1	0	0	**0**	**1**
1	0	1	**1**	**0**
1	1	0	**1**	**0**
1	1	1	**1**	**1**

Figure 1: Schematic and truth table of full adder.

The circuit illustrated in Fig. 1 receives two bits, A and B, and a third bit representing the eventual carry-in (Cin), returning the value of the output bit in S and the bit that represents the occurrence of carry-out in Cout. Therefore, when associating several full adders, it is possible to perform the sum between two binary operands with any number of bits, as indicated in Fig. 2. An RCA performance (velocity) is determined by the propagation chain of the carry signal, that is, by the number of cascaded FA blocks.

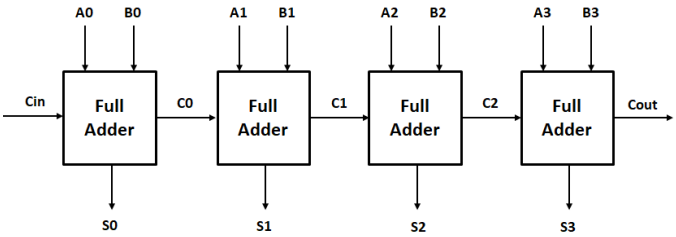


Figure 2: Block diagram of 4-bit ripple carry adder (RCA).

IV. Approximate RCA Analysis

Fig. 2 shows a carry propagation chain through the FA blocks, represented by output signals C0, C1, and C2. In the context of approximate computing, it is possible to perform breaks at strategic points in the carry propagation chain in order to reduce its length, so increasing the adder operation speed at the expense of generating some logic errors at the output [8]. Depending on how these breaks occur, for instance, taking into account the significance of the bit chosen to perform an interference in the respective carry-in, different values of errors are obtained for each of the metrics described above.

Our methodology for analyzing the errors produced by interference in the carry propagation chain in a sum involving two binary operands involves verifying the error metrics produced by cuts (i.e., making carry-in sinal equals to 0 logic value) and inductions (i.e., making carry-in signal equals to 1 logic value) in specific FA chain positions. For each experiment presented in Section V, we obtain the error metrics described in Section II.

- For the first experiment, shown in Table I and Fig. 3, we consider an RCA with 8 input bits. All 8-bit input combinations were considered, totaling 65536 operations. We perform fixed interferences (cuts and inductions) only in a specific bit. Metrics are calculated for each modification in the adder architecture. This experiment shows the relationship between the two types of interferences (cut and induction).
- For the second experiment, shown in Table II and Fig. 4, we performed the same operations described in the previous experiment. However, all operations (with or without carry-in interference) in which overflow occurs are disregarded in metric calculations. Again, this experiment aims to show the relationship between the two types of interference, and allow the analysis of changes in the values of the error metrics (comparing this experiment with the first one) caused by disregarding overflow.
- For the third experiment, we consider an RCA with 16 input bits, as shown in Table III. As the number of operations for a 16-bit adder is very high, ten thousand simulations were considered to obtain the proposed metrics. In this experiment, we also performed only the cut in a specific bit. This experiment seeks to analyze the behavior of the approximation in an adder with more input bits and the impact of non-exhaustive simulation.
- For the fourth experiment, shown in Table IV, in Fig. 5 and in Fig. 6, we also consider a 16-bit RCA, but interferences are performed to break the carry chain in more than once. This way, when one cut is made, the chain is split into two parts; when two cuts are made, the chain is split into three parts, and so on. This experiment aims to understand the relationship between the approximate RCA performance and the number of inserted errors.

V. Experimental Results

By analyzing Table I, it can be seen that the error rate (ER) related to carry-in cuts starts closer to 0.25, while the same metric related to carry-in inductions starts closer to 0.75. As

Table I: Exhaustive stimilus of 8-bit RCA, taking into account overflow cases.

Bit	Cut			Ind		
	ER	MAE	MRE	ER	MAE	MRE
1	0.25	0.99	0.50	0.75	2.97	0.04
2	0.37	2.95	1.38	0.62	4.92	0.07
3	0.43	6.78	2.24	0.56	8.71	0.11
4	0.46	14.06	3.00	0.53	15.93	0.19
5	0.48	27.12	3.54	0.51	28.87	0.31
6	0.49	47.25	3.67	0.50	48.75	0.49
7	0.49	63.5	2.88	0.50	64.50	0.67

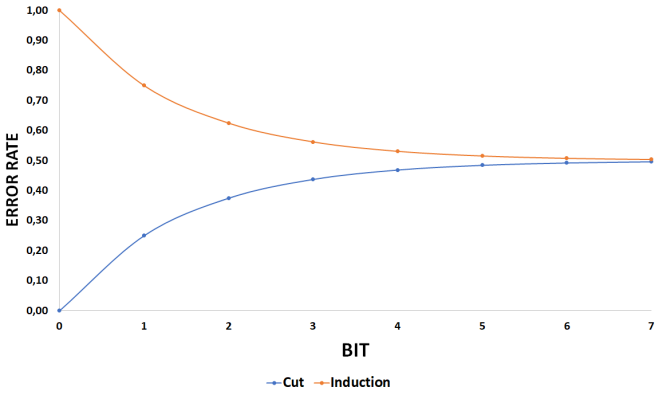


Figure 3: Error rate comparison considering overflow cases.

the interference bit significance increases, both values tend to equal 0.50. This behavior can be seen in Fig. 3, where an intersection between the two curves is shown. In turn, the mean absolute errors (MAE) present relatively close values for both types of interference, which also increase along with the significance of the interference bit. As for MAE, it is noted that the values related to carry-in cuts are substantially higher than those related to carry-in inductions.

By analyzing Table II, one can see no change in the behavior of ER values in cases of carry-in inductions compared to the previous experiments. However, it can be seen that, from a certain point onwards, the ER values of the carry-in cut-off cases begin to decrease with the increase in the significance of the interference bit, as graphically shown in Fig. 4. In addition,

Table II: Exhaustive stimilus of 8-bit RCA, ignoring overflow cases.

Bit	Cut			Ind		
	ER	MAE	MRE	ER	MAE	MRE
1	0.24	0.49	0.00	0.75	1.50	0.01
2	0.36	1.47	0.01	0.62	2.50	0.02
3	0.42	3.37	0.02	0.56	4.50	0.03
4	0.43	7.00	0.05	0.53	8.50	0.07
5	0.42	13.50	0.09	0.51	16.50	0.16
6	0.36	23.53	0.15	0.50	32.50	0.38
7	0.24	31.62	0.19	0.50	64.50	1.04

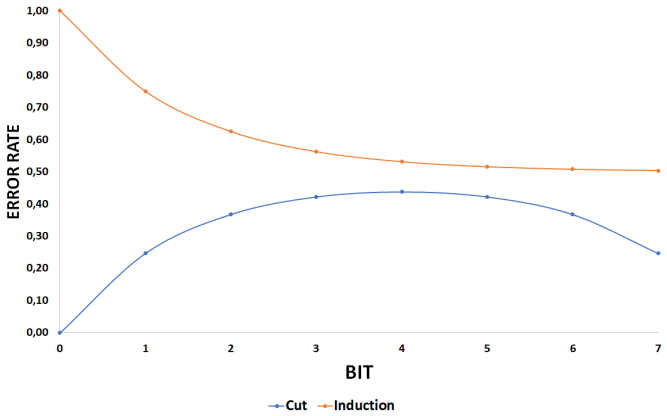


Figure 4: Error rate comparison ignoring overflow cases.

Table III: Evaluation of 16-bit RCA with random operands and carry-in cuts.

Bit	ER	MAE	RMSE	MRE
1	0.24	0.49	0.99	0.00
2	0.37	1.50	2.45	0.00
3	0.42	3.43	5.24	0.00
4	0.47	7.53	10.97	0.00
5	0.47	15.31	22.13	0.00
6	0.48	31.21	44.69	0.00
7	0.49	63.64	90.25	0.00
8	0.49	126.15	179.71	0.00
9	0.49	253.54	360.29	0.01
10	0.51	522.24	731.28	0.02
11	0.50	1034	1455	0.04
12	0.49	2018	2875	0.07
13	0.50	4116	5807	0.14
14	0.49	8155	11559	0.24
15	0.50	16420	23195	0.38

it is possible to observe a greater distance in the MAE values for the two types of interference. Finally, it is noted that the disregard of overflow cases have caused a significant reduction in the values of MAE related to carry-in cuts.

By observing Table III, we have noticed that in the ER calculations, there is a fast convergence to the value 0.50 as we increase the significance of the interference bit. In MAE values, it is possible to notice the proximity of the results with powers of two. The RMSE and MRE metrics, in turn, increase with the significance of the bit as expected, and the growth of MRE occurs slowly compared to the other experiments.

By observing Table IV, it is possible to notice that a limitation in the size of the carry-in chain promotes an increasing in the value of ER, which are close to 1, as indicated in Fig. 5. Increasing the number of carry-in cuts in different bits, there is a significant augmentation in MAE values, with the exception that in the largest possible limitation in the size of the carry-in chain, a brief decreasing occurs, as graphically observed in Fig. 6.

Table IV: Evaluation of 16-bit RCA with random operands and simultaneous carry-in cuts.

Carry Chain Max. Size	ER	MAE	Bits w/ Cuts
8	0.49	62	7
5	0.73	515	5,1
4	0.84	2075	3,7,12
3	0.89	4145	2,5,9,13
2	0.93	8233	2.5.8,11,14
1	0.96	8016	2,4,6,8,10,12.14

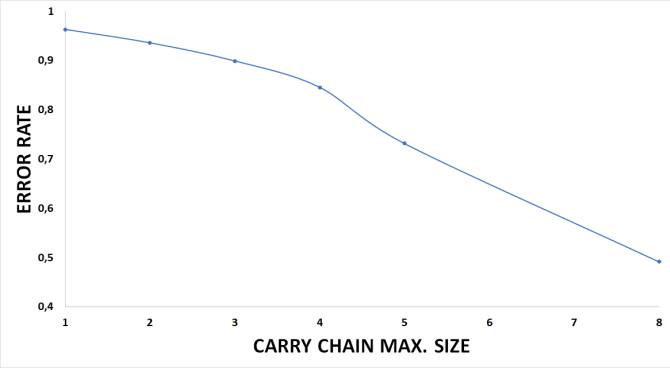


Figure 5: Error rate analysis with simultaneous carry-in cuts.

VI. Conclusions

This paper presented an analysis of the impact of breaking the carry chain of an RCA and a simulator. As a result, the impacts of interference on the RCA carry-in chains could be graphically and metrically visualized through the use of error metrics related to the approximate computation. The different ways of interfering with carry-in chains guarantee several different analysis options, guaranteeing the need for different forms of analysis on the subject. The potential applicability of adder circuits implies the importance of analyzing its operation in approximate computing, justifying the performance of the experiments demonstrated and experiments to be carried out in future projects. In addition to breaking the carry chain, another possible approach is to modify the functionality of the blocks used, such as the full adders (FA).

In future work, extending the simulator developed to consider eventual logic behavior modifications in FA is interesting. Another natural extension of this work is considering other adder architectures, such as carry-select (CSelA), carry-skip (CSkipA), and carry lookahead (CLA) adders. Such an extension of research can reveal new efficient ways to approximate adder circuits, contributing to the development of approximate computing.

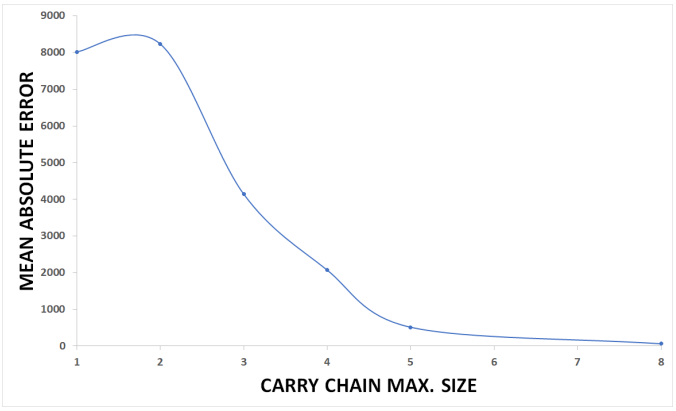


Figure 6: Mean absolute error analysis with simultaneous carry-in cuts.

References

[1] Qiang Xu, Todd Mytkowicz, and Nam Sung Kim. "Approximate Computing: A Survey". In: *IEEE Design & Test* 33.1 (2016), pp. 8–22.

[2] Sparsh Mittal. "A survey of techniques for approximate computing". In: *ACM Computing Surveys* 48.4 (2016).

[3] Vinay Chippa et al. "Analysis and characterization of inherent application resilience for approximate computing." In: *2013 Design Automation Conference (DAC)*. 2013.

[4] Honglan Jiang et al. "Approximate Arithmetic Circuits: A Survey, Characterization, and Recent Applications". In: *Proceedings of the IEEE* 108.12 (2020), pp. 2108–2135.

[5] Vaibhav Gupta et al. "IMPACT: IMPrecise adders for low-power Approximate CompuTing". In: *International Symposium on Low Power Electronics and Design (ISLPED))*. 2011.

[6] Zhixi Yang et al. "Approximate XOR/XNOR-based Adders for Inexact Computing". In: *IEEE International Conference on Nanotechnology*. 2013.

[7] Zdenek Vasicek. "Formal Methods for Exact Analysis of Approximate Circuits". In: *IEEE Access* 7 (2019), pp. 177309–177331.

[8] Israel Koren. *Computer Arithmetic Algorithms*. A K Peters/CRC Press, 2001.