Bounded Model Checking for Multiplexer implemented Approximate Circuit

S. ROOBAN[1,*], ARSHAD BASH AS[1], SHARATH BABU NUNAVATHU[2],
G. SREENIVASA RAJU[2], M. KIRAN KUMAR[2], AMRITA SAJJA[2]
[1]Department of Electronics and Communication Engineering,
Koneru Lakshmaiah Education Foundation,
INDIA

[2]Department of Electronics and Communication Engineering,
School of Engineering, Anurag University,
INDIA

**Corresponding Author*

Abstract: – Approximate Circuits are circuits that can tolerate known errors. The error tolerance is a tradeoff for achieving low power consumption, low area and increased speed of circuits. This unique quality of approximate circuits makes a huge impact on circuit design and is highly in demand in sectors like image processing, signal processing, etc. Due to this, performing verification for these circuits has become an essential part. However, formal verification techniques like formal error metrics, Mean Squared Error (MSE), Mean Absolute Error (MAE), and Maximum Error (MXE) cannot be used to prove 100% accuracy of approximate circuits due to error tolerant nature. Thus, in this work, we propose a formal verification methodology to generate a 100% verification assurance of these approximate techniques and reduce the verification time by 0.01 sec. Approximated Ripple Carry Adder (RCA) has been used as a case study and has performed verification for $(2^8)^2$ input patterns and has proved for all input patterns that the approximate circuit stands true for the desired functional specifications. A Formal Verification technique called Bounded Model Checking (BMC) is used to perform the verification using Symbiyosys formal verification tool.

Key-Words: – Formal Verification, Approximate Circuits, Bounded Model Checking, Functional Specifications, Mean Squared Error, Mean Absolute Error.

Received: May 9, 2025. Revised: August 17, 2025. Accepted: November 21, 2025. Published: April 6, 2026.

1 Introduction

Approximate computing is a design technique that focuses on generating error-resilient circuits. This approximation computing is used when accuracy is not the major challenge. By using approximate computing, similar results are obtained as the original (fully functional) circuits. Approximate computing can obtain low-power, area-effective, delay, and cost-efficient circuits compared to their exact counterparts. The approximations are performed during different levels of circuit design [1], [2]. Approximate circuits contain known errors that are deliberately introduced in the circuit during circuit design as per the applications required. These error-resilient circuits are highly desired in sectors that do not require 100% accuracy in the circuit result, for example, image processing, signal processing, etc. The approximations are introduced by various methods like over-sampling, over-clocking, etc., making these circuits more desirable.

This high demand for approximate computing techniques leads to the need of performing verification to ensure that these circuits follow the functional specification and contain only the known approximations. There have been a variety of formal techniques that are used to assess these circuits, such as formal error analysis, which uses Boolean satisfiability (SAT) and Boolean Decision Diagrams (BDD) to perform MSE, MAE, and MXE. These formal error analysis techniques, however, cannot ensure 100% working of these circuits. This is crucial because there may be external errors that are unwanted or undesired by the developers that may occur and go unnoticed. These occur because the error analysis metrics perform verification on an average of all output values and work by targeting a certain margin for error. This error margin includes the errors that are desired and undesired. The external errors that occurred will go unnoticed and cannot be rectified, leading to a larger-than-expected faulty circuit, resulting in a complete failure of this circuit. It became a serious issue in modern-day technology due to the complex operations and applications, along with an exponential increase in the number of transistors per circuit. The complexity of modern-day technology is understood more with the help of Moore's theorem, which states that the number of transistors of a circuit is doubled over 18 months or 2 years. Over the years, complexity has defied this law and has grown exponentially, alarming the need for a verification result with an accuracy of 100%.

However, in safety-critical systems, even small errors in approximate circuits lead to catastrophic failures. Therefore, ensuring the functional correctness of such circuits while preserving their advantages is a key research challenge. One example of showing the impact of this error in the overall design is the intel's Pentium FDIV bug in the mid-1990s. Due to a small error in the division algorithm of the Pentium processor, some division operations produced incorrect results. Although the error only occurred in rare cases, it led to substantial financial loss and damage to Intel's reputation, costing the company around 475 million dollars to replace faulty processors [3]. There are works trying to prove the accuracy and reliability of these circuits by using different approximation techniques and formal verification techniques [4].

2 Motivation

Performing the formal error analysis techniques is not enough for the complete 100% assurance of the circuit. As discussed earlier, the formal error analysis only performs verification for a certain threshold value, which can also be called a yield in testing methodologies. The higher the yield value, the higher the quality of the circuit. These values suggest that the circuit is working for the threshold percentage of the overall tests and shall be able to proceed with further steps. This, however, suggests that the circuit may contain some unknown errors which are undesired, [5], [6]. By using these verification techniques, it is hard to assert whether the errors present in the circuit are only the desired ones. To understand this better, an example has been presented for formal error analysis and a practical analysis.

2.1 Formal Error Analysis

For a structurally preserved circuit, a simple full adder is introduced with an error for the input pattern "*010*" as "*0*", where the original output is "*1*". One more error has occurred, that is not introduced intentionally, for the input pattern "*000*" as "*1*" where the original output is "*0*". This is understood by creating two circuits where one circuit does not contain any error other than the desired error, which is for the input pattern "*010*". The other circuit is a buggy circuit in which an external error occurred for the input pattern "*000*". The known errors, otherwise called approximations, are considered only for the s1m output. The carry outputs are the same as the exact adder and do not include any approximations in them. This is represented in *Table* 1. As depicted in the truth table mentioned in *Table* 1, there is an unwanted error that occurred in the functional specifications of the full adder. It is normally very hard to identify small errors like this since the functional specifications are far larger and more complex in industry-standard circuits [7]. Hence, the formal error analysis, which uses MXE, MAE, and MSE, will provide a substantial verification report and does not guarantee the 100% verification result for these approximate circuits.

Table 1- Truth Table Representation

A	B	Cin	Exact Sum	Exact Carry	Desired Approx. Sum	Buggy Approx. Sum
0	0	0	0	0	0	1
0	0	1	1	0	1	1
0	1	0	1	0	0	0
0	1	1	0	1	0	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	1	0	0	1	0	0
1	1	1	1	1	1	1

"Source: created by the authors"

The Formal Error Analysis MSE, MAE, and MXE are performed by using Equations 1, 2, and 3, respectively.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - x_i)^2 \quad (1)$$

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - x_i| \quad (2)$$

$$\text{MXE} = \max(|y_i - x_i|) \quad (3)$$

After performing the verification, the results obtained for MSE, MAE, and MXE are *1006.375, 16.75*, and *127,* respectively, for both the desired and the buggy approximated adder. Hence, it is understood that the buggy approximation is not detected by performing formal error analysis.

2.2 Practical Analysis

To understand the impact of this error, a practical application, an image processing technique, is taken. Image Steganography is a technique that adds two pieces of data in the form of binary bits and produces a result that is very similar to the original data [8]. In this, the two binary data represent the *original image* and the *secret message as an image* for performing the encryption of the secret data inside the picture, [9], [10]. However, performing a full-scale Steganography is not an easy task, this example is performed by using simple sparse images that contain the "*0*" and "*255*" binary values as the majority in the image, as shown in Figure 1. These sparse images are added by using the exact adder, the desired approximate adder, and the buggy approximated adder. The quality of the image is being measured by using *peak-signal-to-noise-ratio* (PSNR), and then the results are compared among the three adders' results. The PSNR value is calculated by using Equation 4.

$$PSNR = 10log_{10}(\frac{MAX^2}{MSE}) \quad (4)$$

The obtained PSNR values can be seen in *Table 2*. The standard value of "*45*" is considered in this application, and the value can be varied for different applications. Higher PSNR value results in better quality. Since the MSE is "*0*" for the exact adders, the PSNR value becomes infinity, resulting in the highest quality.
The PSNR value for the desired approximate circuit is "*47*", which is higher than the standard value by "*2*". This is the minimum amount of change generated by the desired approximate circuit and does not affect the original image.

Table 2- PSNR Values of Three Applications

	PSNR Value of Exact Adder	PSNR Value for Desired Approx. Adder	PSNR Value for Buggy Approx. Adder
Exact	45	45	45
Obtained	INF	47	28
Difference	INF	2	17

"Source: created by the authors"

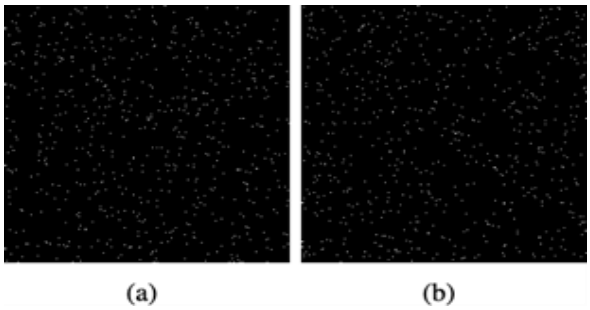

Fig. 1. Two Sparse Images.
"Source: created by the authors"

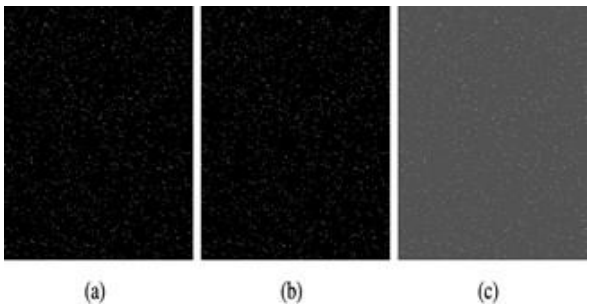

Fig. 2. Generated Results after Adding Sparse Images using a) Exact Adder, b) Desired Approx. Adder, C) Buggy Approx. Adder.
"Source: created by the authors"

The visible difference of the resulting images after performing addition operations on the two sparse images is shown in Figure 2. Note that the result obtained by performing formal error analysis of these

adders is the same, and the error has gone unnoticed using formal error analysis. Thus, performing only the formal error analysis does not provide 100% assurance of the working of the circuit.

3 Preliminaries

3.1 Full Adder

Full Adder is a Digital Circuit which performs binary addition of two multi-bit data. Full Adder is a basic digital arithmetic circuit. It contains 3 inputs, namely input 1, input 2, and *Cin,* and 2 outputs, namely *Sum* and *Carry,* as represented in *Table 3*. This, in terms of Boolean expressions, can be represented in Equations 5 and 6.

$$Sum = A \wedge B \wedge Cin \quad (5)$$

$$Carry = (A \,\&\, B) \mid (B \,\&\, Cin) \mid (Cin \,\&\, A) \quad (6)$$

For circuits tailored for specific input patterns and by using functional specifications, the Boolean expression is represented as shown in Equations 7 and 8.

$$Sum = (\sim A \,\&\, \sim B \,\&\, Cin) \mid (\sim A \,\&\, B \,\&\, \sim Cin) \mid (A \,\&\, \sim B \,\&\, \sim Cin) \mid (A \,\&\, B \,\&\, Cin) \quad (7)$$

$$Carry = (\sim A \,\&\, B \,\&\, Cin) \mid (A \,\&\, \sim B \,\&\, Cin) \mid (A \,\&\, B \,\&\, \sim Cin) \mid (A \,\&\, B \,\&\, Cin) \quad (8)$$

By using the above functional specifications of a full adder, the approximations are introduced to a structurally preserved circuit.

Table 3- Full Adder Truth Table

A	B	Cin	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

"Source: created by the authors"

3.2 Ripple Carry Adder

Ripple Carry Adder (RCA) is a digital circuit used to perform the addition operation on two multi-bit data. It is one of the simplest representations of the addition operation in computer arithmetic. RCA consists of a combination of full adders connected in a chain model. The data bits are all carried through the combination of adders and finally generate the addition operation, [11], [12]. The Ripple part comes from how the carry bit propagates from one stage to the next. A Traditional RCA consists of a combination of full adders that takes 3 inputs, namely *i)* the first bit from the first number, *ii)* the first bit from the second number, and *iii)* the carry from the previous stage. For the first cycle, the carry is assumed as low-bit or "0". These inputs are then carried across the combination of full adders.

As shown in Figure 3, the number of bits present in the binary data is directly proportional to the number of full adder modules to be used. If the RCA is a 4-bit module, that means it consists of 4 full adder modules.

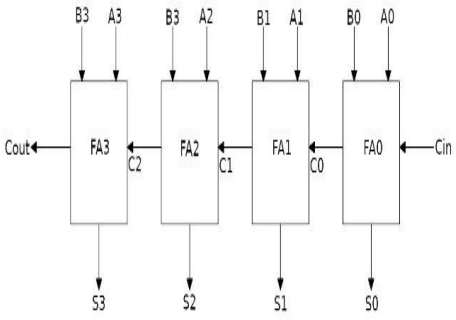

Fig. 3. Ripple Carry Adder
"Source: created by the authors"

3.3 Bounded Model Checking

BMC is a powerful verification technique that is used for formally verifying hardware circuits. BMC verifies whether a circuit satisfies its specifications for a finite number of steps. Unlike simulation, which tests specific input vectors, BMC exhaustively verifies all possible behaviors up to a given bound using rigorous mathematical equations. For combinational circuits, the BMC step count is "1" as it only requires checking of all input patterns once to achieve a complete verification. However, that's not the case for sequential circuits, which require multiple steps due to the presence of clock values in the circuits, [13], [14].

BMC translates the circuit's behavior and its specifications into a logical problem using Boolean satisfiability (SAT). In BMC, these SAT variables are verified exhaustively for all input patterns for k ("1" for combinational circuits) number of steps, and checks for any violations in any input patterns. If it finds any violations, it prints the output as" True" and gives us a counter- example with the status as" FAIL",

otherwise it prints" False" and gives us output as" PASS" ,[15], [16].
The proposed method uses the Symbiosis tool to automate this process and compute the verification results.

3.4 Functional Specifications

These describe the input-output behavior of the digital circuits and the conditions under which they operate. Functional specifications are the specifications that represent the functionality of the circuit. These outline the necessary functions that the circuit has to achieve for a fully functional circuit. These functional specifications are often represented in circuit design as Boolean expressions or truth tables. Hence, the working of the circuit is understood by using functional specifications. Similarly, these functional specifications are used to introduce approximations into the circuit. As the input-output patterns are known, approximations are introduced for any particular or tailored input patterns for our desired outputs, [17], [18]. The proposed method uses the same functional specifications to generate a circuit to formally verify the digital circuit.

4 Methodology

As discussed in the previous section, the adders are the basic building blocks of arithmetic circuits, and by introducing approximations in the full adder modules, different approximate circuits are generated. The proposed verification methodology consists of 3 major steps, which are *i)* generating approximate circuits by using functional specifications, *ii)* generating an MUX-based approximate circuit, and *iii)* creating a miter circuit.

4.1 Approximate Circuit Generation

By using the functional specifications, the desired approximate circuit for a particularly tailored input pattern has been generated. The approximated circuit is generated by altering the Boolean Equation generated from the functional specifications. The approximations are understood by the depicted Boolean equations in Equations 9 and 10.

$$Sum = (\sim A \,\&\, \sim B \,\&\, \sim Cin) \mid (\sim A \,\&\, \sim B \,\&\, Cin) \mid (\sim A \,\&\, B \,\&\, \sim Cin) \mid (A \,\&\, B \,\&\, Cin) \quad (9)$$

$$Carry = (\sim A \,\&\, \sim B \,\&\, \sim Cin) \mid (\sim A \,\&\, B \,\&\, Cin) \mid (A \,\&\, \sim B \,\&\, Cin) \mid (A \,\&\, B \,\&\, Cin) \quad (10)$$

This Boolean Equation describes the functional specifications, also structurally preserving the circuit, so that the output patterns are obtained as the desired approximations. The Boolean Equations generated approximations for the input pattern of "000" as "1" instead of "0", which is the original output for both sum and carry. For input vector "100", where the output of the sum is "1", it has been approximated to "0". For input vector "110", where the exact output of carry is "1", it has been approximated to "0". These are the 4 approximations that are introduced in a full adder by using the functional specifications. The same has been depicted in *Table 4*. The approximate outputs are represented with an underline. Hence, the approximate circuit with the desired approximations is obtained by using functional specifications.
The approximations introduced in the full adder module are used to generate an approximated RCA. The approximations are introduced to the Least Significant Bits (LSBs) of the RCA, and the Most Significant Bits (MSBs) are left unchanged, as shown in Figure 4. This is to ensure that the overall results of the RCA do not incline towards any major change from the exact circuit.

Table 4- Truth Table of Approximated Full Adder

A	B	Cin	Exact Sum	Exact Carry	Approx. Sum	Approx. Carry
0	0	0	0	0	<u>1</u>	<u>1</u>
0	0	1	1	0	1	0
0	1	0	1	0	1	0
0	1	1	0	1	0	1
1	0	0	1	0	<u>0</u>	0
1	0	1	0	1	0	1
1	1	0	0	1	0	<u>0</u>
1	1	1	1	1	1	1

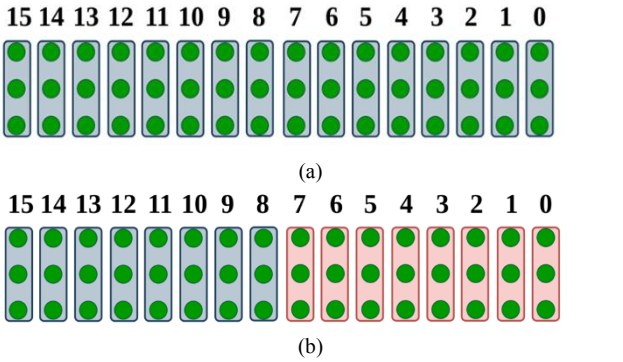

Fig. 4. Bit Representation of a) Exact RCA, b) Approximated RCA.
"Source: created by the authors"

4.2 Mux Based Circuit Generation

Multiplexers are digital logic components that are driven by selection lines. Mux designs basic functionality based on data selection, and the same is used in this methodology to create an approximated circuit, which follows the functional specifications and is used while performing verification [19]. As stated, Mux circuits operate mainly depending on selection bits, which can be altered as desired. This is implemented by using the functional specifications of the approximated adder. In Figure 5, the representation of how the selection bits are manipulated in the mux, which thus leads to the introduction of the approximations by using the functional specifications, is represented. As shown in *Table 4*, the number of input vectors required to obtain the 100% assured verification result is 2^n, n being the number of bits. However, by using Mux as a replacement, the total number of vectors used for 100% verification has been reduced by *half*, thus increasing the verification speed and reducing the overall verification delay. The truth table for the mux-based RCA is represented in *Table 5*.

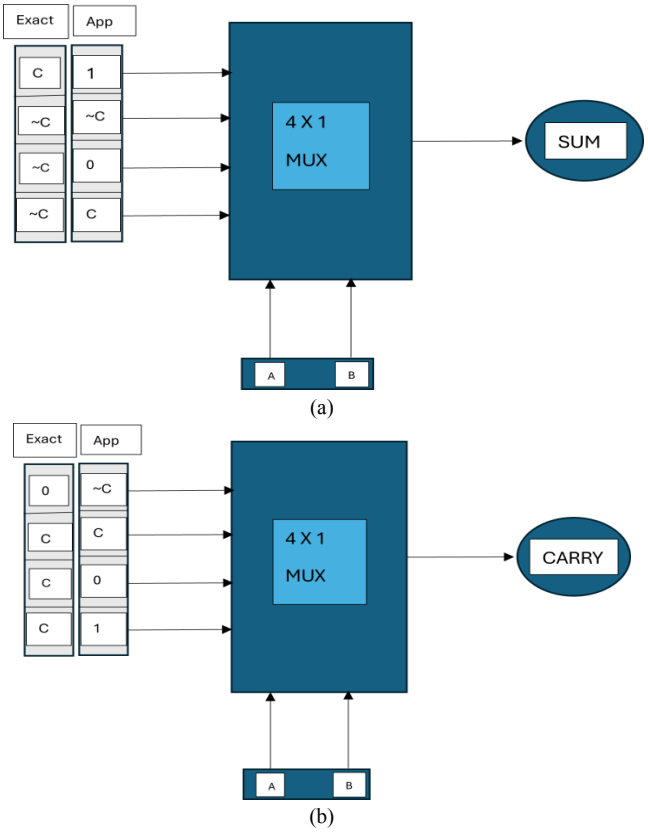

Fig. 5. Proposed MUX Representation of Exact and Approximated a) SUM, b) Carry.
"Source: created by the authors"

Table 5- Mux Generation Methodology
"Source: created by the authors"

Input bits of the sum	Input bits of Carry	Selection bit (A)	Selection bit (B)	Output (Sum)	Output (Carry)
1	~C	0	0	1	~C
~C	C	0	1	~C	C
0	0	1	0	0	0
C	1	1	1	C	1

As shown in *Table 6,* the functional specification of the truth table is maintained and generates a mux-based RCA. The methodology follows the selection lines *A* and *B,* which are inputs of RCA, and generates the output *sum* and *carry* all while structurally preserving the circuit. The *Cin* is used as input data following the functional specifications of the required approximations. The similar input bits of *A* and *B* are categorized into a group, neglecting *Cin* for input data, and the output *sum* and *carry* are obtained by using the same functional specifications as the approximate adder.

Table 6- Mux Generated Truth Table

A	B	Cin	Exact Sum	Exact Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

"Source: created by the authors"

For input vector "00", the *Cin* value represents the same for sum, and for carry, the *Cin* value is grounded (fixed to 0). For input vector "01", for sum, the *Cin* value is inverted, and for carry, the *Cin* remain same. For a "10" input vector, the *Cin* value is inverted and for carry the *Cin* values remain the same. And for input vector "11", the *Cin* value is the same for sum, and for carry, the *Cin* value is always high or connected to *vdd* (fixed to 1). By using the mentioned functional specification, a similar output is obtained as the desired approximate adder while reducing the vectors required for a complete 100% assurance.

There are many works presented that support the fastness and versatile use of MUX-based circuits. The approximate circuits, by using functional specifications, act as a golden reference. The proposed approximated MUX circuits are compared with the golden reference. The selection of bits is done by using the desired functional specifications of the approximate circuit.

4.3 Miter Circuit Generation

A miter is a circuit that takes the output of two circuits and performs an XNOR operation. The miter circuit takes the outputs from approximate and Mux-generated circuits as inputs and performs an XNOR operation. The XNOR operation generates 'TRUE" only when both the inputs are "TRUE" (same) and generates "FALSE" when the inputs are different, as shown in *Table 7*. Thus, by performing the XNOR operation, it is stated that both circuits generate the same outputs. Hence, when the miter circuit results in true for every input pattern, it assures that the approximate circuit follows the functional specifications and can provide 100% verification assurance.

Table 7- Miter Truth Table

Output of approx. circuit	Output of Golden reference (MUX)	XNOR output (miter output)
0	0	1
0	1	0
1	0	0
1	1	1

"Source: created by the authors"

4.4 Verification Using Symbiyosys

After generating the necessary circuits, *i)* approximate circuit, *ii)* Mux circuit, *iii)* miter circuit, and *iv)* Symbiiosys file (*.sby* file). The Symbiyosys file is necessary for using the Symbiyosys verification tool. The *sby* file contains all the details, which helps in operating the verification process in Symbiyosys tool. The *sby* file specifies the search engine, which is *smtbmc* in our example, the verification methodology BMC, and the three Verilog files, namely the Approximate, Mux generated, and Miter circuit.

After obtaining the *sby* file, the *sby* command, which is *Sby -f <file name>.sby,* is used in the Symbiosis verification tool environment. The Symbiyosys generates the verification result, and in case the verification process generates a file that contains suggestions for improvement. The overall design flow is shown in Figure 6.

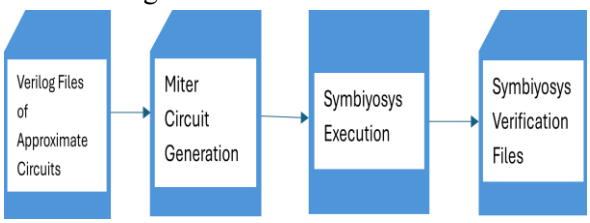

Fig. 6. Overall Design Flow.
"Source: created by the authors"

5 Results

The results obtained from the Symbiosys verification tool are represented in Figure 7. The verification has been completed successfully for one step of the BMC. In combinational circuits, the step count is 1 because the *smtbmc* covers all the input patterns exhaustively for every step.

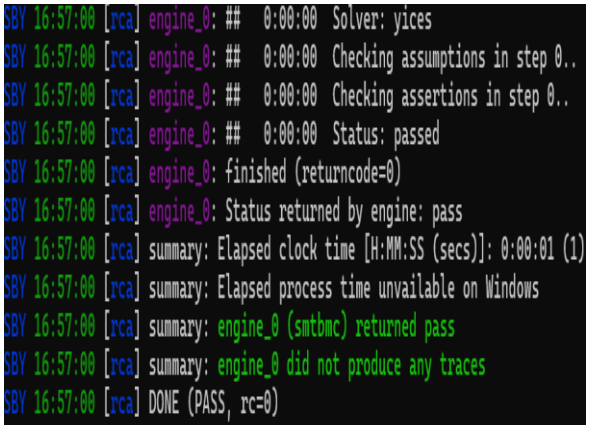

Fig. 7. Generated Results from Symbiosys Verification Tool.
"Source: created by the authors"

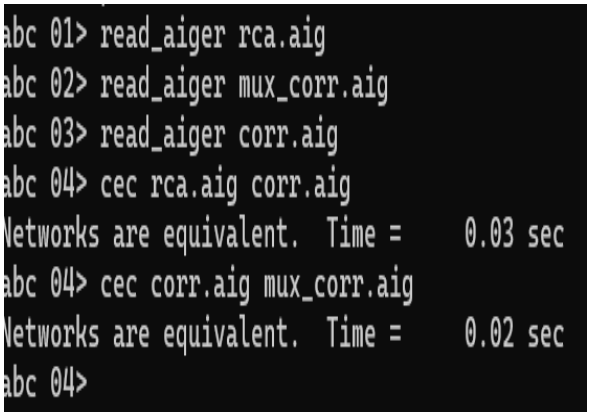

Fig. 8. Results from cecApprox.
"Source: created by the authors"

Table 8- Comparison of Verification Time

Bit width	Approximated adders	Normal Adders Verification time	MUX-based Adders Verification Time
8	1	0.01	0.01
	2	0.01	0.01
	3	0.01	0.01
	4	0.02	0.01
	5	0.02	0.01
	6	0.02	0.02
	7	0.02	0.02
16	1	0.02	0.01
	2	0.02	0.01
	3	0.02	0.01
	4	0.02	0.02
	5	0.02	0.02
	6	0.02	0.02
	7	0.02	0.02
	8	0.02	0.02
	9	0.02	0.02
	10	0.02	0.02
	11	0.03	0.02
	12	0.03	0.02
	13	0.03	0.02
	14	0.03	0.03
	15	0.03	0.03

"Source: created by the authors"

The solver used for the verification is *Yices* and performed BMC by using the *smtbmc* engine. This states that the approximate circuit does not contain any external error except for the errors that are intended. By this process, a 100 percent verification result for the approximation can be obtained. By using this methodology, the drawbacks of the formal error analysis have been overcome, and the verification time has been reduced.

As shown in Figure 7, the approximate full adder and the mux-based full adder generated from the same specifications are generating the same output when performing BMC. *Approx* verification technique is also used to verify the same, and the result is shown in Figure 8.

The results are obtained by performing the *cecAprox* verification methodology. The comparison of both circuits is shown in *Table 8*. The verification change, when the number of approximations increased, is shown in *Table 8*. Thus, MUX based full adder reduces the required verification delay.

6 Conclusion

Formally verified a structurally preserved approximate circuit generated by using functional specifications. Generated a MUX-based full adder circuit by using the same functional specifications as the approximated circuit. Created a miter circuit for the approximate circuit and mux-based full adder circuit, and successfully performed formal verification by using BMC and obtained 100% verification result. The BMC has performed verification exhaustively for every input pattern by using the *smtbmc* verification engine. Reduced the required verification time by *0.01* sec compared with the previous works. Obtained the verification results by using the Symbiopsys formal verification tool.

References

[1] A. M. Dalloo, A. Jaleel Humaidi, A. K. Al Mhdawi and H. Al-Raweshidy, "*Approximate Computing: Concepts, Architectures, Challenges, Applications, and Future Directions*," in IEEE Access, vol. 12, pp. 146022-146088, 2024. 10.1109/ACCESS.2024.3467375

[2] A. Bosio, A. Virazel, P. Girard and M. Barbareschi, "*Approximate computing: Design and test for integrated circuits*," 2017 18th IEEE Latin American Test Symposium (LATS), Bogota, Colombia, pp. 1-1, 2017. doi:10.1109/LATW.2017.7906737

[3] P. Manolios, "*Refinement and theorem proving*," Springer, Berlin, Heidelberg, pp. 176–210, 2006.ISBN 978-3-540-34304-2 (Accessed Date: April 15, 2025).

[4] L. Qiu, Z. Zhang, J. Calhoun, and Y. Lao, "*Towards Data-Driven Approximate Circuit Design*," 2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Miami, FL, USA, pp. 397-402, 2019. Doi 10.1109/isvlsi.2019.00078 (Accessed Date: April 15, 2025).

[5] J. Wang, J. Shao, Y. Li and J. Ding, "Survey on Formal Verification Methods for Digital IC," *2009 Fourth International Conference on Internet Computing for Science and Engineering*, Harbin, China, 2009, pp. 164-168, 10.1109/ICICSE.2009.46.

[6] I. Saha, S. Roy and S. Ramesh, "*Formal Verification of Fault-Tolerant Startup Algorithms for Time-Triggered Architectures: A Survey,*" in Proceedings of the IEEE, vol. 104, no. 5, pp. 904-922, May 2016. 10.1109/JPROC.2016.2519247

[7] K. -y. Khoo, "*Formal Verifications in Modern Chip Designs*," 2006 IEEE International High Level Design Validation and Test Workshop, Monterey, CA, USA, pp. 38-38, 2006. 10.1109/HLDVT.2006.320001

[8] H. Jiang, F. J. H. Santiago, H. Mo, L. Liu and J. Han, "*Approximate Arithmetic Circuits: A Survey, Characterization, and Recent Applications*," in Proceedings of the IEEE, vol. 108, no. 12, pp. 2108-2135, Dec. 2020. 10.1109/JPROC.2020.3006451

[9] C. N. Raghuram, G. K. Kumar, V. Sreeja, C. Tharani, A. Vivaswanth and N. V. L. N. Murty, "*Efficient Approximate Adder Design for Image Brightness Enhancement: Balancing Area, Power, and Delay*," International Conference on Computer, Electronics, Electrical Engineering and their Applications (IC2E3), Srinagar Garhwal, Uttarakhand, India, pp. 1-6, 2024. 10.1109/IC2E362166.2024.10827439

[10] J. V., M. P. and R. C. K., "Cognitive Approximate Adder Design for Image Processing Applications," 2023 International Conference on Recent Trends in Electronics and Communication (ICRTEC), Mysore, India, pp. 1-6, 2023. 10.1109/ICRTEC56977.2023.10111918

[11] W. Yang, L. Wang and K. Li, "*An Approximate Adder Design Based on Inexact Full Adders*," 2021 IEEE 15th International Conference on Anti-counterfeiting, Security, and Identification (ASID), Xiamen, China, pp. 154-158, 2021. 10.1109/ASID52932.2021.9651713

[12] Z. Wang, G. Zhang, J. Ye, J. Jiang, F. Li and Y. Wang, "*Accurate reliability analysis methods for approximate computing circuits,*" in Tsinghua Science and Technology, vol. 27, no. 4, pp. 729-740, Aug. 2022. 10.26599/TST.2020.9010032

[13] L. Liu, W. Kong, T. Ando, H. Yatsu and A. Fukuda, "*A Survey of Acceleration Techniques for SMT-Based Bounded Model Checking,*" 2013 International Conference on Computer Sciences and Applications, Wuhan, China, pp. 554-559, 2013. 10.1109/CSA.2013.135

[14] Z. Vasicek, "*Formal Methods for Exact Analysis of Approximate Circuits*," in IEEE Access, vol. 7, pp. 177309-177331, 2019. 10.1109/ACCESS.2019.2958605

[15] B. Zhao, L. Qiu and Y. Lao, "*Data-Driven Feature Selection Framework for Approximate Circuit Design*," in IEEE Transactions on Computer- Aided Design of Integrated Circuits and Systems, vol. 42, no. 11, pp. 3519-3531, Nov. 2023. 10.1109/TCAD.2023.3260160

[16] M. Traiola, A. Virazel, P. Girard, M. Barbareschi and A. Bosio, "*A Survey of Testing Techniques for Approximate Integrated Circuits,*" in Proceedings of the IEEE, vol. 108, no. 12, pp. 2178-2194, Dec. 2020. 10.1109/JPROC.2020.2999613

[17] P. K. Sharma and N. K. Singh, " *BDD-based area and power efficient digital circuit design using 2T and 4T MUX at 90 nm technology*," 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), Kanyakumari, India, pp. 7-11, 2014. 10.1109/ICCICCT.2014.6992920

[18] A. Kumar, S. Sinha and D. Bansal, "*Design of Efficient Approximate Adders for Error Analysis and Mitigation*," 2023 International Conference on Ambient Intelligence, Knowledge Informatics and Industrial Electronics (AIKIIE), Ballari, India, pp. 1-5, 2023. 10.1109/AIKIIE60097.2023.10390198

[19] Mandrumaka, K.K., Noorbasha, F. A low-power 10-bit SAR ADC with variable threshold technique for biomedical applications. SN Appl. Sci. 1, 918 (2019). https://doi.org/10.1007/s42452-019-0940-3.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)
The authors equally contributed to the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or the Scientific Article Itself
No funding was received for conducting this study.

Conflict of Interest
The authors have no conflicts of interest to declare that are relevant to the content of this article.