ISSN 2349-6193
Impact Factor (PIF): 2.243

IJESMR
International Journal OF Engineering Sciences & Management Research

FAULT DIAGNOSIS OF SUBTHRESHOLD LEAKAGE CURRENT DEFECTS IN DRAM

K. Manju Priya*, S.Malini
*1SVS COLLEGE OF ENGINEERING.
2 SVS COLLEGE OF ENGINEERING.

Keywords: Functional Fault Models (FFM), Address decoder Fault (AF), Stuck At Fault (SAF), Transition Fault (TF), Coupling Fault (CRF),Complex Read Fault CRF), Automatic Repeat Query (ARQ), Forward Error Correction (FEC), In Line Test (ILT), Bose Chaudhuri Hocquenghem code (BCH), Syndrome Storing-based error Detection (SSD).

ABSTRACT

The minimum feature size of dynamic RAM has been down-scaled, so several studies have been carried out to determine ways to protect cell data from leakage current in many areas. Due to leakage sources, permanent error occurs. Many permanent faults are a result of manufacturing defects, which can be detected during manufacture testing. These errors can also occur at runtime. A self-contained adaptive system for detecting and bypassing permanent errors in on-chip interconnects is proposed. This system reroutes data on erroneous links to a set of spare wires without interrupting the data flow.

INTRODUCTION

Functional fault models (FFMs) is the deviation of the observed memory behavior from the functionally specified one, under a given sequence of performed memory operations [1]. Therefore, two basic ingredients are needed to define any FFM: (1) a sequence of performed memory operations, and (2) a list of corresponding deviations in the observed behavior from the expected one. The observed memory behavior that deviates from the expected one is called a faulty behavior or simply a fault, which can be denoted by fault primitive (FP). RAM faults are of two types. They are static faults and dynamic faults. Static fault models consist of static FPs, sensitized by at most a single memory operation. In other words, these are faults that describe an incorrect behavior of either the data stored in a cell, or a single memory operation performed on it. Dynamic faults are faults that are sensitized by performing two or more memory operations on the memory.

DRAM faults have two main causes. Improperly set voltages resulting in voltage dependent faults, and leakage currents resulting in time dependent faults. Faults caused by improper voltages stem from the inability of a memory operation in a defective memory to set the full voltage levels expected at different nodes of the memory, resulting in two different fault modes: 1. improper voltages present within the memory cell, and 2. Improper voltages on the nodes of the peripheral circuits . Improper voltages within the cell cause partial faults, while improper voltages in periphery cause dirty faults. Leakage currents, on the other hand, cause time dependent faults to take place, and depending on the direction of the leakage with respect to the performed operation, either soft faults or transient faults take place due to a supporting or an opposing leakage current, respectively.

The number of faults in on-chip links is expected to increase as technology scales further into the nano scale regime. While most faults are temporary, about 20% of all errors are caused by permanent or intermittent faults. Many permanent faults are a result of manufacturing defects, which can be detected during manufacture testing, however, these errors can also occur at runtime (e.g., from electromigration or aging). Error control coding (ECC) techniques are commonly used to address reliability issues in on-chip interconnect, but these techniques generally target transient errors rather than permanent errors. A single permanent fault can drastically reduce or even eliminate the correction capabilities of the commonly used codes. In order to maintain coding strength in the presence of permanent errors, spare wires can be used to replace permanently erroneous wires.

The introduction of spare wires requires the following: 1. Reconfiguration control and logic for bypassing erroneous wires and 2. A protocol for synchronizing information between receiver and transmitter. We present a system that uses spare wires to replace permanently erroneous wires without interrupting the data flow. To detect these permanent errors, we propose an in-line test (ILT) method to test each adjacent pair of wires in a link for opens, shorts and leakage. These tests can be run periodically to ensure that each link's ECC capability is not

IJESMR
International Journal OF Engineering Sciences & Management Research

being crippled by permanent errors. By testing every wire in the link, the ILT method also recovers resources from intermittent errors that were incorrectly flagged as permanent. In addition to the ILT method, we describe a syndrome storing-based error detection (SSD) method, which is based on evaluation of consecutive code syndromes at the receiver. Syndromes are calculated during the decoding procedure and contain information on errors in the received words.

LITERATURE REVIEW

A. INTRODUCTION

Conventional DRAM testing can be grouped into retention testing and functional testing. Retention testing is a test method that screens leakage-current defects by operating read and write functions containing a particular delay time. In functional testing, March elements that are a finite sequence of read or write operations applied to a cell in memory before proceeding to the next cell are conducted on each memory cell in order to detect the cell-to-cell bridge and coupling noise [2]. Due to the flexibility of these March elements, most built-in self test (BIST) architectures adopt functional test algorithms using the conventional March elements ; but with the downscaling of the device, conventional March testing cannot properly detect the leakage-current defects, and these defects have to be assessed using the time-consuming stress method. To test for leakage-current defects, several studies have attempted to implement retention testing using special techniques.

The word-line-pulsing technique has been proposed as a means of detecting weak cells by coupling nearby neighbor word lines. This technique results in an adjustable test stress based on setting the word-line enable time. This method can be used to detect sub threshold leakage-current defects; but when it is used in DRAM, it has to consider stress equality according to the cell location during the stress enable time. March complex read faults (CRF, which is also suggested in static RAM) detects faulty cells induced by the leakage current using the voltage gap between the bit line and the target cell with opposite data. However, according to our experimental results, March CRF has lower screen ability than the word line pulsing technique. *X*-direction-Extended March *C*- and *Y* -direction MATS are proposed to screen retention faults using self-refresh and time delay in eDRAM , but these studies mainly deal with detecting retention faults and analyzing the relationship between the leakage current and temperature[3]-[4].

B. MEMORY TESTING

The exponential increase in the integration density of memory components and the increase in the memory complexity, faulty behavior have made fault analysis and memory testing significantly important. Conventional DRAM testing can be grouped into retention testing and functional testing. Retention testing is a test method that screens leakage-current defects by operating read and write functions containing a particular delay time. In functional testing, March elements that are a finite sequence of read or write operations are applied to a cell in memory before proceeding to the next cell. It is conducted on each memory cell in order to detect the cell-to-cell bridge and coupling noise. A test algorithm is a finite sequence of test elements. A test element consists of number of memory operations.

C. TEST ALGORITHM

A test algorithm is defined by the test components. The various test algorithms are March tests, word line pulsing technique, large Vds Data Retention Test Pattern.

1. March tests

In order to verify whether a given memory cell is good, it is necessary to carry out a sequence of write and read operations to the cell. The number of read and write operations and the order of the operations depend on the target fault model. Most commonly used memory test algorithms are March tests, in which there are finite sequences of March elements. A March element is a finite sequence of read (**r**) or writes (**w**) operations applied to a cell in memory before processing the next cell. The address of the next cell can be in either ascending or descending address order. March tests depend on ‘n’ number of bits in the chip.

When a test algorithm reads a cell, the response will be either 0 or 1, and they are denoted as r_0 and r_1, respectively. Similarly, writing a 1 (0) into a cell is denoted as w_1 (w_0). But with the downscaling of the device, conventional March testing cannot properly detect the leakage-current defects, and these defects have to be assessed using the time-consuming stress method.

TABLE 1 COMPARISON OF MARCH TESTS

March algorithm	Test length	Fault coverage
MATS	4n	Some AFs, SAFs
MATS+	5n	AFs, SAFs
Marching1/0	14n	AFs, SAFs,TFs
MATS++	6n	AFs, SAFs,TFs
March X	6n	AFs, SAFs,TFs,some CFs
March Y	10n	AFs, SAFs,TFs,some CFs
March C-	15n	AFs, SAFs,TFs,some CFs
March A	8n	AFs, SAFs,TFs,some CFs
March B	17n	AFs, SAFs,TFs,some CFs

D. ERROR CORRECTION

Data that are transmitted over a link can be damaged, their bits can be masked or inverted by noise. Detecting and correcting these errors is important. Some simple codes can detect but not correct errors, others can detect and correct one or more errors [5]. Error control coding (ECC) techniques are commonly used to address reliability issues in on-chip interconnect, but these techniques generally target transient errors rather than permanent errors. Some methods of protecting against transient errors can also be used to protect against permanent and intermittent errors, but this can severely limit the code's ability to protect against transient errors.

1. **Parity checking**

One simple way to detect errors is:
1. Count the number of ones in the binary message.
2. Append one more bit, called the parity bit, to the message
3. Set the parity bit to either 0 or 1, so that the number of ones in the result is even. For example, if the original message contained 17 ones, the parity bit would be a one; if there had been 16 ones, the parity bit would be a zero.
4. Count the number of ones in the received message, including the parity bit. The result will always be even if no errors were encountered. (This approach also works if the parity bit is set to make the count come out odd, as long as the receiver checks for an odd count.)

This simple check does have two limitations: it only detects errors, without being able to correct them; and it can't detect errors that invert an even number of bits. In general, two recovery techniques are used when an error on a link has been detected.

2. Automatic Repeat Query (ARQ):

ARQ is also known as Automatic Repeat reQuest. In ARQ, Every block of data received is checked using the error detection code used, and if the check fails, retransmission of the data is requested which may be done repeatedly, until the data can be verified. Hybrid approaches combine the best properties of ARQ and FEC [9]. It uses acknowledgements (messages sent by the receiver indicating that it has correctly received a data frame or packet) and timeouts (specified periods of time allowed to elapse before an acknowledgment is to be received) to achieve reliable data transmission over an unreliable service. If the sender does not receive an acknowledgment before the timeout, it usually re-transmits the frame/packet until the sender receives an acknowledgment or exceeds a predefined number of re-transmissions.

Limitations:

The drawback of ARQ is the retransmission latency, as the number of retransmissions depends on error conditions; in persistent noise environments, a large number of retransmissions may result, making ARQ less energy efficient than FEC. The ARQ method fails in the presence of permanent errors. The retransmission is only useful for avoiding transient errors.

3. Forward Error Correction (FEC):

In FEC, the check bits that are transmitted together with the data are used to correct errors without the need for retransmission. Binary forward error correcting (FEC) block codes, means that the symbol alphabet consists of just two symbols (which we denote 0 and 1) and the receiver can correct a transmission error without asking the sender for more information or for a retransmission [6]. The transmissions consist of a sequence of fixed length blocks, called code words. This code is single error correcting (SEC), and a simple extension of it, also discovered by Hamming, is single error correcting and, simultaneously, double error detecting (SEC-DED).

3.1 Hamming code:

Hamming code is a very direct construction of a code that permits correcting single-bit errors. It assumes that the data to be transmitted consists of a certain number of information bits u, and also adds a number of check bits p such that if a block is received that has at most one bit in error, then p identifies the bit that is in error (which may be one of the check bits). Specifically, in Hamming code, p is interpreted as an integer which is 0 if no error occurred, and otherwise is the 1-origined index of the bit that is in error. Let k be the number of information bits, and m the number of check bits used. Because the m check bits must check themselves as well as the information bits, the value of p, interpreted as an integer, must range from 0 to m+k which is m+k+1 distinct cases values.

Because m bits can distinguish 2^m cases, we must have

$$2^m \geq m+k+1$$

This is known as the Hamming rule.

Limitations:

FEC codes can detect and correct permanent errors, but each permanent error will reduce the code's capability to tolerate transient or intermittent errors.

4. BCH:

BCH abbreviation stands for the discoverers, Bose, Chaudhuri and Hocquenghem. BCH codes are cyclic codes. These codes are multiple error correcting codes and a generalization of the Hamming codes. These are the possible BCH codes for $m \geq 3$ and $t < 2^{m-1}$.

Block Length: $n=2^m-1$

Parity Check Bits: $n-k \leq mt$

Minimum Distance :$d \geq 2t+1$

This code can correct t or fewer random errors over a span of $2m - 1$ bit positions.

The code is a t-error-correcting BCH code.

Limitations:

BCH codes can detect and correct multiple errors but have large power and area overhead costs, motivating the need for a different type of solution to handle permanent errors.

ADAPTIVE SYSTEM

The proposed adaptive link framework is shown in Fig.7. It consists of a transmitter, a link, and a receiver. The incoming k-bit-wide data word is encoded in the transmitter to a codeword of width n, which is transmitted through the link and decoded in the receiver. The decoder is responsible for correcting any errors and outputs the original bit data word [7]-[8]. A number of spare wires are available. Reconfiguration units at the transmitter and receiver determine which of the lines carry data and which are left idle.

The reconfiguration control units pass reconfiguration information between the receiver and transmitter and synchronize reconfiguration. The error detection *and* reconfiguration central control unit detects permanent errors and initiates reconfiguration. The inputs to this unit depend on which detection method is used. For the SSD method, the syndrome (synd) and error vector (err_vec) from the decoder are needed. For the ILT method, test outputs (test_out) from the spare wires under test are needed.

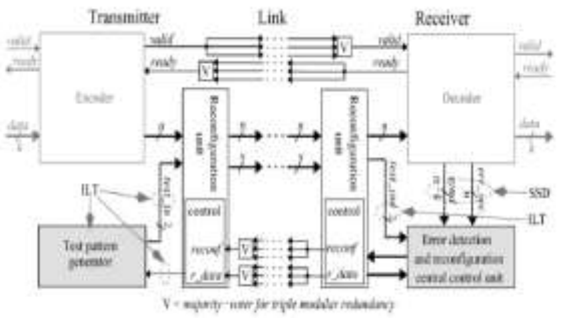

Transmitter
Link
Receiver
Encoder
Decoder
Reconfiguration unit
control
ILT
SSD
Test pattern generator
Error detection and reconfiguration central control unit
V = majority voter for triple modular redundancy

Fig.1 Reconfigurable link system.

The ILT method requires a test pattern generator (TPG) block and test inputs (test_in) to produce test signals. We apply our techniques to permanent and intermittent errors in the link. Two methods are presented here, namely, the proposed ILT method and the improved SSD method.

A. Encoder and Decoder

The encoder and decoder are responsible for implementing the tolerance against transient faults. The encoder calculates check bits that are transmitted together with the data word over the link, and these check bits are used in the decoder to detect and correct possible errors.

B. Reconfiguration Units

The function of the reconfiguration unit is to route the data around the erroneous wires or wires under test. To balance the delay within routed wires, the reconfiguration ripples through the bus, as shown in Fig.8, which presents the core capability of the reconfiguration unit. The number of spare wires to be inserted into the system depends on the probability of a permanent error in a wire, the number of wires, and the desired probability for correct operation of the link.

C. Reconfiguration Unit Control

The error detection circuits as part of the error detection and reconfiguration central control unit provide the location of the erroneous wire, which should then be bypassed. The control parts of the reconfiguration units at each end of the link are used to transmit the reconfiguration information from the receiver to the transmitter, as well as synchronize the reconfiguration so that both the transmitter and receiver do the reconfiguration in the same cycle. The control unit provides a separate control signal for each wire, reducing the amount of logic inserted into the critical path. The control signal provides the number of reconfigurations that have occurred at all indexes less than or equal to the wire address, so one signal can directly be used for controlling the wire routing [9]. A difference between the control values of a wire i and wire i-1 indicates that wire is erroneous. Fig.8 shows the usage of control signals, shown just below each control input. In this example, there are three spare wires.

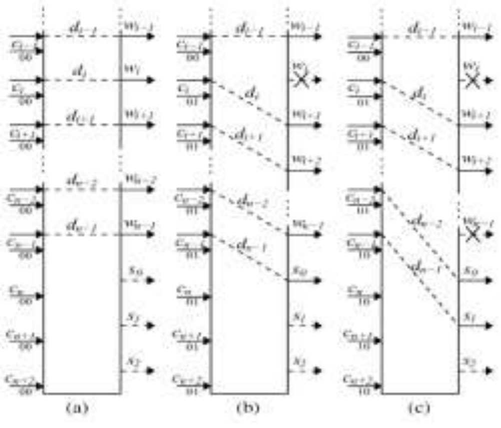


Fig.2 Reconfiguration unit. (a) No permanent errors. (b) Permanent error at location i. c) Permanent errors at locations i and n-1.

D. Transmission Protocol

To minimize the link area and energy, the transmission of control data is serial, using only one signal r_data, which is protected with TMR and spatial separation. Synchronization is done using the reconf signal (also protected by TMR), which enables the transmission of control data at the appropriate time. The transmission protocol can be divided into three phases: 1) the initialization of a transmission; 2) the transmission of the error location 1 b at a time (width [$\log_2(n+s+1)$], where the +1 in the equation is a result of reserving the all 1s location in the ILT system for the purpose of starting a test without doing any reconfiguration); and 3) the end of the transmission. The reconfiguration procedure follows immediately after the transmission.

E. ILT Method

The proposed ILT method sequentially routes data from each pair of adjacent wires to a set of available spare wires, allowing each pair to be tested for intermittent and permanent faults. This is achieved during normal operation, without interrupting data transmission, by making use of the reconfiguration system. To protect against runtime permanent errors, the ILT is run periodically, with a period that can be shortened to improve error resilience or increased for energy efficiency [10]. In addition to this periodic testing, the ILT can be triggered when an error is detected beyond the error correction capability of the code protecting the link. The trigger can use a simple timer or be adaptively controlled by an upper protocol layer (e.g., application) to save energy during idle periods.

1. ILT Procedure:

To begin the test, the ILT control unit reconfigures the link such that the first pair of wires is connected to the TPG (the data on those lines are rerouted to spare wires). The TPG issues a series of test patterns using the (test_in) signal. The ILT control unit compares the received (test_out) signal to a lookup table to determine if there is a permanent error in that pair of wires. The lookup table indicates which line(s) need(s) to be flagged as erroneous. Functional wires are reconfigured to carry data once again, and the process is repeated for each pair of wires (i.e., the test is shifted from wires 1 and 2 to wires 2 and 3, etc.). Note that, during each test, wires that were flagged as faulty are retested to prevent intermittent errors from wasting wire resources. To provide even greater protection against permanent errors, the system is designed to take into account the number of spare wires when running a round of tests. The ILT control unit determines the number of remaining spares by checking the status of the last wire in the link. If one spare remains, the upper layer system can be alerted to the lack of spare resources, and the system only reroutes one wire at a time to that remaining spare. Instead of the two-wire test, the system performs a single-wire test that can detect opens in the line but cannot detect a short between the wire under test and its neighbors. If a wire adjacent to the wire under test is disabled due to a previously detected error, the ILT unit will perform the two-wire test on that pair. If no spare wires remain, the system will periodically retest each disabled wire in an effort to recover from intermittent errors.

2. Analysis of Open and Shorted Wires:

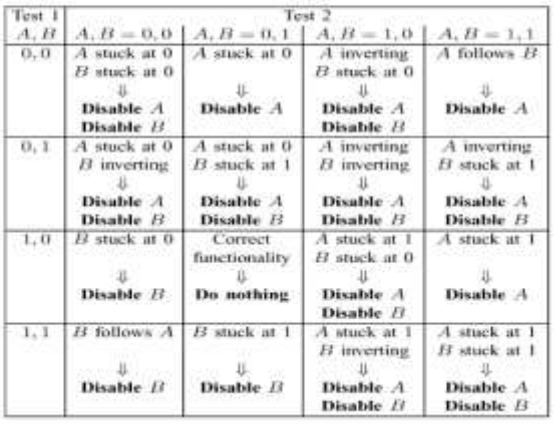
Here, we analyze the impact of open and shorted interconnect faults on circuit operation to determine the test patterns that will be used in the test procedure. Based on these patterns, a lookup table is created to determine if wires are erroneous using the (test_out) signal.

The potential for a short to only affect the output under certain conditions requires two test cases to evaluate whether a short exists between two wires. Each combination of contention currents must be examined (i.e., the case where A=1 and B=0, as well as the case where A=0 and B=1). These two tests are also capable of detecting an open in a line, as both values (0 and 1) of each line are examined. Each test has four possible output combinations, listed in Table 4 along with potential causes of that output response. The output from the two test cases can result in 16 possible combinations, listed in Table 4 along with potential causes of each response and the necessary action to take (shown in bold). Rows indicate results of Test 1 (A=1 and B=0), while columns indicate results of Test 2 (A=0 and B=1). For example, the box where the result of Test 1 is (1, 0) and the result of Test 2 is (0, 1) represents the condition where the outputs of A and B behave correctly, so no wires need to be disabled. There are two possible fault modes, including stuck-at faults, possibly from a break in the wire, or inverted response, which may be a result of delay errors or a combination of a broken wire and a short to another wire.

Table 3 Possible output responses for each of the two fault tests

Test 1: A = 1, B = 0		Test 2: A = 0, B = 1	
A, B = 0,0	- Open on line A - Wire shorted, B dominating	A, B = 0,0	- Open on line B - Wire shorted, A dominating
A, B = 0,1	- A & B open	A, B = 0,1	- Correct
A, B = 1,0	- Correct	A, B = 1,0	- A & B open
A, B = 1,1	- Open on line B - Wire shorted, A dominating	A, B = 1,1	- Open on line A - Wire shorted, B dominating

Table 4 Diagnosing and correcting open and shorted wires



Test 1	Test 2			
A, B	$A, B = 0, 0$	$A, B = 0, 1$	$A, B = 1, 0$	$A, B = 1, 1$
0, 0	A stuck at 0 B stuck at 0 $\Downarrow$ **Disable** A **Disable** B	A stuck at 0 $\Downarrow$ **Disable** A	A inverting B stuck at 0 $\Downarrow$ **Disable** A **Disable** B	A follows B $\Downarrow$ **Disable** A
0, 1	A stuck at 0 B inverting $\Downarrow$ **Disable** A **Disable** B	A stuck at 0 B stuck at 1 $\Downarrow$ **Disable** A **Disable** B	A inverting B inverting $\Downarrow$ **Disable** A **Disable** B	A inverting B stuck at 1 $\Downarrow$ **Disable** A **Disable** B
1, 0	B stuck at 0 $\Downarrow$ **Disable** B	Correct functionality $\Downarrow$ **Do nothing**	A stuck at 1 B stuck at 0 $\Downarrow$ **Disable** A **Disable** B	A stuck at 1 $\Downarrow$ **Disable** A
1, 1	B follows A $\Downarrow$ **Disable** B	B stuck at 1 $\Downarrow$ **Disable** B	A stuck at 1 B inverting $\Downarrow$ **Disable** A **Disable** B	A stuck at 1 B stuck at 1 $\Downarrow$ **Disable** A **Disable** B

F. SSD Method

Syndrome decoding is a common technique for decoding linear block codes. The syndrome is calculated by matrix multiplication s = uH^T, where u is a received code word vector of length n (u=c + e, where c is a transmitted code word and e is an error vector, both of length n), H^T is the transpose of the (n-k) x n parity-check matrix, and is a syndrome vector of length n-k. The syndrome gives the minimum weight error vector index, so the error vector can easily be determined. The correction is done by c = u+e, eliminating the error from the received data word. The basic idea behind SSD is that the error syndrome of an error control code contains information about the errors of a received code word. If the syndromes of a number of consecutive received code words are the same, then it can be concluded that there is a permanent error in the link. The error location can be extracted from the syndrome using the normal decoding procedure. The effectiveness of this approach comes from the fact that it takes advantage of the code and decoder already present at the system.

If there are more errors in the link than the error correction code is capable of correcting, the syndrome will be decoded incorrectly, providing a wrong error location. An important design decision for the SSD method is to determine how many syndromes to consider before deciding that an error is permanent. We refer to this number of cycles as the observation period t_{op}. In early methods, this period was set to three transmissions. We now present a more detailed analysis of the tradeoffs between t_{op} and reliability and provide guidelines for selecting t_{op}. If an intermittent error is misdiagnosed as a permanent error, a spare wire is consumed. In SSD, there is no method for recovering spare wires once they have been assigned, so the error observation period can result in wasted wire resources if set too short. On the other hand, too long an observation period may result in a large number of cycles before the error is detected or may even leave errors undetected. This is because the detection of stuck-at faults is data dependent; in order to be detected, the error must occur in all data words during the observation period. For example, a stuck-at-1 fault can only be detected if all the data bits passing through that wire over t_{op} cycles are 0. The upper limit for the number of cycles before the permanent error should be detected can be derived from the transient bit error rate (BER). Since a permanent error in a link may prohibit the detection and correction of a transient error, the number of cycles to detect a permanent fault should be much smaller than the mean time between transient errors.

When combining SSD with FEC codes, the error extraction circuitry already present in the FEC decoder can be used. Error extraction is the process of obtaining the error vector from the syndrome s. The error extraction circuit needs to be expanded to also extract check-bit errors. Certainly, if there is a permanent error in a wire used for a check bit, it should be corrected since it affects the overall error correction capability of the code. This is generally not implemented in FEC decoders since only the data bits are output. The structure of the SSD circuit is shown in Fig 9.

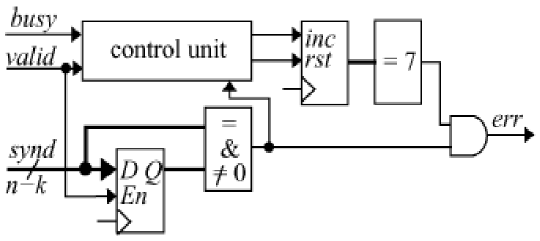


Fig.3 General Structure of the SSD unit.

It contains a register to store the syndrome, a comparator between the input and the last syndrome, and a counter for counting the number of identical but nonzero syndromes. The counter is reset during a reconfiguration procedure, which is achieved using the busy signal. By changing the width of the counter, t_{op} can be easily changed, in contrast to the existing method, where the observation period was set with a fixed number of parallel registers for storing syndromes. In this example implementation, an observation period of nine transmissions is used. The counter only counts to seven since only eight comparisons are necessary. The first comparison is between the first and the second data sample. An error to the output is signaled when the seven most recent comparisons have been equal and the eighth comparison is also equal, thus resulting in an observation period of nine. The valid signal connected to the enable input of the register is used to ensure that syndrome checking is done only for new values. The enable is implemented as a clock gating signal instead of using registers with enable inputs in order to reduce power consumption. In existing method, there was one fixed spare wire for each interleaving section, while in this method, there are s spare wires, and their usage is not restricted. A counter is used to count the number of spare wires used and shuts down when no more spares are available.

EXPERIMENTAL RESULTS

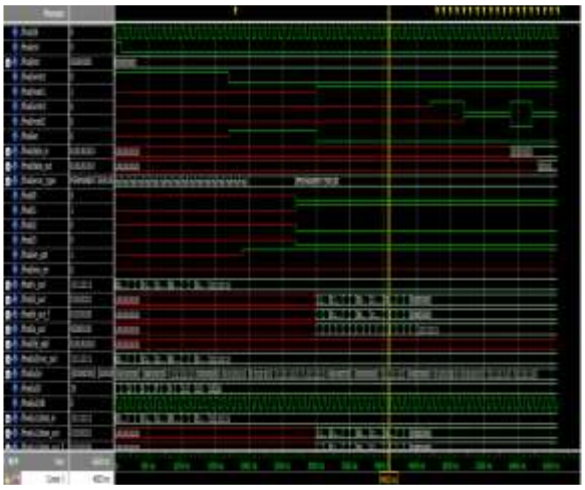


Fig.4 Detection and correction of permanent error

CONCLUSION

A complete reconfigurable system utilizing spare wires to replace erroneous wires and enabling reconfiguration without interfering with data transmission has been used. Two methods for error detection and correction have been evaluated, namely, interleaving algorithm, rotating ILT and improved SSD. The results show that our approach provides tolerance against a number of permanent errors equal to the number of spare wires in the system. Thus the protection against permanent errors can be achieved using spare wires than using complex coding schemes.

REFERENCES

[1]. Book on DRAM Fault Analysis and Test Generation (online).
[2]. M.C.-T. Chao, Y. Hao-Yu, H. Rei-Fu, L. Shih-Chin, and C. Ching-Yu, "Fault models for embedded-DRAM macros," in Proc. Design Autom. Conf., Jul. 2009, pp. 714–719.
[3]. AD J. VAN DE GOOR, "Using March Tests to Test SRAMs,"in proc Design and Test of Computers 1993.
[4]. C.-M. Chang, M. C.-T. Chao, H. Rei-Fu, and C. Ding-Yuan, "Testing methodology of embedded DRAMs," in Proc. Int. Test Conf., Oct. 2008, pp. 1–9.
[5]. M. Meterelliyoz, H. Mahmoodi, and K. Roy, "A leakage control system for thermal stability during burn-in test," in Proc. Int. Test Conf., Nov. 2005, pp. 991–1000.
[6]. Book on Hamming Error correction codes by Allen.D.Holliday.
[7]. D. Bertozzi, L. Benini, and G. De Micheli, "Error control schemes for on-chip communication links: The energy-reliability tradeoff," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 24, no. 6, pp. 818–831, Jun. 2005.
[8]. L. Li, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, "Adaptive error protection for energy efficiency," in Proc. ICCAD, San Jose, CA, Nov. 2003, pp. 27
[9]. S. Murali, T. Theocharides, N.Vijaykrishnan, M. J. Irwin, L. Benini, and G. De Micheli, "Analysis of error recovery schemes for networks on chips," IEEE Des. Test Comput., vol. 22, no. 5, pp. 434–442, Sep./ Oct. 2005.
[10]. Teijo Lehtonen, David Wolpert, Pasi Liljeberg, "Self-Adaptive System for Addressing Permanent Errors in On-Chip Interconnects," in IEEE transactions on Very Large Scale Integration (VLSI) systems, vol. 18, no. 4, April 2010.