**Indonesian Journal of Electrical Engineering and Computer Science**
Vol. 35, No. 1, July 2024, pp. 90~101
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v35.i1.pp90-101 

# ADKNN fostered BIST with Namib Beetle optimization algorithm espoused BISR for SoC-based devices

**Suleman Alnatheer, Mohammed Altaf Ahmed**
Department of Computer Engineering, College of Computer Engineering and Sciences, Prince Sattam bin Abdulaziz University, Al-Kharj, Saudi Arabia

**Article Info**

***Article history:***

Received Jan 24, 2024
Revised Mar 13, 2024
Accepted Mar 21, 2024

***Keywords:***

ADKNN
BIST and BISR
Built-in redundancy analysis
Memory under test
NBOA
System-on-chips

**ABSTRACT**

Redundancy analysis is a widely used method in fault-tolerant memory systems, and it is essential for large-size memories. In current security operations centers (SoCs), memory occupies most of the chip space. To correct these memories using a conventional external equipment test approach is more difficult. To overcome this issue, memory creators utilize redundancy mechanism for substituting the columns and rows along with a spare one to increase output of the memories. In this study, a built-in-self-test (BIST) to test memories and built-in-self-repair (BISR) mechanism to repair the faulty cells for any recent SoC devices is proposed. The BIST, based on adaptive activation functions with a deep Kronecker neural network (ADKNN), not only detects the defect but also determines the kind of defect. The BISR block uses the Namib Beetle optimization algorithm (NBOA) to fix the mistakes in the memory under test (MUT). The study attempts to determine how the characteristics of SoC-based devices change in the real world and then contributes to the suggested controller blocks. Performance metrics such as slice register, region, delay, maximum operating frequency, power consumption, minimum clock period, and access time evaluate performance. Comparing the proposed ADKNN-NBOA-BIST-BISR scheme to existing BIST, BISR, and BISD-based methods reveals its significant performance.



***Corresponding Author:***

Suleman Alnatheer
Department of Computer Engineering, College of Computer Engineering and Sciences
Prince Sattam bin Abdulaziz University
Al-Kharj, Saudi Arabia
Email: s.alnatheer@psau.edu.sa

## 1. INTRODUCTION

The initial need for memory in SOCs is to store a large number of data. The memories cover a huge area in the system-on-chips design, developed using complementary metal–oxide–semiconductor (CMOS) technology [1]. The system-on-chips are smaller in size, denoting that the memories are necessary for the greater result. The memory repair mechanism consists of both column and row repair. The memory repair mechanisms are of 2 phases. The 1st phase verifies the failure detected by the memory built-in self test (MBIST) controller via repaired memory testing. The 2nd phase obtains the signature of repair to repair the memory [2], [3]. Furthermore, the built-in redundancy analysis approach evaluates the repair signatures based on memory failure information and implements the memory redundancy technique [4]. The repair signature is saved in the BIRA registers here for processing by the memory built in self test controllers [5]. Then, the scan chain of the repair register is used to apply the repair signature [6]. The read-write test access port for fuse boxes is controlled. A specific repair log keeps track of the scan chains connecting the memory

and the fuse [7]. While using the greater voltage nut, the repair data is scanned [8]. Following an on-chip reset, restoration data is imported as well as debugged in repair logs. And each recollection is held together by redundancy. Finally, the memory built in self-test runs on restored memory to test memory accuracy [9]. Memory does not, as is customary, consist of logic gates and flip-flops. Here, numerous fault algorithms and test models are necessary to test memories. System-on-chips' overall outcome is affected by any memory defects [10]. To overcome this issue, the spare columns and rows are included in memory. Using built-in self-repair logic, spare columns/rows are employed in place of damaged cells [11]. Repair logic includes either line/column repair or both. As a result, MBIST and repair tools rapidly assess memory to find defects in normal memory cells [12]. Coupling, transition, stuck-at-fault, neighborhood design sensitive fault (NBSF), and address decoder faults are appropriate fault models for testing memory [13]. The error detection model gives a way to identify the faulty sites, and RA state-of-the-art models. The repaired solutions consist of spare columns and rows for memory replacement [14]–[16]. As a result, while implementing field programmable gate array (FPGA), it does not minimize power consumption, region, or access time [17]. It tests memory using a memory built-in-self-test (BIST) controller and a built-in-self-repair (BISR) algorithm developed on application-specific integrated circuits (ASIC), and the March approach is utilized to test memory under test [18].

The memories in system-on-chip consist of different sizes and lower accessibility. Hence, repairing this memory is not as much easier with the conventional external equipment test model. As a result, memory designers frequently employ redundancy. Mechanism to replace the columns-rows with spare ones which is utilized to increase the better output result of the memories.

In this study, we presented a BIST mechanism to test memory and a BISR method to fix damaged cells for any modern (ADKNN-NBOA-BIST-BISR) based on security operations center (SoC) devices. The BIST based on adaptive activation functions with deep Kronecker neural network not only detects the defect as well as determines the kind of defect. BISR block, based on the Namib beetle optimization algorithm (NBOA), it remedies MUT faults based on the fault signature of built-in redundancy analysis. The study gives empirical insights into how changes in SoC-based device characteristics are brought about after incorporating the suggested controller blocks. This work's contribution is noted below.

- In this manuscript, adaptive activation functions with deep Kronecker neural network fostered BIST with NBOA espoused built-in self-repair for SoC based devices suggested for SRAM.
- The suggested hybrid approach is then the combination of both the adaptive activation functions with deep Kronecker neural network (ADKNN) [19] NBOA [20]; hence it is named the ADKNN-NBOA technique.
- The proposed adaptive activation functions with deep ADKNN based on BIST is utilized for verifying memory array circuit and injecting fault in memory.
- Furthermore, based on built-in redundancy analysis, NBOA-based BISR is used to repair faulty memory cells, and fault memory is corrected with ADKNN for the BISR mechanism after introducing faults in the memory cell.
- Verilog hardware description language (HDL) was used to write the register transfer logic (RTL) code.
- The experiment was carried out on the Xilinx simulator. Moreover, the design and implementation of FPGA architecture by MBIST and BISR hardware structure for SRAM are tested using the Xilinx simulator.
- The simulation outcomes of the ADKNN-NBOA-BIST-BISR method are compared to the art methods, namely, BIST, BISR, and BISD-based BFTCP-RLSCD-BIST-BISR [21], BIST and BISR for AI Accelerator (STRAIT-BIST-BISR) [22], BIST based saboteur and mutant and BISR based counting threshold for memories (VHDL-FIT-BIST-BISR) [23], respectively.

Several studies have been conducted on BIST and BISR approaches. A few of the selected BIST approaches are [24], [25], and some of the selected BISR [26]–[28] approaches are targeted in this research. A survey [29] on the special session, machine learning in test: analogue, digital, memory, and RF integrated circuits, was covered. The remaining section of this manuscript is organized as follows: Segment 2 the method of the proposed approach is described. Segment 3, outcomes and discussion are demonstrated. In segment 4, the conclusion is presented.

## 2. METHOD

This section describes the proposed methodology (adaptive activation functions with deep Kronecker neural network fostered Built in self test with NBOA espoused built-in self-repair for SoC based devices). The block diagram of ADKNN-NBOA-BIST-BISR is represented in Figure 1. It contains two stages, BIST and BISR. Thus, a detailed description of each step is given.

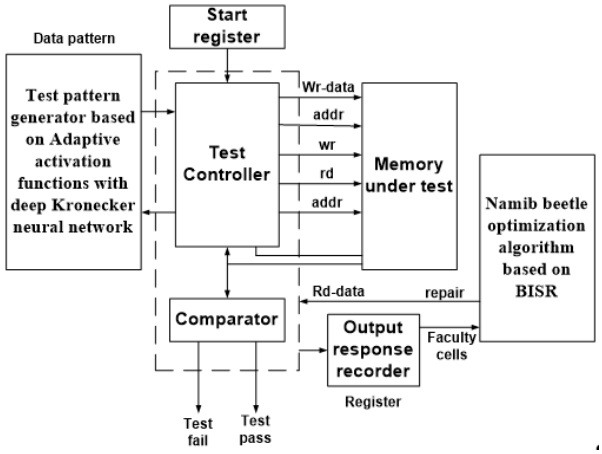



Figure 1. Block diagram of the proposed (ADKNN-NBOA-BIST-BISR) method

### 2.1. BIST and BISR approaches

Before the system is marketed, the problem is added and evaluated to improve its performance. The process of adding flaws to a system is called fault injection. Memory takes up a lot of space in SOC, and memory errors will impair the SOC's results. Here, spare columns and rows are present in the memory. In this framework, adaptive activation functions with deep Kronecker neural network (AFKNN) and NBOA for BIST and BISR for SRAM are proposed. For verifying the memory array unit, the AFKNN-based BIST is used, and defects are injected into the storage via the AFKNN defect injection method. After putting faults into the memory cell, the defect memory is repaired utilizing the BISR approach based on ADKNN. The operations used in BIST and BISR for the embedded memories block are listed in Table 1.

Table 1. Notation in BIST and BISR for embedded memories block

| S.No | Operation | Description |
|---|---|---|
| 1 | Wr-data | Write the information to the storage position specified by the address. |
| 2 | addr | Specifies the address of the area in the memory where the memory's data will be accessed. |
| 3 | wr | To write to memory, use the writes-enabled signaling. |
| 4 | rd | The read enable signal indicates that the memory is being read. |
| 5 | Rd-data | The read data bus holds the read data from a specified memory address. |
| 6 | w0 | To the memory address, write the logic value '0'. |
| 7 | w1 | To the memory address, write the logic value '1'. |
| 8 | r0 | Read the storage cell's logic value '0'. |
| 9 | r1 | Read the storage cell's logic value '1'. |

A test controller, comparator, start register, test pattern generator, memory under test, output BISR and response recorder are all part of the suggested architectural model. The suggested AFKNN approach is used for test controller and test pattern generator. Furthermore, the suggested AFKNN model is used to inject faults, while the proposed ADKNN model is used to restore fault memory.

### 2.2. BIST

BIST is a low-cost integrated circuit that is combined into SRAM memories to test for faults that arise during memory read or write operations. BIST neglects the requirements for automated test equipment (ATE), an expensive as well as time-consuming external hardware component. Since it is more expensive, the BIST includes more structures than the external ATE. The BIST circuit comprises several blocks, including a number of blocks, a level-shifting buffer circuit, an amplifier circuit for amplifying defect signals, and an operational amplifier for increasing weak signals, which acts as a phase amplifier and comparator circuit for comparing faulty output and fault-free SRA memory. The memory test controller utilizes adaptive activation functions with a deep Kronecker neural network to maximize fault coverage and is used to detect the coupling faults. The adaptive activation function with deep Kronecker neural network obtains the test pattern that performs the test controller to implement defects to the memory. After implementing memory faults, defect memory is constant utilizing ADKNN for the BIRST.

**2.2.1. Test pattern generator**

An adaptive activation function with a deep Kronecker neural network is utilized to analyze memory array units for faults. The test pattern generator performs on the ADKNN approach, which designs the patterns that are necessary to inject defects and propagate effect to the result. The memory array unit is expressed in the (1).

$$L_1(Z^{l-1})\underline{\Delta}w^l Z^{l-1} + B^l \tag{1}$$

Here, $w^l$ represents the weight matrix $B^l$ represents the bias vector associated with the $l^{th}$ layer. The fault in adaptive activation function with deep Kronecker neural network is expressed in (2),

$$U^{ff}(Z) = (L_d \circ \phi_1 \circ L_{d-1} \circ \ldots \circ \phi_1 \circ L_d) \tag{2}$$

here, $d$ represents the depth of the feed-forward neural network $\phi_1$ represents the fault memories. After an output layer, the process of activation function is an identity function. The fault injection process is expressed in the (3),

$$\|V\|_1 = \sum_{i=1}^{n}|V_i|, \|V\|^2 = \sum_{i=1}^{n} V_i^2, \|V\|_{\propto 1} = \underset{1\le i\le n}{max}|V_i| \tag{3}$$

here, $V$ represents the fault injection process. The memory array unit for faults is expressed in the (4),

$$\|M\| = \underset{\|X\|=1}{max}\|MX\|, \|M\|_f^2 = \sum_{i=1}^{m}\sum_{j=1}^{n} M_{ij}^2 \tag{4}$$

here, $M$ represents the memory array unit for faults, $M_{ij}$ represents the $i,j$ component of $M$. The total weight and block in the memory is expressed in the (5),

$$1_{K\times K} \otimes w^l = \begin{bmatrix} w^l & \cdots & w^l \\ \vdots & \ddots & \vdots \\ w^l & \cdots & w^l \end{bmatrix} \in \Re^{N_l K \times N_{l-1} K}, 1_{K\times 1} \otimes B^l = \begin{bmatrix} B^l \\ \vdots \\ B^l \end{bmatrix} \in \Re^{N_l K} \tag{5}$$

here, $N_l$ represents the number of chips. The activation function is expressed in the (6),

$$\vec{\phi}(Z) = \begin{bmatrix} \phi_1(Z_1) \\ \vdots \\ \phi_K(Z_K) \end{bmatrix} \tag{6}$$

here, $\phi$ represents the block activation function. The memory chip is expressed in the (7),

$$Z^l = (1_{K\times K} \otimes w^l)\vec{\phi}(Z^{l-1}) + 1_{K\times 1} \otimes B^l \tag{7}$$

the weight and blocks in the memory were expressed in the (8),

$$\widetilde{w}^l = (\varpi^l \otimes \alpha^l) \otimes w^l, \tilde{B}^l = \varpi^l \otimes B^l, 1 \le 1 < d \tag{8}$$

here, $\otimes$ represents the Kronecker product, $\phi_K$ represents the fixed activation functions. The Kronecker neural network using composition operator is expressed in the (9),

$$U^K(Z) = \left(L_d \circ \tilde{\phi}^{d-1} \circ L_{d-1} \circ \ldots \circ \tilde{\phi}^1 \circ L_1\right) \tag{9}$$

here, $\circ$ represents the composition operator. The activation function is expressed in the (10),

$$\tilde{\phi}^l(L_1(Z); \varpi^l, \alpha^l) = \sum_{K=1}^{K} \alpha_K^l \phi_K\left(\varpi_K^l L_1(Z)\right), l = 1, \ldots, d-1 \tag{10}$$

Here, $\varpi^l$ represents the column vector, $\alpha^l$ represents the row vector. To improve the training, the weight of target network regularly recorded via repetition. The produced test patterns stored and applied at the execution of BIST. Here, the patterns obtained at random by an AFKNN-related test pattern creator perform as a test pattern. A key focus of registration architecture is the smaller area that is built with as many designs as feasible.

#### 2.2.2. Test controller

The MUT controllers in the test controller have registers that track failure data details. When the start register's start signal has been programmed, starts the test controller. In this case, the start register comprises clock, stop, start, resume, reset, halt-on-error, and memory ID. The AFKNN technique-based test pattern generator's operation produces patterns and sends them to the memory block under test during the reading process. The read data was analyzed during the testing procedure to determine the produced patterns. Finally, the output response recorder monitors the ID of memory, count of faulty cells, and faulty MUT address. Furthermore, unsuccessful data are sent to a BISR block for the use of spare memory at defective cell substitution.

### 2.3. MUT

The created pattern is applied to the MUT during read and write operations. Several states are covered here: idle, write 0 (w0) in the memory position, read 0 write 1 (r0w1), r1w0r0, w0r0w1, r1w0, r0, failed record status. The test controller waits at an idle state to receive a start signal only once, and then it jumps into the w0 state and starts the memory test operation from the w0 state. When memory under test goes to another state like r0w1, it is full with '0' patterns. A test controller is used to execute read and write activities. Finally, the test controller performs all operations in sequence at all states.

### 2.4. Comparator

From the MUT block, the comparator examines the result and pattern. Furthermore, when executing testing under-read operation, read data are evaluated using chosen patterns. The test controller advances into the level of failure-registration to save the result if the comparison result is unsuccessful, then returning to the original position.

### 2.5. Output response recorder

The output response is evaluated by the output response recorder. Here, the computer reaction to the test vector that has to be validated. Furthermore, the choice is made as to while the PC is defective or not. The output response recorder stores defective cells, fail address, memory ID, and memory under the test count of a faulty cell. In addition, previously determined sequences are fed into the MUT, and the result is recorded in the response output recorder.

The result is obtained at the failed record state when all tasks are finished. Furthermore, the test controller identifies the failure memory ID, fault position, and hand number of defective cells. The present approach's experimental outputs analyze a variety of defect types, including SF, CF, RDF, WDF, TCF, SCF, DCCF, IRF, DRDF, and ICF.

### 2.6. Fault modelling of SRAM

SRAM fault modeling is defined as an explanation of potential failure modes. This research study targeted several failure modes, or fault types, in SRAM. We intend to target various fault types with the AFKNN technique. The fault types in SRAM memory are described as follows:

- Stuck fault: the single-cell error known as a "stuck fault (SF)" occurs when an SRAM memory cell's logic value is locked at 0/1.
- Coupling faults: this is a fault type of SRM cells as a result of its contact with another cell is known as coupling faults (CF). This is the result of dual cell faults, it involves 2 states: (i) growing state implies "0" to "1" (ii) falling state that does the same.
- Read destructive error: the single-cell errors called read destructive error (RDF) that occurred when SRAM cell values are reversed. The incorrect value is output, while a read operation is performed in the cell. It is stated as 0r0/1/1 at the end and reads '0' take place if memory is '0,' and the cell memory becomes '1' if memory is '1,' reads '1' take place, and memory becomes '0.' Finally, it is found to be 1r1/0/0.
- Write destructive faults: the single-cell fault category known as write destructive faults (WDF) includes non-transitional actions when memory cells begin to flip. Additionally, two additional models with the write destructive defect are introduced in WDF. If memory is zero, the cell becomes zero, and the write '0' is obtained. Last but not least, it is defined as 1w1/0/, which means that if memory is "1," then writing "1" will occur, and the cell will become "0."
- Transition coupling fault: a double cell error known as a "transition coupling fault (TCF)" occurs when cells from the victim word are used in a write transition operation. The fault is 0W1/0/ in the up transition and 1W0/1 in the down transition.
- Static coupling faults: static coupling faults (SCF) are dual cell faults that occur at 0/1 and then force the aggressor word into the victim word's cells when the cell is given a 0 to 1 value.

- Disturb cell coupling faults: "disturb cell coupling faults (DCCF)," a subset of dual cell faults, arise when writing otherwise read operation is carried out over word that is being attacked, disrupting the cell of the word that is being attacked.
- Incorrect read fault (IRF): when a read operation occurs at an SRAM cell and returns an inaccurate value while the memory cell's status is stable, this is referred to as an IRF, a kind of double-cell fault. When memory is '0,' reading happens; however, when cell memory is '0,' reading changes to '0' and is stated as 0r0/0/1. Memory reads as '1' if it is '1,' but when a cell's memory changes to '1', it goes back to '0' and defines '1r1/1/0.' The read procedure produces an aggressor value at the conclusion.
- Deceptive read destructive fault (DRDF): a single-cell problem called DRDF happens when the cell value is inverted and receives the correct value during a read operation. When memory is "0," a read "0" occurs, and cell memory changes to "0" after the reading process completes. If memory is "1," a read "1" occurs. After the read operation, the value is reversed, changing from '0' in the cell memory to '1' indicated as 1r1/0/0.
- Idempotent coupling fault (ICF): a double cell defect is called an ICF that performs when pushed through the aggressor text cell contains minimum/maximum transition of write operation for producing the victim word displays the result in the cell.

### 2.7. BISR

This work deals with two certain processes that are explained in the flowchart of Figure 2. The NBOA is utilized to perform a BISR operation. The NBOA actually improves its work performance. The spare columns and rows are utilized in preference to faulty cells utilizing the NBOA BISR method. The primary focus of NBOA work will minimize the power dissipation through minimizing alteration procedure on conventional NBOA without compromising its efficacy. The NBOA performs repair analyzer owing to minimum energy consumption. At first, the information on memory loss, namely memory ID, faulty cell, faulty position, and defect information, is transmitted to BISR to repair the failed memory faulty cells. Redundancy logic (RL) and extra memory or column-row blocks are used in NBOA BISR.

The problematic addresses discovered during the memory test procedure are stored using redundancy management logic. Therefore, it analyzes the faulty addresses along with earlier saved addresses in the defect table based on numerous defects while memory's read and write operations coincide, the NBOA BISR model starts working and data is accessed through spare memory. The defective address is saved in the fault database in case of numerous faults. The address is kept in memory operations that are read and written but are not recorded at the fault table. BIRA planned that the spare column or row be allocated with information on a number of defective cells at a certain address.

In NBOA built-in self-repair, the analysis or problem diagnosis is repaired utilizing a pre-charge process. Therefore, relocating the XOR gate will reduce the circuit's latency and power usage. By enhancing the circuit's performance, the maximum operating frequency is attained. Additionally, the typical repairing works in a simple manner. If rows consist of huge defects, then it must be repaired, and if a column consists of more defects, then the columns and rows are constant. The number of defective cells is determined based on the memory test controller used as a benchmark through NBOA to calculate the repairing procedure.

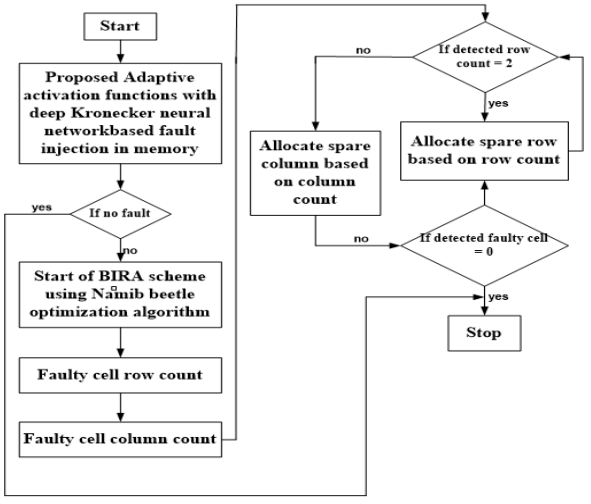



Figure 2. Flow chart of proposed BIST and BISR

In NBOA built-in self-repair, the analysis or problem diagnosis is repaired utilizing a pre-charge process. Therefore, relocating the XOR gate will reduce the circuit's latency and power usage. By enhancing the circuit's performance Namib beetles tries to collect the water and moves towards the highest hills. All over the hill, they search for places that are good for moisture to access huge amounts of water. Early in the morning, they reach the highest hills and raise their bodies up to feel the humid air. After that, they take in moisture from the air and put it in their mouths. The moisture settles on the beetle's body and then forms water droplets. This process repeats, and the water droplets become bigger and heavier. This water droplet directly enters the beetle's mouth as a fresh drink. The beetle's tough, oily skin and solitary mechanism for maintaining the foot in the air prevent water loss. The Namib beetle optimization approach performs the analysis of redundancy that affects the faulty cells using spare columns and rows instead of defective cells. The Namib beetle is expressed in (11).

$$N = \{X_1,\ X_2,\ X_3, \ldots,\ X_d\} \tag{11}$$

Here, $d$ represents the decision variable. The initial population of the Namib beetle is expressed in the (12).

$$P = \begin{bmatrix} N_{1,1} & N_{1,2} & \cdots & N_{1,d} \\ N_{2,1} & N_{2,2} & \cdots & N_{2,d} \\ \vdots & \vdots & \vdots & \vdots \\ N_{n,1} & N_{n.2} & \cdots & N_{n.d} \end{bmatrix} \tag{12}$$

Here, $P$ represents the initial population of the Namib beetle. The population of beetles for the objective function is evaluated and expressed in (13).

$$F = \begin{bmatrix} F(N_{1,1} & N_{1,2} & \cdots & N_{1,d} \\ F(N_{2,1} & N_{2,2} & \cdots & N_{2,d} \\ \vdots & \vdots & \vdots & \vdots \\ F(N_{n,1} & N_{n.2} & \cdots & N_{n.d} \end{bmatrix} = \begin{bmatrix} F(N_1) \\ F(N_2) \\ \vdots \\ F(N_n) \end{bmatrix} \tag{13}$$

Here, $F$ represents the objective function of the Namib beetle. The capacity of the numerous beetles in a place is expressed in (14).

$$C_b = M_c . sin\left(\frac{F(N_i)-F_{min}}{Fmin_{max}.\frac{\pi}{2}()}\right) \tag{14}$$

Here, $C_b$ represents capability of the beetle count in the region, $M_c$ represents maximum capability of the beetle count at 1 region, $F(N_i)$ represents competence of the beetle, $F_{min}$ represents the minimum competencies of the population beetles, $F_{max}$ represents the maximum competencies of the populace beetles. The total population of beetle that seek water is expressed in (15) and (16),

$$C = \sum_{i=1}^{n} C_b = C_1 + C_2 + \ldots + C_n \tag{15}$$

$$C \sum_{i=1}^{n} M_c . sin\left(\frac{F(N_i)-F_{min}}{F_{max}-F_{min}} . \frac{\pi}{2}\right) = M_c . sin\left(\frac{F(N_1)-F_{min}}{F_{max}-F_{min}} . \frac{\pi}{2}\right) + \cdots + M_c . sin\left(\frac{F(N_n)-F_{min}}{F_{max}-F_{min}} . \frac{\pi}{2}\right) \tag{16}$$

the method through which 1 beetle attracts another by using its present location and moisture sensing coefficient is expressed in (17),

$$N_j^{new} = N_j^{old} + Humidity . \left(N_i - N_j^{old}\right) + V \tag{17}$$

in the (17), perform the redundancy operation. The redundancy operation detects faults in the memory. Gravity regions are the regions with the greatest potential for determining water, as expressed in (18).

$$N_i^{new} = N^{old} + rnd . (N^* - \bar{N}) + V \tag{18}$$

After injecting the defective memory cells, the defective memory is repaired using (18). Additionally, the threshold value utilizes spare columns and rows, which are planned according to the number of defective cells in defective rows or columns. The predetermined threshold exceeds or is similar to 'two.' The spare row will be assigned first if the number of rows with defects is more than or equal to two, or

the spare column will be assigned next. * If the allocated spare memory (row or column) is full, the verification procedure will continue. The spare row distributes the verifying process if it is not zero by continuing to check until it achieves the null state. Hence, spare memory is maximized according to the number of defected cells. Therefore, the proposed approach gradually produces a memory test and fault repair through utilizing control flow.

## 3. RESULTS AND DISCUSSION

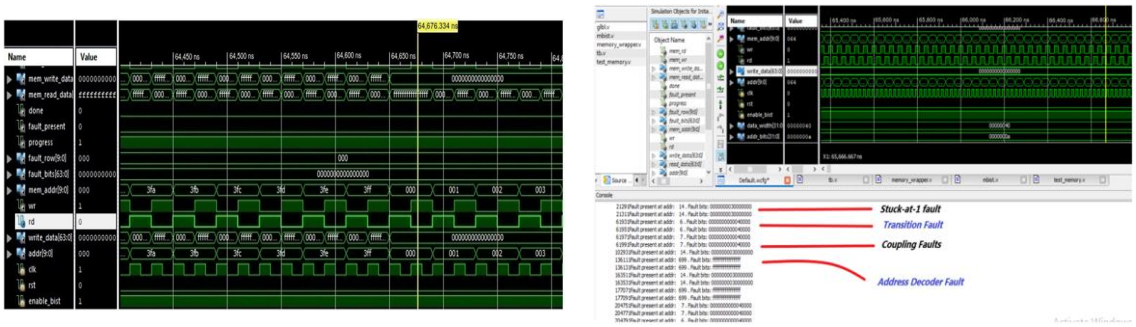
The experimental result of the proposed ADKNN-NBOA-BIST-BISR method is discussed in this section. The Verilog HDL was used to write RTL code. The simulation was carried out using the Xilinx simulator. The performance metrics, like access time, region, energy consumption, maximum operating frequency, minimum clock period, slice register, and delay, are compared with the efficacy of the proposed approach. The obtained results of the IoT-IDS-BWOA-ASRNN algorithm are analyzed with its state-of-the-art model, namely, BIST, BISR, and BISD-based BFTCP-RLSCD-BIST-BISR [21], BIST and BISR for AI accelerator (STRAIT-BIST-BISR) [22], BIST based saboteur and mutant, and BISR based counting threshold for memories (VHDL-FIT-BIST-BISR) [23], respectively.

### 3.1. Simulation output

The proposed IoT-IDS-BWOA-ASRNN technique attains higher performance based on power, region, and delay compared to the other existing approaches. A simulation and synthesis result are achieved utilizing Xilinx ISE 14.5 design suite and mentor graphics, where the Virtex-5 FPGA is executed in design. During the testing period, several defects are injected inside the memory using BWOA along with test design generation. The testing equipment created to check the memories is in charge of controlling the test patterns and defect injection into the memory. By utilizing multiple test designs and then inserting several defects for negative testing, numerous test scenarios are taken into consideration to test the memory. For 8-bit memory, 256 distinct test patterns are needed. Also, the created patterns are employed to identify any potential memory faults. The proposed IoT-IDS-BWOA-ASRNN algorithm yields the injected faults and creates test patterns. Here, the numbers of faulty cells are detected through the IoT-IDS-BWOA-ASRNN algorithm for assigning spare rows and columns.

Simulation outcomes for controller functional checking is shown in Figure 3, where read write operations are represented in Figure 3(a), and defect kind detection are displayed in Figure 3(b). The writing and reading data patterns within the MUT are represented by this simulated waveform. The writing and reading tasks were completed in accordance with the approach. The fault table for BIRA computes and writes the coupling fault, transition fault, SA fault, and address decoder faults.

The repair procedure starts as soon as information about the defective cell is collected from the test controller. After receiving the output, the simulator's screens are displayed with simulation output for fixing the faulty cells in rows and columns. The spare rows and columns are assigned for defective cell information obtained through the IoT-IDS-BWOA-ASRNN block.




(a) (b)

Figure 3. Simulation result (a) read write operations and (b) faults and fault type detection

#### 3.1.1. Performance metrics

Performance metrics are created for the obtained results. It targets the key parameters of speed, power, and area. The parameters access time, region, power consumption, slice register, delay, minimum clock period, and maximum operating frequency are compared to evaluate the efficacy of the proposed method.

### 3.1.2. Power consumption

The quantity of energy utilized per unit of time is known as power consumption. Power consumption in digital systems is crucial. It limits the battery life span of portable devices, including smart phones and laptops. The average power consumption of this ADKNN-NBOA-BIST-BISR approach is expressed in the (19).

$$P_a = \eta S_{vol}^2 (L_c . C_f) \quad (19)$$

Here, $C_f$ represents the clock frequency, $L_c$ represents the load capacitance, $\eta$ represents the activation factor, and $S_{vol}^2$ represents the supply voltage.

### 3.1.3. Delay

The duration of time required to send and receive a packet. It is measured in seconds and calculated by (20),

$$delay = T_S - T_R \quad (20)$$

where, $T_S$ indicates message sending time, $T_R$ denotes message receiving time.

### 3.1.4. Performance measures

The proposed model of ADKNN-NBOA-BIST-BISR attains greater results in the case of slice register, maximum operating frequency, area, power consumption, access time, minimum clock period and delay while analysed with their state of art models. Area analysis is shown in Figure 4. Here, the proposed ADKNN-NBOA-BIST-BISR method attains 33.88%, 35.75%, and 36.16% lower area while comparing to the existing BFTCP-RLSCD-BIST-BISR, STRAIT-BIST-BISR, VHDL-FIT-BIST-BISR methods respectively. Delay analysis is shown in Figure 5. Here, the proposed ADKNN-NBOA-BIST-BISR method attains 31.78%, 36.45%, and 32.87% lower delay while comparing to the existing BFTCP-RLSCD-BIST-BISR, STRAIT-BIST-BISR, VHDL-FIT-BIST-BISR methods respectively.

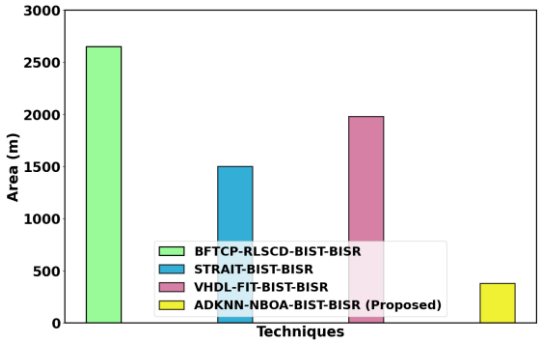



Figure 4. Performance of area analysis

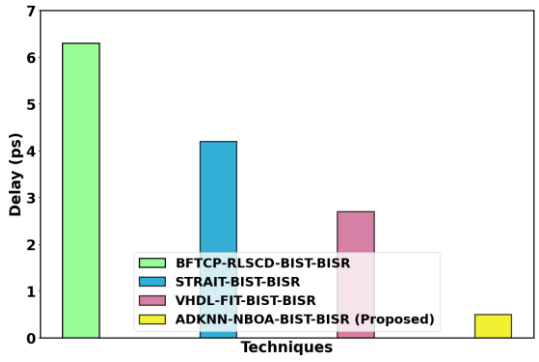



Figure 5. Performance of delay analysis

The consumption analysis is shown in Figure 6. Here, the proposed ADKNN-NBOA-BIST-BISR method attains 34.66%, 32.67%, and 35.86% lower power consumption while compared with existing methods such as, BFTCP-RLSCD-BIST-BISR, STRAIT-BIST-BISR, VHDL-FIT-BIST-BISR, respectively. Figure 7 shows slice register analysis. Here, the proposed ADKNN-NBOA-BIST-BISR method attains 35.44%, 34.57%, and 32.76% lower slice register while compared with existing methods such as, BFTCP-RLSCD-BIST-BISR, STRAIT-BIST-BISR, VHDL-FIT-BIST-BISR, respectively.

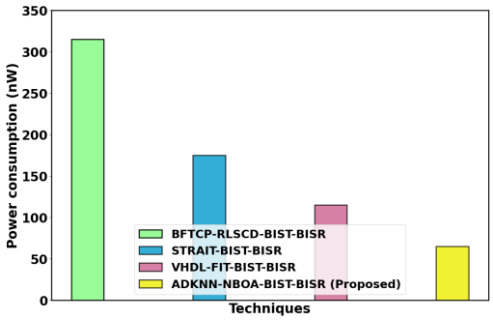



Figure 6. The performance analysis of power consumption

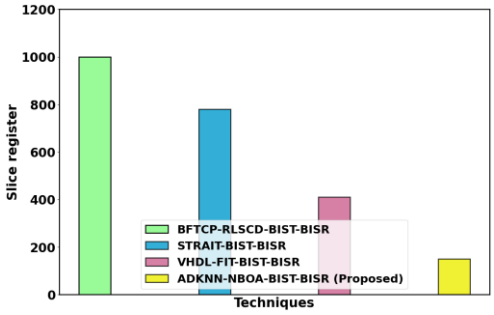



Figure 7. The performance of slice register analysis

Figure 8 shows maximum operating frequency analysis. Here, the proposed ADKNN-NBOA-BIST-BISR method attains 34.99%, 32.89%, and 33.78% higher maximum operating frequency while compared with existing methods such as, BFTCP-RLSCD-BIST-BISR, STRAIT-BIST-BISR, VHDL-FIT-BIST-BISR, respectively. Time analysis is shown in Figure 9. Here, the proposed ADKNN-NBOA-BIST-BISR method attains 35.51%, 37.21, and 32.99% lower access time while compared with existing methods such as, BFTCP-RLSCD-BIST-BISR, STRAIT-BIST-BISR, VHDL-FIT-BIST-BISR, respectively. Figure 10 shows minimum clock period analysis. Here, the proposed ADKNN-NBOA-BIST-BISR method attains 35.73%, 33.18%, and 32.01% lesser minimum clock period while compared with existing BFTCP-RLSCD-BIST-BISR, STRAIT-BIST-BISR, VHDL-FIT-BIST-BISR, methods respectively.

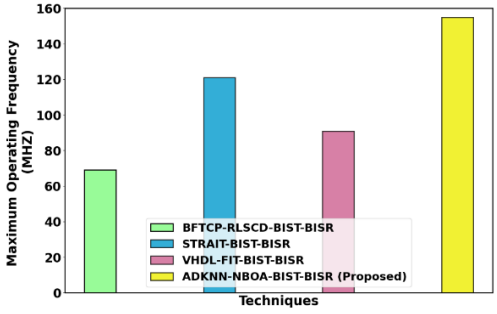



Figure 8. Performance of maximum operating frequency analysis

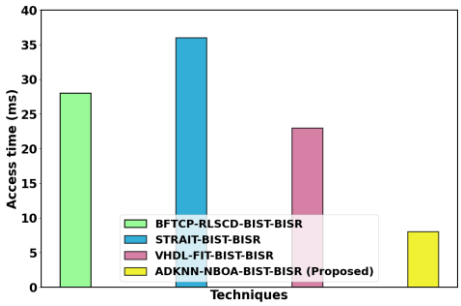



Figure 9. the performance analysis of access time

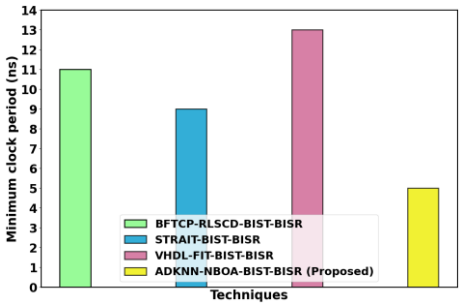



Figure 10. The performance analysis of minimum clock period

## 4. CONCLUSION

This research presented BIST is to test the memories and BISR methods to repair faulty cells for any recent (ADKNN-NBOA-BIST-BISR) based on SoC devices was successfully implemented. The proposed ADKNN-NBOA-BIST-BISR approach is performed in Xilinx simulator. The performance of the proposed ADKNN-NBOA-BIST-BISR approach attains 7.69%, 9.504%, and 10.805% lower minimum clock period; 6.04%, 10.749%, and 13.73% higher maximum operating frequency; 5.23%, 4.305%, and 5.75% lower access time; 9.52%, 10.28%, and 8.14% high lower power consumption; while compared with the existing methods such as, BFTCP-RLSCD-BIST-BISR, STRAIT-BIST-BISR, VHDL-FIT-BIST-BISR, respectively. The proposed study can open up many research tracks where further improvements are possible to reduce the cost and improve parameters like area, power, and frequency of the products.

## ACKNOWLEDGMENT

The authors extend their appreciation to Prince Sattam bin Abdulaziz University for funding this research work through the project number (PSAU/2023/01/ 25578).

## REFERENCES

[1] A. Pavlidis, M.-M. Louerat, E. Faehn, A. Kumar, and H.-G. Stratigopoulos, "SymBIST : symmetry-based analog and mixed-signal built-in self-test for functional safety," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 6, pp. 2580–2593, Jun. 2021, doi: 10.1109/TCSI.2021.3067180.

[2] V. G. Ryabtsev and S. V. Volobuev, "Built-in self-repairing system-on-chip RAM," *Russian Microelectronics*, vol. 50, no. 7, pp. 504–508, Dec. 2021, doi: 10.1134/S1063739721070118.

[3] H. Lee, H. Oh, and S. Kang, "On-chip error detection reusing built-in self-repair for silicon debug," *IEEE Access*, vol. 9, pp. 56443–56456, 2021, doi: 10.1109/ACCESS.2021.3071517.

[4] G. Karthy and P. Sivakumar, "Optimized counting threshold built-in redundancy analysis for memories," *Microprocessors and Microsystems*, vol. 81, p. 103682, Mar. 2021, doi: 10.1016/j.micpro.2020.103682.

[5] S.-K. Lu, J.-S. Shih and S.-C. Huang, "Design-for-testability and fault-tolerant techniques for FFT processors, " in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 6, pp. 732-741, June 2005, doi: 10.1109/TVLSI.2005.844306.

[6] G. Hantos, G. Simon, and M. P. Y. Desmulliez, "On the use of acoustic methods for the detection of electrostatic capture of diaphragm in capacitive MEMS microphones," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 12, no. 3, pp. 454–461, Mar. 2022, doi: 10.1109/TCPMT.2021.3107225.

[7] M. Costa and S. Beerla, "Enabling ECC and repair features in an eFuse box for memory repair applications," in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, IEEE, Apr. 2021, pp. 221–226, doi: 10.1109/ISQED51717.2021.9424327.

[8] C. Cui and J. Huang, "A 3DIC interconnect interface test and repair scheme based on Hybrid IEEE1838 die wrapper register and BIST circuit," in *2021 IEEE European Test Symposium (ETS)*, IEEE, May 2021, pp. 1–2. doi: 10.1109/ETS50041.2021.9465378.

[9] K. H. Ng, N. E. Alias, A. Hamzah, M. L. P. Tan, U. U. Sheikh, and Y. A. Wahab, "A March 5n FSM-based memory built-in self-test (MBIST) architecture with diagnosis capabilities," in *2022 IEEE International Conference on Semiconductor Electronics (ICSE)*, IEEE, Aug. 2022, pp. 69–72, doi: 10.1109/ICSE56004.2022.9863160.

[10] S.-K. Lu, S.-C. Tseng, and K. Miyase, "Fine-grained built-in self-repair techniques for NAND flash memories," in *2022 IEEE International Test Conference (ITC)*, IEEE, Sep. 2022, pp. 391–399, doi: 10.1109/ITC50671.2022.00047.

[11] V. Midasala, G. Lakshminarayana, V. P. C. Reddy, B. H. Krishna, P. M. Kumar, and N. V. Krishna, "Design of hybrid memory built in self test using linear feedback shift registers," in *2022 6th International Conference on Electronics, Communication and Aerospace Technology*, IEEE, Dec. 2022, pp. 564–568, doi: 10.1109/ICECA55336.2022.10009449.

[12] P.-Y. Tan, C.-H. Tung, C.-W. Wu, M. Lee, and G. Liao, "A memory built-in peer-repair architecture for mesh-connected processor array," in *2022 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, IEEE, Apr. 2022, pp. 1–4, doi: 10.1109/VLSI-DAT54769.2022.9768060.

[13] S. Im, G. Nam, S. Park, and M. Noh, "Advanced safety test solution for automotive SoC based on in-system-test architecture," in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, May 2022, pp. 2290–2293, doi: 10.1109/ISCAS48785.2022.9937235.

[14] H. Lee, Y. Yoo, S. H. Shin, and S. Kang, "ECMO: ECC architecture reusing content-addressable memories for obtaining high reliability in DRAM," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 6, pp. 781–793, Jun. 2022, doi: 10.1109/TVLSI.2022.3153894.
[15] L. Zhang, Z. Li, X. Huang, and K. Chakrabarty, "Enhanced built-in self-diagnosis and self-repair techniques for daisy-chain design in MEDA digital microfluidic biochips," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 10, pp. 3236–3249, Oct. 2023, doi: 10.1109/TCAD.2023.3244524.
[16] W. Kang, C. Lee, H. Lim and S. Kang, "Optimized built-in self-repair for multiple memories," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 6, pp. 2174-2183, June 2016, doi: 10.1109/TVLSI.2015.2499387.
[17] A. Chaudhuri, S. Banerjee, J. Kim, S. K. Lim, and K. Chakrabarty, "Built-in self-test of high-density and realistic ILV layouts in monolithic 3-D ICs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 31, no. 3, pp. 296–309, Mar. 2023, doi: 10.1109/TVLSI.2022.3228850.
[18] P.-H. Lee *et al*., "33.1 A 16nm 32Mb embedded STT-MRAM with a 6ns read-access time, a 1M-Cycle write endurance, 20-year retention at 150°C and MTJ-OTP solutions for magnetic immunity," in *2023 IEEE International Solid- State Circuits Conference (ISSCC)*, IEEE, Feb. 2023, pp. 494–496. doi: 10.1109/ISSCC42615.2023.10067837.
[19] A. D. Jagtap, Y. Shin, K. Kawaguchi, and G. E. Karniadakis, "Deep Kronecker neural networks: a general framework for neural networks with adaptive activation functions," *Neurocomputing*, vol. 468, pp. 165–180, Jan. 2022, doi: 10.1016/j.neucom.2021.10.036.
[20] M. Chahardoli, N. O. Eraghi, and S. Nazari, "Namib beetle optimization algorithm: a new meta-heuristic method for feature selection and dimension reduction," *Concurrency and Computation: Practice and Experience*, vol. 34, no. 1, Jan. 2022, doi: 10.1002/cpe.6524.
[21] X. Li, G. Yan, and C. Liu "Built-in fault-tolerant computing paradigm for resilient large-scale chip design: a self-test, self-diagnosis, and self-repair-based approach," Pre-order Softcover, 2023.
[22] H. Lee, J. Kim, J. Park, and S. Kang, "STRAIT: self-test and self-recovery for AI accelerator," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 9, pp. 3092–3104, Sep. 2023, doi: 10.1109/TCAD.2023.3236875.
[23] K. Gopalan and S. Pothiraj, "Retracted article: a saboteur and mutant based built-in self-test and counting threshold-based built-in self repairing mechanism for memories," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 6, pp. 6651–6663, Jun. 2021, doi: 10.1007/s12652-020-02284-5.
[24] M. A. Ahmed and S. Alnatheer, "Deep q-learning with bit-swapping-based linear Feedback shift register fostered built-in self-test and built-in self-repair for SRAM," *Micromachines*, vol. 13, no. 6, p. 971, Jun. 2022, doi: 10.3390/mi13060971.
[25] M. A. Ahmed and A. M. Abuagoub, "MBIST controller based on March-ee algorithm," *Journal of Circuits, Systems and Computers*, vol. 30, no. 09, p. 2150160, Jul. 2021, doi: 10.1142/S0218126621501607.
[26] S. Alnatheer and M. A. Ahmed, "Optimal method for test and repair memories using redundancy mechanism for SoC," *Micromachines*, vol. 12, no. 7, p. 811, Jul. 2021, doi: 10.3390/mi12070811.
[27] M. A. Ahmed, A. E. M. Eljialy, and S. Ahmad, "Memory test and repair technique for SoC based devices," *IEICE Electronics Express*, vol. 18, no. 8, pp. 20210092–20210092, Apr. 2021, doi: 10.1587/elex.18.20210092.
[28] D. Han, H. Lee, and S. Kang, "Effective spare line allocation built-in redundancy analysis with base common spare for yield improvement of 3D memory," *IEEE Access*, vol. 9, pp. 76716–76729, 2021, doi: 10.1109/ACCESS.2021.3082949.
[29] S. Roy, S. K. Millican, and V. D. Agrawal, "Special session – machine learning in test: a survey of analog, digital, memory, and RF integrated circuits," in *2021 IEEE 39th VLSI Test Symposium (VTS)*, IEEE, Apr. 2021, pp. 1–14, doi: 10.1109/VTS50974.2021.9441051.

## BIOGRAPHIES OF AUTHORS

**Dr. Suleman Alnatheer** is an assistant professor in computer engineering at Prince Sattam bin Abdulaziz University. In this role, he conducts research on the topic of statistical learning theory that can be applied to next-generation wireless communication technologies. In addition, he teaches undergraduate courses in various electrical and computer engineering topics. Prior to joining Prince Sattam bin Abdulaziz University, he worked as a lecturer in the College of Telecommunication and Information, where he was lucky enough to work with amazing people. He holds a Ph.D. from Stevens Institute of Technology and an M.Sc. and B.Sc. from George Washington University. He can be contacted at email: s.alnatheer@psau.edu.sa.

**Dr. Mohammed Altaf Ahmed** received his Ph.D. in Engineering from GITAM University, Vishakhapatnam, India, for the thesis entitled "design of a built-in self-test controller using the march algorithm for fault diagnosis in embedded memories" in the year 2018. He received the best thesis award and the gold medal for his Ph.D. He completed a master's in engineering and technology with a specialization in embedded systems in 2008 from Jawaharlal Nehru Technological University, Hyderabad, India. He received his bachelor's in engineering (Electronics and Communications) in 2000 from Swami Ramanand Teerth Marathwada University, Nanded, India. He is presently focusing his research on VLSI design, embedded systems, and IoT devices. He has successfully implemented several research grant projects in the same area and published research articles in leading academic journals indexed in Scopus and Science. He can be contacted at email: m.altaf@psau.edu.sa.