International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Arithmetic Adders for Approximate Computing

Joshi Viraj Vilas[1], Pravin Mane[2]

Submitted:11/03/2024 **Revised**: 26/04/2024 **Accepted**: 03/05/2024

Abstract: When the latest research trends in approximate arithmetic circuits are reviewed, it is observed that importance has been laid out for the novelty and development of certain architectures, designs, and layouts to make them universally adaptable and applicable in real-time applications. This area has been explored deeply by discussing various existing architectures and approximation techniques in arithmetic circuits, novel architectures and techniques, and comparative test reports. There is a huge scope of work to add novel approximation techniques in arithmetic circuits like adder and multiplier for error resilient and image processing applications. To provide solutions and guidelines to the existing and novel architectures and applications, this thesis addresses the work at the circuit simulation and synthesis level of abstractions. Through simulations, this work discusses the comparative resource parameter analysis for energy-efficient applications, and error analysis for error-resilient applications while introducing a new architecture for both said applications.

Keywords: *Approximate Computing, Error Resilient Applications, resource parameter analysis, Error Analysis*

1.Introduction

ADDITION is a key operation and basic building block for approximately all arithmetic circuits. Full adders are versatile building blocks in approximate adders because the approximation schemes are mostly applied at full adders [1]. For any fast adder, the key part is its carry network. Analysis of carry generation and propagation is done with the help of probability. After ripple carry adder, many techniques are introduced to design high-performance adders. Block-based adders are a common approach used in designing high-performance adders. These adders are composed of smaller sub-adder units, also called blocks or stages, which operate in parallel to compute the sum bits of the output. Each stage can handle smaller numbers of bits than the full adder, reducing the carry propagation delay and allowing for faster operation. Additionally, block-based adders can be optimized for different performance metrics such as area, power consumption, delay, and error characteristics. Various approximate adders have been proposed to improve the overall circuit's performance by reducing the delay associated with the carry propagation problem in conventional adders. Ten approximate adders are discussed and compared with the exact adder: CLA. All ten adders have unique architectures and are compared based on resource parameters analysis.

2.Carry Lookahead Adder

The Carry Look Ahead logic generates two output bits for each (i) th bit addition using equation (1) to get Genrt_i and Propgt_i using equation (2). These two bits signify the carryout from (i)th bit that goes as carry-in for the (i+ 1)th bit. If Genrt_i is 1 for (i)th bit, the carry-in to the (i + 1)th bit will always be 1. If the Propgt i is 1, then the carry-in to (i+ 1)th bit is the same as the carry-in to the (i) th bit.

$$Genrt\_i = Ai \times Bi \quad (1)$$

$$Propgt\_i = Ai \oplus Bi \quad (2)$$

where Ai and Bi refer to the two inputs. The Si and Ci+1 for the ith bit are found using the equations (3) and (4), respectively.

$$Si = Propgt\_i \oplus Ci \quad (3)$$

$$Ci+1 = Genrt\_i + Propgt\_i \times Ci \quad (4)$$

As we calculate Propgt_i and Genrt_i for all the values of i from 0 to the maximum of N-1 for an N-bit adder, these values are independent of the previous Propagate and Generate signals by invoking parallelism and opening the Ci+1 can be directly made dependent on C0 as shown in equation (5).

$$Ci+1 = Genrti + Genrti-1 \times Propgti + \ldots + C0 \prod_{i=1}^{i} Pi \quad (5)$$

An AND-OR circuit can simply form the above equation and has a delay of 2 gates. Now, we can get the Ci for all the N bits together in a 3-gate delay, and it will just require another 1-gate delay to find all the Sum (Si) bits. Equation (5), which presumably describes the critical path in the CLA design, has a fan-in of i+2i + 2i+2. This indicates that as you increase the number of inputs (N) for higher bit-width adders, the fan-in increases. Real gates typically have a practical limit on fan-in (often around 5), which can be exceeded as the bit-width of the adder increases. To build higher bit-width CLA adders, designers often cascade smaller CLA adders (e.g., cascading 4-bit CLA adders to

[1] *Department of Electrical and Electronics Engineering, Birla Institute of Technology and Science Pilani, Goa, 403726, India*
[2] *Department of Electrical and Electronics Engineering, Birla Institute of Technology and Science Pilani, Goa, 403726, India*
[3] ** Corresponding Author Email: p20180018@goa.bits-pilani.ac.in*

create a 32-bit CLA adder). This approach helps to manage the fan-in and design complexity. Cascading introduces delays because the carry and sum outputs of each stage must propagate through the previous stages. For instance, in a 32-bit CLA composed of 4-bit CLA adders, the delay for finding C32 (carry out of the 32nd bit) is 17 gates, and S31 (sum of the 31st bit) is settled after 18 gate delays. Despite these delays, this approach is still superior to a 32-bit ripple-carry adder, which has significantly higher delays of 63 gates for S31 and 64 gates for C32. Ripple-carry adders (RCA) have a linear delay proportional to the number of bits, which makes them less efficient for large bit-widths due to the long critical path delays [2],[3],[4].

Figure 1 shows the detailed architecture of CLA adder.

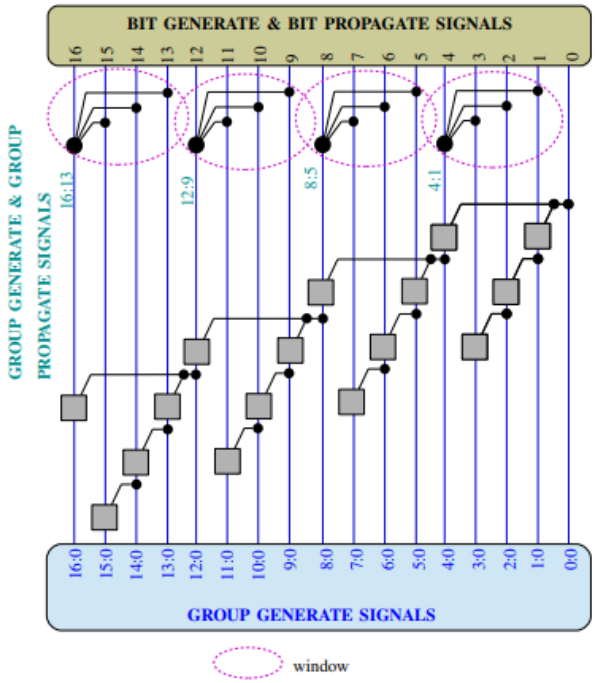


Fig. 1 Carry Lookahead Adder

3. ACA1: Almost Correct Adder-I

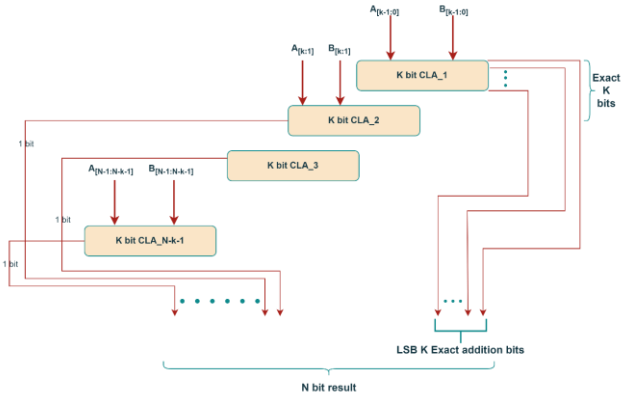


Fig. 2 Almost Correct Adder-I

ACA-I exploits the idea that for most additions the longest carry propagation chain is smaller compared to the total number of bits. Hence by using multiple sub-adders of the length of the maximum carry propagation chain, an accurate result can be produced. It is proposed to use the multiple overlapping sub-adders with one resultant bit per sub-adder. Due to the fixed single-bit shift operation, sub-adder count increases along with the fan-out of the input, which in turn increases the area overhead. Error detection and correction were also proposed for ACA-I [5]. The detailed architecture of ACA-I is shown in the figure 2.

It is one of the first approximate adders. The ACA-1 architecture exploits optimization in speed from an assumption that the maximum carry chain propagation (maximum distance between the carry generation bit followed by a long propagation chain right up to the concerned bit) is somehow known to us. If we are concerned about the result at some mth bit and suppose that the carry is generated at $(m - k)^{th}$ bit. If the carry generated at the $(m - k)^{th}$ bit can propagate to the m^{th} bit, then the maximum carry chain propagation length is k. By considering the above steps, one assumes that the result at the m^{th} bit is solely dependent on the value from $(m-1)^{th}$ bit to $(m-k)^{th}$ bit, which is not the case in reality. To get a 100 percent correct result, we need to be aware that the result at the m^{th} location depends on the $(m - 1)^{th}$ location to the 0^{th} location. The ACA-I, being an approximate adder, assumes and, therefore, gives incorrect results in a few cases depending on the length of maximum carry propagation used to select the size of the sub-adder. If the carry propagates even from $(m - k - 1)^{th}$ bit, in reality, it contributes to an error case for the ACA-I adder.

Further, ACA-I adder is used in ACA-I Multiplier. ACA-I has proved to be the fastest approximate adder among selected adders at the cost of more area utilization. The said adder can be used in GPUs and will be a great addition to ML in the hardware domain. The Approximate Circuit extensions can be implemented on RISC-V ISA to provide variable frequency operating modes.

4. ACA-II: (Accuracy configurable adder-II)

Approximate adder ACA-II architecture is as shown in figure 3, that uses multiple overlapping sub-adders with half of sub-adder length resultant bits per sub-adder. Error detection and correction scheme was also proposed for ACA-II [6].

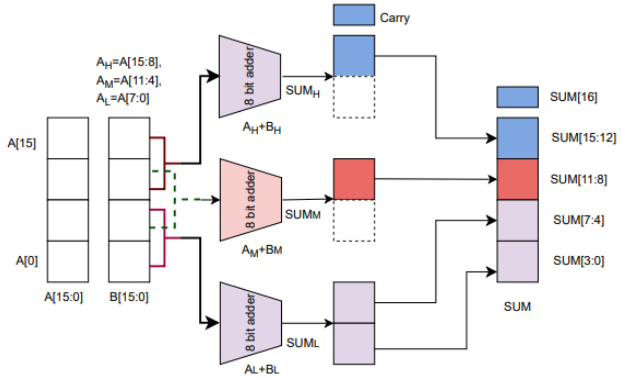
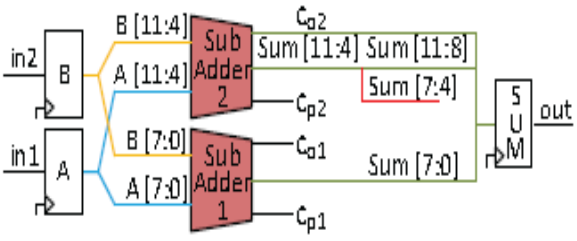
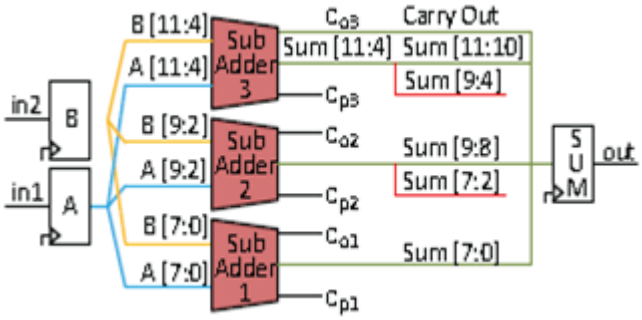


Fig. 3 Proposed approximate adder (ACA-II)- 16 bit adder case

Each sub-adder within the ACA adder architecture checks for errors by comparing its output with the carry-in signal received from the previous sub-adder. This mechanism helps identify inaccuracies introduced during propagation. Upon detecting an error, the ACA adder employs a simple error correction strategy. It adds '1' to the approximate output to rectify the error. This correction is implemented using an incrementor circuit, which minimizes the overhead while ensuring accuracy improvements. The ACA adder utilizes 'AND' gates for error detection, keeping the overhead minimal. This design choice ensures that the adder remains efficient even when operating in accuracy-configurable modes. In the future, the concept of accuracy-configurable designs isn't limited to adders alone. There are ongoing efforts to extend this approach to other arithmetic components such as multipliers and multi-input adders. This expansion aims to create a flexible framework where different parts of the arithmetic logic unit (ALU) can adapt their accuracy dynamically to match the requirements of diverse computational tasks. In essence, the ACA-II adder represents a significant advancement in arithmetic circuit design by offering flexibility in accuracy trade-offs without compromising performance. It provides a practical solution for applications where computational accuracy requirements vary dynamically or can be predefined based on specific workload characteristics.

5. GeAr : (Generic Accuracy Configurable Adder)





GeAr adder with N=12, R=4, P=4, and k=2

GeAr adder with N=12, R=2, P=6, and k=3

Fig.4 Generic Accuracy Configurable Adder

GeAr (Generic accuracy configurable adder) is a modified version of ACA-II. As shown in figure 4, the total adder is divided into sub-adders and the internal sub-adders are working parallel. All sub-adders are strictly of same size. The first adder is always an exact adder and remaining operate in approximate mode. Carry Look ahead Adder (CLA) is used for addition [7].

If 'N' is the total size of the adder , 'K' is number of result bits for each sub adder except the first (lowest) one. Then the number of sub adders (n) are calculated using this formula:

$$n = [N/K - 1] \quad (6)$$

If 'p' is number of prediction/overlap bits and 'R' is the result bits then ACA-II works on the condition of K=P=R .The sub adder size=2K and should be completely divisible to total size to produce remainder zero.

$$\text{Total adder size } (N) = R+P+(n-1) *R \quad (7)$$

As said above, this adder is the modified version of ACA-II adder. The architectural difference between ACA-II and GeAr is that GeAr is flexible to have result bits (R) equal to prediction bits (P) or not. ACA-II allows only R=P.

The number of sub adders (k) can be calculated as :

$$K = ((N-L)/R)+1 \quad (8)$$

GeAr adder is designed using three key parameters N, R, and P.

By setting parameters R=1 and P=L−1, GeAr can be configured to resemble ACA-I, an approximate adder design. This configuration prioritizes lower latency an d reduced complexity at the cost of accuracy. Setting R=L/2 and P=L/2 aligns GeAr with ACA-II and ETAII configurations. These configurations strike a balance between accuracy and performance by employing moderate error prediction and correction mechanisms. GeAr can also mimic GDA adders, where the number of carry prediction bits is uniform across all sub-adders. This approach ensures consistency in error prediction and correction strategies

throughout the adder structure. Implemented with AND gates, GeAr identifies errors introduced during computation, ensuring the integrity of results, particularly when high accuracy is required. For error correction, GeAr employs sub-adders, OR gates, and multiplexers tailored to the adder size.

6. 6. GDA (Gracefully degrading adder)

A reconfiguration-oriented design methodology for approximate circuits is presented in figure 5, which proposes a reconfigurable approximate adder design that degrades computation quality gracefully [8].

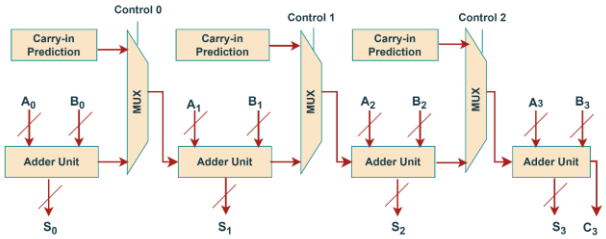


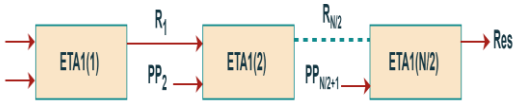
Fig. 5 Gracefully Degrading Adder (4-bit)

GDA, while offering some configurability in carry prediction, is constrained by its fixed architectural decisions and tends to have higher area overhead and potential latency issues compared to more modern designs like GeAr. The choice between these designs depends largely on specific application requirements for accuracy, speed, and resource utilization. GDA employs a hierarchical carry look-ahead structure for carry prediction, where the number of carry prediction bits is constrained to multiples of the sub-adder length R. It uses ripple carry adders (RCA) for sub-adders combined with configurable carry prediction circuits akin to carry look-ahead (CLA) adder components. Error recovery is facilitated even in configurations like ETAII and GDA, which originally lack error correction mechanisms. Only thing is that the complexity of GDA's carry prediction circuitry, similar to CLA adders, results in significant area overhead compared to more streamlined designs like GeAr.

GDA's design limits the flexibility in configuring prediction bits, which must align with the sub-adder length RRR. This restriction may impact its adaptability across different computational demands. GDA's performance, particularly in terms of latency, can be less optimal compared to other adder types like GeAr, due to its fixed structure and reliance on hierarchical carry look-ahead techniques.

7. ETA-I: (Error Tolerant Adder-I)

This is the proposed architecture of ETA-I adder [9]. As described in figure 3.6 and 3.7, ETA-I is a block-based adder. It is divided into two sub parts, accurate part and inaccurate part. Out of that the accurate part consists of ripple carry adder with carry-in always equal to zero and inaccurate part consists of Modified EXOR gates. Modified EXOR gates are simple logical EXOR gates working with the principle of Error Tolerance. The inaccurate part acts as an EXOR Gate till a combination of two 1's is found at a particular bit position. When this combination is detected, all the trailing bits from that bit position are set to 1 irrespective of their actual value and this is done with the help of a 2:1 Multiplexer whose select line is controlled set to 1 when the combination of two 1's is found in the operands' bit stream. The biggest advantage of ETA-I architecture is that there is no carry propagation in the inaccurate part. The critical delay path is because of the circuitry used for generating the control signal for the select line of the multiplexer. The adder's critical delay is due to carry chain propagation. ETA-I uses a lesser number of accurate bits than the total size of the adder hence carry propagation chain is significantly reduced. As the ETA-I uses an exact adder like Ripple Carry Adder for an accurate part, carry is propagated only through the accurate part instead of from LSB to MSB. Accurate and inaccurate parts work parallelly to compute the end results generating lesser delay compared to the conventional adders.



R----Result of previous stage

PP---Partial Product Terms

R0---Is always zero

Fig. 6 Error Tolerant Adder-I

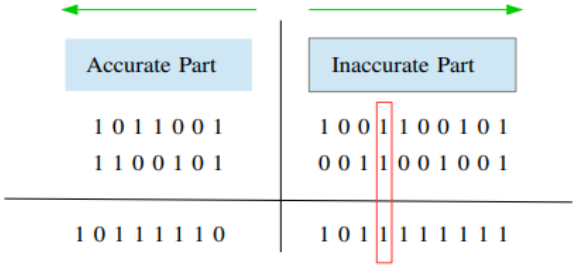


Fig.7 Example of ETA-I addition

The accuracy of the addition operation can be adjusted by changing the number of the accurate bits. Higher the accurate bits, better the accuracy. By eliminating the carry propagation path in the inaccurate part and simultaneously performing addition in two separate parts, the overall delay time is greatly reduced. So is the power consumption.

Not all digital systems can employ the error-tolerant circuit. In some digital systems, such as a control system, the correctness of the output signals is extremely important and hence eliminating the use of error-tolerant circuit. For small inputs, ETAI produced inaccurate results as the inaccurate

part of the adder was used. To handle the limitations of ETAI another adder ETAII was proposed.

8. ETA-II: (Error Tolerant Adder-II)

Different from ETAI, ETAII does not eliminate the entire or part of the carry propagation path. Instead, it splits the entire carry

propagation path into a number of short paths and completes the carry propagations in these short paths concurrently as shown in figure 8. In this way, the speed performance of the adder can be significantly improved and with almost no degradation in its power consumption [10].

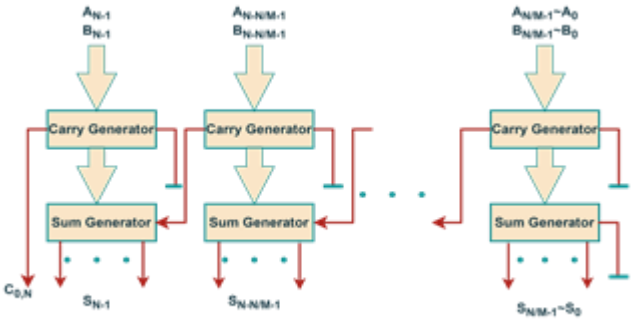


Fig. 8 Error Tolerant Adder-II

ETAII is designed to reduce carry propagation delay by dividing the total number of bits into multiple sub-adders. Each sub-adder operates independently with its own carry generation and propagation, which limits the maximum carry propagation to twice the length of each sub-adder. This approach aims to improve performance by reducing overall delay.

However, ETAII faces challenges related to accuracy, especially with larger inputs, where its accuracy can degrade. To address these issues, ETAIIM was proposed as a modified version. ETAIIM retains the division of the adder into sub-adders but modifies the carry and sum generation mechanisms. Notably, the carry generator in ETAIIM does not receive carry-in signals from the previous block, whereas the sum generator does, thereby limiting carry propagation between neighboring blocks rather than across the entire adder structure [11].

The choice of how to divide the adder into these blocks (M sub-adders of equal size) affects its performance characteristics. Fewer blocks (larger sub-adders) can reduce the likelihood of errors but increase delay, while more blocks (smaller sub-adders) can improve speed but may increase error potential.

Both ETAII and ETAIIM utilize different strategies for carry and sum generation: ETAII uses CLA (Carry Look Ahead) for carry and RCA (Ripple Carry Adder) for sum, while ETAIIM appears to modify this approach to address specific issues encountered in ETAII.

It's also noted that neither ETAII nor ETAIIM include an error recovery scheme, suggesting that their design assumes error avoidance rather than correction, which is a critical consideration in reliable computing systems.

9. SARA (Simple Accuracy Reconfigurable Adder)

Figure 9 represents the basic structure of SARA. SARA represents a significant advancement in the design of accuracy configurable adders, leveraging carry-prediction techniques and efficient use of sub-adders to achieve high performance with reduced area and power consumption. Its simplicity, coupled with effective optimization strategies, positions it as a competitive choice for applications requiring both speed and accuracy in arithmetic operations [12].

SARA utilizes a carry-prediction-based approach, distinguishing it from traditional adder designs like CLA (Carry Look Ahead). This approach typically aims to predict carry bits based on previous computations, thereby reducing delay and potentially improving overall performance. Unlike some previous designs that use CLA (which can be more area-intensive and complex),

SARA employs CRA for its sub-adders. CRA is generally simpler and more area-efficient when compared to CLA, aligning with SARA's goal of reducing area while maintaining performance. SARA optimizes its carry prediction mechanism by reusing parts of the sub-adders their selves, rather than incorporating dedicated prediction circuitry. This approach helps to avoid additional overheads associated with prediction circuits, which can contribute to both area and power savings.

SARA offers significant advantages over previous designs like GDA and RAP-CLA. It claims to achieve 50% less Power Delay Product (PDP) compared to GDA while maintaining a comparable Peak Signal-to-Noise Ratio (PSNR). Additionally, SARA demonstrates better accuracy-power-delay trade-offs than RAP-CLA, showcasing its efficiency in balancing these critical metrics. SARA incorporates a delay-adaptive reconfiguration technique, which enhances its ability to adjust and optimize performance characteristics based on operational requirements. This technique likely contributes to its improved accuracy-power-delay trade-off, making it adaptable to varying computational needs.

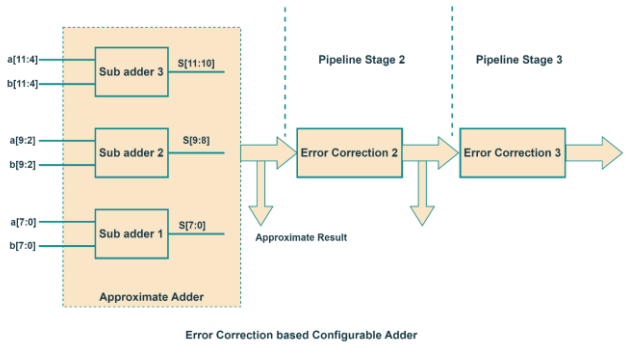


Fig.9 Simple Accuracy Reconfigurable Adder (SARA)

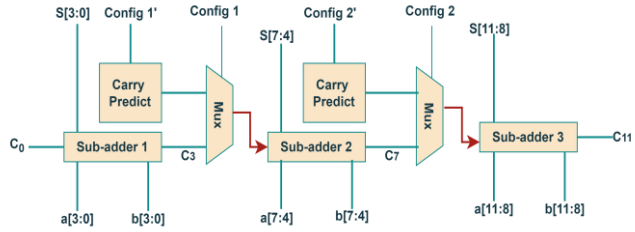


Fig.10 Carry-prediction-based-configurable adder

As shown in fig.10, an N-bit adder is composed of K segments of L-bit sub-adders, where K = [N/L]. Each sub-adder is almost the same as a CRA except that the MSB (Most Significant Bit) of a sub-adder, which is bit i, provides a carry prediction as:

$$c\ prdt\ i = gi \tag{9}$$

For the LSB (Least Significant Bit) of the higher-bit sub-adder, which is bit i + 1, its carry-out ci+1 can be computed using one of two options: either by the conventional:

$$ci+1 = gi+1 + pi+1 \cdot ci \tag{10}$$

or by using the carry prediction as:

The selection between the two options is realized using MUXes and the MUX selection result is denoted as cˆi. The carry prediction is a truncation-based approximation to carry computation.

Therefore, cˆi can be configured to either accurate mode or approximation mode, i.e., cˆi ← (c prdt i , if approximation mode ci , if accurate mode. It should be noted that the carry prediction c prdt i reuses gi in an existing full adder instead of introducing an additional dedicated circuit. This prediction scheme makes a very simple modification to the conventional full adder. One can connect cˆi to its higher bit i + 1 to compute both carry ci+1 and sum si+1, as in GDA and RAP-CLA.

SARA achieves improved accuracy in sum computation with minimal additional overhead compared to GDA and RAP-CLA. This is advantageous because it enhances computational reliability without significantly increasing design complexity or area usage.

SARA's overhead mainly consists of MUXes, which are essential for its configurability. This overhead is minimal compared to more complex adders like CLA and is optimized to be as efficient as possible for configurable adders.

When c^i is configured to equal ci for all K sub-adders, SARA operates similarly to a CRA. In this mode, the critical path extends along N-bit full adders. Configuring c^i to be cprdt_i across all sub-adders shortens the critical path to approximately L-bit full adders. This reduction in critical path length not only improves performance by reducing delay but also enables potential power savings through supply voltage scaling. In worst-case scenarios where the predicted carry (cprdt_i) differs from the actual carry (ci), errors can occur. The likelihood of errors typically increases with the number of sub-adders K, which introduces a trade-off between error rate, area, power consumption, delay, and accuracy. Larger values of K imply smaller values of L, leading to shorter critical paths and potentially reduced power consumption. However, this needs to be balanced against the increased error rate and area overhead associated with more sub-adders. The goal is to find an optimal configuration that maximizes accuracy while minimizing power consumption and delay. When faced with multiple configurations with the same critical path length, prioritizing the configuration that offers higher accuracy becomes crucial. This ensures that the design meets performance requirements while maintaining reliability in computations. SARA represents a versatile approach to configurable adder design, leveraging efficient configurations and carry-prediction techniques to enhance accuracy and performance. Its ability to adapt critical path lengths and configurations based on operational needs makes it a competitive choice in modern computing systems where efficiency and reliability are paramount.

$$ci+1 = gi+1 + pi+1 \cdot c\ prdt\ i = gi+1 + pi+1 \cdot gi \tag{11}$$

SARA is area efficient than RAP_CLA and GDA.

SARA's self-configuration technique recognizes that the worst-case path delay in adder operations often depends on the specific values of the addends being processed. When the actual carry propagation chain is short due to addend values , there is no necessity to activate the approximation configuration. The approximation configuration typically aims to shorten the carry propagation chain to reduce delay and potentially save power, but it introduces a risk of introducing errors. SARA-DAR (Delay-Adaptive Reconfiguration) builds upon this principle. It extends the self-configuration concept by dynamically adjusting its configuration based on real-time assessment of the addends' impact on carry propagation. By implementing self-configuration, SARA and its derivative SARA-DAR enhance flexibility and adaptability in adder design. They mitigate the complexity associated with static configuration

decisions and ensure that computational resources are utilized optimally based on actual operational conditions [12] In conclusion, SARA's self-configuration technique, particularly in the context of SARA-DAR, represents a significant advancement in the field of reconfigurable adder design.

10. SARA-DAR: (Simple Accuracy Reconfigurable Adder with Delay-Adaptive Reconfiguration)

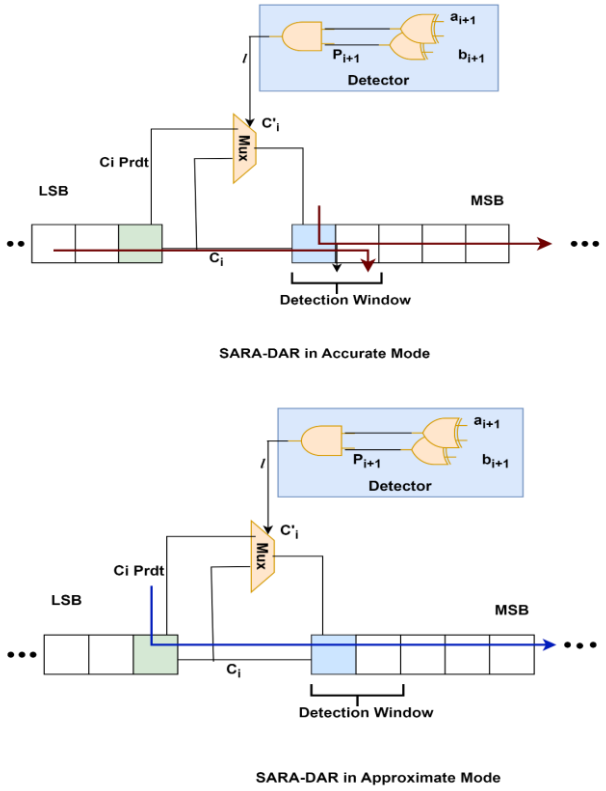


Fig.11: Design of DAR for SARA operating in two different modes

The SARA-DAR (Simple Accuracy Reconfigurable Adder with Delay Adaptive Reconfiguration) introduces a Delay Adaptive Reconfiguration (DAR) technique to optimize performance and accuracy dynamically based on the characteristics of carry propagation through adder circuits.

As shown in figure 11, a MUX determines whether to operate in approximation mode or accurate mode based on detected carry propagation characteristics. The MUX switches to approximation mode only when a potentially long carry chain is detected, which helps in reducing delay and power consumption. By dynamically switching modes, SARA-DAR can avoid errors that may occur in approximation mode for short carry chains, while still benefiting from delay and power reduction for longer carry chains that can be shortened. The MUX adjusts its detection window based on the presence of false propagate bits. If a false propagate bit is detected, the MUX remains in accurate mode to maintain the carry chain length effectively. The detection mechanism involves minimal overhead, typically just one NAND gate per MUX. This ensures efficient operation with minimal additional hardware.

In approximate mode, it limits the effective carry chain length to no greater than L+1 bits, where L is the length of the full adder. In accurate mode, it allows for carry chain lengths within L+2 bits, providing higher accuracy. The size of the detection window W determines the trade-off between accuracy and the effective carry chain length in accurate mode. A larger W reduces error rates but increases the critical path length in accurate mode.

SARA-DAR is primarily designed for unpipelined implementation, though it can support pipelined operation. Unpipelined designs simplify performance evaluation compared to pipelined designs, which require additional clock cycles for error correction and computation. The addition of DAR hardware introduces some area overhead, mainly due to the NAND gates used for detection. However, this overhead is kept minimal to maintain efficiency.

SARA-DAR's ability to dynamically adjust its configuration based on real-time carry propagation characteristics optimizes both performance and accuracy. By utilizing simple detection mechanisms and minimal additional hardware, SARA-DAR achieves effective delay reduction without compromising accuracy unnecessarily. SARA-DAR represents a sophisticated approach to accuracy configurable adder design, leveraging Delay Adaptive Reconfiguration to enhance performance and maintain reliability in varying computational scenarios. Its focus on dynamic mode switching and minimal overhead makes it a compelling choice for applications where both speed and accuracy are critical [12].

11. RAP-CLA (Reconfigurable approximate-Carry lookahead adder)

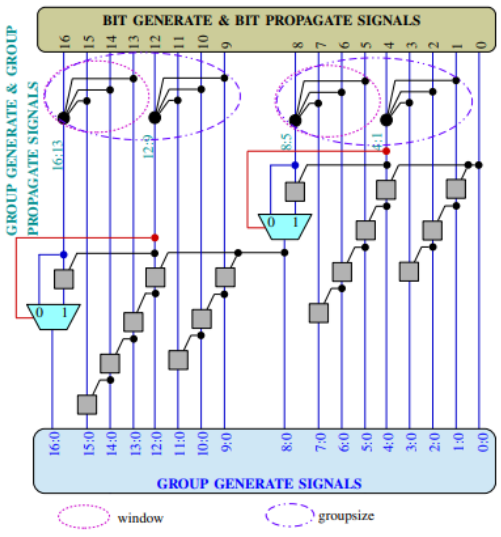


Fig. 12 Reconfigurable approximate-Carry lookahead adder

The GDA and RAP-CLA methods represent two different approaches to designing accuracy configurable adders, each

with its own strengths and trade-offs. RAP-CLA builds upon the accurate Carry Look-Ahead (CLA) adder design by incorporating reconfigurable elements. It uses a portion of the CLA circuitry for carry prediction, which helps in reducing overall area compared to a full CLA implementation. RAP-CLA benefits from the accuracy of CLA during normal operation. It utilizes the reconfigurable carry prediction to switch to approximate mode when necessary, thereby balancing performance and power efficiency. RAP-CLA achieves a smaller area footprint compared to a full CLA adder, yet it typically occupies more area than simpler adder designs like basic CLA due to the added complexity of reconfigurable elements. RAP-CLA manage errors through carry prediction mechanisms eliminating the need for traditional error correction codes and minimizing data stalls. RAP-CLA, while more complex due to its reconfigurable nature, offers a balance between accuracy and area efficiency by leveraging the existing CLA architecture. RAP-CLA is preferred for applications demanding finer control over accuracy and area efficiency [13].

The structure of RAP-CLA was based on some modifications to the structure of the conventional CLA as shown in figure 12. RAP-CLA uses CLA as its baseline architecture. In CLA, each bit's carry-out is computed in parallel by considering the carry-in from lower-order bits, making it efficient for high-speed addition. RAP-CLA optimizes area efficiency by reusing a portion of the CLA circuitry for carry prediction rather than introducing dedicated prediction circuitry. The reconfigurable carry prediction mechanism in RAP-CLA dynamically adjusts based on input conditions or configuration settings, enabling the adder to switch between accurate and approximate modes as needed. By integrating reconfigurable carry prediction within the CLA framework, RAP-CLA achieves a practical balance between performance, area efficiency, and flexibility. This makes it suitable for applications where both speed and adaptability are crucial factors.

In summary, RAP-CLA stands out as a robust solution for configurable adder design, providing superior accuracy, performance efficiency, and versatility compared to GeAr. Its ability to operate effectively in both approximate and exact modes ensures optimal performance across a range of computational tasks and application requirements.

12. Modified RAP-CLA

Proposed modified RAP-CLA architecture is as shown in figure 13. It makes use of PG technique in a same way like RAP-CLA I with more architectural flexibility. Proposed circuit is a well combination of CLA with cognizance to error resilience. When compared with RAP-CLA described by A. Omid et.al, proposed RAP-CLA has a small architectural difference w.r.t. multiplexer component. Multiplexer select line is controlled by external signal instead of circuit generated signal. Mux select line is generated by lower 4 bits valency black cell in each individual group with size equal to (groupsize-windowsize). In proposed adder architecture

1. the multiplexer select line is independently set/reset by external signal (ApproxRCON). ApproxRCON signal which is equal to Adder Size/Group Size.

2. parallel computations of carries to reduce the delay of the n-bit additions.

3. carry prediction circuit reuses a part of look ahead circuit rather than building extra dedicated prediction circuitry.

4. while operating in approximate mode, part of a circuit is getting bypassed which will reduce hardware complexity with improved resource parameters.

Other than this, proposed approximate adaptive carry-lookahead adder for error resilient applications is flexible to work in approximate as well as exact mode. Proposed architecture differs from previous RAP-CLA design. The architectural differences are highlighted as: 1. The multiplexer input ”1” is fed by output of higher 4 bits valency black cell and input ”0” is fed by gray cell output connected at highest bit-position in lower group of group size. Inputs to this gray cell are one from lower 4 bits valency black cell and another from output of previous group or Cin in case of lowest group. 2. Select line of multiplexer is defined as ”ApproxRCON” and it is controlled by external signal either 0/1. Proposed adder works in exact mode when select line signal set to zero and in approximate mode when ApproxRCON set to 1. During approximate mode, part of circuit is bypassed [3 bits at LSB side in each group] due to which resource parameters improve with reduced hardware complexity but error rate exceeds at some extent [13].

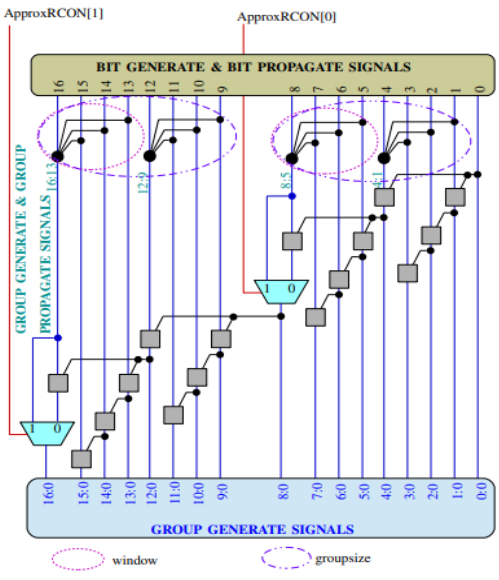


Fig.13 Modified RAP-CLA

13. Results and discussion

The role of approximation in arithmetic circuit like adder was demonstrated through a comparative analysis of the ten approximate adders with different architectures and one exact adder for bit width of 16 bits and tabulated in table no. 1. The simulation results expressed in figure number 14, 15 and 16 illustrate that area wise ETA-I shows the best performance with least area requirement of 95.418 um2 and it is 35.41% efficient when compared with CLA. When power consumption is considered, again ETA-I utilizes the least power of 2.8584 uW and 30.30% power efficient than CLA. In case of delay, ACA-I proved to be the fastest among all with 847 pS. It is 63.64% faster than CLA. It concludes the importance of approximate adders for resource parameter improvements over accurate adders.

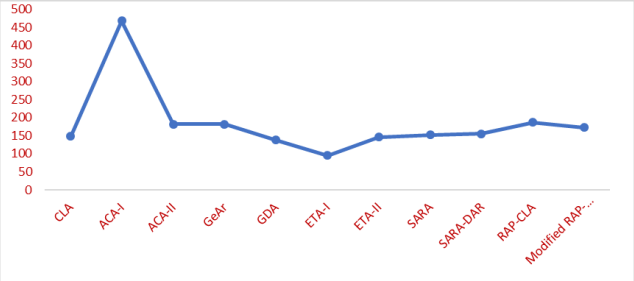


Fig.14 Area plot for selected adders

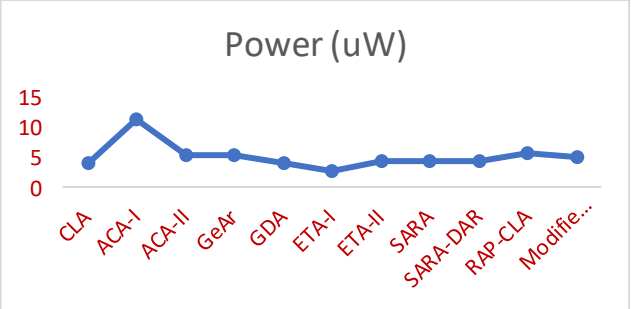


Fig.15 Power plot for selected adders

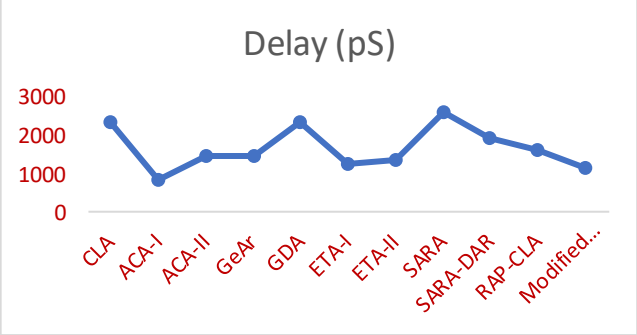


Fig.16 Delay plot for selected adders

Table 1. TAP Analysis of Different adder's architectures

Parameter	Exact Adder	Approximate Adders									
	CLA	ACA-I	ACA-II	GeAr	GDA	ETA-I	ETA-II	SARA	SARA-DAR	RAP-CLA	Modified RAP-CLA
	N=16, Valency =4	N=16, K=8, P=7, R=1	N=16, K=8	N=16, R=8, P=8	N=16,K=4, T=8 K=G.S. T=CLA _approximation _bits	N=16, A=8, IA=8, K=9	N=16, G.S.=8	N=16, G.S.=8	N=16, G.S.=8, W=4	N=16, G.S.=8, W=4	N=16, G.S.=8, W=4, ApproxRcon=1
Area	147.744	468.5	181.944	181.944	138.51	95.418	146.034	152.532	154.926	186.732	172.368
Power (uW)	4.089	11.44	5.311	5.311	4.17	2.8584	4.44	4.266	4.324	5.601	5.04
Delay (pS)	2330	847	1444	1444	2337	1257	1359	2604	1916	1584	1152

14. Conclusion

Approximate circuits itself designed to produce inaccurate results which are not wrong but slightly deviated from accurate one. The purpose is to improve system's performance parameters. Arithmetic adders are the integral parts of digital circuits. The comparative study concludes with the use of a particular arithmetic adder for a specific application. But in few situations it is mandatory to switch from inaccurate to accurate results during run time applications and usually it is called as dynamic approximate computing. This is the future scope of the approximate adders.

References

[1] S. Purohit and M. Margala , "Investigating the Impact of Logic and Circuit Implementation on Full Adder Performance," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 7, pp. 1327-1331, July 2012, doi:

10.1109/TVLSI.2011.2157543

[2] V. Vijay et al. “A Review on N-bit Ripple Carry Adder, Carry Select Adder, and Carry-Skip Adder”. Journal of VLSI Circuits and Systems,2022, vol. 4, no. 1, pp. 27-32.

[3] Singh, Gurinder & Nidhi,. (2014). Efficient Design of Ripple Adder and Carry Skip Adder with Low Quantum Cost and Low Power Consumption. Int. Journal of Engineering Research and Applications. 4.

[4] Hamacher, C., Vranesic, Z., Zaky, S., Manjikian, N. “Computer Organization and Embedded Systems”, 2012, (6th ed.), McGraw Hill; Standard Edition (9 January 2023); McGraw Hill Education (India).

[5] Verma, A. K., Brisk, P., Ienne, P. “Variable Latency Speculative Addition: A New Paradigm for Arithmetic Circuit Design”. 2008 Design, Automation and Test, Europe, 2008, pp. 1250-1255.

[6] B. Kahng and S. Kang. “Accuracy-configurable adder for approximate arithmetic designs”. DAC Design Automation Conference 2012, San Francisco, CA, USA, 2012, pp. 820-825.

[7] M. Shafique, W. Ahmad, R. Hafiz and J. Henkel, "A low latency generic accuracy configurable adder," *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, San Francisco, CA, USA, 2015, pp. 1-6, doi: 10.1145/2744769.2744778.

[8] R. Ye, T. Wang, F. Yuan, R. Kumar and Q. Xu, "On reconfiguration-oriented approximate adder design and its application," *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, San Jose, CA, USA, 2013, pp. 48-54, doi: 10.1109/ICCAD.2013.6691096.

[9] Kyaw, K. Y., Goh, W. L., and Yeo, K. S. (2010). Low-power high-speed multiplier for error tolerant application. In 2010 IEEE International Conference of Electron Devices and SolidState Circuits (EDSSC), pages 1–4.

[10] Zhu, N., Goh, W. L., and Yeo, K. S. (2009). An enhanced low-power high-speed adder for error-tolerant application. In Proceedings of the 2009 12th International Symposium on Integrated Circuits, pages 69–72.

[11] R. Ye, T. Wang, F. Yuan, R. Kumar and Q. Xu, "On reconfiguration-oriented approximate adder design and its application," *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, San Jose, CA, USA, 2013, pp. 48-54, doi: 10.1109/ICCAD.2013.6691096.

[12] W. Xu, S. S. Sapatnekar, and J. Hu, “A simple yet efficient accuracy configurable adder design,” Transactions on Very Large Scale Integration (VLSI) Systems, IEEE vol. 26, no. 6, pp. 1112–1125, 2018.

[13] O. Akbari, M. Kamal, A. Afzali-Kusha and M. Pedram, "RAP-CLA: A Reconfigurable Approximate Carry Look-Ahead Adder," in *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 8, pp. 1089-1093, Aug. 2018, doi: 10.1109/TCSII.2016.2633307.