Testing and Fault Tolerance Techniques for CNT-Based FPGAs

Siyuan Lu[1] [a,d], Kangwei Xu[1] [b], Peng Xie [a], Rui Wang [c] and Yuanqing Cheng [a,d,*]

[a]*School of Integrated Circuit Science and Engineering, Beihang University, Beijing, 100190, China*

[b]*Department of Electronic Design Automation, Technical University of Munich (TUM), Munich, 80333, Germany*

[c]*School of Computer Science and Engineering, Beihang University, Beijing, 100190, China*

[d]*Shenzhen Institute of Beihang University, Shenzhen, 518000, China*

ABSTRACT

As the semiconductor manufacturing process technology node shrinks into the nanometer-scale, the CMOS-based Field Programmable Gate Arrays (FPGAs) face big challenges in scalability of performance and power consumption. Multi-walled Carbon Nanotube (MWCNT) serves as a promising candidate for Cu interconnects thanks to the superior conductivity. Moreover, Carbon Nanotube Field Transistor (CNFET) also emerges as a prospective alternative to the conventional CMOS device because of high power efficiency and large noise margin. The combination of MWCNT and CNFET enables the promising CNT-based FPGAs. However, the MWCNT interconnects exhibit significant process variations due to immature fabrication process, leading to delay faults. Also, the non-ideal CNFET fabrication process may generate a few metallic CNTs (m-CNTs), rendering correlated faulty blocks. In this article, we propose a ring oscillator (RO) based testing technique to detect delay faults due to the process variation of MWCNT interconnects. Furthermore, we propose an effective testing technique for the carry chains in CLBs, and an improved circuit design based on the lookup table (LUT) is applied to speed up the fault testing of CNT-based FPGAs. In addition, we propose a testing algorithm to detect m-CNTs in CLBs. Finally, we propose a redundant spare row sharing architecture to improve the yield of CNT-based FPGA further. Experimental results show that the test time for a 6-input LUT can be reduced by 35.49% compared with conventional testing, and the proposed algorithm can achieve a high test coverage with little overhead. The proposed redundant architecture can repair the faulty segment effectively and efficiently.

1. Introduction

Field-programmable gate arrays (FPGAs) have been the most popular reconfigurable fabrics in the past few decades and are widely used in numerous commercial applications [1, 2]. The high programmability of FPGAs is implemented by FPGAs' interconnects and configurable logic. However, the resistivity of copper (Cu) interconnect increases due to electron surface scattering. Besides, these configurable logic blocks built with conventional MOSFETs suffer from large area overheads and high leakage power [3]. As shown in Fig. 1 (a), electromigration-resistant multi-walled carbon nanotube (MWCNTs) is a prospective candidates to replace Cu interconnect due to its superior conductivity and ampacity [4]. Moreover, as shown in Fig. 1 (b), carbon nanotube-based field-effect transistor (CNFET) is a promising alternative to MOSFET due to the extremely low power dissipation and high endurance [5] [6] [7]. It shows that CNT-based FPGAs can provide 2.67× performance improvement compared with CMOS-based FPGAs at the same technology node [8]. Recently, a microprocessor called N3XT is introduced, in which the logic and peripheral circuits are manufactured using CNT technology, which can achieve 850× EDP (Energy-Delay product) improvement compared to MOSFET technology[9].

While CNFET has many benefits compared with conventional MOSFETs, it can be widely adopted only if the design and manufacturing cost are commercially viable [10]. One
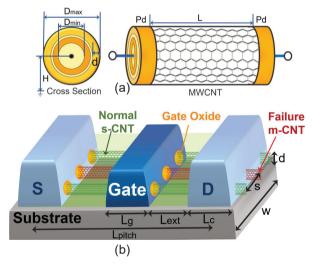

Figure 1: (a) Cross-sectional view and 3D view of an MWCNT structure (b) 3D view of a CNFET structure

of major challenges on the path to large-scale CNT-FPGAs is the process variation of multi-walled carbon nanotube (MWCNT) bundled interconnects [11]. Growing CNTs introduces several feature size variations (e.g., shell diameter and chirality), which affect the propagation delay of signals [4]. Many delay-fault testing methods for FPGAs are supported by built-in self-test (BIST) [12] structures, such as [13] [14]. These works compared timing differences among the paths under test (PUTs). However, the test accuracy was degraded due to the uncertainty introduced by clock skew.

Another major challenge for CNT-based FPGAs is the presence of metallic CNTs (m-CNTs) in the channels of CNFETs devices [5]. As shown in Fig. 1 (b), the normal semiconducting CNTs (s-CNTs) are grown together with m-CNTs in the fabrication process. The s-CNTs are promising channel materials for building CNFETs, while an m-CNT can induce a stuck-on fault in a CNFET device, leading to the malfunction of circuits. Since an m-CNT can be as long as hundreds of micrometers [18], a large number of correlated blocks can be faulty blocks once m-CNTs appear in CNT-based programmable gate arrays. The key open-ended problem of efficient testing is not solved satisfactorily in existing works [19]. Therefore, it is of great significance to propose fast and effective test techniques for CNT-based FPGAs. Moreover, it is necessary to propose novel redundant architectures to repair faulty components containing m-CNTs in order to improve yield further.

In this article, combining with the unique fault patterns in carbon nanotube-based programmable gate arrays, we propose a novel delay testing technique for multi-walled carbon nanotube (MWCNT) interconnects, and a fast diagnosis of the faulty segments in configurable logic blocks (CLBs). Besides, a redundant spare row sharing architecture is proposed to repair faulty segments. The main contributions of this work can be summarized as follows.

1. We propose a ring oscillator (RO) based testing technique to identify delay faults on MWCNT interconnects.
2. Then, we establish the fault models induced by metallic CNTs (m-CNTs) that may occur in a configurable logic block (CLB), and we propose the carry chain testing methodology supplementing the traditional test configurations. An improved testing circuit design based on a lookup table (LUT) in a CLB is also explored to speed up the delay fault testing.
3. Considering these correlated faulty CLBs induced by m-CNTs, an effective technique to diagnose the faulty segments is proposed.
4. Considering these faulty segments induced by m-CNTs, we propose a redundant architecture to repair the faulty segments effectively.

The rest of this article is organized as follows. Section 2 introduces the background of the CNT-based FPGAs and the related work. Section 3 presents the MWCNT-based delay fault testing, illustrates the test technique for a single CLB considering m-CNT induced defects, and introduces the diagnosis methodology for the overall CLBs. Section 4 describes the redundant architecture of m-CNT-induced faulty tiles. Section 5 presents the experimental evaluations of our proposed testing and fault tolerant techniques. Section 6 concludes the paper.

2. Preliminaries and Motivation

In this section, we introduce the preliminaries of CNT technology. Then, we present the motivation of the CNT-based FPGA architecture.
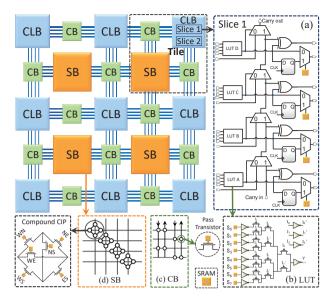

Figure 2: The CNT-based FPGA architecture (a)The diagram of a slice in a CLB, which contains LUTs, carry chain and triggers (b) The CNT-based LUT (c) The connection block (d) The switch block

2.1. Preliminaries
2.1.1. MWCNT and CNFET

The multi-walled carbon nanotube (MWCNT) interconnect is considered as a promising alternative to Cu interconnect in terms of performance. MWCNT has many superior properties such as near ballistic transport, high conductivity and ampacity. As shown in Fig. 1 (a), an MWCNT interconnect has several concentric shells with diameters ranging from several nanometers to tens of nanometers [11].

The structure and operation of a CNT-based field-effect transistor (CNFET) are analogous to those of a CMOS device. As shown in Fig. 1 (b), semiconducting CNTs (s-CNT) form the conducting channel between source and drain, and can be controlled by a gate electrode. Based on intrinsic CV/I gate delay, CNFET devices can be up to 13× and 6× faster than pMOS and nMOS devices with the same gate length [20].

In this work, we assume the MWCNTs serve as the interconnects, and CNFETs are used for transistors to construct the CNT-based FPGA.

2.1.2. CNT-based FPGA Architecture

Modern FPGAs typically use an island-based architecture, which is mainly composed of configurable logic blocks (CLBs) connected via programmable connection blocks (CBs) and switch blocks (SBs) [21], and a tile is composed of a CLB connected via two CBs and a SB, as shown in Fig. 2. A CLB consists of the combinational and sequential elements that are needed to implement specific logic functions. Static Random-Access Memory (SRAM) is used to store truth table in a lookup table (LUT) [22, 23] in a CLB. In a CBs/SBs, SRAM stores the configuration bits to control interconnects. The selection signals of multiplexers are also controlled by SRAM.

Fig. 2 (a) shows the schematic of a slice in a CLB, the programmability of a CLB is controlled by the lookup table (LUT). A slice is composed of LUTs, carry chain, flip-flops, and some multiplexers (MUXs) used to select the immediate output or registered output. An LUT with K inputs can implement any K-input Boolean functions. As shown in Fig. 2 (b), for a K-input LUT, 2^K SRAM cells are used to store the configuration bits, and a 2^K-to-1 MUX is used to select the output bit.

In Fig. 2 (c), a CB consists of a few programmable switches and MWCNT interconnects. It serves to connect the interconnect channels and the inputs/outputs of CLBs. The pass transistors (PTs) instead of transmission gates are commonly used to implement programmable interconnects in FPGAs because each PT requires only one P-CNFET.

As shown in Fig. 2 (d), for each SB, its connection is controlled by six configurable interconnect points (CIPs). Individual CIP in a switch blocks is denoted by the relative directions of two to-be-connected wires. For example, a WN CIP connects a wire located in the west (W) to another wire in the north (N). To provide high flexible programmability, lots of programmable CLBs, SBs and CBs are integrated in an FPGA.

2.2. Motivation

The use of ring oscillators is an effective technique to measure variations in FPGA manufacturing processes. Li *et al.* used an array of ring oscillators in an FPGA to measure the gate length variation [15], which was then used to improve the fabrication process and reduce the negative effect of process variations on circuit performance. For the traditional testing scheme of CMOS-based CLBs, according to horizontal testing and vertical testing [24], the faulty CLBs can be identified by intersecting faulty columns with faulty rows. But this scheme cannot test the cascaded faulty CLBs effectively. In the technique presented in [14], every CLB used in the mapped design is reconfigured as transparent logic to construct scan chains. Also, fanout branches of a net are tested in different test configurations, resulting in a number of test scenarios. Due to the complexity of configuration generation algorithm, it cannot be applied to large designs.

Besides, although m-CNTs can be removed by electrical burning [25], sorting [26], and selective etching [27], these techniques can not achieve perfect metallic CNT removal. The m-CNT removal process may also create an open circuit if some s-CNTs under the FET are removed (see Fig. 6 (b)). Therefore, testing unique faults on CNT-based FPGA becomes increasingly important to guarantee the performance and the yield.

In addition, for the CMOS-based faulty memory, it is common to use the Built-in Self-Repair (BISR) method [28], which is a good way to repair faulty memory rows or columns and increase memory yield. But this method is suitable for discrete random failures, and not effective for continuous failures caused by random distribution of m-CNTs in CNFETs. Considering the CNT-based faulty
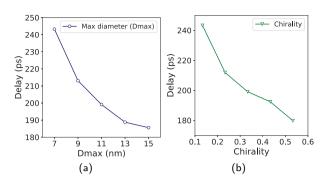

Figure 3: The delay between two adjacent CNT-based CLBs with (a) Dmax variation (b) chirality variation

SRAMs, the adjacent CLBs sharing technology (SSS-2) based on Divide BitLine (DBL) was proposed to repair faulty segments [29]. But when the number of CLBs is small, the repair rate is not satisfied. Furthermore, this method is only suitable for SRAM repair, and will cause significant overhead when applied for tile-based system level fault tolerance.

3. Ring Oscillator-based Delay Fault Testing

In this section, we explore the delay fault of MWCNT interconnects and propose a Ring Oscillator (RO)-based BIST scheme, which can effectively test delay faults induced by the MWCNT interconnects.

3.1. The Delay Fault of MWCNT Interconnects

As the line width and interconnect pitch scale down to nanometers, the propagation delay becomes a major performance concern. Related work shows that the CNT diameter and chirality variations play essential roles in determining the performance of MWCNT interconnects [4].

To evaluate the effects of above parameters on MWCNT delay, we plot the delay between two adjacent CNT-based CLBs with MWCNT variation settings as in [4]. As shown in Fig. 3 (a), delay between two adjacent CLBs with the D_{max} variation decreases significantly as the diameter increases. Similarly, as the chirality of MWCNT is improved (see Fig. 3 (b)), the delay between two adjacent CLBs is also reduced. Nearly 37% of delay improvement is observed when chirality is changed from 0.33 (without any chirality optimization during fabrication) to 0.53 (53% metallic CNT).

We evaluate the timing variations due to diameter (D_{max}) and chirality variations by Monte Carlo (MC) simulations. We assume the D_{max} variation obeys Gaussian distribution N ($11nm$, 1.65^2nm^2), and the chirality of each shell obey Bernoulli distribution with each shell of 1/3 probability to be metallic [4].

The interconnect delay variation between two adjacent CLBs is shown in Fig. 4. We can observe that a few interconnect paths exhibit large delay variations. As the FPGA fabrication technology migrates to deep sub-micron regime, the impact of delay faults on interconnect paths will become more acute [13].
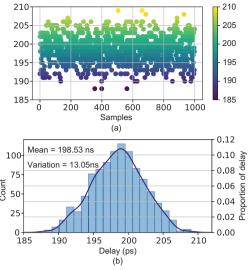
Figure 5: (a) The ring oscillator structure with 7-stages (b) Ring oscillator constructed by seven 6-input LUTs

Table 1
Formation of oscillating path with 6-input LUTs using XNOR logic function

I0	I1	I2	I3	I4	I5 (Input)	Output = I5⊙I4⊙I3⊙I2⊙I1⊙I0
0	0	0	0	0	0	SRAM0 is mapped to value '1'
0	0	0	0	0	1	SRAM0 is mapped to value '0'

Figure 4: The Monte Carlo simulation of propagation delay between two adjacent CLBs considering MWCNT process variations.

3.2. RO-based Delay Fault Testing of MWCNT Interconnects

Routing resources consist of wire segments that are connected or disconnected by configurable interconnect points (CIPs). In this work, the ROs are constructed to measure the delay of MWCNT interconnect path. Note that XOR tree-based testing structures were used to detect delay faults of ASICs in prior works [33], but this method did not deal with the application to FPGAs.

A RO is a circuit that consists of an odd number of inverting logic stages connected in series to form a closed-loop chain. An example where each stage consists of an inverter is shown in Fig. 5(a) The oscillating period is twice the sum of the propagation delay of all elements that compose the loop. ROs can be mapped on FPGAs using LUTs to measure the propagation delay of MWCNT interconnects.

Fig. 5(b) illustrates a possible formation of a ring oscillator with 7-stages using 6-input LUTs. A 6-input LUT consists of 64 SRAMs and a 64:1 multiplexer. A tree of 2:1 multiplexers has been used to build the 64:1 multiplexer. Any 6-input Boolean function can be realized by setting the truth table values in the SRAMs, where the output is determined by the logic values of three-level hierarchical selectors (I5, I4, I3, I2, I1, and I0). The output of each LUT is connected to selection bits of the next LUT to form a closed-loop chain. For each test configuration, we denote the frequency of each 7-stage RO as $f_1, f_2, f_3, ..., f_{N^2/7}$ respectively. N^2 represents the number of tiles in the FPGA array. Proper mapping of SRAM cell data and MUX selection bits are required to obtain the oscillation behavior from LUTs.

A Boolean function that implements inverter logic is needed. Standard logic XNOR and XOR can be used as an inverter when one of the inputs is considered as RO input while other inputs remain stable. An example of XNOR-based mapping for 6-input LUT is illustrated in Table 1, the operator '⊙' represents the XNOR operation. In Sec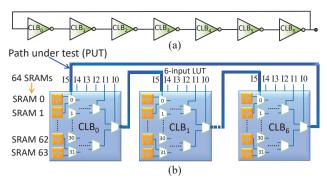tion 4.1, we map odd LUTs (7 LUTs) into a ring oscillator as a RO cell. LUT input pin I0 serves as input to inverter logic, and XNOR-based ROs can satisfy the proposed testing requirement.

Compared with traditional scan-based or MUX-based delay test techniques, the proposed RO-based structure eliminates the need for external pattern generation and I/O measurement, and achieves higher fault coverage with significantly lower hardware overhead.

While prior techniques such as scan/launch methods require complex scan chains and dedicated timing paths, our approach leverages on-chip reconfigurable LUTs to form oscillators that directly measure interconnect delay as a frequency shift.

4. Fault Testing Methodologies for Configurable Logic Blocks

In this section, we first analyze fault models induced by m-CNTs that may occur in a CLB. Then, we propose the test technique for the carry chain in a CLB, and a test circuit design based on an LUT is also explored to speed up the fault testing. Finally, we propose a technique to diagnose and repair the faulty segments within an FPGA tile.

4.1. M-CNT Induced Fault Model

As mentioned in Section 3, s-CNTs form the channel of CNFET. However, m-CNTs may grow together with normal s-CNT, and a typical CNT synthesis process yields roughly 3%-33% of m-CNTs. As shown in Fig. 6 (a), CNFET containing such m-CNTs is no longer controlled by the gate, which can lead to a short defect between source and drain, and cause a failure in the CNFET. Moreover, m-CNTs affect the current flowing through a CNFET when it is switched on (I_{on}) and off (I_{off}). If the ratio $\frac{I_{on}}{I_{off}}$ of a CNFET cannot reach a designated threshold value, the CNFET is regarded as having an open fault.
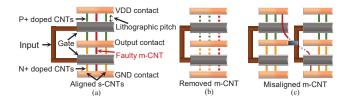
Figure 6: m-CNT in CNT-based inverter (a) m-CNT leading to a short fault (b) s-CNT removal leading to an open fault (c) Misaligned CNTs in a two-stage inverter

Considering a representative design scenario, an FPGA contains hundreds of thousands of tiles, with a 5% m-CNT probability (a typical CNT synthesis process results in a percentage of m-CNTs in the range 3%-33%) and 99.99% m-CNT removal percentage [25]. There may be dozens of faulty tiles in the CNT-based FPGA. The growing length of a CNT can be as long as hundreds of micron meters, an m-CNT can lead to a few correlated faulty CLBs; a misaligned m-CNT may span different rows, which can result in faulty segments in CNT-based FPGAs, and cause several cascaded CLBs to malfunction. The m-CNT can also have length and angle variations, and the induced faults are highly correlated with the CNT growing direction.

As mentioned above, for the CNT-based logic circuits, the growth of m-CNTs may cause incorrect circuit functionality. A CLB in the CNT-based programmable gate array is mainly composed of LUTs (including SRAMs, multiplexers, etc.), carry chains and triggers. An LUT is typically built out of SRAM bits that hold mapped values and a set of multiplexers (MUX) to select the bit that drives the LUT output. Next we analyze the fault model of an LUT, and we verify the fault models of different LUT components according to [31].

1) SRAM fault model: As shown in Fig. 7 (b), we consider the following scenarios: In scenario (1), an m-CNT passes through two horizontal CNFETs in a row, e.g., T_2. In scenario (2), a misaligned m-CNT affects non-vertical CN-FETs in two CNT bundles, e.g., T_2 and T_3. In scenario (3), an m-CNT terminates after it passes through one CNFET, e.g., T_1. Note that the etched m-CNT is drawn with the light red dotted line.

A CNT-based SRAM cell may exhibit various types of faults, depending on the locations of m-CNTs in the SRAM. Table 2 summarizes the fault models considering typical m-CNT induced faults at the transistor level. The first column describes different scenarios mentioned above. In the second column, 'X–Y' means two shorted positions, 'X' and 'Y'. The third column refers to the transistor label in the SRAM that suffers from the short defect, and the last column lists faults to be detected. We observe that all faults can be modeled as the conventional stuck-at fault.

2) MUX fault model: As shown in Fig. 8 (a), a 3-input LUT is composed of 8 SRAM cells and an 8:1 multiplexer. Any logic function of 3-inputs can be realized by setting the appropriate value in the SRAM cells and 3 level hierarchical selectors (I0, I1, I2). The CNT-based MUX can be built by P-CNFETs as transmission transistors. A 3-input LUT needs
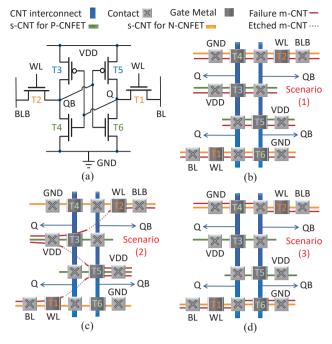

Figure 7: (a) The circuit schematic of a CNT-based SRAM cell. (b) The faulty layout of a SRAM cell induced by m-CNTs, which grow together with s-CNTs. (c) The faulty layout of a SRAM cell induced by misaligned m-CNTs. (d) The faulty layout of a SRAM due to the length variation of m-CNTs.

Table 2
The fault models of CNT-based SRAM.

The growth of m-CNT	Short fault (X-Y) between two points (VDD → 1, GND → 0)	Scenarios Position	SRAM Fault Model
Scenario (1) : m-CNTs grow together with s-CNTs	Q-0, QB-0	T1, T6	Stuck-at 0
	Q-0, QB-0	T2, T4	Stuck-at 0
	Q-1, QB-1	T3	Stuck-at 1
	Q-1, QB-1	T5	Stuck-at 1
Scenario (2) : Misaligned m-CNT in a SRAM	Q-1, QB-1	T2, T3	Stuck-at 1
	Q-1, QB-1	T3, T5	Stuck-at 1
	Q-1, QB-1	T1, T5	Stuck-at 1
Scenario (3) : m-CNT with length variation, and only affecting one CNFET	Q-BL	T1	Stuck-at 1
	QB-BLB/Q-0, QB-0	T2/T4	Stuck-at 0
	Q-0, QB-0	T6	Stuck-at 0

8 CNT bundles as transmission paths. When scaled to K inputs, the LUT will contain 2^K CNT bundles.

The CNT-based MUX may also present various faults depending on positions of m-CNTs, and we analyze the following typical scenarios in Fig. 8 (b): In scenario (1), an m-CNT passes through a whole row, and affects all selection (S) signals, e.g., short between SRAM-0 and output, which leads the MUX to always output the value stored in SRAM-0. In scenario (2), an m-CNT terminates after it passes through one S signal, e.g., a short between SRAM-2 and a CNFET. When the SRAM-3 output is selected, it causes a wired-AND/OR fault of the values stored in SRAM-2 and SRAM-3. In scenario (3), a misaligned m-CNT affects transmission gates in two CNT bundles: the front m-CNT causes a wired-AND/OR fault of SRAM-4 and SRAM-5 when the SRAM-5 output is selected, the end part of the
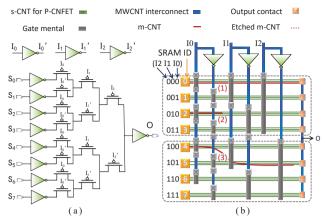
Figure 8: (a) The circuit schematic of a MUX. (b) The faulty layout of a MUX induced by m-CNTs.

m-CNT also causes a wired-AND/OR fault when the output of SRAM-1/SRAM-3/SRAM-7 is selected (i.e., affects the output of SRAMs corresponding to $\overline{I2}$).

4.2. Overview of Fault Types

As mentioned in Section 2, faults may occur in the LUT (including SRAM memories, a MUX and inverters), the carry chain logic (including MUXs and XOR gates) and triggers.

The fault type of a CNT-based SRAM can be considered as the conventional stuck-at faults. If a fault occurs in an SRAM cell, wrong SRAM value will be output.

A multiplexer is a group of switches and only one switch is allowed to be on. The fault type of a CNT-based MUX can be regarded as short fault or the wired-AND (wired-OR) fault as mentioned in Section 3.

For a trigger (D flip-flop), a fault may cause the trigger to receive wrong data or to be incapable of being set or reset.

5. Problem Formulation and Analysis

5.1. Fault Testing for a Single CNT-based CLB

In this work, we adopt the test technique for a single CLB proposed in [34] and we add the carry chain test in the CLB, which is not solved in [34]. Then, an improved design based on LUT is also proposed to speed up the fault testing.

1) Universal test procedure: The configuration SRAM memory cells (CMCs) are used to configure its logic functions. When programming an FPGA, we can load the bit patterns into CMCs [32]. Such a programming process is called a configuration. We denote the procedure for testing CLBs in CNT-FPGAs as the test session TS_{CLB}. Then, we represent TS_{CLB} consisting of a configuration and input test patterns [16, 17] applied to the configuration as follows:

$$TS_{CLB} = [(TC_1, Seq_1), (TC_2, Seq_2), ..., (TC_{k+1}, Seq_{k+1})] \quad (1)$$

where TC_i is the ith configuration, TP is the bit patterns applied to TC_i, and k is the number of inputs of a LUT. For each test configuration, the number of TP is 2^k, so the
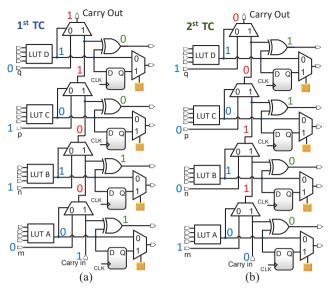

Figure 9: The test configurations applied to the carry chain for fault detection. (a) The first test configuration. (b) The second test configuration.

Table 3
Two test configurations for the carry chain

Label	Input (LUT)	1st (Carry In)	Output (MUX)	Output (XOR)	2st (Carry In)	Output (MUX)	Output (XOR)
MUX A	0		0	1		1	0
MUX B	1	1	0	1	0	1	0
MUX C	0		1	0		0	1
MUX D	1		1	0		0	1

the minimal length of complete input sequence Seq_{k+1} (the number of TPs per TC) applied for TC_i can be expressed as

$$|Seq_1| = |Seq_2| = \cdots = |Seq_{k+1}| = 2^k \quad (2)$$

Since an LUT can realize 2^n ($n = 2^k$) different functions, it is impractical to test each function exhaustively [34].

2) The proposed carry chain test: With the modern FPGA technology, a dedicated carry chain is embedded in every CLB. The carry chain comprises basically MUXs and XOR gates to compute both the carry-out and the sum bits, respectively. Fig. 9 shows a carry chain circuit and the associated LUTs [30]. The labels in Fig. 9 indicate how to configure the outputs of LUTs and the external terminals. The detailed testing configuration is shown in Table 3. At the first logic stage (i.e., the bottommost one), the MUX inputs are set to constant values '0' and '1', forcing it to propagate the output ('0') of LUT A to the next stage. So the output of the first MUX and XOR gate are constant values '0' and '1', respectively. Through the internal routing of the slice, the output of the first MUX is directly connected to the input of the second one.

Then, the two inputs of the second stage's MUX are now set to '1' and '0', respectively. As the selection signal ('1') is the output of LUT B, the second MUX is equivalent to calculating the XNOR function of the external signal ('1') and the signal ('0') from the first MUX. Note that at this moment MUXs in the carry chain are configured to perform XNOR operations. Hence, we can perform the computation of the XOR and XNOR functions of the entire carry chain.
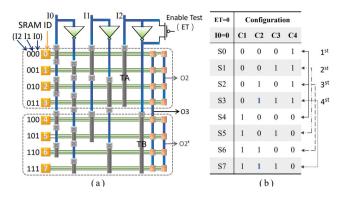
Figure 10: (a) The proposed improved design of LUT. (b) The traditional test configuration scheme [30].

All stuck-at faults within the carry chain can be detected by configuring the MUX and XOR accordingly.

3) The proposed improved design for a LUT: To speed up the testing application time for each configuration, an improved design based on an LUT in a CLB is proposed. As shown in Fig. 10 (a), a P-CNFET is placed on the right of the inverter connected with the signal I2. The inverter enters the normal mode when ET=1, and enters the test mode when ET=0 (The inverter is shorted by the placed P-CNFET).

In addition, contacts are placed after the pull-up and pull-down networks of the MUX, respectively. The MUX is divided into two networks, which can be tested in parallel, i.e., NW1 and NW2. For example, as shown in Fig. 10 (b), the paths corresponding to SRAM-0 and SRAM-3 can be tested simultaneously in the 1st group of test patterns, where the traditional eight test patterns are compressed into four patterns. However, when ET=0, I0=0, the pass transistors (TA and TB) corresponding to selected signal I0 are always on. Therefore, this technique cannot detect the stuck-on faults of these two gates. It needs to add a test pattern in configuration C2, i.e., (I2, I1, I0)=111. Because the values stored in SRAM-3 and SRAM-7 are all logic '1', if any O2 or O2' outputs a logic value '1' in this pattern, the transistor (TA/TB) is considered to have a stuck-on fault, and vice versa.

In summary, for one test session, the traditional method of testing each LUT requires k+1 test configurations with 2^k test patterns per configuration where k and 2^k represent the number of inputs and the configuration bits of a LUT, respectively. For example, as shown in Fig. 10 (b), a 3-input LUT with 8 configuration bits requires 4 configurations and 8 test patterns per configuration. But for the proposed improved design of LUT, each LUT requires only 2 configurations and 4 test patterns per configuration in this case, so the test time is shorter than that of the traditional method.

6. The Proposed Heuristic Algorithm

Step I: Fault Testing Technique for the Overall CLB Array. As shown in Fig. 11, the traditional fault testing of CLBs contains two sessions: 1) horizontal test, the outputs of the rightmost CLBs are compared with correct responses to identify faulty rows. 2) vertical test, the outputs of the bottom
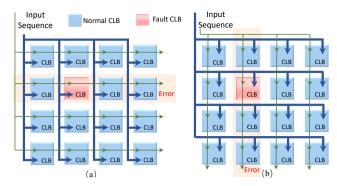

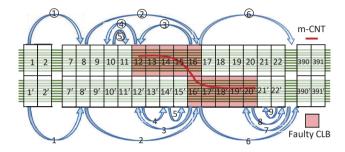**Figure 11:** The traditional method to diagnose the faulty CLBs [37].

Figure 12: An example illustrating recursive jump test.

CLBs are compared with correct responses to identify faulty columns. By intersecting faulty columns with faulty rows, the faulty cells can be identified [37].

However, the correlated fault CLBs induced by several hundred micrometers long m-CNT may span across several rows, resulting in correlated faults faults in these rows. The traditional test technique is only suitable for MOSFET-based FPGAs, and cannot test the cascaded faulty CLBs effectively. Note that diagnosing and locating the faulty CLBs of each row should be sensitized from the external IO ports [30].

In this work, we explore the unique property of these cascaded faults induced by m-CNTs, and propose to make the test "jump" over CLBs. This can reduce the test overhead effectively.

The idea of the recursive jump testing is to dynamically configure the direction and step size of each jump based on previous testing results until we locate both ends of the faulty segments. The algorithm is divided into two phases.

1) Initial phase: In this phase, we jump with an initial jump step size through the columns along the rows, and record each test response of the current test (see the first and second jump in Fig. 12). This phase aims to detect the faulty segment. Note that the jump step is always the same in the initial phase. Once the test response of the current CLB is different from the previous one, indicating that a faulty segment has been detected. Then, we enter the next phase denoted as the recursive phase (after the second jump).

2) Recursive phase: Note that the jump step in this phase is halved in each iteration. Once the test result of the jump is different from the previous one, the test jumps in the opposite

Algorithm 1 Recursive Test

Require: Jump direction *Dir*, jump length *Step*, direction flag *Key*, row *i* column *j* CLB $C_{i,j}$, recursive function Recursive($Dir, Step, Key, C_{i,j}$)
1: **if** $Step = 1$ **then**
2:      Return;
3: **else**
4:      $Step=Step/2$;
5:      $Key = (C_{i,j} \text{ xor } C_{i,j+Dir \times Step})$ and Key
6:      **if** $Key = 1$ **then**
7:          $Dir = -Dir$;
8:          Recursive($Dir,Step,Key,C_{i,j+Dir \times Step}$);
9:      **else**
10:         Recursive($Dir,Step,Key,C_{i,j+Dir \times Step}$);
11:     **end if**
12: **end if**

---

direction, e.g., the third test jumps backward in Fig. 12, but with only a half step size. If the test result of the third test is the same as the previous one, the test will jump forward again with the half jump step (see the fourth test). In our case, the test response of the fourth test is different from the third one, then we jump in the opposite direction and halve jump step size again (see the fifth test).

This phase continues until we cannot divide the jump step further (e.g., the fifth test has jump step 1). The starting point of the faulty segment can be located at the moment. Consequently, we quit the recursive phase and continue the initial phase to detect the endpoint of the faulty segment.

In summary, the pseudo-code of recursive testing algorithm is shown in Algorithm 1, where in the variable *Dir* indicates the jump direction, *Step* means the jump step size, *Key* is to judge whether the two test responses before and after each recursion are equal, $C_{i,j}$ is the location of the CLB.

Step II: Exploration of the Redundant Row Sharing Architecture. In this section, we firstly discuss the traditional method for repairing faulty segments in CNT-based FPGAs, and then explore the redundant architecture to repair faulty CNT-based FPGAs.

Previous studies have proposed a rich set of methods to deal with faults in CNFET circuits [28] [29]. What is currently missing is a way to increase the optimal amount of redundancy by adding redundant rows/columns and determining the appropriate sharing scheme to maximize yield while minimizing hardware overheads. According to the distribution of m-CNTs mentioned in Section 4, the m-CNT grows in the direction of the column and may also span different rows.

The traditional redundancy spare column/row repair scheme is only suitable for MOSFET-based FPGAs, and cannot effectively repair faulty tiles due to m-CNTs in the CNT-based FPGAs. In contrast to the discrete faults in CMOS-based RAM (which occur in a two-dimensional local area), the faulty segments caused by m-CNTs expand along the direction of CNT growth and may affect continuous column blocks. So in order to reduce fault tolerance overhead,
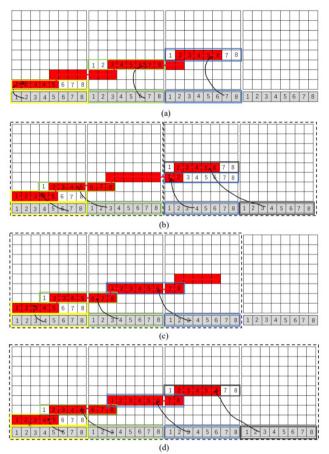

Figure 13: (a) One 8×8 tile shares one spare row. (b) Two 8×8 tiles share two spare rows. (c) Three 8×8 tiles share three spare rows. (d) Four 8×8 tiles share four spare rows.

we propose a redundant spare row sharing scheme to repair the faulty segments induced by m-CNTs. We divide total FPGA tiles into multiple small tiles and share alternative rows in adjacent tiles. The granularity of sharing is one spare row rather than with an entire tile, which can reduce hardware overhead significantly.

The basic concepts of the alternative row-sharing architecture are shown in Fig. 13. Small tiles of multiple adjacent columns are grouped together to form a tile group that can share redundant spare rows. Within a tile group, any faulty segment in a small tile can be replaced by any spare row, and each spare row can span two small tiles for fault tolerance. The redundant spare row sharing schemes considered in our work are given as follows:

- *Scheme 0:* Each 8×8 tile shares one spare row.
- *Scheme 1:* Two 8×8 tile shares two spare rows.
- *Scheme 2:* Two 8×8 tiles share three spare rows.
- *Scheme 3:* Three 8×8 tiles share three spare rows.
- *Scheme 4:* Three 8×8 tiles share four spare rows.
- *Scheme 5:* Four 8×8 tiles share four spare rows.
- *Scheme 6:* Four 8×8 tiles share five spare rows.
- *Scheme 7:* Five 8×8 tiles share four spare rows.

Table 4
CNT Parameters

Definition	Value
The number of CNTs for each CNFET: Mean-Nμ, Variation-Nσ	Nμ=4, Nσ=1 [30]
Probability of m-CNT: Pm	3%~33% [19]
Probability of removing m-CNT: Prm	99.99% [19]
Probability of removing s-CNT: Prs	5% [19]
The angle of CNTs: Mean-Aμ, Variation-Aσ	Aμ=0°, Aσ=10° [30]
The length of CNTs: Mean-Lμ, Variation-Lσ	Lμ=150μm, Lσ=3.33μm [31]

7. Experimental Results and Analysis

In this section, we firstly characterized the oscillation delay of a seven-stage RO array in a CNT-based FPGA. Then, the test configuration overhead of different scales of CNT-based FPGAs were evaluated. Moreover, we simulated the test application time for a single CNT-based CLB constructed by different input LUTs. In addition, the average test coverage, test overheads with different m-CNT distribution probabilities and initial jump sizes were evaluated. Finally, we evaluated the repair rate and hardware overheads under different spare row sharing architectures.

7.1. Experimental Setup

Process parameter settings of CNT in our experiments were the same as [35]. We generated a sample of CNT-based FPGA. Its structure is similar to Xilinx Virtex 7V2000T and consists of 391×391 FPGA tiles. For delay faults testing, the process variation of MWCNT was set according to [4].

To evaluate the effectiveness of our proposed CLB testing technique, we built a simulator with the layout information of the CNT-based FPGAs, and the parameter setting is shown in Table 4. We took the imperfect m-CNT removal process into consideration. The probability of m-CNTs is p_m, and p_{Rm} is m-CNT removal rate. The starting coordinates of m-CNT were randomly generated. The CNT length had a mean length and standard deviation of L_μ and L_σ. The misaligned m-CNTs were randomly generated based on Gaussian distribution with misaligned probability as p_{mis}. According to [21], for an architecture with the cluster size of 8, we estimated the footprint of a baseline CNT-based CLB to be 27698T where T denotes a minimum width transistor area, i.e. $2.2\times10^{-3}\mu m^2$ in 7nm technology node. Then, the area of a CLB can be estimated according to [35]. For a CLB composed of 4 six-input LUTs, m-CNTs in CNT bundles were randomly distributed based on the above parameter settings. We performed Monte-Carlo simulations to generate 1000 basic samples of the CNT-based FPGAs. After applying the recursive jump test with different jump steps and different misaligned angles of m-CNTs in these samples, we can derive the corresponding fault maps. Then, we compared the recursive jump testing with single-step and fixed-step testing schemes in terms of test coverage and test overhead. Finally, to evaluate the effectiveness of the redundant spare row sharing architecture, we evaluated the
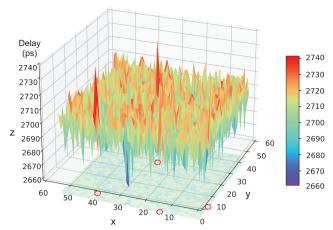

Figure 14: The observed oscillation delay for each 7-stage oscillator in a CNT-based FPGA.

repair rate and hardware overhead for scheme 0~7. Two metrics used in comparisons are defined as follows:

Repair Ratio: The percentage of faulty segments repaired by redundant rows.

Spare segment overhead: The average number of redundant rows allocated per 8×8 tiles.

7.1.1. The Delay Fault Testing of MWCNT Interconnects

In this experiment, each RO had seven-stages and was placed in a CLB. ROs were mapped to the LUT by the XNOR configuration mentioned in Section 3. Related work shows that it is feasible to measure the process variation or aging by mapping RO into FPGA [36]. In this work, the ROs were used to test the delay fault of MWCNT interconnects. To avoid the measurement noise, each frequency was measured three times and the average value was used.

Fig. 14 shows the observed oscillation delay in one test configuration. The mean of oscillation delay was 2.70ns, and the variation was 100ps. Although the total range of variation is reasonably small, there are still a few ROs with large loop delays, which seriously affect the performance of a CNT-based FPGA operating at hundreds of MHz. So with ROs delay testing technique, the timing delay fault in MWCNT can be detected effectively.

These results demonstrate the suitability of RO-based testing for capturing the fine-grained delay variations unique to MWCNT interconnects, which traditional delay fault tests often overlook.

7.1.2. Testing Overhead for m-CNT Faults in CLBs

We evaluated the test overhead for different CLB array sizes, and the length of configuration bitstream and clock were set according to [30].

Since the testing of carry chain was included, the test scheme proposed in this article requires $K + 3$ test configurations on the basis of the traditional $K+1$ test configuration, where K is the number of input ports in a LUT. As shown in Fig. 15, the test configuration overhead of the proposed technique (including the carry chain test) is slightly higher
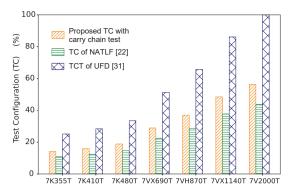
Figure 15: Comparisons of the number of test sessions of different Xilinx-7 devices.
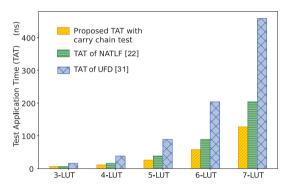

Figure 16: Simulation results for the test time of different CNT-based LUTs.

than the traditional NATLF method without carry chain test (increased by an average of 8.1%) [34], but far less than the traditional UFD method [37]. In summary, adding the test of carry chain in the CLB only incurs little test overhead.

Then, we applied the technique mentioned in Section 4 to a single CNT-based CLB constructed by different input LUTs. We evaluated the test time in one test session by SPICE simulation. As shown in Fig. 16, the test time (including the carry chain test) of the proposed technique is less than the other two traditional methods without carry chain test (i.e., NATLF [34], UFD [37]), and the test time decreases more significantly with the increase of LUT input ports. Compared with the NATLF [34], the test time decreased by 28.77% on average. For the general 6-input LUT, the test time can be reduced by 35.49%.

The proposed RO-based test with carry chain addition requires only marginally more configurations than NATLF, but achieves more comprehensive fault coverage. Therefore, it offers a better coverage-to-cost trade-off than both NATLF and UFD.

7.2. Experimental Results and Discussion
7.2.1. Evaluations of Cascaded Faulty CLB Segment Testing

There are two major methods for FPGA testing: application-independent testing and application-dependent testing. For application-independent testing, since the actual user configuration is unknown during testing, so all FPGA logic units will be tested. The application-dependent testing only partially tests the circuit actually used [38].

Table 5
Comparison of test methods

Method	Fault Coverage	Test Overhead
Single	Baseline (100%)	Baseline (100%)
Fixed	Mid (45.4–78.2%)	Low (2.9–31.55%)
Recursive	High (82.9–100%)	Mid (64.2–69.64%)

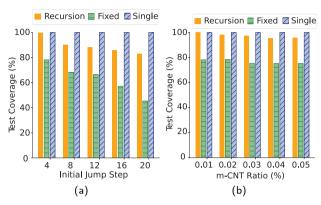**Note: Recursive test starts with jump = 4.**

Figure 17: Simulation results on test converage with varying (a) jump step, (b) m-CNT ratio.
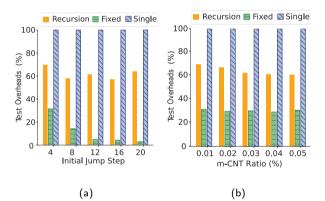

Figure 18: Simulation results on test overheads with varying (a) jump step, (b) m-CNT ratio.

For application-independent testing, we compared the recursive testing with the fixed-step jump testing and single-step testing. The m-CNT probability was assumed to be 0.01%, and the default size of initial jump step size was set to 4. We evaluated the average test coverage obtained by varying initial jump step size and m-CNT ratio. As shown in Fig. 17(a), the proposed jump tests with step size 4 show 100% test coverage. The fixed-step jump testing, however, results in lower test coverage as the initial jump step increases. This is because it has a higher chance to jump over the start/end point of the faulty CLB segment. As shown in Fig. 17(b), the recursive test provides higher test coverage than the fixed-step jump test as m-CNT probability increases. The average test coverage of recursive test is 96.58%, which is much higher than fixed-step jump test (63.80%). Therefore, the recursive jump method strikes the best balance.

Fig. 18 presents the test overheads, which are normalized to the overheads of single-step tests. As expected, the test

Table 6
Application-dependent test

Benchmark	Test Coverage (%)			Test Overhead (%)		
	Recursion	Fixed	Single	Recursion	Fixed	Single
PCI	91	71	100	33	19	100
I2C	84	63	100	31	16	100
SPI	93	69	100	47	23	100
FIR	79	60	100	36	27	100
FPU	82	66	100	44	26	100
VGA	92	70	100	49	29	100
PCM	81	73	100	61	21	100
DMA	87	59	100	54	19	100
USB	82	64	100	32	15	100
MEM	86	61	100	29	16	100

overhead of the recursive test is higher than that of the fixed-step jump test. This is because more jumps are required to guarantee high fault coverage, and the test overhead of the recursive testing slightly increases with the increase of m-CNT probability. Compared with the single-step testing, the test overhead of recursive testing can be reduced by 35.78% on average.

In particular, Fig. 18(a) reveals an interesting phenomenon: the test overhead with jump step size 12 is higher than that with jump step size 8. This is because when the recursive step is reduced to 3 or 5, the algorithm increases the last jump step size by 1 and divides it by 2. For example, for the initial jump step 8, the recursive jump test would use jump steps 8, 4, 2, and 1, while for the initial jump step size 12, recursive jump test would use jump steps 12, 6, 3, 2 and 1, which brings higher test overheads. Similar results can be observed with the initial jump step size 20. Based on this observation, we suggest set the initial test step size as four. Experimental results show that the proposed recursive testing can achieve a high test coverage with low test overhead.

For application-dependent test, we also used the recursive jump test method, and the initial jump step size is four and the m-CNT probability is 0.05%. At the beginning, we chose 10 different benchmarks and imported them into Vivado. Then the benchmarks were synthesized to generate gate netlists, and programmable logic blocks used in the FPGA array were identified. We used recursive testing, fixed-step testing and single-step testing, respectively. Test coverages and test overheads with different benchmarks are shown in Table 6. We can observe that the test coverage of the recursive testing is higher than the fixed-step jump test while the test overhead of the recursive testing is lower than the single-step testing, which validates the effectiveness of our proposed recursive testing technique.

As summarized in Table 5, the recursive jump test consistently achieves high coverage (82.9–100%) with moderate overhead. This observation is further validated in Table 6, demonstrating recursive jump practicality in real-world application scenarios.

Table 7
Comparison of spare row sharing schemes

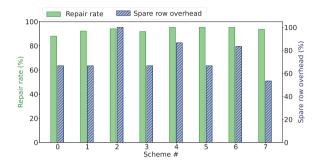| Scheme	Repair Rate	Overhead
Scheme 0	Low(92.4%)	Mid(66.7%)
Scheme 2	High(98.8%)	Baseline(100%)
Scheme 5	High(100%)	Mid(66.7%)
Scheme 7	High(98.4%)	Low(53.3%)

Figure 19: Repair rate and spare row overhead for redundancy architecture.

7.2.2. Testing Overhead and Effectiveness of Redundant Sharing Architecture

Fig. 19 shows the repair rate and hardware overhead of the proposed spare row sharing architecture with different sharing strategies. From the experimental results, we can see that scheme 5: four 8×8 tile sharing four spare rows, has the highest repair rate and less hardware overhead than most of other schemes.

As the number of sharing units increases, circuit area, latency, and power consumption also increase significantly [29], so there should not be too many sharing units. Looking at scheme 7: five 8×8 tiles sharing four spare rows. The repair rate loses 1.2%, which is negligible compared with scheme 5. However, the hardware overhead of scheme 7 is significantly reduced by 13.34%~46.67% compared with other schemes.

As shown in Table 7, Scheme 7 achieves a high repair rate (98.4%) comparable to the best-performing configuration, while reducing hardware redundancy by over 46.7%. This makes it the most efficient and scalable solution among the evaluated schemes.

7.2.3. Fault Injection and Detection Coverage Evaluation

To validate the fault detection capability of the proposed testing framework, we conducted a logic-level fault injection experiment on CNT-based FPGA tile.

Faults were injected directly at the LUT level, where each 6-input LUT was assigned one of three representative fault models:

○ *stuck-at-0*: The LUT output is permanently forced to logic 0, regardless of input selection.

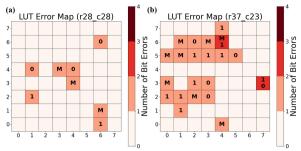○ *stuck-at-1*: The LUT output is permanently forced to logic 1, regardless of input selection.

Figure 20: Visualization of injected logic faults in LUTs. (a) Randomly distributed fault pattern involving all three fault types. (b) Clustered fault region reflecting correlated CNT-induced defects.

Note: "0" indicates stuck-at-0 fault; "1" indicates stuck-at-1 fault; "M" indicates mux override fault.

Table 8
Detected Fault Count and Coverage by Method

Fault Type	Recursion	Fixed	Single
MUX	**13126**	8634	14693
Stuck-at-0	**19651**	13030	22121
Stuck-at-1	**13115**	8843	14764
Total	**45892 (89.0%)**	30407 (58.9%)	51578 (100%)

○ *mux override*: The internal address decoding logic is overridden, causing the output to be fixed to a specific configuration bit regardless of the intended *sel* input.

Fig. 20 presents the spatial distribution of injected faults in a single 8×8 tile block. Each tile contains four LUTs, and the injected faults are marked according to their type. In Fig. 20(a), faults of all three types appear scattered across the block, illustrating the diversity and randomness of possible CNT-induced logic errors. In contrast, Fig. 20(b) exhibits a correlated defect region where adjacent tiles exhibit clusters of faults, which is consistent with physical CNT alignment anomalies and bundle-induced disruptions.

For fault detection, we adopted the recursive jump testing strategy at the *tile level*. That is, test patterns were generated and applied to tiles as atomic units. Internally, to accelerate fault localization within a tile, we leveraged the LUT-level configuration-aware structure introduced in Section 5. This structure enables fast lookup of erroneous LUTs by checking aggregated tile responses. The detection process was applied across all 49×49 simulated blocks, each consisting of 8×8 tiles, thereby covering the full 391×391 tile array. A fault was considered detected if any of the four LUTs within a tile produced an output mismatch under recursive test stimuli. To evaluate effectiveness, we compared the detection performance of the recursive, fixed-step, and ideal single-step test strategies across the entire array.

As summarized in Table 8, the proposed recursive method detects over 88% of injected LUT faults across all types, significantly outperforming fixed-step testing and closely approaching the ideal fault detection coverage achieved by exhaustive single-step tests.

8. Conclusion

CNT-based FPGA is a promising alternative to conventional CMOS-based FPGA. However, due to the imperfect fabrication process of CNTs, CNT-based FPGA may exhibit unique faulty patterns. With the help of an advanced ring oscillator design, CLBs can be connected in series to form a ring oscillator, which can be used to effectively detect the delay fault of MWCNT interconnects. Furthermore, for the faulty CLBs induced by m-CNTs, we propose a carry chain test method based on the traditional test method, and a new technique is proposed to speed up the fault testing. Finally, considering the faulty segments induced by m-CNTs, we propose a fault tolerant architecture by sharing the spare rows to repair the faulty segments, which can improve the repair rate and reduce the hardware overhead effectively.

Acknowledgment

References

[1] X. Chen, L. Yin, B. Liu and Y. Han, "Merging Everything (ME): A Unified FPGA Architecture Based on Logic-in-Memory Techniques," 2019 56th ACM/IEEE Design Automation Conference (DAC), Las Vegas, NV, USA, 2019, pp. 1-2.

[2] Kangwei Xu et al., "HLSRewriter: Efficient Refactoring and Optimization of C/C++ Code with LLMs for High-Level Synthesis," ACM TODAES, 2024.

[3] X. Chen, K. Ni, M. T. Niemier, Y. Han, S. Datta, and X. S. Hu, "Power and Area Efficient FPGA Building Blocks Based on Ferroelectric FETs," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 66, no. 5, pp. 1780-1793, 2019.

[4] R. Chen, J. Liang, J. Lee, V. P. Georgiev, R. Ramos, H. Okuno, D. Kalita, Y. Cheng, L. Zhang, R. R. Pandey, S. Amoroso, C. Millar, A. Asenov, J. Dijon, and A. Todri-Sanial, "Variability Study of MWCNT Local Interconnects Considering Defects and Contact Resistances—Part I: Pristine MWCNT," IEEE Transactions on Electron Devices, vol. 65, no. 11, pp. 4955-4962, 2018.

[5] Nishant Patil, Albert Lin, Jie Zhang, H-S Philip Wong, and Subhasish Mitra. Digital VLSI logic technology using carbon nanotube FETs: Frequently asked questions. In Proceedings of the 46th Annual Design Automation Conference, pages 304–309. ACM, 2009.

[6] R. Chen, et al.,"Carbon Nanotube SRAM in 5-nm Technology Node Design, Optimization, and Performance Evaluation—Part I: CNFET Transistor Optimization" in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol.30, no.04, pp. 432-439, 2022.

[7] R. Chen, et al.,"Carbon Nanotube SRAM in 5-nm Technology Node Design, Optimization, and Performance Evaluation—Part II: CNT Interconnect Optimization" in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 30, no. 04, pp. 440-448, 2022.

[8] C. Dong, S. Chilstedt, and D. Chen, "FPCNA: A Carbon Nanotube-Based Programmable Architecture," Nanoelectronic Circuit Design, N. K. Jha and D. Chen, eds., pp. 307-348, New York, NY: Springer New York, 2011.

[9] Aly M M S, Gao M, Hills G, et al. Energy-efficient abundant-data computing: The N3XT 1,000 x[J]. Computer, 48(12): 24-33 ,2015.

[10] Payman Zarkesh-Ha and Ali Arabi M. Shahi. Stochastic analysis and design guidelines for CNFETs in gigascale integrated systems. IEEE Transactions on Electron Devices, 58(2):530–539, 2011.

[11] V. R. Kumbhare, P. P. Paltani, C. Venkataiah and M. K. Majumder, "Analytical Study of Bundled MWCNT and Edged MLGNR Interconnects: Impact on Propagation Delay and Area," in IEEE Transactions on Nanotechnology, vol. 18, pp. 606-610, 2019.

[12] Kangwei Xu, Dongrong Zhang, Qiang Ren, Yuanqing Cheng, and Patrick Girard, "All-spin PUF: An Area-efficient and Reliable PUF Design with Signature Improvement for Spin-transfer Torque Magnetic Cell-based All-spin Circuits," ACM Journal on Emerging Technologies in Computing Systems (JETC), 2022.

[13] E. Chmelar. FPGA interconnect delay fault testing. In Proc. Int. Test Conf. (ITC), pages 1239–1247, 2003.

[14] M. Abramovici and C. Stroud. BIST-based delay-fault testing in FPGAs. In Proc. IEEE Int. On-Line Testing Workshop, pages 131–134, July 2002.

[15] X. Y . Li, F. Wang, T. La, and Z.-M. Ling, "FPGA as process monitor–an effective method to characterize poly gate CD variation and its impact on product performance and yield," IEEE Trans. Semiconduct. Manufact., vol. 17, no. 3, pp. 267–272, Aug. 2004.

[16] K. Xu et al., "LLM-Aided Efficient Hardware Design Automation," arXiv:2410.18582.

[17] Chandan Kumar Jha et al., "Large Language Models for Verification, Testing, and Design," ETS, 2025.

[18] S. C. Chowdhury, B. Z. Haque, T. Okabe, and J. W. J. C. P. B. E. Gillespie, "Modeling the effect of statistical variations in length and diameter of randomly oriented CNTs on the properties of CNT reinforced nanocomposites," vol. 43, no. 4, pp. 1756–1762, 2012.

[19] K. Xu et al., "Fault Testing and Diagnosis Techniques for Carbon Nanotube-Based FPGAs," IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC), 2022.

[20] J. Deng, et al., "Carbon Nanotube Transistor Circuits: Circuit-Level Performance Benchmarking and Design Options for Living with Imperfections," International Solid-State Circuits Conference, 2007.

[21] Betz, Vaughn, Rose, Jonathan, Marquardt, Alexander, "Architecture and CAD for deep-submicron FPGAs," Springer US, 1999.

[22] Kangwei Xu et al., "Logic Design of Neural Networks for High-Throughput and Low-Power Applications," IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC), 2024.

[23] Kangwei Xu et al., "Automated C/C++ Program Repair for High-Level Synthesis via Large Language Models," MLCAD, 2024.

[24] T. Inoue, S. Miyazaki, and H. Fujiwara, "Universal fault diagnosis for lookup table FPGAs," IEEE Design & Test of Computers, vol. 15, no. 1, pp. 39-44, 1998.

[25] N. Patil et al., "VMR: VLSI-compatible metallic carbon nanotube removal for imperfection-immune cascaded multi-stage digital logic circuits using carbon nanotube FETs," in Proc. IEEE Int. Electron Devices Meeting (IEDM), Baltimore, MD, USA, 2009, pp. 1–4.

[26] R. Krupke, F. Hennrich, H. V . Löhneysen, and M. M. Kappes, "Separation of metallic from semiconducting single-walled carbon nanotubes," Science, vol. 301, no. 5631, pp. 344–347, 2003.

[27] G. Zhang et al., "Selective etching of metallic carbon nanotubes by gas-phase reaction," Science, vol. 314, no. 5801, pp. 974–977, 2006.

[28] Habiby, and Asli. "Design and implementation of a new symmetric Built-in Redundancy analyzer." Csi International Symposium on Computer Architecture & Digital Systems IEEE, 2012.

[29] Li, T. , et al. "Defect tolerance for CNFET-based SRAMs." 2016 IEEE International Test Conference (ITC) IEEE, 2016.

[30] 7 Series FPGAs Configurable Logic Block User Guide. Accessed: 2020. [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug474_7Series_CLB.pdf

[31] C.-S. Lee, E. Pop, A. D. Franklin, W. Haensch, and H.-S. Wong, "A compact virtual-source model for carbon nanotube fets in the sub-10-nm regimepart i: Intrinsic elements," IEEE transactions on electron devices, vol. 62, no. 9, pp. 3061–3069, 2015.

[32] Kangwei Xu et al., "HLSTester: Efficient Testing of Behavioral Discrepancies with LLMs for High-Level Synthesis," ICCAD, 2025.

[33] W. K. Huang, M. Y . Zhang, F. J. Meyer, and F. Lombardi, "A XOR-tree based technique for constant testability of configurable FPGAs," in Proc. 6th Asian Test Symp. (ATS), pp. 248–253, 1997.

[34] S.-K. Lu, F.-M. Yeh, and J.-S. Shih, "Fault Detection and Fault Diagnosis Techniques for Lookup Table FPGAs," VLSI Design, vol. 15, pp. 397-406, 2002.

[35] J. Luo, L. Wei, C. Lee, A. D. Franklin, X. Guan, E. Pop, D. A. Antoniadis, and H. P. Wong, "Compact Model for Carbon Nanotube Field-Effect Transistors Including Nonidealities and Calibrated With Experimental Data Down to 9-nm Gate Length," IEEE Transactions on Electron Devices, vol. 60, no. 6, pp. 1834-1843, 2013.

[36] X.-Y . Li, F. Wang, T. La, and Z.-M. Ling, "FPGA as process monitor–an effective method to characterize poly gate CD variation and its impact on product performance and yield," IEEE Trans. Semiconduct. Manufact., vol. 17, no. 3, pp. 267–272, Aug. 2004.

[37] T. Inoue, S. Miyazaki, and H. Fujiwara, "Universal fault diagnosis for lookup table FPGAs," IEEE Design & Test of Computers, vol. 15, no. 1, pp. 39-44, 1998.

[38] Rozkovec, M. , J Jeníček, and O Novák. "Application Dependent FPGA Testing Method." Euromicro Conference on Digital System Design: Architectures IEEE Computer Society, 2010.