IEICE TRANS. INF. & SYST., VOL.E107–D, NO.1 JANUARY 2024

PAPER

Testing and Delay-Monitoring for the High Reliability of Memory-Based Programmable Logic Device

Xihong ZHOU[†], ***Student Member***, **Senling WANG**[†a)], **Yoshinobu HIGAMI**[†], ***Members***, ***and*** **Hiroshi TAKAHASHI**[†], ***Senior Member***

SUMMARY Memory-based Programmable Logic Device (MPLD) is a new type of reconfigurable device constructed using a general SRAM array in a unique interconnect configuration. This research aims to propose approaches to guarantee the long-term reliability of MPLDs, including a test method to identify interconnect defects in the SRAM array during the production phase and a delay monitoring technique to detect aging-caused failures. The proposed test method configures pre-generated test configuration data into SRAMs to create fault propagation paths, applies an external walking-zero/one vector to excite faults, and identifies faults at the external output ports. The proposed delay monitoring method configures a novel ring oscillator logic design into MPLD to measure delay variations when the device is in practical use. The logic simulation results with fault injection confirm the effectiveness of the proposed methods.

key words: *reconfigurable device, reliability, interconnect defects, aging*

1. Introduction

Reconfigurable devices (e.g., FPGAs) allow users to customize their functions in-field, providing a flexible and scalable platform for system development with faster development cycle times, low design costs, and high performance. Benefiting from such capabilities, FPGAs have achieved great success in the development of IoT (Internet of Things) [1], SDV (self-driving vehicles) [2], and AI (artificial intelligence) [3].

Recently, a new type of reconfigurable device called MPLD (memory-based programmable logic device) [4] is under development for edge computing devices in IoT and AI applications. In contrast to FPGAs, which require large amounts of programmable interconnect resources to achieve programmability, MPLD is constructed only with an array of MLUTs (multiple look-up tables) without any extra programmable interconnect resources. An MLUT is the essential reconfigurable element constructed using general SRAMs and connects with its neighbors via Address-inputs/Data-outputs called AD interconnects. Users can configure wires and logic into MLUTs by writing the corresponding truth tables into the SRAMs. This feature enables high-density reconfigurable devices with low production costs, low power consumption, and minimal delay.

To guarantee the long-term reliability of MPLD, extensive production testing with high quality is first required to identify as many manufacturing defects as possible in the MLUT array. When the device is operating in the field, various hard-to-predict factors, such as aging phenomena [5], [6], and environmental factors including operating temperature, power supply, noise, etc., can cause delay degradation in the MLUT array of MPLD and threaten its long-term reliability [7].

In this paper, we focus on the issues that would affect the long-term reliability of MPLD. We propose a test method to address these reliability concerns to detect and identify interconnect defects in the MLUT array during the production phase. We also propose a delay monitoring technique to detect aging-caused failures in the field.

The proposed test method creates route maps in MPLD for fault propagation by configuring pre-designed test configuration data into the SRAM array, it then excites faults by applying an external walking-zero/one vector to the external input ports of MPLD and identifies any faults through fault effects propagated to the external output ports. The delay monitoring method configures a novel ring oscillator (RO) logic design into MPLD to measure aging-induced delays. We designed an MPLD with a 6x6 MLUT array to evaluate the proposed methods by performing logic simulations. The simulation results with fault injection confirmed the effectiveness of the proposed methods.

The main contributions of this paper are as follows.

1. *We explore the fault models of interconnect defects within the MLUT array of MPLD.*
2. *We propose approaches to test stuck-at faults and bridge faults caused by interconnect defects in the MLUT array during MPLD production. This contributes to high reliability and yield improvement.*
3. *We propose a test method to accurately identify the location of faults. The proposed test method improves the manufacturing process and enables the avoidance of faulty MLUT blocks, thereby ensuring high reliability when the MPLD is put into practical use.*
4. *We investigate the reliability issues induced by aging when the MPLD operates in the field and propose a monitoring technique to measure the aging-induced delay variations by configuring a novel ring oscillator logic design into MPLD.*

The remainder of the paper is organized as follows: Sect. 2 introduces the architecture of MPLD and its basic working principle. Section 3 discusses the reliability concerns in the

Manuscript received May 20, 2023.
Manuscript revised August 22, 2023.
Manuscript publicized October 3, 2023.

[†]The authors are with the Graduate School of Science and Engineering, Ehime University, Matsuyama-shi, 790–8577 Japan.
a) E-mail: wang@cs.ehime-u.ac.jp

DOI: 10.1587/transinf.2023EDP7101

lifecycle of MPLD. Section 4 proposes the production test solution for interconnect defects and shows the simulation results. Section 5 presents the delay monitoring method and shows the simulation results. Section 6 concludes this paper.

2. Preliminary

In this section, we briefly introduce the architecture of MPLD and its working principle.

2.1 MPLD Architecture

MPLD is constructed using an array of reconfigurable cells called MLUTs (multiple lookup tables). It can function in two modes: memory operation and logic operation. Each MLUT has *m-bit* logic address inputs $A_{m-1:0}$ and *m-bit* logic data outputs $D_{m-1:0}$, referred to as the *m-pair* AD interconnects, used for logic operation mode. Figure 1 shows the MPLD structure. The logic address inputs of the inner MLUTs are connected to the logic data outputs of their adjacent MLUTs. The logic address inputs and logic data outputs of the outermost MLUTs are connected to the IO (Input/Output) ports of the MPLD device. A memory address/data bus is used for the memory operation mode to configure each MLUT.

Figure 2 shows the structure of an MLUT under logic operation mode. It consists of two asynchronous SRAMs (SRAM1, SRAM2), two synchronous SRAMs (SRAM3, SRAM4), address transition detector (ATD) circuits, and logic output control circuits.

Each SRAM has $m/2$-*bit* address inputs and *m-bit* data outputs. The low-order $m/2$-*bit* address inputs $A_{m/2-1:0}$ are shared by SRAM1 and SRAM3, and the high-order $A_{m-1:m/2}$ are shared by SRAM2 and SRAM4, respectively. Asynchronous SRAMs are used to create combinational logic functions, and synchronous SRAMs are used to create sequential logic functions.

The ATD circuit is connected to the address inputs of the asynchronous SRAM to detect any value changes coming from the data outputs of its adjacent MLUTs at high speed for combinational logic operation. Figure 3 shows the ATD circuit structure. When no address change is detected, an address Change Detection unit (ACDU) generates an enable signal atd_{in_en} to enable the D-latches in the address Input Switching unit to switch the address inputs of the MLUT (e.g., A_0 to A_3) and performs signal change detection for them. Furthermore, if the ACDU detects an address change, a pulse signal is generated on atd_{pulse} to trigger D-FFs in the address Output Switching unit to switch the address inputs $[a1_3{:}a1_0]$ of the SRAM. Meanwhile, atd_{ce_n} and atd_{clk} signals will be output to drive the SRAM for read operation.

The logic output control circuit controls the logic data outputs of MLUT by using an *m*-bit *output control register* (OCR), OR gates, and XOR gates. Figure 4 shows its functional operation. The C_k of the OCR controls the logic data output of the MLUT D_k, as an input to an XOR gate to enable this XOR logic. The other input of the XOR gate is the output of an OR gate. The inputs of the OR gate are

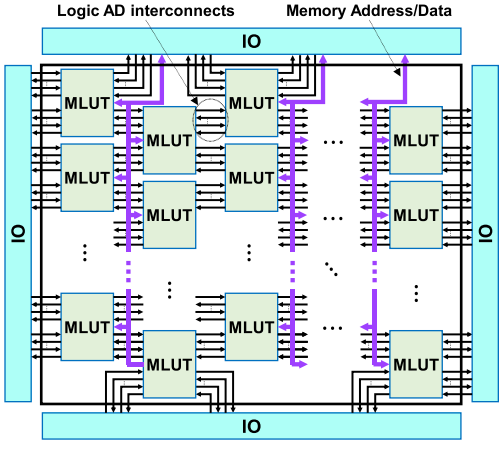

Fig. 1 MPLD Structure: MLUT Array.

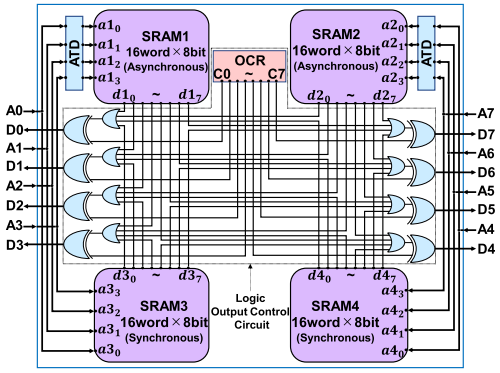

Fig. 2 MLUT Structure in logic operation mode.

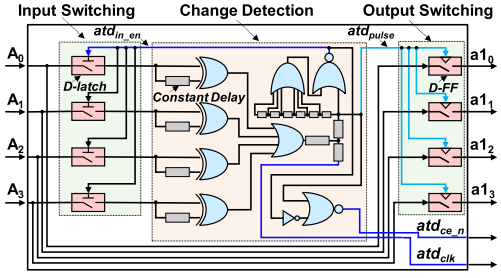

Fig. 3 ATD circuit.

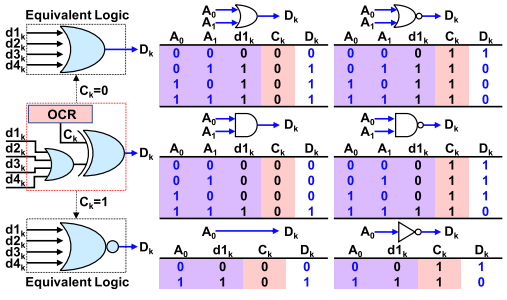

Fig. 4 Functional operation of logic output control circuit.

the k-th data outputs of all SRAMs ($d1_k$, $d2_k$, $d3_k$, $d4_k$). In essence, this output control circuit performs the following logic function:

$$D_k = C_k \oplus (d1_k \vee d2_k \vee d3_k \vee d4_k).$$

When $C_k = 0$, this output control circuit is equivalent to an OR logic for k-th data outputs of all SRAMs:

$$D_k = d1_k \vee d2_k \vee d3_k \vee d4_k.$$

When $C_k = 1$, this output control circuit is equivalent to a NOR logic for k-th data outputs of all SRAMs:

$$D_k = \overline{d1_k \vee d2_k \vee d3_k \vee d4_k}.$$

By using the output control circuit, i.e., OCR, OR, and XOR, various types of output logic functions can be implemented in the MLUT for its logic inputs, such as common logics like OR, NOR, AND, NAND, Inverter, and wiring. The most important role of the OCR is to control the output logic function of the MLUTs in the logic mode without changing the contents of the truth table back to the memory mode.

In this architecture, each MLUT can work in either memory or logic operation mode. In memory mode, users can access (read/write) data as a regular memory block. When the MLUT is used for reconfigurable computing, it is first necessary to put the MPLD in memory mode to write the truth table of the logic function into the corresponding SRAMs, and then switch the device to logic operation mode.

2.2 Working Principle of MPLD

Figure 5 shows an example to configure logic functions and wires in an MLUT. Here, we use two asynchronous SRAMs to configure an AND gate and an OR gate into SRAM1, an XOR gate, a wire, and an INVERTER into SRAM2. We choose the address A_0 and A_1 of MLUT as the AND gate's inputs, A_2 and A_3 as the OR gate's inputs, A_4 and A_5 as the XOR gate's inputs, and A_7 as the INVERTER's input, the data output D_0, D_1, D_2, D_4, D_5, and D_6 as the output of the AND gate, OR gate, wire $A_3 \rightarrow D_2$, XOR gate, wire $A_6 \rightarrow D_5$, and INVERTER, respectively. We represent the AND, OR logic, and the wire $A_3 \rightarrow A_2$ in truth table1, the XOR logic, wire $A_6 \rightarrow D_5$, and the INVERTER in truth table2. In memory operation mode, we write truth table1 and truth table2 into SRAM1 and SRAM2, respectively. Since the data outputs of SRAMs are connected to each other (by OR gate) and controlled by the OCR, we need to set the values of the remaining data outputs of SRAMs to all-zero and the OCR to all-zero. In logic operation mode, the MLUT will execute the configured logic and wires as a combinational logic block.

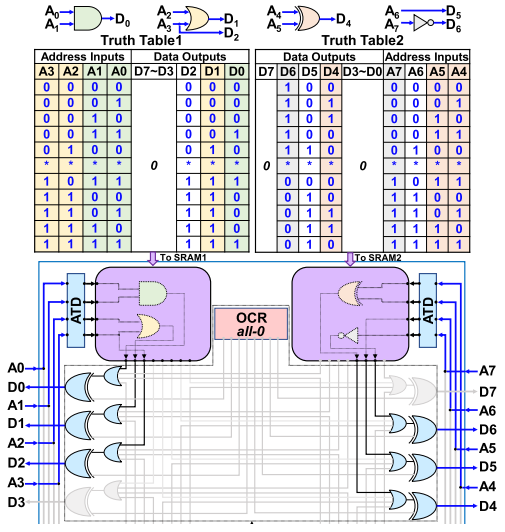

Fig. 5 Logic configuration in a single MLUT.

3. Reliability Issue in MPLD

In this section, we introduce the factors that affect the long-term reliability of MPLD in the production phase and in the field, respectively.

3.1 Manufacturing-Defects-Caused Reliability Issue

In the production of MPLD, the manufacturing defects existing at AD interconnects between MLUTs, including short, bridge, or open, would cause significant yield loss and reliability degradation. As described in Sect. 2, the address inputs of a target MLUT come from the data outputs of its adjacent MLUTs. A defect at the AD interconnect would change the value of the address inputs of MLUTs, causing access errors and resulting in logic faults in the configured circuit.

Figure 6 shows the example of an AD interconnect-defect-caused stuck-at-1 fault, which assumes that an OR-logic is configured in an MLUT ($D_5 = A_1 \vee A_0$) through the truth table. For a defect-free device, when the all-zero is applied to the address inputs of the MLUT, the corresponding contents stored in the SRAM will be readout, and the OR gate will output 0 (D_5: 0). If there is a stuck-at-1 fault at A_0, the value A_0 will be forced to 1, and the normal address of all-zero will be changed to 00000001, which causes an access error where the content of 0010000 is read out, thus the output of the OR gate D_5 will output 1.

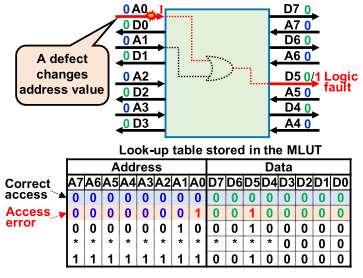

Fig. 6 Interconnect defect causes a stuck-at-1 fault.

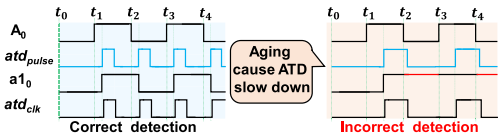

Fig. 7 Aging caused ATD detection error.

3.2 Aging Issues in MPLD

When the MPLD works in the field for a long term or under a severe environment, various aging phenomena such as HCI (hot carrier injection) and BTI (bias temperature instability) [5], [6] would threaten the long-term reliability [7] of the MPLD. As described in Sect. 2, in each MLUT, the asynchronous SRAMs use the ATD circuit to detect input address changes to execute asynchronous operations. The ATD is extremely sensitive to delay variation. Aging phenomena (HCI, BTI) would increase the threshold voltage of the transistors in ATD and slow down the switching speed [8], which might cause false detection of address changes.

As demonstrated in Fig. 3, the ATD circuit will generate an atd_{pulse} signal once detected any value changes in the address inputs ($A_{0:3}$) of MLUT and switches $A_{0:3}$ to the address inputs ($a1_{0:3}$) of the asynchronous SRAM1. Suppose that 01010 is applied to A_0 at time $t_0t_1t_2t_3t_4$, respectively, as shown in Fig. 7. When a transition occurs at A_0, the ATD must detect the value change and transfer the transition to $a1_0$ in a very short delay. Aging-induced delay at the ATD logic would generate an anomalous atd_{pulse} signal to switch the A_0 to $a1_0$, which causes false detections for the A_0 at t_2, t_4 and result in $a1_0$ being 01111 at $t_0t_1t_2t_3t_4$.

4. Interconnect Defect Test: Detection and Localization

4.1 Fault Models in MPLD

A short interconnect defect between the ground (supply) and AD interconnect is modeled as a stuck-at fault, that fixes the address input value of MLUT at logic 0 (1). Figure 8 (a) shows the behavior of a stuck-at fault at $M_1D_5 \rightarrow M_2A_5$, which would fix the M_2A_5 to 1 (0).

A short interconnect defect between the AD interconnects is modeled as a bridge fault. Depending on the employed logic circuitry, a bridge fault may result in a wired-OR (OR-bridge) or wired-AND (AND-bridge) logic function. Figure 8 (b) shows the behavior of the bridge faults. Suppose that a bridge occurs between $M_1D_5 \rightarrow M_2A_5$, $M_1D_4 \rightarrow M_2A_4$, an AND-bridge would cause a faulty value 0 at M_2A_5 (M_2A_4) when M_1 outputs logic 1 (0), 0 (1) at M_1D_5, M_1D_4, respectively. In contrast, an OR-bridge would cause a faulty value 1 at M_2A_5 (M_2A_4) when M_1 outputs logic 0 (1), 1 (0) at M_1D_5 and M_1D_4.

4.2 Test Method

In [9], [10], we have proposed the test method to detect

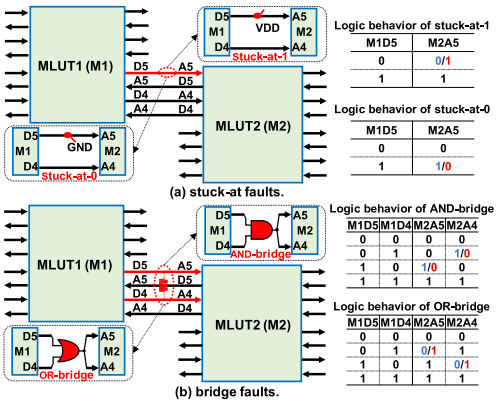

Fig. 8 Fault models.

stuck-at and bridge faults in the MLUT array in fewer test configurations with high fault coverage. Identifying the fault location also plays an essential role in improving the manufacturing process and helps the user to void the use of a faulty MLUT block when the MPLD is put into practical use. Therefore, in this paper, we extend our proposed test methods in [9], [10] to identify the locations of interconnect defects. Here, we define the "Test" of MPLD referred to as the fault detection and fault location to avoid confusion.

In [11], the authors presented a universal fault diagnosis technique to locate faults in the LUT array of an FPGA. The method can locate all faulty points of the LUT array in two steps: horizontal and vertical diagnosis. For MPLD constructed by an MLUT array, the basic idea in [11] would also be applicable. However, since the interconnects between MLUTs are unconfigurable unlike the FPGA, we carefully considered how to implement horizontal and vertical diagnosis in MPLD.

In this paper, we explored a new test method to detect and locate faults caused by AD interconnect defects, including stuck-at fault and bridge fault, based on the idea in [11] and the test method in [9]. The basic idea to detect and locate the faults is as follows.

1) Configuring the truth tables for tests, into the MLUTs to create the fault propagation path, called ***test configurations***.

2) Applying the pre-generated ***external test patterns*** to input ports of MPLD to excite the target faults.

3) Observing the fault effects at the output ports of MPLD for fault detection.

4) Identifying the fault location through the position of the external output ports where faulty effects were observed.

It is worth noting that the proposed scheme must assume that the other faults except the faults on the interconnects of MLUTs do not occur.

The core of the proposed method is to create proper paths on the MLUT array, which can propagate the fault effect to the expected output ports for fault localization.

For an MLUT with *m-bit* address inputs and data outputs, there are ${}_mP_m = m!$ kinds of wiring patterns to connect the address inputs with the data outputs, which we call

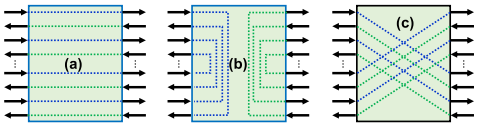

Fig. 9 Route maps; (a) horizontal, (b) vertical (c) diagonal.

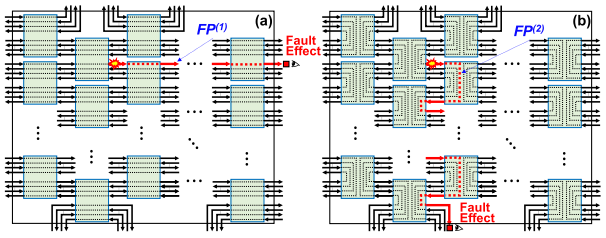

Fig. 10 Test mechanisms under route maps.

route maps. Figure 9 shows three route maps to create data paths between the address inputs and data outputs of MLUT in horizontal, vertical, and diagonal directions, respectively. Figure 10 (a) and Fig. 10 (b) represent the tests under the horizontal route map and vertical route map, respectively. For stuck-at faults at any AD interconnect of the MLUTs array, configuring either the horizontal route map or the vertical route map into the MLUTs can detect all faults and identify the location of the faults by configuring the two maps in order. For bridge faults, an additional diagonal route map is considered necessary; For multiple faults, more route maps are needed; this is the difference with [11].

The procedure of the proposed test method based on the route map is as follows.

Testing Procedure

Definitions:

- N_{rm}: *number of route maps.*
- rm_i: *route map i;* $i \in [1, N_{rm}]$.
- $TC^{(i)}$: *test configurations creating* rm_i.
- $N_{FE}{}^{(i)}$: *number of observed fault effects under* rm_i.
- $FP_k{}^{(i)}$: *fault propagation path k obtained under* rm_i; $k \in [1, N_{FE}{}^{(i)}]$.
- $FP^{(i)}$: *fault propagation path set under* rm_i.
- F_{loc}: *fault location.*

Process:

(1) Test under rm_i *for* $i \in [1, N_{rm}]$:

(a) Configure $TC^{(i)}$ *into each MLUT to create* rm_i.

(b) Apply external test patterns to the input ports of MPLD.

(c) Observe fault effects. If $N_{FE}{}^{(1)} == 0$, *end testing (fault-free).*

(d) Obtain the fault propagation path set: $FP^{(i)} = \bigcup_{k=1}^{N_{FE}{}^{(i)}} FP_k^{(i)}$.

(2) Identify fault location: $F_{loc} = \bigcap_{i=1}^{N_{rm}} FP^{(i)}$.

4.3 Test Configurations and External Test Patterns

For an MLUT with m pairs of AD interconnects ($A_{m-1:0}$ and $D_{m-1:0}$) and constructed by four $2^{m/2}word \times m$-bit size SRAMs, Table 1 shows the test configurations for creating the route map rm_1, rm_2, and rm_3. Each test configuration (TC) consists of two truth tables: truth table1 and truth table2; they are respectively written into two asynchronous SRAMs (or two synchronous SRAMs) for creating the test route between the address inputs $A_{m-1:0}$ and the data outputs $D_{m-1:0}$ of the MLUT.

Table 1 Test configuration in an MLUT with m-pair AD interconnects

Route Maps	Test Configurations		
rm_1: horizontal route map	$TC^{(1)}$	truth table1	$D_{m-1:m/2} = A_{0:m/2-1}$ $D_{m/2-1:0} = all\text{-}0$
		truth table2	$D_{m-1:m/2} = all\text{-}0$ $D_{m/2-1:0} = A_{m/2:m-1}$
rm_2: vertical route map	$TC^{(2)}$	truth table1	$D_{m-1:m/2} = all\text{-}0$ $D_{m/2-1:0} = A_{0:m/2-1}$
		truth table2	$D_{m-1:m/2} = A_{m/2:m-1}$ $D_{m/2-1:0} = all\text{-}0$
rm_3: diagonal route map	$TC^{(3)}$	truth table1	$D_{m-1:m/2} = A_{m/4:m/2-1} : A_{0:m/4-1}$ $D_{m/2-1:0} = all\text{-}0$
		truth table2	$D_{m-1:m/2} = all\text{-}0$ $D_{m/2-1:0} = A_{3m/4:m-1} : A_{m/2:3m/4-1}$

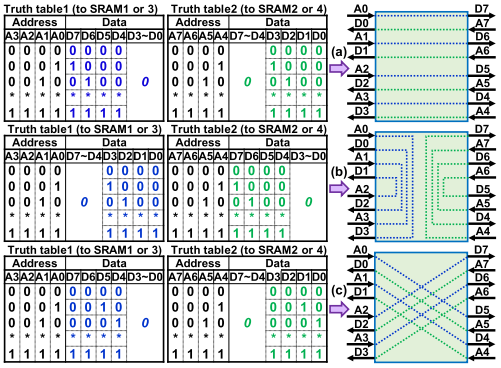

Fig. 11 Test configuration in the MLUT; (a) $TC^{(1)}$, (b) $TC^{(2)}$, (c) $TC^{(3)}$.

Table 2 External test patterns applied to external inputs of MPLD

Fault Types	External Test Patterns
stuck-at-1	*all-zero vector: [0...0]*
stuck-at-0	*all-one vector: [1...1]*
AND-bridge	*walking-zero vector: [1...1$\vec{0}$1...1]*
OR-bridge	*walking-one vector: [0...0$\vec{1}$0...0]*

Figure 11 shows an example of configuring TCs into SRAMs of an MLUT with 8-pair AD interconnects. The truth table1 and the truth table2 are stored in the SRAMs sharing the address with the low-order address ($A_{3:0}$) and the high-order address ($A_{7:4}$) of MLUT, respectively.

Upon writing the TCs into their respective MLUTs during memory mode, the MPLD is transitioned into test mode, followed by the configuration of the route map. For stuck-at faults, applying an *all-zero* (*all-one*) *vector* (a binary vector in which all elements are set to zero/one) to the external input ports of MPLD will activate the stuck-at-1 (stuck-at-0) faults, leading to detection. For bridge faults, applying a *walking-zero/one vector* (a sequence of binary values where a single zero/one "walks" through a series of ones/zeros) to the external input ports of MPLD makes the AND-bridge/OR-bridge faults active and propagates the faulty effects to the external output ports. Table 2 presents the external test patterns for exciting the stuck-at and bridge faults.

Figures 12 and 13 show the example of applying the *all-zero vector* and the *walking-zero vector* to excite a stuck-at-1 (*sa1*) and AND-bridge (*ANDbd*) fault in an MPLD with

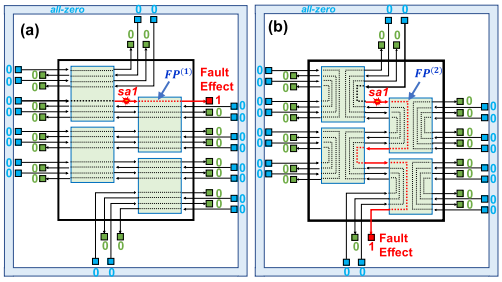

Fig. 12 Apply *all-zero* to excite stuck-at-1 fault.

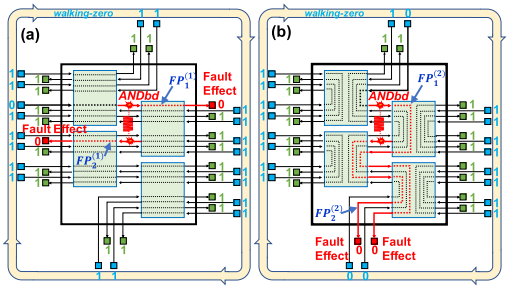

Fig. 13 Apply *walking-zero* to excite AND-bridge fault.

2×2 MLUTs array, respectively. Figures (a) and (b) show the test under rm_1 and rm_2. The rm_1 and rm_2 are respectively created by $TC^{(1)}$ and $TC^{(2)}$ stored in SRAMs to propagate the fault effect along with the horizontal and vertical routes. When applying the *all-zero* (*walking-zero*) *vector* to the external inputs, the *sa1* (*ANDbd*) fault is excited, and the fault effect(s) 1 (0) will be propagated to the external output ports for observation. As shown in the figures, the route that reaches the external output ports is individual, we can thus obtain the fault propagation path set on both the rm_1 and rm_2: $FP^{(1)}$ and $FP^{(2)}$, by observing the fault effects mapped on the external output ports. We can locate the fault by calculate $F_{loc} = FP^{(1)} \cap FP^{(2)}$ (for the *ANDbd*, $FP^{(1)} = \bigcup_{k=1}^{2} FP_k^{(1)}$, $FP^{(2)} = \bigcup_{k=1}^{2} FP_k^{(2)}$).

4.4 Simulation Results

To validate the proposed test method, we conducted logic simulations using ModelSim by injecting fault nodes into the netlist of the MPLD.

The MPLD has been designed with a 6×6 array of 36 MLUTs, as depicted in Fig. 14. The external IO ports consist of 48-bit left/right IOs: *li[47:0]*, *lo[47:0]*, *ri[47:0]*, *ro[47:0]* and 20-bit top/bottom IOs: *ti[19:0]*, *to[19:0]*, *bi[19:0]*, *bo[19:0]*. Each MLUT features 16-pair AD interconnects ($A_{15:0}$, $D_{15:0}$), and comprises four *256word* × 16-bit SRAMs including two asynchronous SRAMs (SRAM1, SRAM2) and two synchronous SRAMs (SRAM3, SRAM4). The simulation was executed using the following procedure.

1: Inject a stuck-at-0 (*sa0*) fault at $\boldsymbol{x_2 y_1 A_2}$ (address line A_2 of MLUT x_2y_1) and an OR-bridge (*ORbd*) fault between $\boldsymbol{x_4 y_2 A_1}$ and $\boldsymbol{x_4 y_4 A_3}$.

2: Configure the test configurations into SRAMs of each MLUT to create the route maps.

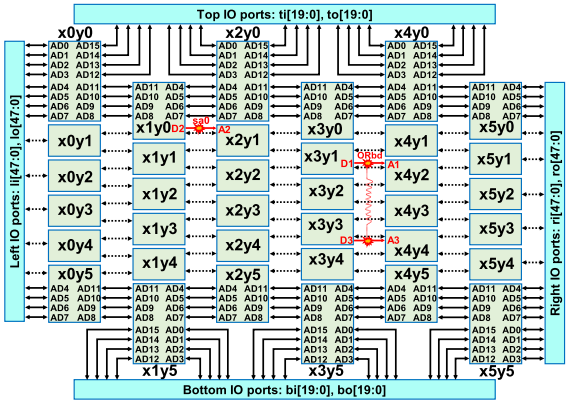

Fig. 14 MPLD with 6×6 MLUTs array.

3: Apply external test patterns *all-one* (*walking-one*) *vector* to external input ports (*li*, *ri*, *ti*, *bi*) to excite injected *sa0* (*ORbd*) fault.

4: Identify the fault location by observing fault effects at the external output ports (*lo*, *ro*, *to*, *bo*).

The simulation results of testing the *sa0* fault under the rm_1 and rm_2 are shown in Figs. 15 and 16. Prior to enabling the injected fault node *sa_fltinj_en* (= 0), the MPLD is fault-free and all output ports exhibit a value of 1. Upon enabling *sa_fltinj_en* (= 1), $\boldsymbol{x_2 y_1 A_2}$ is fixed at 0 and the fault effect value of 0 is propagated along the horizontal (vertical) route to the port *ro[6]* (*bo[14]*) (= 0). The *sa0* propagation path sets on the rm_1 and rm_2, $FP^{(1)}$ and $FP^{(2)}$, can be determined by:

$$\begin{aligned} FP^{(1)} &= \{li[10] \rightarrow x_1y_0A_{13} \rightarrow \boldsymbol{x_2y_1A_2} \rightarrow x_3y_0A_{13} \\ &\quad \rightarrow x_4y_1A_2 \rightarrow x_5y_0A_{13} \rightarrow ro[6]\}, \\ FP^{(2)} &= \{ti[14] \rightarrow x_1y_0A_5 \rightarrow \boldsymbol{x_2y_1A_2} \rightarrow x_1y_1A_5 \\ &\quad \rightarrow x_2y_2A_2 \rightarrow x_1y_2A_5 \rightarrow x_2y_3A_2 \rightarrow x_1y_3A_5 \\ &\quad \rightarrow x_2y_4A_2 \rightarrow x_1y_4A_5 \rightarrow x_2y_5A_2 \rightarrow x_1y_5A_5 \\ &\quad \rightarrow bo[14]\}. \end{aligned}$$

Then, the location of *sa0* fault can be identified by:

$$F_{loc} = \bigcap_{i=1}^{2} FP^{(i)} = FP^{(1)} \cap FP^{(2)} = \boldsymbol{x_2 y_1 A_2}.$$

The simulation results of testing the *ORbd* fault under the rm_1 and rm_2 are shown in Figs. 17 and 18. A bridge fault is injected into the MLUT array by setting *bd_fltinj_en* to 1, where the address inputs $\boldsymbol{A_1}$ of MLUT $\boldsymbol{x_4 y_2}$ and $\boldsymbol{A_3}$ of $\boldsymbol{x_4 y_4}$ are bridged represented in an OR logic function:

$$x_4y_4A_3^{(0\rightarrow 1)} = x_4y_2A_1^{(=1)} = x_4y_2A_1^{(=1)} \vee x_4y_4A_3^{(=0)}$$

The fault effect values 1 are propagated along the horizontal (vertical) route to the port *ro[13]* and *ro[31]* (*bo[5] and bo[7]*), respectively. The *ORbd* propagation paths can be obtained by:

$$FP_1^{(1)} = \{li[17] \rightarrow x_1y_1A_{14} \rightarrow x_2y_2A_1 \rightarrow x_3y_1A_{14}$$

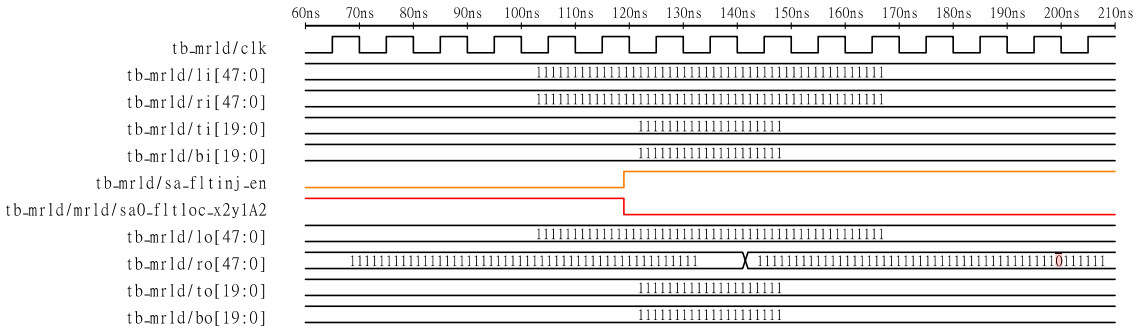

Fig. 15 Simulation result of the test under rm_1 for *sa0*.

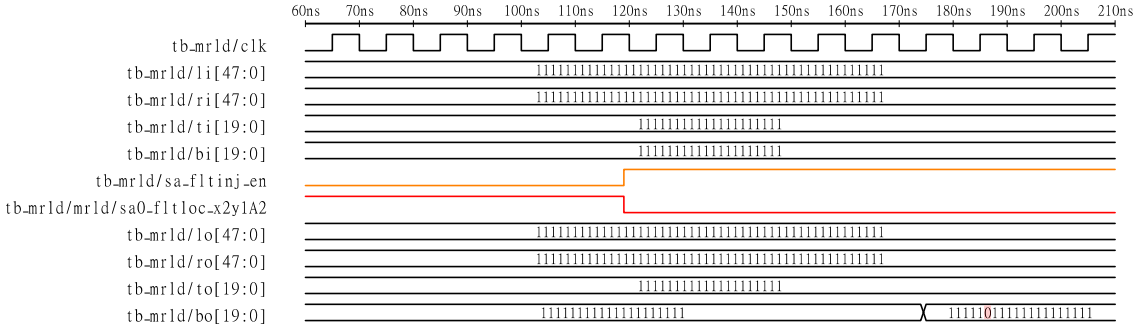

Fig. 16 Simulation result of the test under rm_2 for *sa0*.

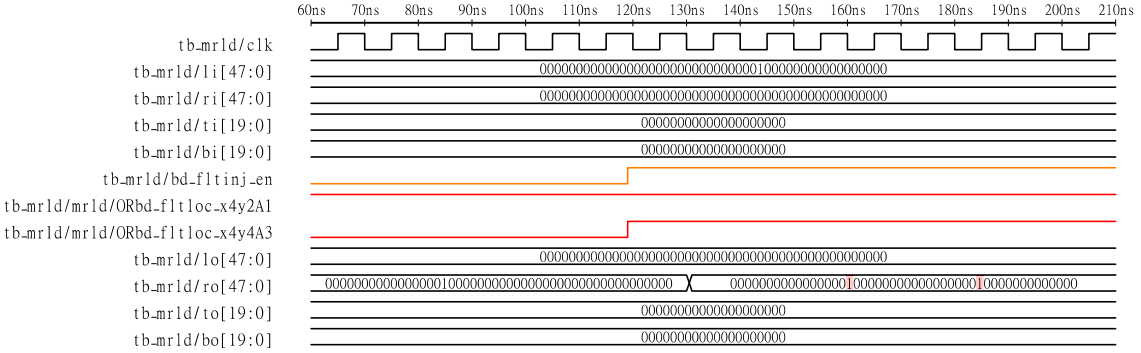

Fig. 17 Simulation result of the test under rm_1 for *ORbd*.

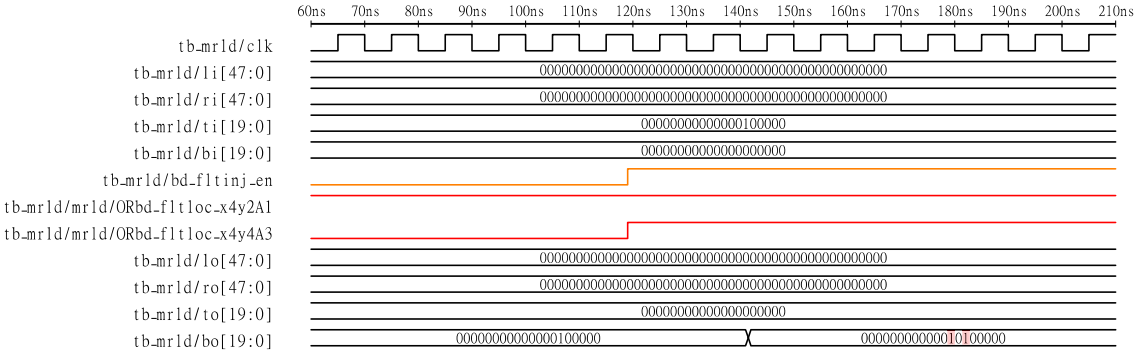

Fig. 18 Simulation result of the test under rm_2 for *ORbd*.

Table 3 Test effectivity for a single fault.

MPLD size	# of MLUTs: N_{lut}	AD-pair	# of input ports: N_{inp}	
	$x \times y$	m	$(y + (x\text{-}1)/2) \times m$	
# of total interconnects	$n=((x+1)y+(x\text{-}1)/2)m$			
# of locatable faults (=total)	sa-0/1: n AND/OR-bd: $n \times (n\text{-}1)/2$			
Time to configure an SRAM	T_{sram} (cycles)			
Time of one config. (cycles)	$T_{conf} = 2 \times N_{lut} \times T_{sram}$			
Time of one EP: T_{ep} (cycles)	all-0	all-1	walking-1	walking-0
Access via Boundary Scan	N_{inp}	N_{inp}	$2N_{inp}$-1	$2N_{inp}$-1
# of times of config. and EP	det. sa	loc. sa	det. bd	loc. bd
	1	2	1	2 or 3

$$\rightarrow \boldsymbol{x_4 y_2 A_1} \rightarrow x_5 y_1 A_{14} \rightarrow ro[13]\},$$

$$FP_2^{(1)} = \{li[35] \rightarrow x_1 y_3 A_{12} \rightarrow x_2 y_4 A_3 \rightarrow x_3 y_3 A_{12} \rightarrow \boldsymbol{x_4 y_4 A_3} \rightarrow x_5 y_3 A_{12} \rightarrow ro[31]\},$$

$$FP_1^{(2)} = \{ti[5] \rightarrow x_3 y_0 A_6 \rightarrow x_4 y_1 A_1 \rightarrow x_3 y_1 A_6 \rightarrow \boldsymbol{x_4 y_2 A_1} \rightarrow x_3 y_2 A_6 \rightarrow x_4 y_3 A_1 \rightarrow x_3 y_3 A_6 \rightarrow x_4 y_4 A_1 \rightarrow x_3 y_4 A_6 \rightarrow x_4 y_5 A_1 \rightarrow x_3 y_5 A_6 \rightarrow bo[5]\}$$

$$FP_2^{(2)} = \{ti[7] \rightarrow x_3 y_0 A_4 \rightarrow x_4 y_1 A_3 \rightarrow x_3 y_1 A_4 \rightarrow x_4 y_2 A_3 \rightarrow x_3 y_2 A_4 \rightarrow x_4 y_3 A_3 \rightarrow x_3 y_3 A_4 \rightarrow \boldsymbol{x_4 y_4 A_3} \rightarrow x_3 y_4 A_4 \rightarrow x_4 y_5 A_3 \rightarrow x_3 y_5 A_4 \rightarrow bo[7]\}.$$

Then, the *ORbd* can be identified by:

$$F_{loc} = \bigcap_{i=1}^{2} FP^{(i)} = \bigcap_{i=1}^{2} \left(\bigcup_{k=1}^{2} FP_k^{(i)} \right) = \left(FP_1^{(1)} \cup FP_2^{(1)}\right) \cap \left(FP_1^{(2)} \cup FP_2^{(2)}\right) = \{\boldsymbol{x_4 y_2 A_1}, \boldsymbol{x_4 y_4 A_3}\}.$$

The proposed test method can detect and locate all single faults at any AD interconnect. Table 3 shows the test effectivity to an MPLD with the size of $x \times y$ MLUTs (having m-pair AD interconnects) array. Each type of fault, present anywhere in all $((x + 1)y + (x\text{-}1)/2)m$ interconnects in MPLD, can be detected and located by the proposed test method. Detecting (det.) stuck-at (*sa*) faults requires only one configuration (config.) and one application of the external test pattern (EP); for bridge (*bd*) faults, it also only involves one time. Locating (loc.) stuck-at faults only requires two configurations and two applications of one identical external test pattern; for bridge faults, it only involves two or three times (where the fault sites of a bridged interconnect locate on the same column or row of the MLUT array only require two times, otherwise, it might require three times).

As shown in Table 3, the time (T_{conf}) for one configuration depends on the number of MLUTs (N_{lut}) and the time (T_{sram}) to configure an SRAM. The time (T_{ep}) for one external pattern application depends on the number (N_{inp}) of external logic input ports of the MPLD. Based on the structure of the MPLD presented in this paper, $N_{lut} = x \times y$; $N_{inp} = (y + (x\text{-}1)/2) \times m$; T_{sram} depends on the size ($2^{m/2} word \times m$-bit) of the SRAM. All in all, the time for one configuration and time for one external pattern application depends on the size of the MPLD.

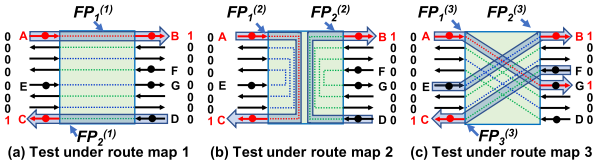

Fig. 19 Example to test triple faults.

4.5 Testing Multiple Faults

Our proposed test method is also available for multiple faults. The reason that any multiple faults can be detected and located is as follows.

(1) Any multiple faults can be detected because all interconnects are on pre-configured routes, and the external test pattern used for fault excitation traverses all these routes, exciting the fault sites on any interconnect. The excited fault responses are then propagated to the external logic output ports along the pre-configured routes, similar to detecting single faults, but capturing multiple sets of fault responses.

(2) Multiple faults can be located because, for the same fault site, its response under different route map tests will propagate along varying paths to different external logic output ports. By increasing the testing of diverse route maps, the intersection of the fault propagation paths narrows down to the precise fault site, enabling the location of multiple faults.

The testing procedure is described as follows. Suppose that *N-faults* exist at different AD interconnects on the MLUT array, where $N > 1$. Let the range of the multiple faults that occur be ${N_F}^R$, the number of faults in the range ${N_F}^R$ be N_F, and the number of faulty effects observed on the test route map i is denoted by ${N_{FE}}^{(i)}$, where $i = 1, 2, 3, \ldots$, denote the horizontal, vertical, diagonal, and so on, route map, respectively.

For *N-faults* that exist at any site on the MLUT array, the fault range can be determined by the following formula:

$$N_F \in {N_F}^R = [max({N_{FE}}^{(1)}, {N_{FE}}^{(2)}), {N_{FE}}^{(1)} \times {N_{FE}}^{(2)}].$$

And they can be detected and located by executing i-times tests with i route maps until the following formula holds.

$$max({N_{FE}}^{(1)}, \cdots, {N_{FE}}^{(i)}) == N_{\bigcap_{i=1}^{Nrm} FP^{(i)}},$$

$$then,\ N_F = N_{\bigcap_{i=1}^{Nrm} FP^{(i)}},\ F_{loc} = \bigcap_{i=1}^{Nrm} FP^{(i)}.$$

For instance, to test *triple faults* (stuck-at-1) at interconnect A, B, and C on the MLUT as shown in Fig. 19. By executing the test under two route maps (1: horizontal, 2: vertical), we can determine the range ${N_F}^R$ of the number of multiple faults N_F, where,

$${N_{FE}}^{(1)} = 2,\ {N_{FE}}^{(2)} = 2,\ max({N_{FE}}^{(1)}, {N_{FE}}^{(2)}) = 2,$$

$$N_F \in {N_F}^R = [max({N_{FE}}^{(1)}, {N_{FE}}^{(2)}), {N_{FE}}^{(1)} \times {N_{FE}}^{(2)}] = [2, 4],$$

Table 4 Test time and test data volume for multiple faults.

# of times of test: i	$i >= 2$
Time of test for detecting	$2 \times (T_{conf} + T_{ep})$
Time of test for locating	$i \times (T_{conf} + T_{ep})$
Test data of test for detecting	$2 \times (D_{conf} + D_{ep})$
Test data of test for locating	$i \times (D_{conf} + D_{ep})$

$$FP^{(1)} = \bigcup_{k=1}^{N_{FE}{}^{(1)}} FP_k^{(1)} = FP_1^{(1)} \cup FP_2^{(1)} = \{A, B\} \cup \{C, D\} = \{A, B, C, D\},$$

$$FP^{(2)} = \bigcup_{k=1}^{N_{FE}{}^{(2)}} FP_k^{(2)} = FP_1^{(2)} \cup FP_2^{(2)} = \{A, C\} \cup \{B, D\} = \{A, B, C, D\},$$

$$\bigcap_{i=1}^{2} FP^{(i)} = FP^{(1)} \cap FP^{(2)} = \{A, B, C, D\}.$$

Since $max(N_{FE}{}^{(1)}, N_{FE}{}^{(2)}) = 2$ *is unequal to* $N_{\bigcap_{i=1}^{2} FP^{(i)}} = 4$, we need to perform additional test using route map 3 on the diagonal direction, then,

$$N_{FE}{}^{(3)} = 3,\ max(N_{FE}{}^{(1)}, N_{FE}{}^{(2)}, N_{FE}{}^{(3)}) = 3,\ and,$$

$$FP^{(3)} = \bigcup_{k=1}^{N_{FE}{}^{(3)}} FP_k^{(3)} = FP_1^{(3)} \cup FP_2^{(3)} \cup FP_3^{(3)} = \{A, G\} \cup \{E, B\} \cup \{F, C\} = \{A, B, C, E, F, G\},$$

$$\bigcap_{i=1}^{3} FP^{(i)} = FP^{(1)} \cap FP^{(2)} \cap FP^{(3)} = \{A, B, C, D\} \cap \{A, B, C, E, F, G\} = \{A, B, C\}.$$

Then, since $max(N_{FE}{}^{(1)}, N_{FE}{}^{(2)}, N_{FE}{}^{(3)}) = 3$ *is equal to* $N_{\bigcap_{i=1}^{3} FP^{(i)}} = 3$, we can determine the number of multiple faults and identify fault locations:

$$N_F = N_{\bigcap_{i=1}^{3} FP^{(3)}} = 3,$$

$$F_{loc} = \bigcap_{i=1}^{3} FP^{(i)} = \{A, B, C\}.$$

Table 4 shows the test time and the amount of test data for detecting and locating multiple faults. Detecting whether multiple faults exist on the interconnect in MPLD requires at most two tests. Therefore, the most time for detecting is $2(T_{conf}+T_{ep})$ cycles and the most test data is $2(D_{conf}+D_{ep})$ bits. Where $D_{conf} = 2 \times N_{lut} \times K$ bits (K is the size of an SRAM: $2^{m/2} word \times m$-bit); D_{ep} $(= T_{ep})$ indicates the bit number of the external test pattern. For Locating multiple faults, as in the above testing procedure, i-times tests need to be performed. Therefore, the most time and test data are $i(T_{conf} + T_{ep})$ cycles and $i(D_{conf} + D_{ep})$ bits, respectively.

5. Delay Monitoring in MPLD

5.1 Delay-Monitoring Techniques

Commonly, aging-induced extra delay can be mitigated through manufacturing tests (such as burn-in test or stress test), redundancy designs, or by setting a certain timing margin in the operating frequency of devices at the design phase [12], [13]. Therefore, a pre-designed timing margin in the operation frequency of MPLD may accommodate the aging-induced delay of MLUTs throughout most of the lifetimes. However, in MPLDs, the rate of aging varies among MLUTs and accelerates in frequently used MLUTs. The delay in MLUTs with faster aging rates may surpass the timing margin sooner, potentially leading to abrupt system failures. Furthermore, optimizing the timing margin for a device is challenging due to variations in fabrication processes, workloads, and operational environments, which may compromise the device's performance [13].

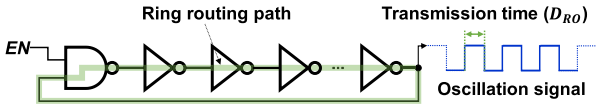

Fig. 20 Ring oscillator.

Delay-monitoring techniques [12], [14] serve as an effective means of ensuring a device's in-field reliability. These techniques measure the delay variation of a circuit influenced by process, voltage, and temperature (PVT) factors in real-time, by incorporating timing-measure circuits such as ring oscillators (ROs) [15] and time-to-digital converters (TDCs) [16] into the target device. If the delay surpasses the safe delay boundary, an early warning or report can be issued to the higher-level system, preventing system failure, or prompting maintenance actions such as repair and diagnosis. In [17], [18], the authors proposed an on-chip digital delay sensor using ROs to monitor the aging-induced delay of application-specific integrated circuit (ASIC) devices. Additionally, in [19], the authors integrated the on-chip digital delay sensor into the field-programmable gate array (FPGA) with the goal of enhancing the reliability of logic reconfigurable devices. As a result, we incorporate ROs into MPLDs for delay monitoring purposes.

5.2 Delay-Monitoring in MPLD

Figure 20 presents a schematic representation of a generic Ring Oscillator (RO) structure, which comprises a 2-input NAND gate and an even number of inverters connected in series to form a ring circuit. One of the inputs to the NAND gate is the oscillation control signal, denoted as *EN*. By setting *EN* to 0, the RO is initialized to a stable state. Conversely, when *EN* is set to 1, the RO operates in the oscillation mode and generates an oscillation signal at a specific frequency. The delay (transmission time) D_{RO} of RO's entire ring routing path is half of the oscillation period T_{RO} of RO. This delay can be calculated using the oscillation number $N_{osc}^{t_{RO}}$ within a certain oscillation operation time t_{RO} as follows:

$$D_{RO} = \frac{T_{RO}}{2} = \frac{t_{RO}}{2N_{osc}^{t_{RO}}} \quad (1)$$

In MPLD, we can deploy RO into specified measurement areas (partial MLUTs or all MLUTs) to measure the average delay (local delay or global delay) of MLUTs within the designated region.

Deploying RO in MLUTs:

First, the measurement area (MLUT cells to be

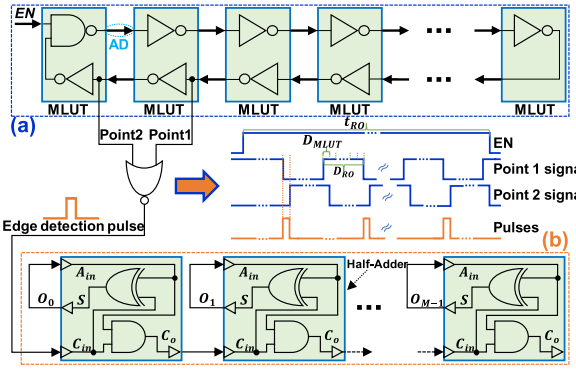

Fig. 21 Delay monitor; (a) RO in MLUTs, (b) counter for RO.

evaluated for delay) is determined, followed by the design of the RO structure in accordance with the subsequent deployment rules:

Rule 1- *RO elements:* A NAND gate and an even number of inverters must be routed in series within a ring.

Rule 2- *Deployment area:* All elements must be situated within the measurement area, and the loop routing path must pass through each MLUT cell within, rather than outside, the designated region.

Rule 1 is essential to ensure that the circuit can oscillate, while Rule 2 guarantees the accuracy of the delay measurement, which corresponds precisely to the delay of the measurement area.

Figure 21 (a) shows an example of deploying a RO circuit into MLUTs. The RO elements are placed within the MLUT cells to be measured and are routed in series in a ring through AD interconnects of the MLUTs. The average delay (D_{MLUT}) of the MLUT cells can be calculated using the transmission time (D_{RO}) of the loop routing path and the number (N_{AD}) of AD interconnects traversed by the loop routing path:

$$D_{MLUT} = \frac{D_{RO}}{N_{AD}} = \frac{t_{RO}}{2\mathrm{N}_{\mathrm{osc}}^{t_{RO}} N_{AD}} \tag{2}$$

Deploying Counter for RO:

Here we describe the design of the counter to calculate the oscillation number ($\mathrm{N}_{\mathrm{osc}}^{t_{RO}}$). Contrary to conventional counter designs composed of synchronous Flip-Flops, we propose a novel and more suitable counter circuit design that is simpler to implement in an MPLD, as depicted in Fig. 21 (b). The proposed counter consists of M half-adders connected in series. Upon setting the RO oscillation mode, pulses can be generated by a signal edge detection logic (in this case, a NOR logic is configured in the MLUT) by comparing the signals of two neighbor AD interconnects on the ring routing path. The $\mathrm{N}_{\mathrm{osc}}^{t_{RO}}$, the number of pulses, can be calculated by the counter through the execution of an addition carry operation for the pulses:

$$\mathrm{N}_{\mathrm{osc}}^{t_{RO}} = (O_{M\text{-}1} \cdots O_1 O_0)_2 \tag{3}$$

The procedure of implementing RO and counter for measuring the delay of MLUT as follows.

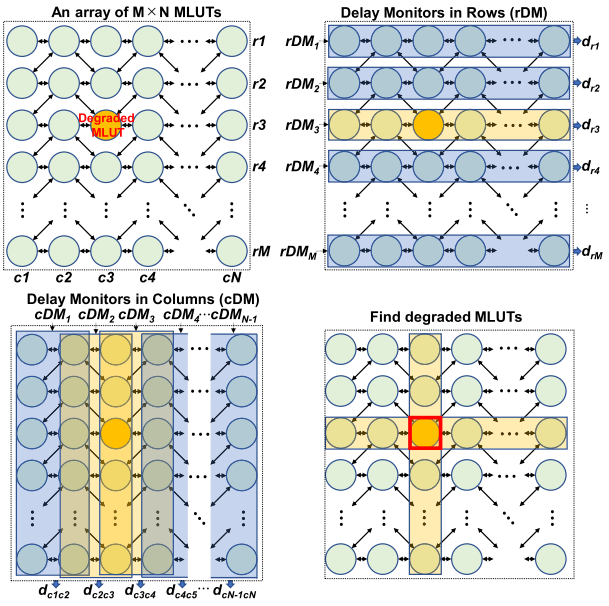

Fig. 22 Delay-monitors deploying to locate degraded MULT.

Implementation Procedure

Step 1: select measurement area (MLUTs);
Step 2: deploy RO and counter;
Step 3: create the truth tables for each MLUT in the area;
Step 4: write the truth tables into corresponding MLUTs;
Step 5: set the MPLD to logic operation mode;
Step 6: set oscillation operation time ($EN = 1$);
Step 7: observe the oscillation number (counter outputs).

The proposed delay monitor aims to periodically detect the degradation state of MLUTs in MPLDs operating in the field. To detect degraded MLUTs in an MPLD with an $M \times N$ array of MLUTs, as shown in Fig. 22, the total number and location of delay monitors are determined according to the detection method described below.

(1) *row detection*: M delay monitors are configured in M rows $(r_1, \ldots, r_M)$ to detect the average delay of MLUTs in each row $(d_{r1}, \ldots, d_{rM})$.

(2) *column detection*: N-1 delay monitors are configured in N columns $(c_1, \ldots, c_N)$, where two adjacent columns are required to configure a delay monitor. This detects the average delay of MLUTs in two adjacent columns $(d_{c1,c2}, \ldots, d_{cN\text{-}1,cN})$.

(3) *locating degraded MLUTs*: MLUTs in the intersection region of rows with delays in M rows and columns with delays in N rows. For example, the MLUT at row 3 and column 3, which is determined by d_{r3}, $d_{c2,c3}$, and $d_{c3,c4}$.

In this setup, row detection and column detection are configured simultaneously and work in parallel. It is worth noting that the proposed delay monitor is configured in the MLUT as a truth table without incurring logical gates and routing costs. The only additional overhead is the time cost of configuring the delay monitor truth table in the MLUT.

5.3 Simulation Results

To assess the effectiveness of the proposed delay

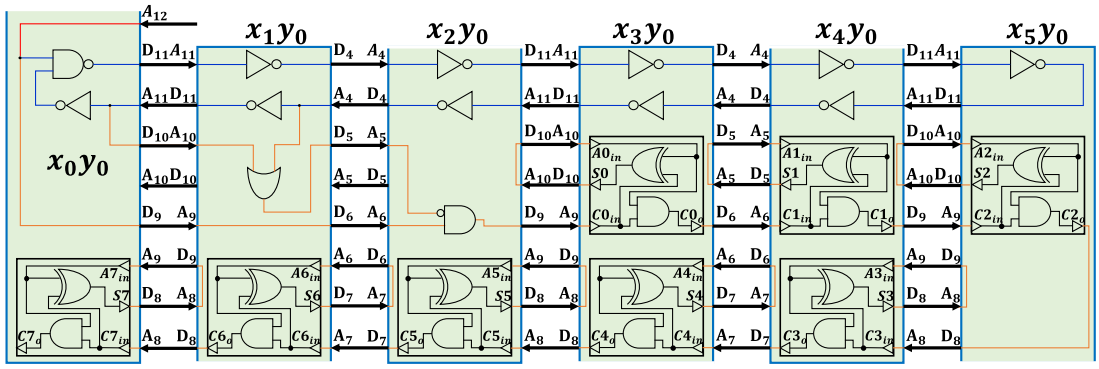

Fig. 23 RO and counter in MLUTs to be measured for delay.

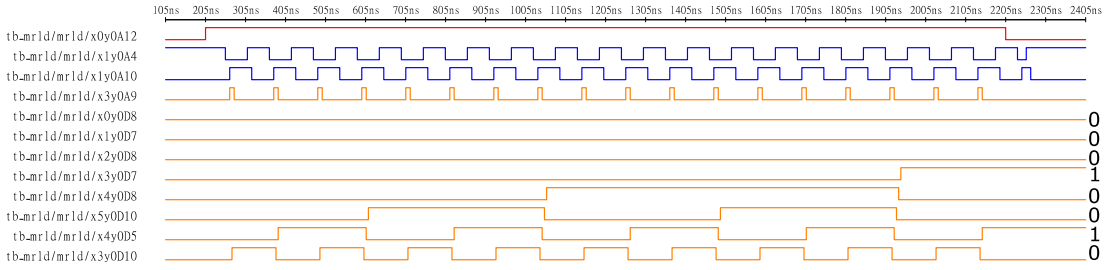

Fig. 24 Simulation waveform to measure delay for MLUT.

monitoring technique, as depicted in Fig. 23, we configured a Ring Oscillator (RO) comprising 11 elements (a NAND gate and 10 inverters) and an 8 half-adder counter within the measurement area (MLUTs: x_0y_0, x_1y_0, x_2y_0, x_3y_0, x_4y_0) of the designed MPLD containing a 6×6 MLUTs array. A logic simulation experiment was conducted using the following procedure.

1: Route the RO pass through 10 AD interconnects in the measurement area ($N_{AD} = 10$).

2: Set the delay of the ATD circuit for each MLUT to *5.5ns* and the overall oscillation operation time of the RO to *2000ns* (t_{RO}).

The waveform of the RO oscillation and the counter is shown in Fig. 24. When setting the oscillation control signal to 1, the RO commences oscillation while the counter records the detected pulse until the oscillation control signal reverts to 0. The pulse number (RO oscillation number) counted by the counter is 18: $N_{osc}^{t_{RO}} = (00010010)_2$. Thus, the average delay of MLUTs within the area can be calculated by formula (2):

$$D_{MLUT} = \frac{t_{RO}}{2N_{osc}^{t_{RO}} N_{AD}} = \frac{2000ns}{2 \times 18 \times 10} = 5.\dot{5}ns \tag{2}$$

Compared the set delay (*5.5ns*) of the ATD circuit with the D_{MLUT} ($5.\dot{5}ns$), the result confirms the effectiveness of the proposed delay monitor method.

6. Conclusions

MPLD is a novel type of programmable logic device that offers the advantages of low production cost, reduced power consumption, and minimal delay. To ensure the long-term reliability of the MPLD device, we propose a test method for identifying interconnect defects during the production phase, as well as a delay monitor technique to detect aging-induced failures in the field.

The proposed test method generates route maps in MPLD for fault propagation by configuring pre-designed test configuration data into an SRAM array. This method excites faults by applying an external walking-zero/one vector to the input ports of the MPLD. Faults are identified when the fault effect is propagated to the external output ports.

The delay monitor technique involves configuring a carefully designed delay measurement circuit, which utilizes a ring oscillator, into the MPLD to monitor delay variations induced by aging.

To evaluate the proposed methods, we designed an MPLD with a 6×6 MLUTs array and conducted logic simulations by injecting faults into the MPLD. The simulation results corroborate the effectiveness of the proposed methods.

We posit that the proposed test method may be applicable to multiple faults. In our future work, we will investigate the effectiveness of the proposed methods for various fault types.

References

[1] A. Rupani, D. Pandey, and G. Sujediya, "Review and Study of FPGA Implementation of Internet of Things," Int. J. of Sci. Technol. & Eng., vol.3, no.2, Aug. 2016.

[2] M.R. Nithin, R. Basheer, and S. Mohan, "Advanced Driver Assistance System using FPGA," QuEST Global Corp., May 2017.

[3] C. Wang, L. Gong, Q. Yu, X. Li, Y. Xie, and X. Zhou, "DLAU: A Scalable Deep Learning Accelerator Unit on FPGA," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Syst., vol.36, no.3, pp.513–517, March 2017.
[4] TAIYO YUDEN CO LTD, "Reconfigurable semiconductor device," Japan Patent JP2016208426A, Dec. 2016.
[5] H. Puchner and L. Hinh, "NBTI reliability analysis for a 90nm CMOS technology," 30th Eur. Solid-State Circuits Conf., pp.257–260, Sept. 2004.
[6] F. Chen and M. Shinosky, "Addressing Cu/Low-*k* Dielectric TDDB-Reliability Challenges for Advanced CMOS Technologies," IEEE Trans. on Electron Devices, vol.56, no.1, pp.2–12, Jan. 2009.
[7] D. Rossi, "The Effects of Ageing on the Reliability and Performance of Integrated Circuits," Ageing of Integrated Circuits: Causes, Effects and Mitigation Techniques, B. Halak, Ed., Springer International Publishing, pp.35–64, 2020.
[8] S.S. Sapatnekar, "What happens when circuits grow old: Aging issues in CMOS design," 2013 Int. Symp. on VLSI Technol., Syst. and Appl. (VLSI-TSA), pp.1–2, 2013.
[9] S. Wang, Y. Higami, H. Takahashi, M. Sato, M. Katsu, and S. Sekiguchi, "Testing of Interconnect Defects in Memory Based Reconfigurable Logic Device (MRLD)," 2017 IEEE 26th Asian Test Symp., pp.17–22, 2017.
[10] S. Wang, et al., "Test Method for the Bridge Interconnect Faults in Memory Based Reconfigurable-Logic-Device (MRLD) Considering the Place-and-Route," 33th Int. Tech. Conf. Circuits/Syst., Comput. Commun., 2018.
[11] T. Inoue, S. Miyazaki, and H. Fujiwara, "Universal fault diagnosis for lookup table FPGAs," IEEE Des. Test Comput., vol.15, no.1, pp.39–44, 1998.
[12] Y. Sato, S. Kajihara, Y. Miura, T. Yoneda, S. Ohtake, M. Inoue, and H. Fujiwara, "A Circuit Failure Prediction Mechanism (DART) for High Field Reliability," 8th IEEE Int. Conf. on ASIC, pp.581–584, Oct. 2009.
[13] M.S. Mispan, et al., "Ageing Mitigation Techniques for SRAM Memories," Ageing of Integrated Circuits: Causes, Effects and Mitigation Techniques, B. Halak, Ed., Springer International Publishing, pp.91–111, 2020.
[14] Y. Tsugita, K. Ueno, T. Hirose, T. Asai, and Y. Amemiya, "An on-chip PVT compensation technique with current monitoring circuit for low-voltage CMOS digital LSIs," IEICE Trans. on Electron., vol.E93-C, no.6, pp.835–841, 2010.
[15] M. Bhushan, A. Gattiker, M.B. Ketchen, and K.K. Das, "Ring oscillators for CMOS process tuning and variability control," IEEE Trans. on Semicond. Manuf., vol.19, no.1, pp.10–18, Feb. 2006.
[16] P. Chen, C.-C. Chen, C.-C. Tsai, and W.-F. Lu, "A time-to-digital-converter-based CMOS smart temperature sensor," IEEE J. of Solid-State Circuits, vol.40, no.8, pp.1642–1648, Aug. 2005.
[17] S. Kajihara, Y. Miyake, Y. Sato, and Y. Miura, "An On-Chip Digital Environment Monitor for Field Test," 2014 IEEE 23rd Asian Test Symp., pp.254–257, Nov. 2014.
[18] Y. Miyake, Y. Sato, S. Kajihara, and Y. Miura, "Temperature and Voltage Measurement for Field Test Using an Aging-Tolerant Monitor," IEEE Trans. on Very Large Scale Integration (VLSI) Syst., vol.24, no.11, pp.3282–3295, Nov. 2016.
[19] Y. Miyake, Y. Sato, and S. Kajihara, "On-Chip Delay Measurement for In-Field Test of FPGAs," 2019 IEEE 24th Pacific Rim Int. Symp. on Dependable Computing (PRDC), Kyoto, Japan, pp.130–137, Dec. 2019.

Xihong Zhou received M.S. from the Department of Computer Science and Electronics, Ehime University, Japan, in 2019. Currently, he is a Ph.D. student in the Department of Computer Science at Ehime University. His research interest includes field testing, fault diagnosis for digital systems, and design for testability. He is a student member of the IEEE and IEICE.

Senling Wang received M.S. and Ph.D. degree from the Department of Computer Science and Electronics, Kyushu Institute of Technology, Japan, in 2011 and 2014, respectively. Currently, he serves as a Senior Assistant Professor in the Graduate School of Science and Engineering, Ehime University, Japan. His research interest includes Field testing, Low power testing, and Design for Testability. He is a member of the IEEE, IEICE, IPSJ and JIEP.

Yoshinobu Higami received his B.E., M.E., and D.E. degrees from Osaka University in 1991, 1993 and 1996, respectively. Currently he is a full professor at the Graduate School of Science and Engineering, Ehime University. In 1998 and 2006, he was also an honorary fellow at University of Wisconsin Madison, U.S.A. He received the IEICE Best Paper Award in 2005 and 2012, and Best Paper Award of IEEE Computer Society Annual Symposium on VLSI in 2014. His research interests include test generation, design for testability and fault diagnosis of logic circuits. He is a senior member of the IEEE and a member of IEICE and IPSJ.

Hiroshi Takahashi received the Ph.D. degree from Ehime University, Japan in 1996. Since 2010, he has been a Full Professor at Ehime University. From 2018 to 2023, he served as Dean of Faculty of Engineering and Graduate School of Science and Engineering at Ehime University. From May 2000 to March 2001, he was a research fellow at the University of Wisconsin-Madison, USA. He received the IEICE Best Paper Award in 2012, the best Paper Award of IEEE Computer Society Annual Symposium on VLSI in 2014 and the Reliability Engineering Association of Japan Takagi Prize in 2016. His research interests are test generation and fault diagnosis for digital systems. He served as the Program Chair of the 2012 IEEE Asian Test Symposium. He also served as the General Co-Chair of the 2016 IEEE Asian Test Symposium. He is a senior member of the IEEE, IEICE, and IPSJ.