Symmetric Encryption Key Generation Method with Approximate Computation

Kun-Lin Tsai[1]*, Feng-Yi Leu[2], Chi-Hung Lo[3], Li-Woei Chen[4], and Jin-Wei Ye[1]

[1] Department of Electrical Engineering, Tunghai University, Taichung, Taiwan
{kltsai, s09360202}@thu.edu.tw

[2] Department of Computer Science and Information Engineering, Tunghai University, Taichung, Taiwan
leufy@thu.edu.tw

[3] Department of Industrial Design, Tunghai University, Taichung, Taiwan
chlo@thu.edu.tw

[4] Department of Computer and Information Sciences, Chinese Military Academy, Kaohsiung, Taiwan
Chen.Li.woei@gmail.com

Abstract

Approximate computing has gained popularity as a technique for designing power-efficient circuits with reduced complexity by sacrificing precision in favor of energy efficiency. Such operations find common use in applications where a high degree of accuracy is not essential, such as image processing on mobile devices. However, in the field of security, precise computation is a fundamental necessity, and integrating approximate computing into cryptography poses significant challenges. As a result, leveraging the advantages of approximate computing in security applications remains a complex endeavor. In this paper, a Symmetric Encryption Key generation method with Approximate computation (SEKA) is proposed to generate a data encryption key for both transmitter and receiver. The SEKA utilizes a system time parameter to control different operation modes of the approximate computing so that a highly random number can be generated to be the symmetric encryption key. This approach combines the advantages of reduced computing complexity and heightened encryption key security. Simulation results highlight SEKA's capability to deliver a highly secure data encryption key while minimizing computational demands, making it particularly well-suited for energy-constrained devices.

Keywords: Symmetric encryption, Approximate computation, Security, Key generation

1 Introduction

In the field of computer science, the adder [1][2] is one of the important components that performs basic operations. Various applications, including digital signal processing, AI computing [3], data encryption and decryption [4], image processing [5], etc. heavily rely on addition operations. Particularly in artificial intelligence-related applications, the need to process extensive data sets is paramount. Therefore, improving the computing performance of the adder and reducing the power consumption required for the adder operation are two important issues that many studies are trying to solve.

Approximate computation is a design method that strikes a balance between computational cost and computational accuracy [6][7]. By intentionally sacrificing a degree of accuracy, the

*Corresponding author: Department of Electrical Engineering, Tunghai University, 407224, Xitun District, Taichung, Taiwan, Email: kltsai@thu.edu.tw circuit complexity can be reduced. Adders designed with this approximation approach exhibit lower accuracy but consume less energy. These adders maintain essential circuit functionality even if their operations are not entirely precise, thus significantly reducing computational expenses such as power consumption, latency, and circuit area.

However, applying approximate operations in the field of information security presents challenges. This is primarily due to the fact that data encryption and decryption often demand precise arithmetic and logical operations to ensure the integrity of the data and adhere to the CIA Triad, i.e., Confidentiality, Integrity, and Availability. Since the outcomes of approximate operations differ from exact operations, employing approximation in data encryption makes it difficult to decrypt ciphertext back into plaintext through the decryption process. Consequently, there has been no prior research on the application of approximate computing in information security, and the potential benefits of low power consumption and reduced latency of approximate computation have remained untapped.

To address these challenges, this study integrates of approximate computing into the field of information security and introduces the Symmetric Encryption Key generation method with Approximate Computation (SEKA). The approximate adder employed in this research is based on the Radix-4 adder, incorporating fault-tolerant application technology and power gating control technology. It assesses the impact on the circuit in terms of power consumption and accuracy, effectively reducing circuit area and power consumption through power gating technology. By utilizing different control modes, including Accurate Mode, Quarter Approximate Mode, Half Approximate Mode, and Three-Quarter Approximate Mode, the approximate adder generates a symmetric encryption key based on time parameters. The key transmission process has the characteristics of mutual authentication and can resist replay and eavesdropping attacks.

The remaining part of this paper is organized as follows. Section 2 reviews related studies of this paper. In Section 3, the approximate adder used in this paper is introduced. The proposed SEKA and key sharing method is presented in Section 4. The security analyses and simulation results of SEKA are detailed in Section 5. In Section 6, we conclude this study and outline our future works.

2 Related Studies

Approximate adders can be broadly categorized into two main types. The first type is an adjustable adder that relies on error correction mechanisms [8][9]. This architecture commences with an initial approximate adder and employs a truncated carry technology within the carry chain to segment sub-additions. This strategy aims to reduce the delay time associated with carry operations. To mitigate truncation errors, some sub-adders incorporate input from previous bits. The configuration approach begins with modest precision enhancements. However, a drawback of this method is that it often leads to numerous overlapping sub-adders. Overlapping implies that operation bits within the adders are duplicated, resulting in redundant circuits. Additionally, this design necessitates the inclusion of detection and correction circuits, which can significantly increase both area and power consumption. Furthermore, since error correction progresses from lower bits to higher bits, the precision gradually improves throughout the correction process.

The second type of approximate adder is the carry-predictive adder [6][10]. These carry-predictive adder architectures consist of multiple groups of sub-adders, each equipped with a rapid prediction circuit that generates an approximate value, thereby shortening the carry chain and reducing operation delays [11][12]. The overall circuit's precision can be tailored to

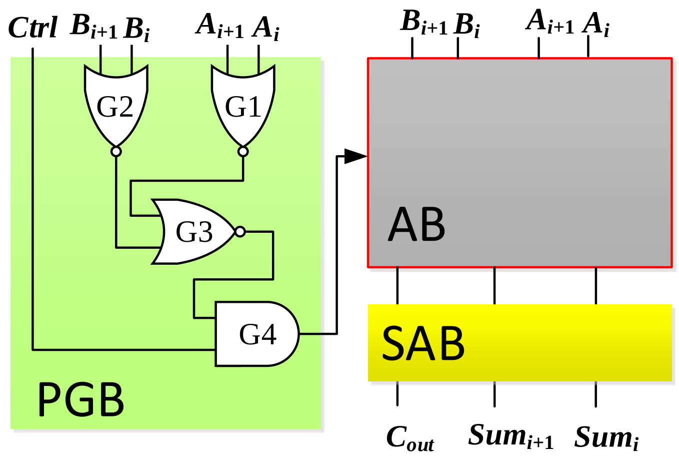


Figure 1: The architecture of the approximate adder

different levels by selecting functions during circuit operation, allowing for the output of precise results from sub-adders or using carry prediction to yield approximate results. Importantly, this approach obviates the need for error detection/correction circuitry. Additionally, the outputs of high-bit and low-bit sections are independent of each other, facilitating swift convergence within the circuit.

3 Approximate Computation

This paper employs the Radix-4 adder as the foundational architecture, considering factors such as delay time, power consumption, calculation accuracy, and circuit complexity. By selectively closing specific circuit blocks and autonomously adjusting addition results, it achieves a reduction in power consumption while enhancing calculation accuracy.

As shown in Figure 1, the architecture outlined in this study comprises three primary blocks, i.e., the Power Gating Block (PGB), the Adder Block (AB), and the Self-Adjustment Block (SAB). The PGB is responsible for managing the power supply to the AB, which functions as a precise addition block. Meanwhile, the SAB automatically generates approximate addition results when the power supply of the AB is turned off.

In Figure 1, A_i and A_{i+1} represent the two-digit summand, while B_i and B_{i+1} represent the addend. Sum_i and Sum_{i+1} denote the two-digit sum, while C_{in} and C_{out} signify the additive Carry-in and Carry-out signals, respectively. The approximate adder employs the Radix-4 calculation method and performs addition operations with 2 bits at a time.

Under the approximate operation architecture, when one or more groups of (A_i,A_{i+1}) or $(B_i,\ B_{i+1})$ equal (0, 0), the power supply to the AB is deactivated. This effectively minimizes both dynamic and static power consumption resulting from the addition circuit.

Table 1 lists the truth table of the approximate adder proposed in this paper. In the table, the text in red highlights cases where there is an error between the approximate operation result and the precise operation result. Out of the 32 possible states, only 7 will yield erroneous

Table 1: The truth table of the approximate adder

C_i	A_1	A_0	B_1	B_0	C_o	S_1	S_0	C_i	A_1	A_0	B_1	B_0	C_o	S_1	S_0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	1	0	0	1	1	0	0	0	1	0	0	1
0	0	0	1	0	0	1	0	1	0	0	1	0	0	1	0
0	0	0	1	1	0	1	1	1	0	0	1	1	0	1	1
0	0	1	0	0	0	0	1	1	0	1	0	0	0	0	1
0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	1
0	0	1	1	0	0	1	1	1	0	1	1	0	1	0	0
0	0	1	1	1	1	0	0	1	0	1	1	1	1	0	1
0	1	0	0	0	0	1	0	1	1	0	0	0	0	1	0
0	1	0	0	1	0	1	1	1	1	0	0	1	1	0	0
0	1	0	1	0	1	0	0	1	1	0	1	0	1	0	1
0	1	0	1	1	1	0	1	1	1	0	1	1	1	1	0
0	1	1	0	0	0	1	1	1	1	1	0	0	0	1	1
0	1	1	0	1	1	0	0	1	1	1	0	1	1	0	1
0	1	1	1	0	1	0	1	1	1	1	1	0	1	1	0
0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1

operation results, resulting in an error rate of 21.87%. The text in blue signifies states in which the power supply to the AB block is switched off. Among the 32 input combinations, 14 combinations will deactivate the power supply to the AB block, accounting for 43.75%. It is worth noting, as Table 1 illustrates, that when $C_{in} = 0$, there are no errors in the operation results. Errors only occur when C_{in} is equal to 1, with an error magnitude of 1. Consequently, during the addition operation, since there is no carry input to the lowest bit (equivalent to $C_{in} = 0$), there are no errors in the operation result of the lowest bit.

To enhance the flexibility of the addition operation, a control signal with n bits can generate a total of 2^n distinct operation modes. Let's take the example of using a 2-bit control signal to generate four operation modes, as illustrated in Figure 2. In this setup, the 8-bit addition operation is divided into four equal segments, and the 2-bit control signal is employed to create these four operation modes:

1. Accurate Mode (AM): In this mode, all bits undergo precise addition.
2. Quarter Approximate Mode (QA): In QA mode, the lowest 1/4 bits undergo approximate addition, while the remaining bits undergo precise addition.
3. Half Approximate Mode (HA): HA mode involves using approximate addition for the lowest 1/2 bits, while the remaining bits are subjected to precise addition.
4. Three Quarter Approximate Mode (TQ): TQ mode exclusively employs the precise adder for the highest 1/4 of the bits, with the remaining bits undergoing approximate addition.

4 Symmetric Key Generation and Delivering

Once the approximate adder design is finalized, we can employ it to craft a symmetric secret key. In the key generation and delivering process outlined in this paper, we utilize system

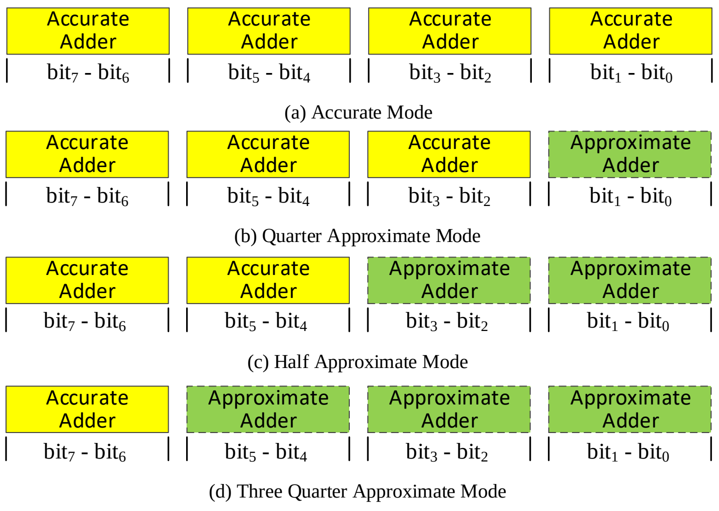


Figure 2: Different operation mode of an 8-bit adder

time parameters to formulate the control pattern for approximate operations. The ensuing explanation elucidates the interplay between system time and control modes.

Upon retrieval of a 64-bit system time value, the following methods can be employed to generate control signals:

- Bit Pairing: Starting from the highest bit and proceeding sequentially, every pair of adjacent bits forms a group that generates a corresponding control signal.
- Bit Pairing with Opposite Ends: In this method, the highest bit pairs with the lowest bit, the second highest bit with the second lowest bit, and so forth, creating control signals for each pair.
- Quad-Bit XOR: Starting from the highest bit, groups of four bits are selected successively. Within each group, the highest two bits are XORed with the lowest two bits to generate a distinct control signal.

To summarize, these methods provide diverse approaches for deriving control signals from the system time, allowing for the selection of specific operation modes: AM for ‘00’, QA for ‘01’, HA for ‘10’, and TQA for ‘11’.

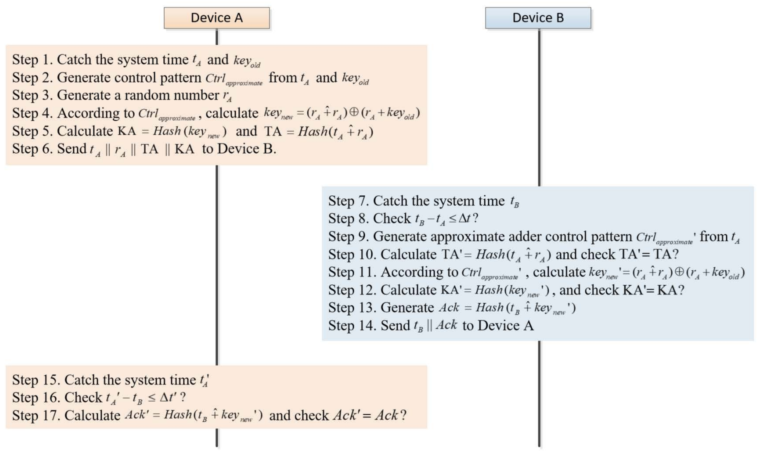
The symmetric encryption key generation and delivering process is shown in Figure 3. In the beginning, the device A performs the following 6 steps.

Step 1. Catches the system time t_A and current symmetric encryption key key_{old}.

Step 2. According to the abovementioned method, generates approximate adder’s control pattern $Ctrl_{approximate}$ from t_A and key_{old}.

Step 3. Generates a random number r_A.

Step 4. Calculates $key_{new} = (r_A \hat{+} r_A) \bigoplus (r_A + key_{old})$, where $\hat{+}$ indicates the approximate addition according to the $Ctrl_{approximate}$.



Step 1. Catch the system time t_A and key_{old}
Step 2. Generate control pattern $Ctrl_{approximate}$ from t_A and key_{old}
Step 3. Generate a random number r_A
Step 4. According to $Ctrl_{approximate}$, calculate $key_{new} = (r_A \hat{+} r_A) \oplus (r_A + key_{old})$
Step 5. Calculate KA $= Hash(key_{new})$ and TA $= Hash(t_A \hat{+} r_A)$
Step 6. Send $t_A \| r_A \|$ TA $\|$ KA to Device B.

Step 7. Catch the system time t_B
Step 8. Check $t_B - t_A \le \Delta t$?
Step 9. Generate approximate adder control pattern $Ctrl_{approximate}'$ from t_A
Step 10. Calculate TA'$= Hash(t_A \hat{+} r_A)$ and check TA'= TA?
Step 11. According to $Ctrl_{approximate}'$, calculate $key_{new}' = (r_A \hat{+} r_A) \oplus (r_A + key_{old})$
Step 12. Calculate KA'$= Hash(key_{new}')$, and check KA'= KA?
Step 13. Generate $Ack = Hash(t_B \hat{+} key_{new}')$
Step 14. Send $t_B \| Ack$ to Device A

Step 15. Catch the system time t_A'
Step 16. Check $t_A' - t_B \le \Delta t'$?
Step 17. Calculate $Ack' = Hash(t_B \hat{+} key_{new}')$ and check $Ack' = Ack$?

Figure 3: Symmetric encryption key generation and delivering process

Step 5. Calculates $KA = Hash(Key_{new})$ and $TA = Hash(r_A \hat{+} r_A)$, where $Hash$ is a one-way hash function.

Step 6. Sends $t_A||r_A||TA||KA$ to device B, where $||$ cascades different parameters

Once device B receives four parameters sent from device A, it executes the following eight steps.

Step 7. Catches its system time t_B and current symmetric encryption key key_{old}.

Step 8. Checks to see whether $t_B - t_A \le \Delta t$ or not, where Δt represents a pre-defined period of time. If not, device B sends an error message to device A and terminates key generation process. Else, device B continues the following steps.

Step 9. Generates approximate adder's control pattern $Ctrl_{approximate}$ from t_A and key_{old}.

Step 10. Calculates $TA' = Hash(r_A \hat{+} r_A)$ and checks to see whether $TA' = TA$ or not. If not, device B sends an error message to device A and terminates key generation process. Else, device B continues the following steps.

Step 11. Calculates $key'_{new} = (r_A \hat{+} r_A) \bigoplus (r_A + key_{old})$.

Step 12. Calculate $KA' = Hash(Key'_{new})$, and checks to see whether $KA' = KA$ or not. If not, device B sends an error message to device A and terminates key generation process. Else, device B continues the following steps.

Step 13. Generates $Ack = Hash(t_B \hat{+} key_{new})$.

Step 14. Sends $t_B||Ack$ to device A.

When device A receives the acknowledgement from device B, it does the following three steps.

Step 15. Catches it system time t'_A.

Step 16. Checks to see whether $t'_A - t_B \leq \Delta t'$ or not, where $\Delta t'$ means a pre-defined period of time. If not, device A sends an error message to device B and terminates key generation process. Else, device A performs the final step.

Step 17. Calculates $Ack' = Hash(t_B \hat{+} key_{new})$ and checks whether $Ack' = Ack$ or not. If not, device A sends an error message to device B, terminates key generation process, and drops key_{new}. Else, it finishes the key generation process.

5 Security and Performance Analysis

5.1 Security features of the SEKA

(1) Mutual Authentication

In the SEKA, two steps ensure mutual authentication between device A and device B. In Step 12, device B verifies KA. Only the legitimate device B, possessing key_{old}, can successfully pass this verification. This step ensures that device A is communicating with the correct and authorized device B. In Step 17, device A verifies the message received from device B by checking the equality of Ack and Ack'. Ack is a hash result generated using $t_B \hat{+} key_{new}$. Since only the legitimate device A, with the correct key_{new} and the appropriate approximate addition control pattern, can generate Ack, this step further confirms the identity of device A. Therefore, the SEKA indeed provides a robust framework for mutual authentication between device A and device B. This means that both devices can verify each other's identity, enhancing the security and trustworthiness of their communication.

(2) Resist Replay Attack

A replay attack occurs when a hacker duplicates a valid message originally sent by either device A or device B. The attacker then impersonates one of these devices by sending the duplicated message to the other, with the aim of illicitly acquiring sensitive information. Importantly, the hacker forwards the unaltered original message to either device A or device B. However, it's crucial to note that in Step 8, the condition $t_B - t_A \leq \Delta t$ is not met due to the retransmission delay, which results in $t_B - t_A > \Delta t$. Here, Δt represents the predefined maximum transmission time required for sending a message from device A to device B. This modification is necessary to account for potential delays. The same protective mechanism is also implemented in Step 16 to ensure that the SEKA remains resilient against replay attacks.

(3) Resist Eavesdropping Attack

In the context of the SEKA, security measures are in place to protect against eavesdropping attacks and ensure the confidentiality of sensitive data.

Dynamic Key Generation: The SEKA protocol generates a new key (key_{new}) based on time parameter t_A and key_{old} for communication process. This means that for every new session or process, a completely different key is created. Since the hacker cannot obtain key_{new} from previous messages, even if they capture a large number of past messages, they won't be able to deduce the current encryption key. This dynamic key generation helps thwart eavesdropping attempts.

Random Number Generation (r_A): In addition to dynamic key generation, the protocol also relies on random numbers (r_A) generated for each process. These random

Table 2: Power consumption for calculating new encryption key. (unit: μW)

	r_A		
	64 bits	128 bits	256 bits
Accurate Computation	4.447	6.758	10.189
Approximate - Bit Pairing	3.605	5.211	7.671
Approximate - Bit Pairing with Opposite Ends	3.681	5.304	7.624
Approximate - Quad-Bit XOR	3.988	5.797	8.566

numbers introduce an element of unpredictability into the encryption process, making it extremely difficult for a hacker to derive the key or predict the encryption patterns.

By combining dynamic key generation and the use of random numbers, the SEKA significantly enhances security and effectively guards against eavesdropping attacks. This ensures that even if a hacker captures a large number of messages from the network, they would not be able to extract sensitive data or deduce the current encryption key, maintaining the confidentiality of communication.

5.2 Performance Analysis

5.2.1 Power Consumption

In the Symmetric encryption key generation and delivering process, we employ an approximate adder in place of a precise one. As outlined in Section 3, varying control modes yield different approximate results. In the experiment, we randomly generated 1000 t_A and r_A, among which r_A can be divided into 64 bits, 128 bits and 256 bits. Table 2 illustrates that, across three different key lengths, power consumption for calculating the symmetric encryption key (from Step 1 to Step 4) varies depending on the method used to generate control signals for the Approximate adder.

Table 2 reveals that the power consumption between the two approximation methods, Bit Pairing and Bit Pairing with Opposite Ends, is notably similar. On average, they offer a substantial 22% reduction in power consumption when compared to Accurate computation. However, Quad-Bit XOR, due to its increased XOR operations, consumes more power than the previous two methods. Nevertheless, it still offers a respectable 13% reduction in power consumption on average when contrasted with Accurate computation. The data in Table 2 convincingly demonstrates the efficacy of approximate computation in power savings during key generation.

5.2.2 Computational latency

To ascertain the computational time needed from Step 1 to Step 17, we conducted simulations on a computer. It's important to emphasize that the transmission time between device A and device B has not been factored into the simulations. Figure 4 presents the time required to execute SEKA with varying key length. As depicted in Figure 4, the use of approximate operations has resulted in a substantial reduction in calculation delay time when compared to accurate computation. On average, this reduction achieves 22.4% when key length is 256 bits.

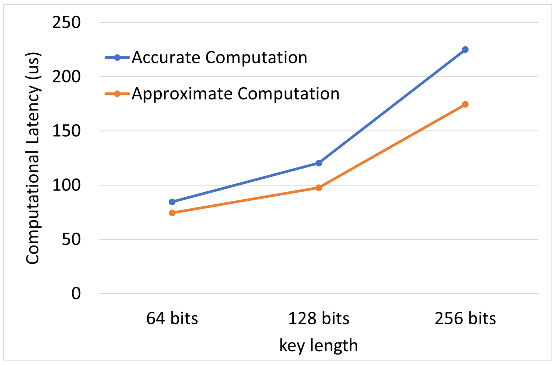


Figure 4: Computational latency of the SEKA with different key lengths

6 Conclusion and Future Studies

Approximate computing has the benefits of low power, low cost, and high performance. In this paper, we propose a symmetric encryption key generation method based on an approximate adder. The proposed approximate adder turns off the power consumption of general adder and outputs an approximate result when summand or addend is zero. The SEKA utilizes the feature of the proposed approximate adder to generate a symmetric encryption key which is very suitable for energy-constrained devices. In the near future, we will enhance the architecture of approximate adder and design approximate multiplier. Besides, new encryption method with approximate computing will also be proposed.

References

[1] Jyoti Kandpal, Abhishek Tomar, Mayur Agarwal, and Kamal Kumar Sharma. High-speed hybrid-logic full adder using high-performance 10-t xor–xnor cell. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 28(6):1413–1422, 2020.

[2] Hao Geng, Yuzhe Ma, Qi Xu, Jin Miao, Subhendu Roy, and Bei Yu. High-speed adder design space exploration via graph neural processes. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(8):2657–2670, 2021.

[3] Sofiene Mansouri. Application of neural networks in the medical field. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, 13(1):69–81, 2023.

[4] A. Sonya1 and G. Kavitha. A data integrity and security approach for health care data in cloud environment. *Journal of Internet Services and Information Security (JISIS)*, 12(4):246–256, 2022.

[5] Jose-Luis Cabra, Carlos Parra, and Luis Trujillo. Earprint touchscreen sensoring comparison between hand-crafted features and transfer learning for smartphone authentication. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, 12(3):16–29, 2022.

[6] Jungwon Lee, Hyoju Seo, Hyelin Seok, and Yongtae Kim. A novel approximate adder design using error reduced carry prediction and constant truncation. *IEEE Access*, 9:119939–119953, 2021.

[7] Kun-Lin Tsai, Yen-Jen Chang, Chien-Ho Wang, and Cheng-Tse Chiang. Accuracy-configurable radix-4 adder with a dynamic output modification scheme. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 68(8):3328–3336, 2021.

[8] Andrew B Kahng and Seokhyeong Kang. Accuracy-configurable adder for approximate arithmetic designs. In *Proceedings of the 49th annual design automation conference*, pages 820–825, 2012.

[9] Muhammad Shafique, Waqas Ahmad, Rehan Hafiz, and Jörg Henkel. A low latency generic accuracy configurable adder. In *Proceedings of the 52nd Annual Design Automation Conference*, pages 1–6, 2015.

[10] Costas Efstathiou, Zaher Owda, and Yiorgos Tsiatouhas. New high-speed multioutput carry look-ahead adders. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 60(10):667–671, 2013.

[11] B Ramkumar and Harish M Kittur. Low-power and area-efficient carry select adder. *IEEE transactions on very large scale integration (VLSI) systems*, 20(2):371–375, 2011.

[12] Rong Ye, Ting Wang, Feng Yuan, Rakesh Kumar, and Qiang Xu. On reconfiguration-oriented approximate adder design and its application. In *2013 IEEE/ACM international conference on computer-aided design (ICCAD)*, pages 48–54. IEEE, 2013.