International Journal of Scientific Research in Science and Technology
Print ISSN: 2395-6011 | Online ISSN: 2395-602X (www.ijsrst.com)
doi : https://doi.org/10.32628/IJSRST

Design of Approximate Adder Using Error Reduced Carry Prediction and Constant Truncation Employing Parallel Prefix Adder

Nivedita[1], Dr. Anuradha Savadi[2]
[1]M. Tech Scholar, Department of ECE (VLSI & EMBEDDED SYSTEM), Sharnbasva University, Kalaburgi, India
[2]Associate Professor, Department of ECE, Sharanbasva University, Kalaburgi, India

Article Info
Volume 9, Issue 4
Page Number : 272-278

Publication Issue
July-August 2022

Article History
Accepted : 05 July 2022
Published : 20 July 2022

ABSTRACT

This article offers a unique approximation adder which uses error-reduced carry prediction and constant reduction in conjunction with error reduction techniques. The suggested adder design actually improve computing accuracy while also improving hardware effectiveness. In comparison to the approximate adders studied in this research, the proposed carry forecast technique can elevate prediction error rates. The error reduction technique also enhances overall computation efficiency by lowering the error distance (ED). The suggested adder is one of the most economical because it has the effective design compromise of the adders beneath analysis. We also show that whenever the postulated adder is used in application areas such as digital image processing and machine learning, the approximation errors caused by the adder have even less influence on quality attributes.

Keywords :- Error Distance, Approximate adder.

I. INTRODUCTION

For error-tolerant multimedia applications coding, approximate computing is a new design strategy that trades accuracy for performance, size, as well as/or power consumption. Because the main criterion is to produce output of sufficient quality to give a decent user experience, the video compression is error-tolerant in nature. As a result, approximation computing provides a lot of promise for improving optimal design performance, area, as well as/or energy usage. Due to its rapid time-to-market, increased flexibility, as well as run-time re-configurability, FPGAs are a great platform for a variety of applications ranging from small-scale embedded devices to high-performance computing systems.

FPGA-based systems, on the other has well as, generally consume more power as well as/or energy than ASIC-based systems, despite providing specialized hardware accelerators as well as co-processors. As just a consequence, alternative routes in energy-efficient computation for FPGA-based systems should be explored in addition to existing energy optimization techniques.

An Approximation Computation paradigm is however one appealing tendency, which is re-emerging as a result of Moore's law as well as Dennard scaling collapsing, as well as the growing desire for high-performance as well as energy efficiency. To obtain

significant improvements in critical chain latency, area, energy, as well as/or energy consumption, approximate computing compromises the accuracy as well as precision of intermediate or final computations.

This trade-off is advantageous for applications that have inherent application resilience, or the capacity to provide usable output even if part of the computations are wrong due to approximations. This property could be found in a wide range of identification, mining, as well as synthesis applications such as image as well as video processing, information retrieval, machine learning, as well as whatever else.

Current approximation computation methodologies as well as concepts could be implemented to various levels of the computer stack, from hardware circuitry as well as architectures to software compilers as well as computer languages. Both at the hardware as well as software layers, there is a lot of study on computations. The 2 significant approximation computation knobs employed at the hardware level are power over-scaling as well as functional approximation. Loop incision, linear regression, as well as computer language options are indeed the two major kinds of software estimates. Basic compute modules, such as adders but also multipliers, are indeed the subject of approximations at the hardware level. Modelling the prediction error of present state-of-the-art ASIC-based adders as well as recursive multiplier architectures is the topic of research works like this.

We introduce a novel approximation adder architecture based on new approximate FA cells, improved carry predictions, as well as continuous truncation with error reduction in this study. In comparison to the existing approximate adders evaluated here, the suggested carry prediction system significantly reduces prediction error rate. Additionally, truncate with error reduction logic improves overall computing accuracy while consuming less energy and electricity. Our adder also enhances overall computation accuracy as compared to conventional approximate adders. Whenever the proposed design is compared to the other adders in terms of hardware and precision, this is the most competitive.

In conclusion, this study contributes to the design of approximation adders in the following ways: Through systematic analysis, we provide an unique efficient approximate adder design that efficiently balances off in between hardware cost and computation accuracy, and we establish that our design surpasses the competition by evaluating it to 12 approximation adders. We propose

- Especially compared to the others, a new carry prediction system minimizes prediction error rate.
- Approximate FA cells that improves accuracy
- A constant truncation approach with an error reduction scheme that saves money on hardware while yet providing high accuracy

The remainder provides a brief review of existing approximate adders. Then we present the suggested adder incorporates the proposed approximate FAs, new carry prediction, and constant truncation with error reduction. There are additional instances of the adder operation illustrated, as well as a quantitative study of the carry prediction error as total mistake rate. The test findings and analytical process of the hypothesized adder, as well as a thorough comparison with the 12 current approximation adders, then are explained. In addition, there is a joint examination of a adders in terms of hardware and accuracy. Ultimately, we'll look at how approximate adders can be used in digital image processing as well as machine learning. The task is finally finished.

II. EARLIER WORK

This section defines our suggested FA fibroblast approximate adder, which reduces the ED and improves overall computation accuracy by employing a novel carrier prediction system and a constant truncation technique. The error-reduced carry

prediction approximate adder is the title we gave to our adder (ERCPAA). An1:0, Bn1:0, and Sn1:0 refer to a adder's two n-bit input operands and one n-bit output. Furthermore, Ai, Bi, and Si denote the (i) th LSBs of An-1:0, Bn-1:0, and Sn-1:0, respectively.

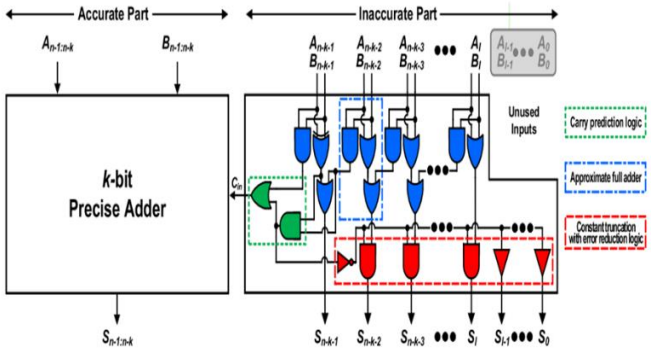

Figure 1 : Block diagram for ERCPAA

Figure 1 the planned approximation adder with n-bit inputs' overall hardware design is shown. An n-bit adder is composed of two components: a k-bit accurate element as well as a (n- k) bit erroneous element, with n-k denoting the number of bits. The exact part is made up of a k-bit precise adder that generates an effective outcome (Sn-1: n-k) using k MSB inputs (An-1: n-k and Bn-k: n-k) plus a carry input (i.e. Cin). To provide an approximation output and a carry input to the exact adder, the inaccurate component takes a few of the remaining LSB inputs. The exact adder can be applied in any sort of classic adder, such as RCA and CLA, and the sizes of the precise and inaccurate parts do not have to be equal.

An array of the proposed approximate FA cells, a carry prediction logic, and a continuous truncation with error reduction logic make up the incorrect section. The suggested FA cell (shown in blue) reduces the traditional single-bit FA cell to generate an approximation sum and carry, and this is positioned in some higher-order bit positions of the incorrect section. The carry input to the precise adder is produced by the carry prediction logic, which itself is highlighted in green.

While conventional FA-based approximate adders construct the carry input by AND ing the MSB inputs of the incorrect section, the prediction logic uses two MSB inputs to increase carry prediction accuracy just at cost of two more logic gates. And save hardware costs, the constant truncation and error mitigation logic marked in red sets l LSB outputs (i.e., Sl1:0) to either a steady 0 or a constant 1 depending on input conditions. To put it another way, the l LSB inputs aren't utilized to calculate approximation summaries. To lower the ED under particular input circumstances, it sets the other output bits to a constant 0 except for the MSB of the incorrect section. The criterion which decides whether the output bits are fixed to 0 or 1 will be explained.

For carry propagate adders, the FA is indeed a necessary component (e.g., RCA). A classic 1-bit FA requires 2 inputs, Ai and Bi, plus a carry from of the preceding bit position Ci1, as well as generates a sum Si and a carry output Ci.

- $S_i = A_i \oplus B_i \oplus C_{i-1}$
- $C_i = A_i B_i + A_i C_{i-1} + B_i C_{i-1}$

Even though the FA takes 2 XOR gates to generate a sum, we use OR equivalents to substitute the XOR gates. Even though the FA calls for two XOR gates to yield a sum, we approximation the FA by replacing the XOR gates with OR counterparts. Furthermore, the FA produces a carry output Ci from both the Ai and Bi inputs, as well as the carry from of the prior bit position Ci1. In those other words, the level are considered position's carry can be propagated to the next bit position through the current FA, causing a long critical path latency as well as poor equipment performance in the carry propagate adders. Similar to in our FA, we remove the carry's need on the preceding bit position to generate the carry output, which reduces the critical path time and hardware overhead. As a result, our estimated FA's Boolean equations are as follows.

- S_i, ERCPAA = $A_i + B_i + C_{i-1}$
- C_i, ERCPAA = $A_i B_i$

As a result, the approximation portion employing the suggested FA cell does not create the carry propagation chain from lower to higher-order bit positions, and the approximation part is actually delay is consistent, despite its larger size (i.e., k decreases under a given n). The MSB location of an erroneous component has a different FA configuration, that produces the total of the two input operands Ai and Bi using an XOR gate instead of an OR gate. The XOR-based FA offers a much more accurate sum, and that also enables the carry prediction logic to produce a more accurate carry input to the exact adder than that of the OR-based FA. The traditional and proposed FAs' truth tables are shown in Table 1. If one of the operands is a 1 and the carry of the preceding bit position is a 1 , the proposed FAs may cause errors. If both operands are 1 and the carry of the preceding bit position is 1, the OR-based FA creates an unexpected error at sum Si.

- $P_i = A_i \oplus B_i$
- $C_i = C_{i-1}$ if $P_i = 1$

A carry could be produced either from the (nk 1)th or (nk 2)th bit position, because our carry prediction scheme utilizes the inputs of two bit locations. A carry input Cin is simply Cnk1 when a carry is produced in the (n k 1)th bit position. If a carry is formed in the (n k 2)th bit position, the carry Cnk2 should be passed to the exact part through the (n k 1)th bit position. As a result, Cin is obtained as the carry input.

- $C_{in} = Cn–k–1 + Pn–k–1Cn–k–2$

To create the carry input Cin, you'll need one XOR, three AND, and one OR gate. Pnk1 can be calculated using the XOR gate of the FA in the MSB position of the incorrect section, and Cnk1 and Cnk2 can be retrieved from the suggested FAs in the appropriate bit positions. One reason for replacing the OR in the FA at the MSB with an XOR is to generate a Pnk1 signal. As a result, implementing the proposed carry prediction logic only requires two extra gates.

Because lower-order outputs have a smaller impact on accuracy than higher-order outputs, the suggested adder outputs a constant to a few LSBs to reduce hardware overhead while losing overall accuracy significantly. Mostly in illustration beneath, the adder architecture specifications n = 16, k = 8, and l = 4 are utilized to demonstrate constant truncation procedures involving error prevention. Irrespective of the inputs to the relevant bit positions, the adder sets the respective LSB outputs to 1, as seen from Figure a. Our adder accomplishes error reduction when a carry is formed from the (n k 2)th bit position and subsequently propagated through the (n k 1)th bit position.

In a nutshell, the decrease occurs when Pnk1Cnk2 = 1. The correct output of the (n k 1)th bit under this input circumstance is 0, however our FA, as illustrated in Figure b, outputs 1 at this bit position. This implies that in this example, the estimated sum will be greater than the correct one. Rather than pushing the l LSB outputs to 1, the suggested adder changes all of the erroneous part's outputs to 0, (i.e., Snk2:0 = 0), bringing the approximation output closer to the actual addition. The ED, defined by |S approximate – S correct| where S approximate and S correct are approximate and precise mathematical abstractions, falls from 211 to 84 under the given input shown in Figure b. This reduction technique allows up to a 2n–k–1 – 1 decrease in the ED.

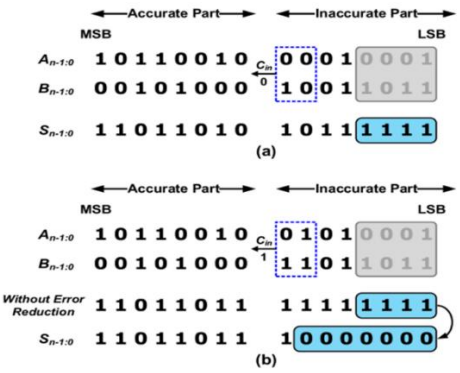

Fig.2: Operations of the adder; (a) constant 1 truncation and (b) constant 0 truncation with error reduction.

III. PROPOSED WORK

To improve performance, we'll substitute the Ripple Carry Adder with a Parallel Prefix Adder in this section. Brent-Kung adder, Han Carlson adder, Skylansy adder, Ladner Fischer adder, while others are all examples of Parallel prefix adders. For the precise part of addition, we're using a Ladner Fischer adder rather than a ripple carry adder. This is the 4-bit Ladner Fischer Adder which we've developed. For the erroneous portion of adding, the [4:0] bits of input a and b have been used. For the right section of adding, the [7:5] bits of input a&b are required. In order to execute the adding of [7:5] bits, the Ladner Fischer adder is made up of three bits. A 4-bit Ladner Fischer adder having extra full adders with 0 as inputs will be used to accomplish this.

Ladner Fischer Adder:

The projected Ladner-Fischer adder is versatile in terms of speeding up binary addition, as well as the organisation resembles a tree structure for excellent arithmetic performance. Due to their capacity to boost the speed of microcomputer platforms such as mobile communication, DSP, and telephony, FPGAs (field programmable gate arrays) have grown increasingly popular in recent years. Electronics development is focused on study into binary operations as well as motivation. There are two stages to building a good Ladner-Fischer adder. Pre-processing and generating are the two stages.

Pre-Processing Stage:

Throughout the pre-processing stage, every pair of inputs is generated as well as propagated. The propagate operation performs a XOR on the input bits, while the generate action performs a AND b on the input bits. Equations 4 & 5 demonstrate how to propagate (Pi) and create (Gi).

$$P_i=A_i \text{ XOR } B_i \text{ ------ (4)}$$

$$G_i=A_i \text{ AND } B_i \text{ ------ (5)}$$

Generation Stage:

Carry is created and transmitted for every bit in this phase, which would be known as carry generator (Cg) (Cp). Carry propagation as well as carry generate are produced for the data flows, but every bit unit's last column offers carry. The last bit carry will aid in the simultaneous summation of the next bits until the last bit is reached. Equations 6 & 7 show how to produce and propagate carry.

$$C_p=P_1 \text{ AND } P_0 \text{ ---------- (6)}$$

$$C_g=G_1 \text{ OR } (P_1 \text{ AND } G_0) \text{ ------ (7)}$$

In equations 6&7, the black cell is the carry propagate Cp, while the grey cell is the carry generation Cg. For the following procedure, a carry propagate is formed. Carry is determined by the final cell in each bit operation. The last bit carry will cause the following bit's sum to be added to the previous bit's sum until the last bit is reached. The carry produced is given in equations 8 and used for the following bit sum operation.

$$C_g=G_1 \text{ OR } (P_1 \text{ AND } G_0) \text{ ------ (8)}$$

A first bit's carry is XOR'ed with following bit of propagates, and the result is summation, as stated in equation 9.

$$S_i=P_i \text{ XOR } C_{i-1} \text{ ------- (9)}$$

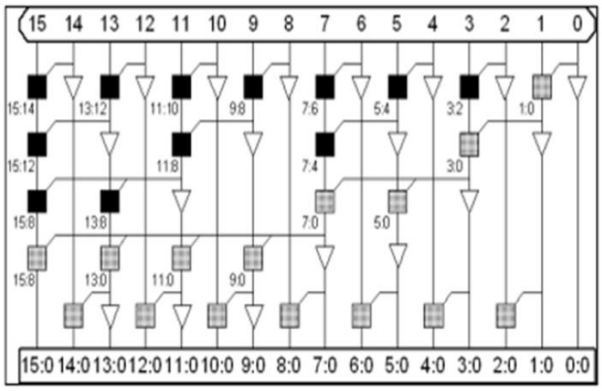

Fig.3: Structure of Ladner Fischer Adder for 16-bit

The mistake reduction carry prediction approximation adder is what we term our adder (ERCPAA). An1:0, Bn1:0, and Sn1:0 are the two n-bit input operands and one n-bit output of the adder,

respectively. Also, the I th LSBs of An1:0, Bn1:0, and Sn1:0 are represented by Ai, Bi, and Si, respectively.

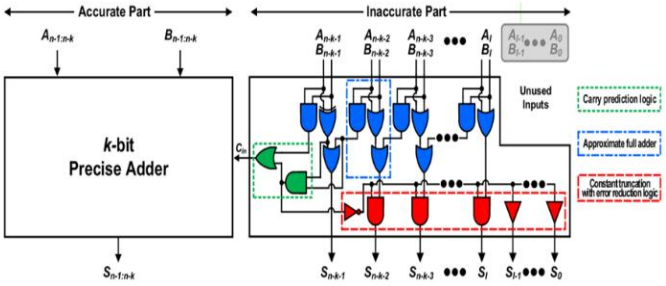

Fig.4 : Block diagram for ERCPAA using Ladner Fischer.

IV. EXPERIMENTAL RESULTS

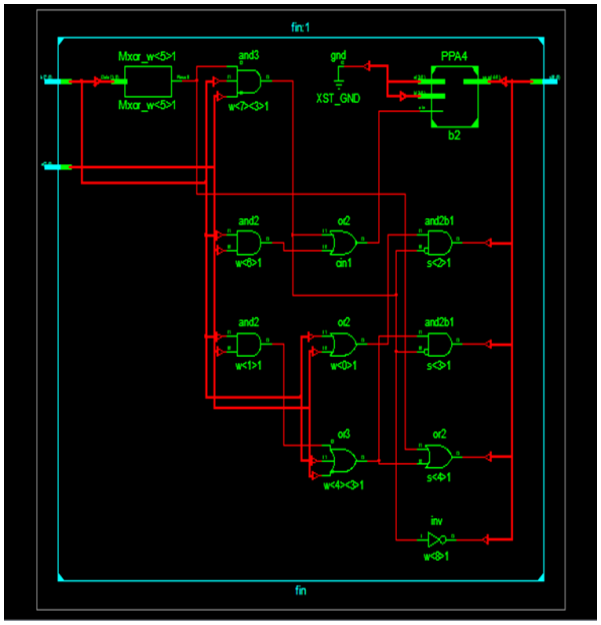

Fig.5: RTL schematic of proposed circuit.

Figure 5 shows RTL schematic of Proposed approximate adder

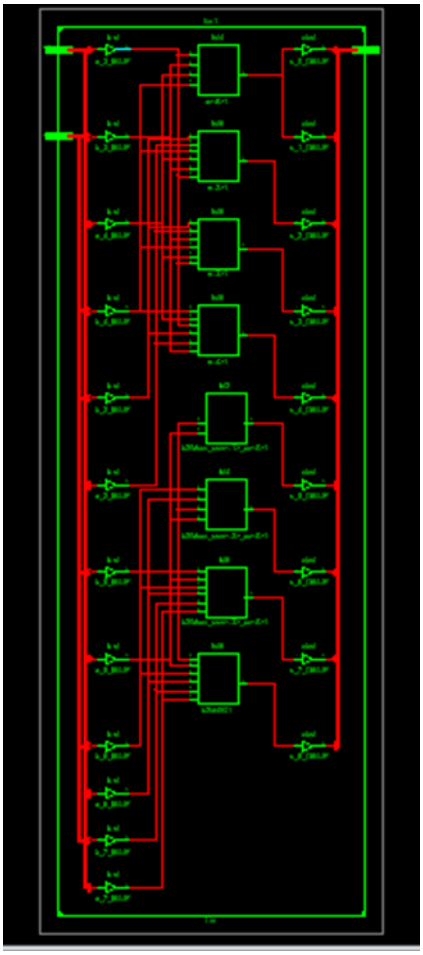

Fig.6 Technology schematic

Figure 6 displays the Technology schematic of our suggested circuit.

Fig7: Simulation results of proposed circuit.

Comparison between Existing and Proposed architectures.

Functionality	Area	Delay
Existing	9 out of 204000	1.460ns
Proposed	8 out of 63400	1.213ns

Table-1: comparison in terms of Area and Delay

From the results, mentioned in table-1, we conclude that Area and delay is reduced in the proposed architecture by using Ladner Fischer adder in the accurate part of addition.

V. CONCLUSION

Throughout this study, we offer a new approximation adder which integrates error-reduced carry forecasting plus consistent reduction techniques alongside error countermeasures. In comparison to the present approximate adder, the suggested carry prediction strategy reduces error rates, and the proposed error visualizations increases overall computing accuracy by reducing mistake distance. By modifying the adder design parameter, we were able to rigorously assess our architecture and determine the best balance of hardware costs as well as precision. In order to attain better performance results, the Ladner Fischer adder, one of the parallel prefix adders, was used.

VI. REFERENCES

[1]. Y. Kim, Y. Zhang, and P. Li, A reconfigurable digital neuromorphic processor with memristive synaptic crossbar for cognitive computing, ACM J. Emerg. Technol. Comput. Syst., vol. 11, no. 4, pp. 38:1–38:25, Apr. 2015.

[2]. B. Liu, Z. Wang, W. Zhu, Y. Sun, Z. Shen, L. Huang, Y. Li, Y. Gong, and W. Ge, An ultra-low power always-on keyword spotting accelerator using quantized convolutional neural network and voltage-domain analog switching network-based approximate computing, IEEE Access, vol. 7, pp. 186456–186469, 2019.

[3]. I. Khan, S. Choi, and Y.-W. Kwon, Earthquake detection in a static and dynamic environment using supervised machine learning and a novel feature extraction method, Sensors, vol. 20, no. 3, p. 800, Feb. 2020.

[4]. Q. Wang, P. Li, and Y. Kim, A parallel digital VLSI architecture for integrated support vector machine training and classification, IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 23, no. 8, pp. 1471–1484, Aug. 2015.

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

[6]. Y. S. Yang and Y. Kim, Approximate digital leaky Integrate-and-fire neurons for energy efficient spiking neural networks, IEIE Trans. Smart Process. Comput., vol. 9, no. 3, pp. 252–259, Jun. 2020.

[7]. A. Raha, H. Jayakumar, and V. Raghunathan, Input-based dynamic reconfiguration of approximate arithmetic units for video encoding, IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 24, no. 3, pp. 846–857, Mar. 2016.

[8]. T. Moreau, A. Sampson, and L. Ceze, Approximate computing: Making mobile systems more efficient, IEEE Pervasive Comput., vol. 14, no. 2, pp. 9–13, Apr. 2015.

[9]. S. Mittal, A survey of techniques for approximate computing, ACM Comput. Surv., vol. 48, no. 4, pp. 1–33, May 2016.

Cite this article as :

Nivedita, Dr. Anuradha Savadi, "Design of Approximate Adder Using Error Reduced Carry Prediction and Constant Truncation Employing Parallel Prefix Adder", International Journal of Scientific Research in Science and Technology (IJSRST), Online ISSN : 2395-602X, Print ISSN : 2395-6011, Volume 9 Issue 4, pp. 272-278, July-August 2022.
Journal URL : https://ijsrst.com/IJSRST229443