<Final Report: EEE4610-01>

An Effective Built-In Redundancy Analysis Design for Memory with Multiple Banks

Report Advisor: Sungho Kang

December 2023

Byeongyun Kim, Jinkwan Kim,
Taeheon Kim, Kyeongseok Oh
School of Electrical and Electronic Engineering
College of Engineering
Yonsei University

ABSTRACT

An Effective Built-in Redundancy Analysis Design for Memory with Multiple Banks

As chip integration continues to increase, memory dice are being stacked like HBM to reduce size and increase memory capacity. In addition, as with automotive chips, stability issues are so important that the number of devices that require internal testing continues to increase. In this situation, BIRA (Built-In Redundancy Analysis) has gained greater importance. In the case of the previous BIRA, faults are stored using CAM (Content Addressable Memory), and the fault is analyzed by several modules to determine whether repair is possible and to output a repair solution. Herein, we refer to this previous BIRA and design an improved BIRA structure that can handle 3 spare structures with 2 banks. This proposed BIRA allows the repair rate to be increased by adding a reanalyze process rather than the previous BIRA. Additionally, it is expected that the RA time may be reduced for signals that can process results quickly by adding early termination signals, etc.

Key words: Built-in Redundancy Analysis (BIRA), Spare Structure, Repair Rate, Reanalysis

1. Introduction

Recently, as the importance of memory increases, the importance of repairing memory is also increasing. Due to the development of semiconductor process technology, attempts to stack memory in 3D increased. 3D memory has advantages in terms of integration and capacity. However, when a fault occurs in a layer, it affects the entire memory. So, the importance of a fault repair method is greater in 3D memory.

The memory repair methods include BIST (Built-In Self-Test) and BIRA (Built-In Redundancy Analysis). First, BIST which detects the faults and transmits its information, is internal testing in memory unlike the existing testing methods using external test equipment such as ATE (Automatic Test Equipment). Therefore, BIST can be tested without the connection of external equipment. This means that tests can be done at startup or during operation, and therefore BIST is mainly used in fields such as automotive and military where product reliability is important.

BIRA exists on the base layer of 3D memory, and it stores and analyzes the fault addresses from BIST and finds the repair solutions that cover all faults by given spares. Memory has some extra capacity (spare), which is used for repairing faults. BIRA checks whether the spares can cover all faults in memory. If spares cover all faults, it is repairable. Otherwise, it is not repairable.

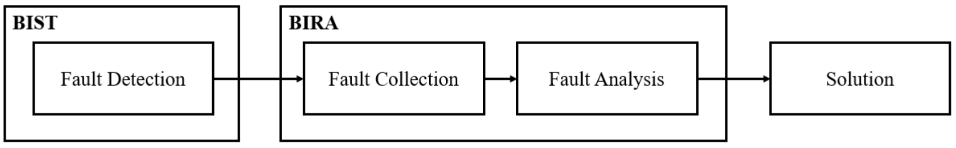


Figure 1. Memory Repair Process with BIST and BIRA.

To improve the performance of BIRA, H/W overhead and RA (Redundancy Analysis) time must be reduced, and the repair rate of BIRA must be increased. These factors are important because they determine the cost and performance of a semiconductor. However, there is a trade-off because these factors cannot be improved at the same. Even if all factors are important, the most important factor in current semiconductors is yield. Therefore, the repair rate, which is related to yield, is the most important feature in memory.

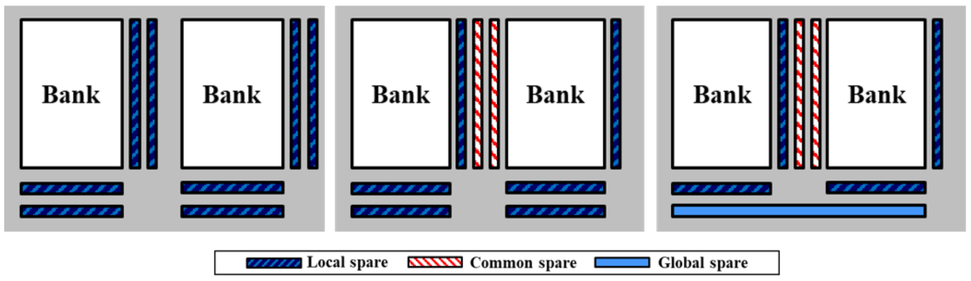


Figure 2. Given Spare Structure of This Project

So, the objective of this project is to design an effective BIRA with a 100% normalized repair rate in the memory with multiple banks. The repair rate is different when using various spare structures [1]. Therefore, proposed BIRA was designed to operate on these three spare structures (**Fig.2**) and checked how the repair rate and analysis time are changed with these structures.

2. Background

2.1 Spec

2.2.1. Repair rate

$$Reapir\ rate = \frac{\#\ of\ repaired\ memories}{\#\ of\ tested\ memory} \tag{1}$$

$$Normalized\ repair\ rate = \frac{\#\ of\ repaired\ memories}{\#\ of\ repairable\ memory} \tag{2}$$

Repair rate and Normalized repair rate are defined in (1) and (2). The repair rate is defined as the ratio of repaired memory to tested memory. Normalized repair rate is defined as the ratio of repaired memory to repairable memory. Normalized repair rate is useful for comparing repair rates between multiple BIRAs. Also, if the normalized repair rate is 100%, the BIRA can be said to have an optimal repair rate.

2.2.2. Hardware overhead

Hardware overhead refers to the additional hardware required to implement BIRA in memory. Minimizing this overhead is essential for creating a more cost-effective BIRA, so algorithms need to be developed to achieve this goal.

2.2.3. Analysis speed

Analysis speed refers to the speed of the process of finding a repair solution that can cover all faults through BIRA. This is also an important factor because the faster it is, the faster memory repair can be completed.

2.2 Classification of Fault

2.2.1. Pivot Fault

A pivot fault is a fault that does not share a row address or column address with other pivot faults. A pivot fault is stored as a pivot fault if the fault in the corresponding order does not share a row address or column address. CAM receives the fault information from BIST and sequentially stores its information. And the faults stored in the CAM must be covered by spares.

2.2.2. Non-pivot Fault

A non-pivot fault is a fault that shares a row address or column address with the pivot fault stored in the Pivot CAM (PCAM) during the process of sequentially receiving memory fault information and storing it in the CAM. In this case, it is stored as a fault dependent on the pivot fault, and the address is stored using a pointer and descriptor, so information about the fault can be stored efficiently from a hardware overhead perspective.

2.2.3. Must Repair Line

Must repair line indicates a line that must be repaired unconditionally with information on the relevant faults. Depending on the spare structure, the rules for determining must repair line vary. For example, if the number of extra column cells that can be allocated in one block is 2, and if the number of faults sharing a row is 3 or more, it is designated as a must-repair line. The generally expressed expression is as following equation [3].

$$\begin{gathered}\#\ of\ faults\ on\ the\ row(column)\ faulty\ lie\ in\ a\ block \\ > \#\ of\ usable\ column(row)\ spares\ in\ a\ block\end{gathered} \tag{3}$$

2.3 Classification of Spares

In the case of Cost-Efficient spare structures [2], spares can be classified from various perspectives. Spares can be divided according to length, coverage area, and direction. First, in the case of length, it is divided into single and double. A single is a spare that can cover one memory bank per spare, and a double can cover two memory banks per spare. Second, the scope of coverage can be divided into local, common, and global. A local spare is a spare that can cover only one block per spare, and the blocks that can be covered are predetermined. A common spare is a spare that can cover two adjacent memory blocks. Therefore, local and common spares have the advantage of fast access speed because their physical location is close to the memory block. And if you use a lot of common spares, The BIRA can repair problems more efficiently, so the repair rate increases. In the case of a global spare, it is a spare that can cover all lines of memory. However, the access speed is slow due to the physical distance from the memory block, but the repair rate increases because all locations can be accessed.

Table 1. Classification of spares

Length	Coverage range	Direction
Single (s) Double (d)	Local (l) Common (c) Global (g)	Row (r) Column (c)

3. Proposed BIRA

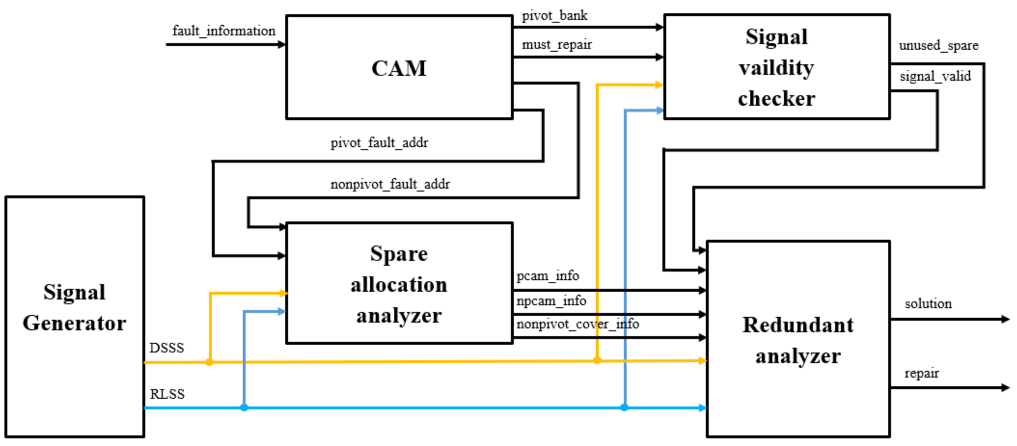


Figure 3. Block diagram for proposed BIRA.

The proposed BIRA is a BIRA that can be used in all three spare structs in **Fig.2** The spare structs in **Fig.2**, from the left, will be referred to as spare struct 1, spare struct 2, and spare struct 3, respectively. As shown in **Fig.3**, the proposed BIRA consists of Content Addressable Memory (CAM), Signal Generator (SG), Signal Validity Checker (SVC), Spare Allocation Analyzer (SAA), and Redundant Analyzer (RA).

In CAM, information about faults is input from BIST, classified into pivot faults and non-pivot faults, and stored in PCAM and NPCAM. Afterward, the saved information is output to SVC and SAA.

SG outputs DSSS and RLSS, which are signals used for repair. Each signal is created and output appropriately according to the spare struct.

SVC determines whether a repair is possible through the input spare struct and DSSS and RLSS signals. It also outputs an *unused_spare* signal that indicates whether each given spare is used.

SAA determines whether each non-pivot fault is covered when using the input DSSS and RLSS signals. Afterwards, the obtained information is output to RA.

In RA, the final repair decision is made by examining the cover status of non-pivot faults identified by SAA and checking the *unused_spare*. If there are non-pivot faults with unused spares that have not been covered yet, the information of those faults is output to CAM, instructing a reanalysis. If, in the end, all non-pivot faults are covered, indicating repair feasibility, the solution is output. Otherwise, if repair is not possible, a termination signal is generated to proceed with the analysis of the next DSSS and RLSS.

As such, the proposed BIRA was designed to be reanalyzed multiple times until results were derived to achieve a normalized repair rate of 100%.

3.1. CAM

CAM is a memory structure for storing fault information. The CAM structure proposed in this project consists of PCAM and NPCAM. Dividing the CAM like this and storing fault information has the advantage of reducing hardware overhead. The CAM module receives fault information from BIST every clock. CAM stores faults in PCAM and NPCAM according to its internal algorithm. The CAM receives fault information from the time fault information begins to be input from BIST until the *test_end* signal is received. After the test is completed, CAM uses a SAA to output the information stored in PCAM and NPCAM every clock. At the same time, the pivot's bank and must repair signals for validation are output using the SVC. Then, the fault analysis and repair process for the memory is completed by receiving the information that needs to be re-analyzed from the RA and storing it back in the CAM. The contents of PCAM and NPCAM are shown in the following figure.

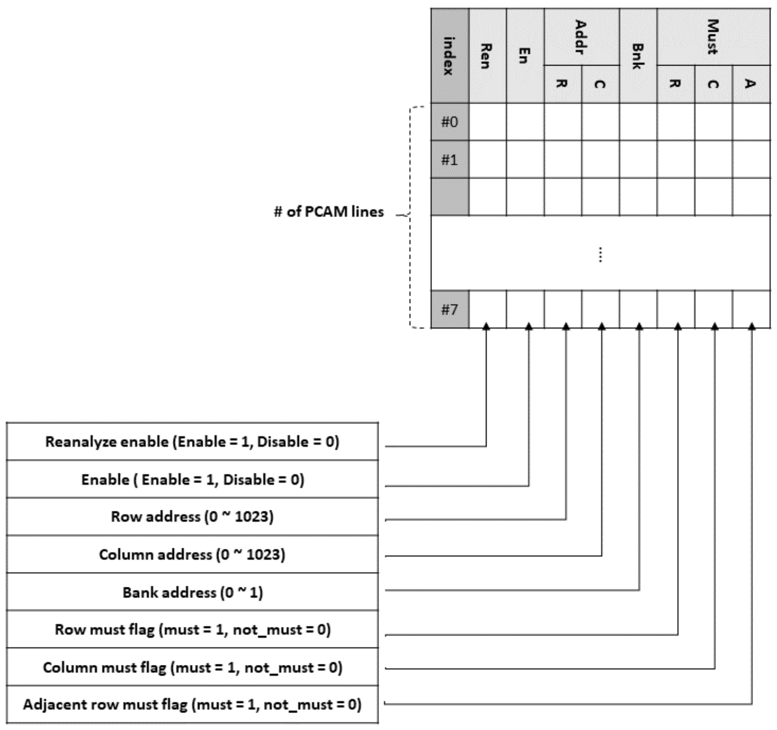

index
Ren
En
Addr
R
C
Bnk
Must
R
C
A
#0
#1
#7
# of PCAM lines
Reanalyze enable (Enable = 1, Disable = 0)
Enable (Enable = 1, Disable = 0)
Row address (0 ~ 1023)
Column address (0 ~ 1023)
Bank address (0 ~ 1)
Row must flag (must = 1, not_must = 0)
Column must flag (must = 1, not_must = 0)
Adjacent row must flag (must = 1, not_must = 0)

Figure 4. PCAM Structure

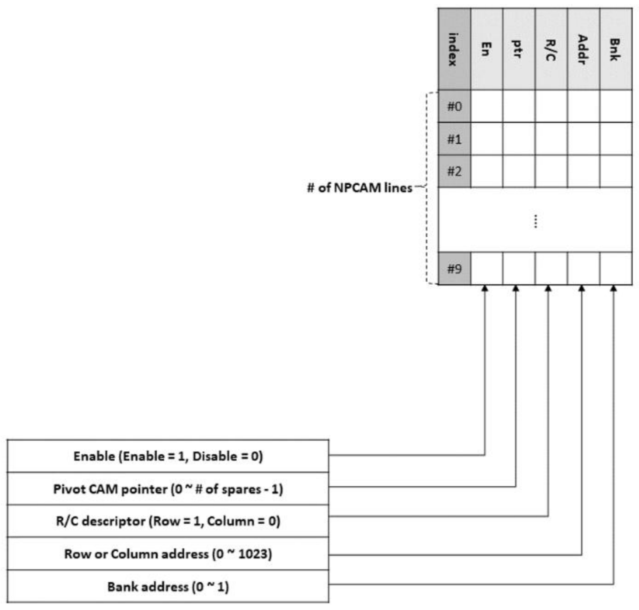

index
En
ptr
R/C
Addr
Bnk
#0
#1
#2
#9
# of NPCAM lines
Enable (Enable = 1, Disable = 0)
Pivot CAM pointer (0 ~ # of spares - 1)
R/C descriptor (Row = 1, Column = 0)
Row or Column address (0 ~ 1023)
Bank address (0 ~ 1)

Figure 5. NPCAM Structure

They have 8 and 10 rows, respectively, and the fault contents stored in PCAM have a size of 27 bits, and the fault contents saved in NPCAM have a size of 17 bits. The number of rows of PCAM and NPCAM is determined by the following equation [4].

$$\#\ of\ PCAM\ lines = \#\ of\ total\ spares$$

$$\#\ of\ NPCAM\ lines = (S_{slc} + S_{scc} + S_{sgc} + 2S_{dgc} + 2S_{dlc}) \times (\#\ of\ usable\ row\ spares\ for\ a\ block)$$

$$+(S_{slr} + S_{scr} + S_{sgr} + 2S_{dgr} + 2S_{dlr}) \times (\#\ of\ usable\ column\ spares\ for\ a\ block) \quad (4)$$

Additionally, the CAM structure is used again during the reanalysis process. After all the information received and stored from BIST is stored and output, the information of the non-pivot fault that has not been covered is input again from the RA while maintaining the saved information, and the information is stored in the remaining space of the PCAM and re-analysis is performed. Through this method, the CAM in BIRA can be said to be a module that operates for storing and outputting fault information in the memory structure.

Table 2. Input Signals of CAM Module

From	Signal Name	Bits	Description
BIST, Redundant Analyzer	*row_addr*	10	Row address of the fault
	col-addr	10	Column address of the fault
	bank_addr	2	The bank address of the fault
	col_flag	8	Since BIST sends a signal every 8 bits to CAM, col_flag determines the exact location of the fault.
	test_end	1	A signal indicating that the test is over.
Top Module	*rst*	1	reset signal
	clk	1	clock signal (100MHz)
	spare_struct	2	Given spare structure. 1 = structure 1 2 = structure 2 3 = structure 3

Table 3. Output Signals of CAM Module

To	Signal Name	Bits	Description
Spare Allocation Analyzer	*pivot_fault_addr*	27	Information stored in the PCAM structure
	nonpivot_fault_addr	17	Information stored in the NPCAM structure
Signal Validity Checker	*pivot_bank*	2	Pivot bank address stored in PCAM for testing whether the signal is valid
	must_repair	3	Must repair signal stored in PCAM for testing whether the signal is valid

3.2 Signal Generator

The SG is a module that generates DSSS and RLSS signals, which are decision signals used in RA, SAA, and SVC.

The DSSS signal is 8 bits equal to the size of the PCAM. Each bit determines whether to use a row or column spare for repairing each PCAM entry. When each bit is set to 1, it is repaired with a row spare, and when each bit is set to 0, it is repaired with a column spare.

The RLSS signal is 3 bits, which is the maximum number of row spares for spare struct 3. Each bit determines global spare will be used or not for repair when using a row spare. When each bit is set to 1, the global spare is used, and when each bit is set to 0, the local spare is used.

The SG outputs the same DSSS/RLSS signal until the signal requesting the next DSSS/RLSS signal comes in. Here, requesting the next DSSS/RLSS signal is early_term_SVC2SG and termination. If these come in 1, the DSSS/RLSS signal changes to the next signal and outputs it.

Table 4. Input Signals of SG Module

From	Signal Name	Bits	Description
Top Module	*rst*	1	reset signal
	clk	1	clock signal (100MHz)
	spare_struct	2	Given spare structure. 1 = structure 1 2 = structure 2 3 = structure 3
BIST	*test_end*	1	A signal indicating that the BIST test is over. After it comes 1, SG starts output DSSS/RLSS.
SVC	*early_term_SVC2SG*	1	When repair is not possible with the given structure, the next signal is requested.
RA	*termination*	1	Be 1 when reanalyzing is finished and a test for the next signal is required

Table 5. Output Signals of SG Module

To	Signal Name	Bits	Description
RA SVC SAA	*DSSS*	8	Determine whether the spare at the i-th PCAM address is the row or column spare.
	RLSS	3	Determine which row spare is a global spare.
SVC	*Start_SVC*	1	Request SVC to start when the next signal is output.

3.3 Signal Validity Checker

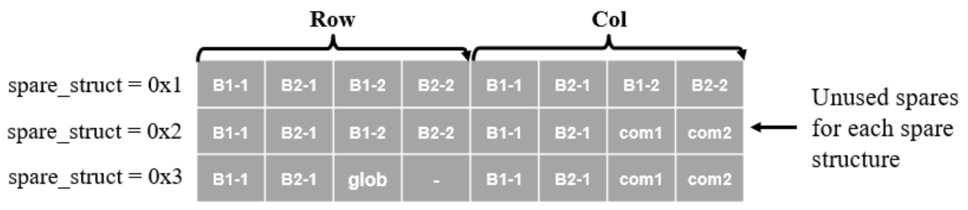


Figure 6.

The SVC module determines the spare used to cover each pivot fault by combining the DSSS/RLSS signal and the *bnk_addr* signal and outputs an *unused_spare* signal.

For the *unused_spare* signal, the spare that each bit represents varies depending on the *spare_struct*. If each bit is 1, the spare is not used, and if each bit is 0, the spare is used. The *unused_spare* signal is initially set to 1111_1111, and whenever *bnk_addr* comes from CAM, spares are allocated according to priority and the *unused_spare* signal is continuously updated.

To allocate spares more efficiently, local spares that can be used in each bank are allocated first, and if there are no more local spares available, common spares are allocated.

For the *unused_spare* signal, all bits of the row/column spare of each bank is 0, but if additional use is required, the repair is impossible and the *signal_valid* is output as 0.

In addition, check whether all must-repair lines are covered through *bnk_addr* and must-flag, and if not covered, determine whether each must-repair line can be covered through *unused_spare*, and output the *signal_valid* signal as 1 if possible, and output 0 if not.

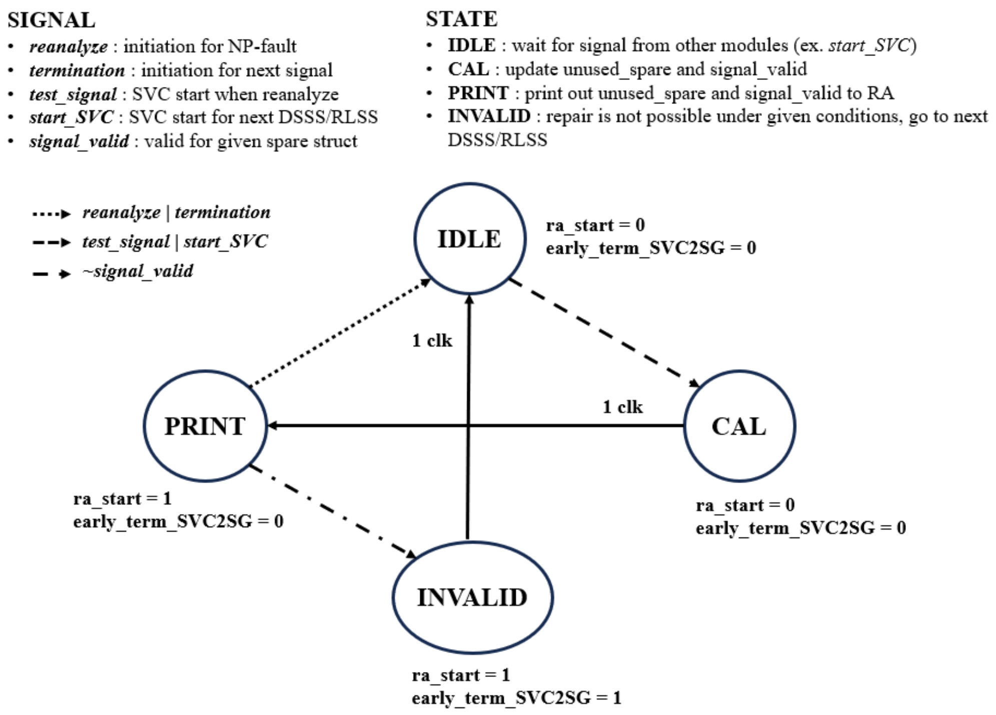

SIGNAL
• reanalyze : initiation for NP-fault
• termination : initiation for next signal
• test_signal : SVC start when reanalyze
• start_SVC : SVC start for next DSSS/RLSS
• signal_valid : valid for given spare struct
STATE
• IDLE : wait for signal from other modules (ex. start_SVC)
• CAL : update unused_spare and signal_valid
• PRINT : print out unused_spare and signal_valid to RA
• INVALID : repair is not possible under given conditions, go to next DSSS/RLSS
reanalyze | termination
test_signal | start_SVC
~signal_valid
IDLE
ra_start = 0
early_term_SVC2SG = 0
1 clk
PRINT
1 clk
CAL
ra_start = 1
early_term_SVC2SG = 0
ra_start = 0
early_term_SVC2SG = 0
INVALID
ra_start = 1
early_term_SVC2SG = 1

Figure 7. State Diagram of SVC.

SVC is designed as a Finite State Machine (FSM). The figure above represents the state diagram, consisting of four states: IDLE, CAL, PRINT, and INVALID.

IDLE is the state where signals are initialized, and the system is waiting. CAL is the state where output signals are calculated and updated based on the input signals. After the calculation, the system transitions to the PRINT state in just one clock cycle. In the PRINT state, the system outputs the *ra_start* signal for the RA to operate and continues to output the calculated *unused_spare* and *signal_valid* signals. If a reanalyze signal is received, the system returns to the IDLE state to initialize signals for further analysis. Similarly, if a termination signal is received, the system goes back to the IDLE state to reset signals for the next DSSS/RLSS signal analysis. If *signal_valid* is 0 in the PRINT state, it indicates that the repair is not possible for the current DSSS/RLSS. In this case, the system transitions to the INVALID state, and outputs the early_term_SVC2SG signal to instruct SG to output the next

DSSS/RLSS signal. After 1 clock cycle, it returns to the IDLE state to initialize signals and wait. The process repeats these based on input signals.

Table 6. Input Signals of SVC Module

From	Signal Name	Bits	Description
Top Module	*rst*	1	reset signal
	clk	1	clock signal (100MHz)
	spare_struct	2	Given spare structure. 1 = structure 1 2 = structure 2 3 = structure 3
SG	*DSSS*	8	Determine whether the spare at the i-th PCAM address is row or column spare.
	RLSS	3	Determine which row spare is a global spare.
	start_SVC	1	
CAM	*bnk_addr0-7*	2	A signal indicating the bank of the pivot faults stored in PCAM.
	must_flag0-7	3	A signal indicating whether the pivot fault stored in PCAM is a must-repair line.
	test_signal	1	
RA	*reanalyze*	1	Be 1 When reanalyzing for non-pivot fault is required
	termination	1	Request SVC to start when the next signal is output

Table 7. Output Signals of SVC Module

To	Signal Name	Bits	Description
RA	*signal_valid*	1	Check whether must-repair lines are covered and generated signal satisfies the number of spares assigned to each bank.
	unused_spare	8	Notice which spares are not in use.
	ra_start	1	Whether SVC is end?

3.4 Spare Allocation Analyzer

The Spare allocation analyzer is responsible for delivering the repaired state to the RA based on the information of DSSS and RLSS signals received from the Signal generator and Pivot faults and non-pivot faults received from the CAM. It was designed by referring to the Analyzer structure of [2],[3].

In pivot fault address' enable and re-enable signals, Pivot faults with 1 enable signal are initially treated as repair, and in situations where the re-enable signal is 1, the repair is performed based on the re-enable signal. The MUX to which the *DSSS* is connected outputs the selected row/column address of the corresponding signal. Using this output, it is divided into Rows and Columns and compared with *Npr / Npc* values to see if non-pivot faults are covered by the selected row/column in RAC/CAC, which is a comparator [2]. In this process, not only the row/column address but also the bank and spare type should be considered to check whether the fault can be covered.

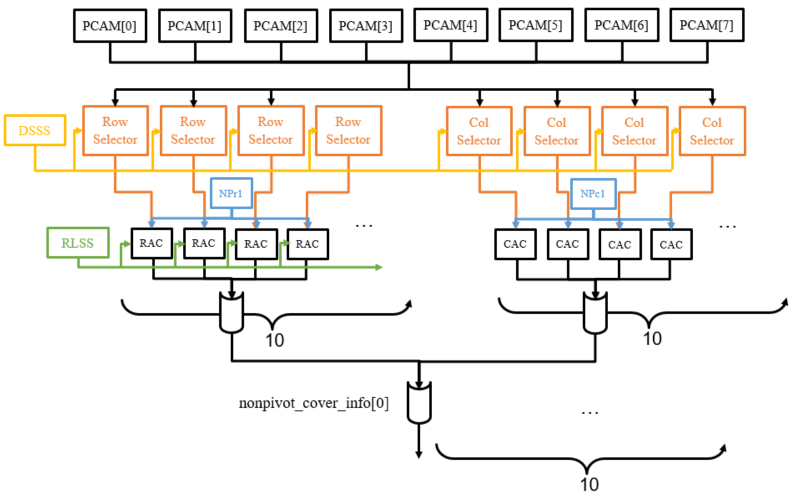


Figure 8. This is the Algorithm for SAA.

Since RAC should also take into account global spares of Row addresses, RLSS should also be processed by receiving *RC_addr, NPr_addr, and NP_bank* signals as input. In this way, after checking whether the non-pivot address is repairable in the Comparator, the signal is collected to the Or gates, and if at least one of them is repairable, the corresponding i-th non-pivot address signals that it is repairable to store the value of *nonpivot_cover_info[i]*. *uncover_nonpivot_addr* saves and sends both row and column address to immediately retrieve the row/column address of non-pivot displayed by the descriptor when saving the address of *nonpivot_fault_addr* received from the CAM to the Pivot CAM to Reanalyze from the RA.

Table 8. Input Signals of Spare Allocation Analyzer Module

From	Signal Name	Bits	Description
Top Module	*Clk*	1	clock signal (100MHz)
Signal Generator	*DSSS*	8	Determine whether the spare at the i-th PCAM address is row or column spare.
	RLSS	3	Determine which row spare is a global spare.
	Start_SVC	1	Check the timing of DSSS changing
CAM	*pivot_fault_addr*	27	Recognize Row and column address and must flag information of Pivot faults to be processed first by the Enable or Re-enable signal.
	nonpivot_fault_addr	17	Recognize the Cover status with the address as Row/Column of Pivot faults.

Table 9. Output Signals of Spare Allocation Analyzer Module

To	Signal Name	Bits	Description
Redundant Analyzer	*pivot_en*	8	Whether pivot fault is stored?
	pivot_ren	8	Whether pivot fault is stored during reanalyze?
	pivot_row_addr	10	Pivot row address
	pivot_col_addr	10	Pivot column address
	pivot_bnk_addr	2	Pivot Bank address
	nonpivot_cover_info	10	Indicate uncovered non-pivot fault
	nonpivot_en	10	Whether non-pivot fault is stored
	nonpivot_row_addr	10	Row address of non-pivot fault
	nonpivot_col_addr	10	Column address of non-pivot fault

	nonpivot_bnk_addr	2	Bank address of non-pivot fault

3.5 Redundant Analyzer

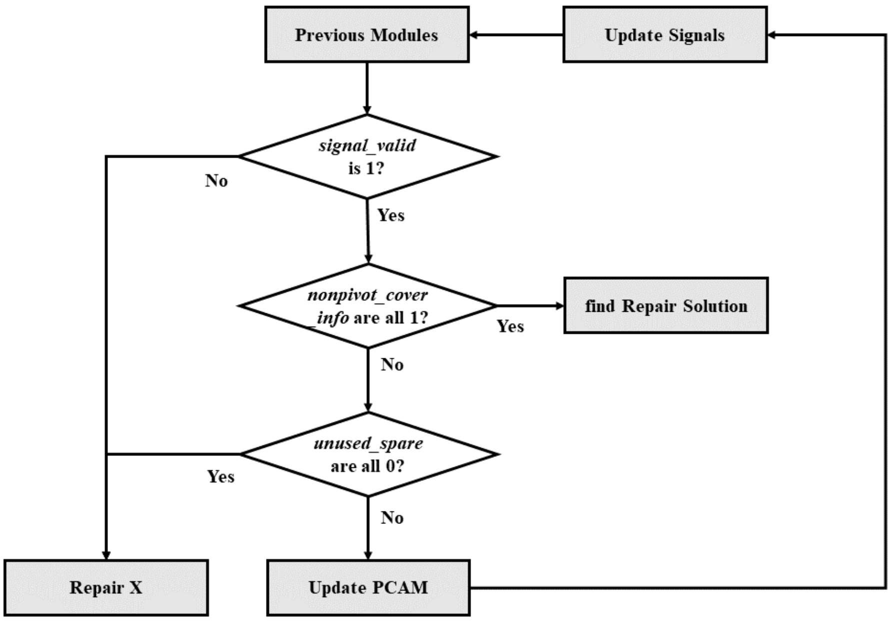


Figure 9. flowchart for RA.

Based on the results of the previous module, the Redundant Analyzer (RA) determines whether repair is possible or impossible and whether to reanalyze the current signal. RA first checks whether *signal_valid* is 0 or 1. If it is 0, RA skips reanalyze because the signal is invalid. Otherwise, RA proceeds with reanalysis.

Next, RA checks if all *nonpivot_cover_info* is all 1. Because the spare is placed in the row or column direction of the pivot fault, the pivot fault is automatically covered. Therefore, RA only needs to check whether non-pivot faults are covered. If *nonpivot_cover_info* is all 1, then all the non-pivot faults in memory are covered. Then RA finds the solutions. However, if there is any 0 at *nonpivot_cover_info*, this means that an uncovered fault exists. Then, RA checks if there are any spares left.

If there are no spares, it is not repairable because the spare cannot be used even if an uncovered fault exists. Then RA informs SG to move the next signal. Otherwise, RA sends the address of uncovered non-pivot faults to CAM. Then, CAM updates the values based on the address received from RA. when the updates in other modules are completed, RA repeats the first process again.

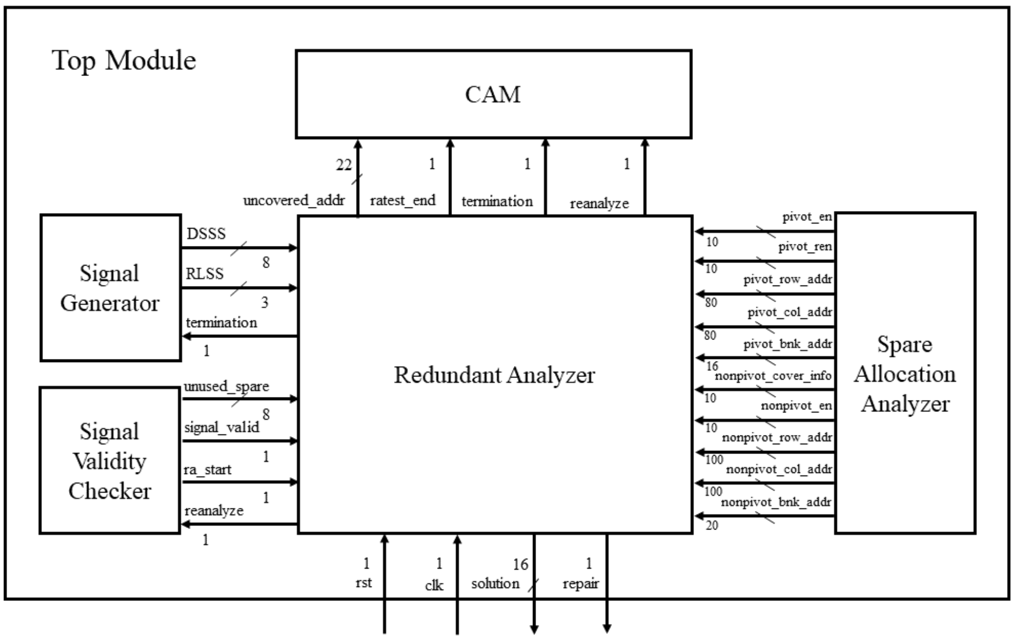


Figure 10. This is block diagram for RA perspectives.

The block diagram of RA is as follows. RA receives fault information and *nonpivot_cover_info* from SAA. DSSS and RLSS signals are received from SG. From SVC, *unused_spare* which corresponds to the remaining spare, and *signal_valid* which indicates whether the signal is valid, are received. After the SVC is completed, RA receives *the ra_start* signal from the SVC.

RA sends out *uncovered_addr* which contains the row, column, and bank address of uncovered non-pivot faults. When repair is not possible, it sends signal *termination* indicating that it is needed to move next signal. If reanalyze is required, then RA transmits signal *reanalyze* for *clk* with the number of uncovered faults. At the last *clk* for reanalyze, 1 *clk* pulse *ratest_end* occurs.

The operation of RA is divided into two types. The first one is where reanalyzing is necessary because all faults are not covered. And the other is finding solutions. Therefore, RA was implemented as a state machine because it must operate differently depending on the inputs.

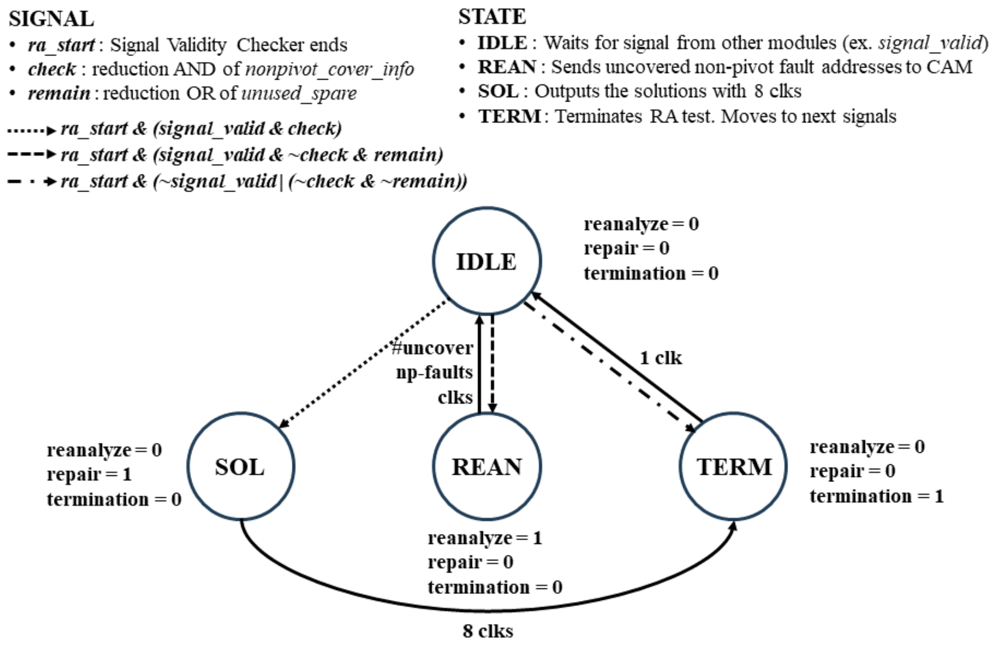

SIGNAL
• ra_start : Signal Validity Checker ends
• check : reduction AND of nonpivot_cover_info
• remain : reduction OR of unused_spare
ra_start & (signal_valid & check)
ra_start & (signal_valid & ~check & remain)
ra_start & (~signal_valid | (~check & ~remain))
STATE
• IDLE : Waits for signal from other modules (ex. signal_valid)
• REAN : Sends uncovered non-pivot fault addresses to CAM
• SOL : Outputs the solutions with 8 clks
• TERM : Terminates RA test. Moves to next signals
IDLE
reanalyze = 0
repair = 0
termination = 0
#uncover
np-faults
clks
1 clk
SOL
reanalyze = 0
repair = 1
termination = 0
REAN
reanalyze = 1
repair = 0
termination = 0
TERM
reanalyze = 0
repair = 0
termination = 1
8 clks

Figure 11. This is state diagram for RA.

The state diagram of RA is as follows. RA maintains the IDLE state until *ra_start* becomes 1. If *ra_start* is 1, then RA can change its state. If *signal_valid* is 0, then the *DSSS* (and *RLSS*) signal is invalid. So, the state is changed to TERM to end the test for the current signal and BIRA moves to the next signal. If *signal_valid* is set, there is no problem with the current signal and RA checks *nonpivot_cover_info* to know whether the current signal is repairable.

If all non-pivot faults are covered (*check* = 1), state changes from IDLE to SOL. RA determines the repair solution and transmits it to the Top Module. The solution is transmitted for 8 *clks* with spare structures S1 and S2, and for 7 *clks* with spare structure S3.

However, if any non-pivot fault remains uncovered (check = 0), and there is no any spare to use, then RA changes its state to TERM. with state TERM, RA ends the test of the current signal and tells other modules to move to the next signal.

If any non-pivot fault remains uncovered (*check* = 0), and spares are still available (*remain* = 1), the state is changed from IDLE to REAN. In this state, the addresses of uncovered non-pivot faults are sent to the CAM for the number of uncovered non-pivot fault *clks*. Subsequently, RA returns to the IDLE state and waits for the value updates in other modules. Once updates are complete, then RA reanalyzes the signal through the same process again.

Table 10. Input Signals of Redundant Analyzer Module

From	Signal Name	Bits	Description
Signal Generator	*DSSS*	8	Determine whether the spare at the i-th PCAM address is row or column spare.
	RLSS	3	Determine which row spare is global spare.
Singal Validity Checker	*signal_valid*	1	Check whether must-repair lines are covered and generated signal satisfies the number of spares assigned to each bank.
	unused_spare	8	Which spares are not in use?
	ra_start	1	Whether SVC is end?
Spare Allocation Analyzer	*pivot_en*	8	Whether pivot fault is stored?
	pivot_ren	8	Whether pivot fault is stored during reanalyze?
	pivot_row_addr	10	Pivot row address
	pivot_col_addr	10	Pivot column address
	pivot_bnk_addr	2	Pivot Bank address
	nonpivot_cover_info	10	Indicate uncovered non-pivot fault
	nonpivot_en	10	Whether non-pivot fault is stored?
	nonpivot_row_addr	10	Row address of non-pivot fault

	nonpivot_col_addr	10	Column address of non-pivot fault
	nonpivot_bnk_addr	2	Bank address of non-pivot fault
Top Module	*rst*	1	reset signal
	clk	1	clock signal (100MHz)
	spare_struct	2	Given spare structure. 1 = structure 1 2 = structure 2 3 = structure 3

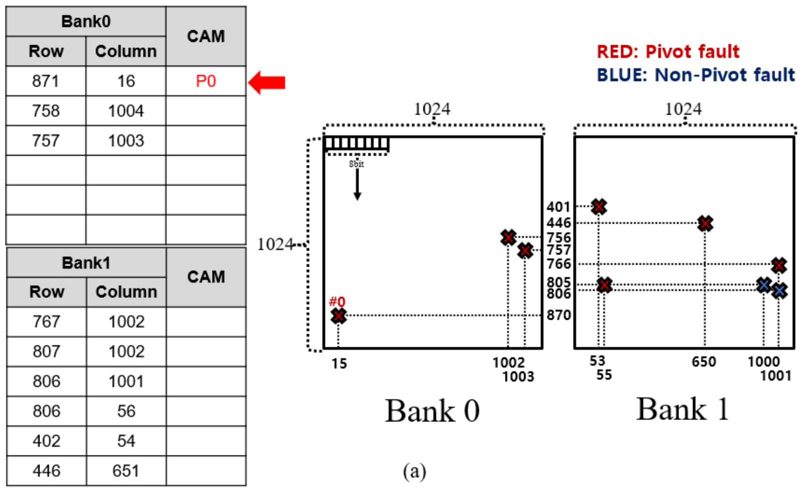
Table 11. Output Signals of Redundant Analyzer Module

To	Signal Name	Bits	Description
CAM	*uncovered_addr*	22	Bank, row, column address of uncovered non-pivot fault with
	termination	1	Be 1 when reanalyze is finished and test for next signal is required
	ratest_end	1	1 clk pulse for last uncovered address
	reanalyze	1	Be 1 when reanalyze for non-pivot fault is required
Signal Generator	*termination*	1	Be 1 when reanalyze is finished and test for next signal is required
Signal Validity Checker	*reanalyze*	1	Be 1 When reanalyze for non-pivot fault is required
Top Module	*solution*	16	Spare type, flag, bank, address of solution
	repair	1	Can be repair with the signal

3.6 Example

(1) CAM

It shows how faults are repaired through an example of the operation of Proposed BIRA's modules. First, it is about the process by which fault information enters and is stored from BIST to CAM. As shown in Fig.12, when faults come in from BIST, they are stored in each location of Bank 0 and Bank 1. Afterwards, starting from Bank 0, each column is divided into 8 bits and the rows are repeatedly scanned from 0 to 2023. When 8 bits arrive at row 2023, it moves to Bank 1 and scans the faults in the same way. Among them, the fault that is discovered first is entered into PCAM starting from 0, and if the row or column value is the same as the previously entered PCAM value, the fault is entered into NPCAM. Finally, in the example, a total of 6 PCAMs and 2 NPCAMs are input.

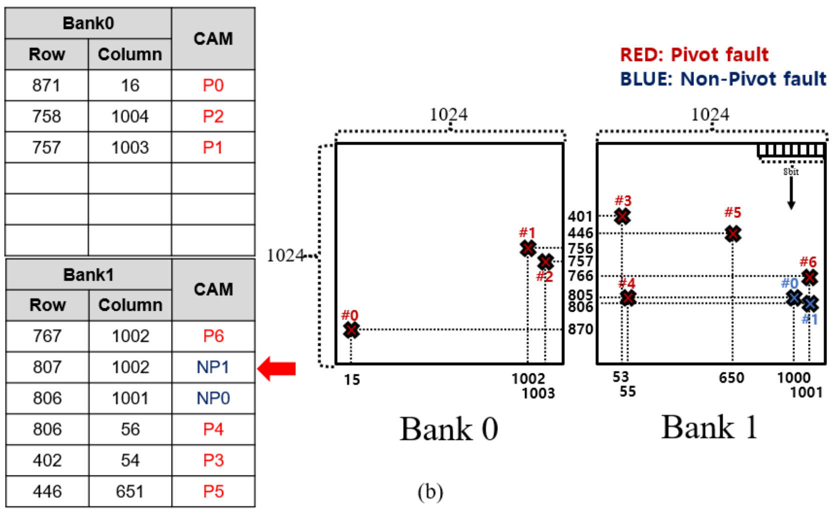


Bank0		CAM
Row	Column	
871	16	P0
758	1004	
757	1003	

Bank1		CAM
Row	Column	
767	1002	
807	1002	
806	1001	
806	56	
402	54	
446	651	

(a)

Figure 12.



Bank0		CAM
Row	Column	
871	16	P0
758	1004	P2
757	1003	P1

Bank1		CAM
Row	Column	
767	1002	P6
807	1002	NP1
806	1001	NP0
806	56	P4
402	54	P3
446	651	P5

(b)

Figure 13.

(2) Fault Repair

The fault repair process involves receiving DSSS signal and RLSS signal, and when the spares at each row and column cover all faults, those spares are output as a solution. First, CAM and SVC select cases of early termination through signal and fault information, and in other cases, DSSS continues to change to find a solution. This example is an example of a signal with DSSS of 00011100 and RLSS of 010. In DSSS, i-th 0 signal means that the i-th PCAM covers the column, and i-th signal 1 means it covers the row. RLSS exists only in spare structure 3 and indicates which position of the global spare is among the three-row spares. Therefore, in the example, P0, P1, P2, and P6 are covered by the column spare, and P3, P4, and P5 are covered by the row spare, and since the second-row in the RLSS is 1, it shows that P4, the second-row spare, is a global spare. If all PCAMs are covered in this way, it shows faults being covered as shown in Figure 15. At this time, if there is any remaining NPCAM, it means that the DSSS signal can't repair and the Reanalyze process is initiated using the remaining Nonpivot faults. When all faults for the DSSS signal are repaired, a solution is output using the spare information. In this example, all faults have been covered by DSSS and RLSS signal, so a solution is output.

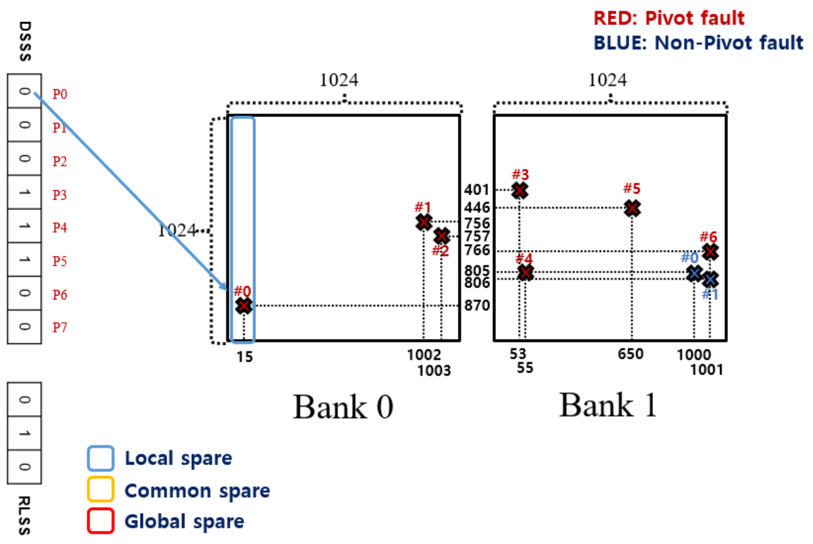


Figure 14.

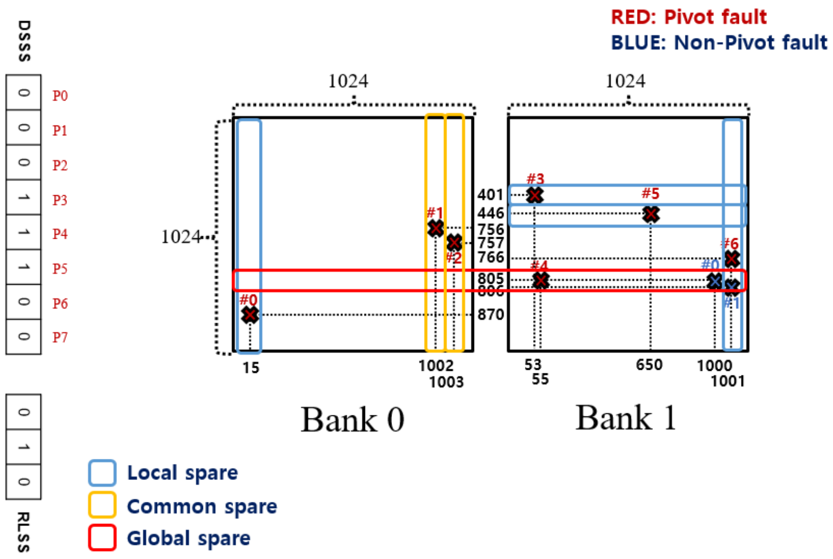


Figure 15.

4. Synthesis result

Since the project's required specifications were a clock frequency of 100 MHz, the synthesis process was conducted at 100MHz clock frequency.

4.1. Schematic

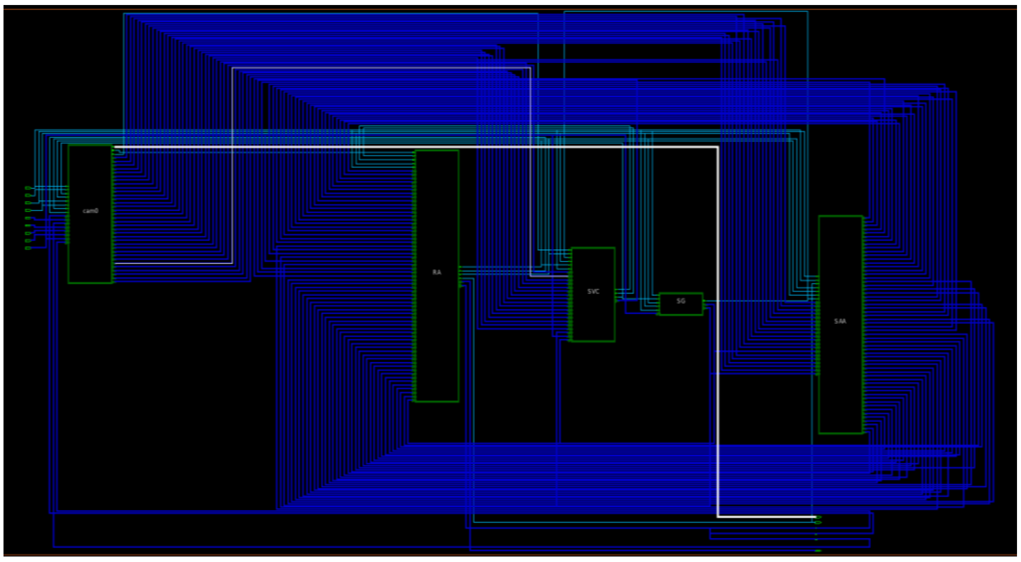


Figure 16. Schematic View of Proposed BIRA.

Fig.16 is a schematic view of the proposed BIRA. It consists of 5 blocks which are CAM, RA, SVC, SG, SAA. Each block has inputs and outputs.

4.2. Area & Timing

Table 12. Area and Timing for 100MHz Clock Frequency

Clock Frequency	Total Cell Area	Gate Count	Slack
100MHz	39912	15705	6.28

Table 1 is the area & timing information of the proposed BIRA at 100MHz clock frequency. The total cell area is 39912. To calculate the gate count that constitutes the proposed BIRA, it must be divided by the area of the 2-input NAND gate. Since the 2-input NAND gate area of the library used is 2.54144, the gate count of the the proposed BIRA is 15705. Slack is 6.28, so proposed BIRA will operate well in 100MHz clock frequency.

5. Result and Discussion

5.1 Normalized Repair rate

$$Normalized\ repair\ rate = \frac{\#\ of\ repaired\ memories}{\#\ of\ repairable\ memory}$$

The normalized Repair rate is as follows. Repairable memory is a memory that can be repaired, while repaired memory is a memory that has been repaired through testing. Using the repair rate without a normalized repair rate would consider cases where repair is originally impossible, so it would not give a meaningful repair rate. Therefore, in this project, we used a normalized repair rate to evaluate the performance of BIRA.

Table 13. Test Result of 1000 Fault Patterns

Spare Structure	# Repaired Memory	# Repairable Memory	# Test Case
S1	455	459	1000
S2	643	650	1000
S3	497	530	1000

Therefore, to confirm the normalized repair rate, we first conducted a test on 1,000 fault patterns. The following results were obtained. Based on the above results, the normalized repair rate was calculated as follows.

Table 14. Normalized Repair Rate of Each Spare Structures

Spare Structure	Normalized Repair Rate
S1	99.129%
S2	98.923%
S3	95.472%

As follows, S1 has the highest rate, followed by S2 and S3. Although the proposed BIRA was initially designed for the normalized repair rate to be 100%, it is unable to achieve this goal. The results are because the spare structure affects the type and number of available spares, which affects the operation of BIRA. And because the consideration of timing issues and spare condition was insufficient, the proposed BIRA cannot achieve 100% normalized repair.

5.2 Analysis Speed

The analysis time ($T_{solution} - T_{start}$) is the time it takes for the BIRA to find a solution after all of the BIST tests have finished. In the BIRA, the following three cases occur:

1) The *early_term* occurs when the number of faults that can be stored in the CAM is exceeded during the TEST in the BIST.
2) The repair is not possible because the BIRA cannot find a solution.
3) The repair is possible because the BIRA can find a solution.

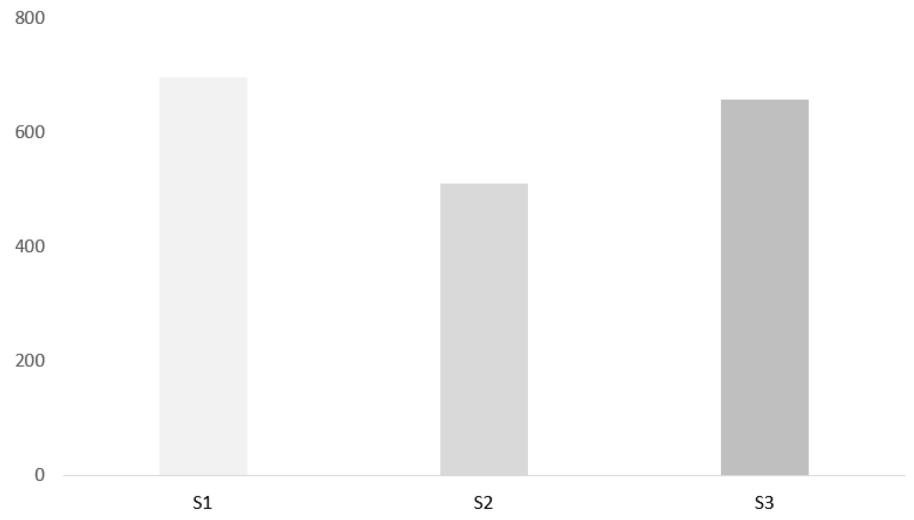


Figure 17. Average Analysis Time of Each Structures

Table 15. Average Analysis Time with Spare Structures

Spare Structure	Average Analysis Time (ns)
S1	696
S2	510
S3	658

The analysis time is in case 3) of the above. According to the analysis of analysis time, S2 found the solution the fastest. And S3 and S1 were the next, in that order.

5.3 H/W overhead

Table 16. H/W Overhead Ratio of Each Modules

Module	Total Cell Area	Gate Count	Ratio(%)
CAM	23108	9092	58.3
SG	510	201	1.29
SVC	1256	494	3.17

SAA	12163	4786	30.7
RA	2566	1010	6.5

Table 16 shows H/W overhead of each block. CAM and SAA account for a large portion of the area. CAM receive and store fault information, so many registers are used, and its portion is large. SAA is large because it must analyze 8 PCAM information and 10 NPCAM information in parallel.

5.4 Different Clock Frequency

In the timing report of 4.3, it was confirmed that there was a large margin for slack. It means the proposed BIRA will operate faster clock frequency.

Table 17. Area and timing for 250MHz Clock Frequency

Clock Frequency	Total Cell Area	Gate Count	Slack
250MHz	40047	15758	0.54

Table 2 is the synthesis result conducted at 100MHz and 250MHz clock frequency. In this case, slack is still bigger than 0. So proposed BIRA will operate 250MHz. If 250MHz is used instead of 100MHz, the proposed BIRA operating speed will be 2.5 times faster.

5.5 Improvement Points

The reason why the normalized repair rate was not achieved at 100% was because the timing issue was not well resolved. It was difficult to figure out what value to read when a signal transitions, and to resolve the issue when a signal is output 1 clock later than expected. To resolve these timing issues, trade-off such as speed and hardware overhead must be considered.

PPA is an abbreviation for Power, Performance, and Area, which represents three important aspects of hardware design. There are several ways to consider how to design hardware by optimizing these three aspects. First, there is the selection of an efficient algorithm. Designing using an efficient algorithm allows efficient design from all aspects of the PPA. In this graduation research, it was necessary to use an efficient algorithm in the RA stage. There may be a more efficient algorithm than the proposed RA method and finding it can be an important point. And code optimization will be necessary in the design of each module. This study was designed using the verilog language. Many conditional statements and variables are used within one module and optimizing them will be a key point in reducing the area. And by finding the critical path, you can give other modules as much slack as the corresponding clock. In this way, the area can be reduced. Although we do not measure power in this graduation research, when designing digital circuits, we must find ways to reduce power consumption, such as using low-power logic, pipelining, optimizing data paths, and designing low-power modes, etc.

6. Conclusion

As memory density increases, the BIRA structure for analyzing repair will become more complex. However, the Proposed BIRA can increase the level of analysis without the need to add a complex structure by attempting a reanalysis method. The Proposed BIRA will show a higher repair rate because it can directly check whether the location can be repaired with a spare by attempting analysis again rather than comparing the number of faults that could not be repaired with an unused spare after one analysis. In fact, except for a few fault types, high repair rates were shown, and it is expected that the normalized repair rate could also reach 100% if enough time exists. Since this Proposed BIRA is a BIRA that focuses on repair rate, it can be used usefully and efficiently in chip test technology for devices where stability is important.

Reference

[1] J. Kim, W. Lee, K. Cho and S. Kang, "Hardware-Efficient Built-In Redundancy Analysis for Memory With Various Spares," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 3, pp. 844-856, March 2017, doi: 10.1109/TVLSI.2016.2606499.

[2] W. Jeong, I. Kang, K. Jin and S. Kang, "A Fast Built-in Redundancy Analysis for Memories With Optimal Repair Rate Using a Line-Based Search Tree," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 17, no. 12, pp. 1665-1678, Dec. 2009, doi: 10.1109/TVLSI.2008.2005988.

[3] W. Jeong, J. Lee, T. Han, K. Lee and S. Kang, "An Advanced BIRA for Memories With an Optimal Repair Rate and Fast Analysis Speed by Using a Branch Analyzer," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 29, no. 12, pp. 2014-2026, Dec. 2010, doi: 10.1109/TCAD.2010.2062830.