Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Effective fault detection in 3D ICs: cluster-based BIST for enhanced inter-layer via fault coverage

Hadi JAHANIRAD‡[1], Ahmad MENBARI[2], Hemin RAHIMI[1], Daniel ZIENER[2]

[1]*Department of Electronics and Communication Engineering, University of Kurdistan, Kurdistan 90210, Iran*

[2]*Department of Computer Architecture and Embedded Systems, Ilmenau University of Technology, Ilmenau 98693, Germany*

E-mail: h.jahanirad@uok.ac.ir; ahmad.menbari@tu-ilmenau.de; hemn.rahimi@uok.ac.ir; daniel.ziener@tu-ilmenau.de

Received Dec. 26, 2024; Revision accepted June 24, 2025; Crosschecked

Unedited

Abstract: Monolithic three-dimensional integrated circuits (M3D ICs) have emerged as an innovative solution to overcome the limitations of traditional 2D scaling, offering improved performance, reduced power consumption, and enhanced functionality. inter-layer vias (ILVs), a crucial component of M3D ICs, provide vertical connectivity between layers but are susceptible to manufacturing and operational defects, such as stuck-at faults, shorts, and opens, which can compromise system reliability. These challenges necessitate advanced built-in self-test (BIST) methodologies to ensure robust fault detection and localization while minimizing the testing overhead. In this paper, we introduce a novel BIST architecture tailored to efficiently detect ILV defects, particularly in irregularly positioned ILVs, and approximately localize them within clusters, using a walking pattern approach. In the proposed BIST framework, ILVs are grouped according to the probability of fault occurrence, enabling efficient detection of all stuck-at faults and bridging faults, and most multiple faults within each cluster. This strategy empowers designers to fine-tune fault coverage, localization precision, and test duration to meet specific design requirements. The new BIST method addresses a critical shortcoming of existing solutions by significantly reducing the number of test configurations and overall test time using multiple ILV clusters. The method also enhances efficiency in terms of area and hardware utilization, particularly for larger circuit benchmarks. For instance, in the LU32PEENG benchmark, where ILVs are divided into 64 clusters, the power, area, and hardware overheads are minimized to 0.82%, 1.03%, and 1.14%, respectively.

Key words: 3D monolithic integrated circuits; Inter-layer vias (ILVs); Built-in self-test (BIST); Fault detection and localization
https://doi.org/10.1631/FITEE.2401094 **CLC number:**

1 Introduction

The demand for high-performance computing (HPC) is rapidly increasing due to technological advancements (Garcia et al., 2024). Extensive computing power is essential for efficiently processing and analyzing large datasets. Consequently, there is a growing need for high-performance processors to support the large-scale computational requirements of high-demand applications ranging from AI-based systems and edge computing to IoT devices (Zhou et al., 2023). However, due to physical limitations, transistor shrinking is reaching its limit, leading to the end of Moore's Law. Monolithic three-dimensional integrated circuits (M3D ICs) have emerged as an innovative solution to overcome the limitations of traditional 2D scaling, offering improved performance, reduced power consumption, and enhanced functionality (Kim and Park, 2024). High bandwidth memory (HBM) is a prime example and one of the first technologies to use this approach to increase performance (Huang et al., 2024).

One of the key features of M3D ICs is the use of vertical vias, known as inter-layer vias (ILVs). These vias are essential for establishing vertical connections

‡ Corresponding author

© Zhejiang University Press 2025

between different layers, facilitating the integration of heterogeneous technologies, and enabling efficient routing of signals and power. However, while ILVs offer significant advantages, they can also pose a critical vulnerability, potentially affecting system reliability.

The fabrication of ILVs presents several challenges, particularly regarding defects that may arise during manufacturing or operation. Issues such as missing or misaligned vias, poor metal fill, and the formation of voids can degrade signal integrity and compromise overall system performance. Therefore, the detection of these defects is crucial to maintaining the functionality and reliability of M3D ICs (Chen et al., 2024).

Traditional testing methods often fall short in detecting ILV defects due to their vertical orientation and high aspect ratio, necessitating the development of new, non-destructive, and efficient testing methods. To meet these demands, built-in self-test (BIST) has emerged as a promising solution. BIST offers an efficient, low-cost, and non-destructive approach to IC testing, making it particularly suitable for ILVs in M3D ICs. By providing targeted and efficient testing of vertical connections, BIST reduces the time and costs associated with traditional testing methods while ensuring the quality and reliability of M3D ICs. As M3D IC technology continues to evolve, advancing ILV testing methodologies remains a critical area of research and development (Pentapati et al., 2024).

While the BIST methods proposed by Chaudhuri et al. (2019 and 2022) provide a promising approach for testing ILVs in M3D ICs, their assumption of a 1-D arrangement of ILVs in a bus with adjacent left and right ILVs is a significant limitation. In reality, during the automatic algorithm-driven place-and-route process used by commercial tools, the ILVs may not be arranged in a 1-D array. This can lead to problems with the testing process, as the ILVs may be connected in a way that makes it difficult to detect faults or defects using the proposed BIST methods. Therefore, it is important to develop testing techniques that can effectively handle the complex 3D arrangements of ILVs in modern IC designs.

Chaudhuri et al. (2023) presented a BIST framework that can detect and localize faults in irregularly placed ILVs.

The framework identifies stuck-at faults (SAFs), multiple faults, shorts, and opens in both up-going and down-going ILVs. The proposed method uses a shared-BIST architecture, optimizing the area and power overhead by enabling multiple faults to use a BIST engine. However, the number of test iterations required to effectively test ILVs increases significantly as the number of ILVs increases. This issue leads to increased testing time, particularly for ICs containing a large number of ILVs.

Here, we present a novel BIST design that efficiently detects SAFs, multiple faults, and bridging faults using a walking pattern approach. Testing a chain of ILVs in a single cluster results in detecting all bridging faults and almost all potential multiple faults simply in a few test configurations. To effectively perform testing on irregularly positioned ILVs using this methodology, two main stages are involved. In the initial stage, an optimization algorithm is introduced to group ILVs into clusters, facilitating streamlined testing procedures. Subsequently, the ILVs within the same cluster are interconnected sequentially to transmit a binary 0/1 value from the input. The resulting output is then analyzed to identify any potential faults.

The main contributions of this paper are as follows:

1. A BIST design is proposed to detect SAFs, bridging faults, and multiple faults in irregularly placed ILVs which simplifies the testing process and allows for more efficient and accurate detection of faults.
2. The proposed algorithm allows designers to cluster ILVs into multiple groups to optimize testing time, approximately localize faults within clusters, and achieve a fault coverage percentage based on their requirements.
3. The proposed approach enables efficient and accurate detection of almost all multiple faults.
4. The proposed BIST approach can detect all bridging faults in a single cluster and all bridging faults with a high probability of occurring in multiple clusters.
5. The proposed BIST design can minimize test time by testing a very limited number of ILVs in each cluster.

The remainder of this paper is structured as follows. Section 2 provides an overview of M3D technology, fault models associated with ILVs, and a brief literature review. Section 3 presents the proposed BIST structure for detecting different types of faults and introduces the

Unedited

algorithm for clustering ILVs into several groups to reduce test time. In Section 4, the proposed method is evaluated using different experiments. Finally, the paper concludes in Section 5.

2 Background

2.1 Manufacturing and routing of M3D ICs

To fabricate M3D ICs, the process begins with integrating transistors and their interconnects in the bottom tier using a standard high-temperature process. A thin insulating layer is then created over the bottom tier's metal stack, followed by low-temperature molecular bonding of the silicon-on-insulator (SOI) substrate to obtain the top tier's transistors. ILVs are fabricated to connect the metal stacks of the top and bottom tiers. This procedure is repeated for any additional tiers that need to be fabricated (Yonehara, 2015; Batude et al., 2015). M3D routing optimizes ILV placements based on wirelength minimization and obtaining timing closure, requiring more complex strategies due to their 3D nature (Park et al., 2020). Pseudo-3D CADs extend commercial 2-D place-and-route tools for M3D designs, adopting either partitioning-first or partitioning-last strategies, which partition a 2-D netlist before or after commercial 2-D tool-driven placement, respectively (Dang et al., 2019; Park et al., 2020).

The resulting ILV locations in the layout depend on the pin locations of standard cells in different tiers, cut locations during tier-partitioning, and the 3D routing procedure, along with associated power, performance, and area (PPA) optimization objectives. These factors lead to potential fault sites, especially shorts, increasing drastically and resulting in a higher area overhead for on-chip test and localization methods. Consequently, a low-cost BIST framework is needed to detect and localize faults for realistic ILV placements (Chang et al., 2017). The tier-partitioning algorithm determines the ILV count, with the number of cuts made to divide a netlist graph into two partitions equating to the number of ILVs (Park et al., 2020). Typically, a min-cut algorithm is used to limit the number of ILVs, reducing the likelihood of ILV faults causing yield loss. After logic cells are partitioned and placed in different tiers, the physical placement of ILVs occurs during the global 3D routing procedure, with ILVs positioned closer to their driving logic gates to reduce timing delay (Park et al., 2020; Chang et al., 2017).

2.2 ILV fault model

The typical fault models for ILVs include shorts, opens, and SAFs. These faults can be classified into hard and resistive categories based on the underlying defects that cause them. Hard shorts in ILVs can occur due to imperfections in the design rules during circuit layout or particle contamination during fabrication. Resistive shorts, on the other hand, can occur when the ILV metal diffuses through the insulating layer to make partial contact with another ILV nearby, or due to defects at the bonding interface between two tiers. In the case of hard opens, a gap exists between the bottom end of an ILV and its landing pad. A resistive open typically occurs due to bonding defects, hairline cracks, and pinhole defects (Koneru et al., 2016; Campregher et al., 2005; Gefken et al., 1999).

2.3 Related studies

Recent studies have highlighted the challenges associated with testing ILVs in M3D ICs due to the high ILV integration density, which introduces a significant overhead for conventional interconnect BIST approaches (Pendurkar et al., 2001). Retrofitting these approaches requires dedicated scan elements and large test application times, and ATPG-based interconnect test methods are less effective due to limited I/O pins on one layer in an M3D IC (Rajski et al., 2013). Multiple faults are likely for dense ILV layouts, leading to test escapes under the single-fault assumption. Moreover, the current test standard for 3D-stacked ICs lacks on-chip ILV-fault localization capabilities (Jutman, 2004).

To address these challenges, researchers have proposed several ILV testing solutions. Some methods, such as those by Thuries et al. (2020), require die-wrapper register cells on both ends of the ILV, but this adds a high area overhead and reduces diagnostic accuracy. Other solutions, like those of Cron and Marinissen (2021), use interface scan cells and a twisted ring counter, which need a dedicated test layer, increasing the fabrication steps and area overhead. The BIST architecture of Thuries et al. (2020) cannot localize shorts between up-going and down-going ILVs and assumes a 1-D ILV arrangement, limiting its applicability to realistic 2-D or irregular ILV layouts after PPA-optimized place-and-route. The post-bond TSV testing methods of Lee et al. (2019) and Mok et al. (2021) detect resistive defects using response compaction but do not support on-chip fault localization.

Unedited

Recent advancements in M3D ICs have influenced design automation and DFT methodologies. Chaudhuri et al. (2020) discuss how M3D technology leverages ILVs for higher transistor density and greater design flexibility than TSV-based architectures, covering design and test techniques for M3D-enabled systems, and addressing challenges like defect tolerance and thermal management. Kim et al. (2020) introduced an RTL-to-GDS design flow for M3D ICs using carbon nanotube field-effect transistors and resistive memory, integrating 2D tools with M3D extensions for post-route optimization to enhance PPA. Their approach includes a low-overhead BIST module for ILVs and logic circuitry, demonstrating significant improvements in power, wirelength, and area with the RISC-V Rocketcore benchmark while maintaining acceptable power and thermal integrity.

3 Proposed method

This section introduces the proposed BIST method, which targets the detection of various hard faults within chains of ILVs. In this context, a chain refers to a serial connection of ILVs with one input and one output (Fig. 1a). To test each chain, we use a simple yet effective technique referred to as a walking pattern. This involves injecting a known bit (e.g., 0 or 1) at the input ILV and propagating it through the chain. Following this, the efficiency of the method in detecting weak faults will be discussed. Subsequently, the procedure for clustering ILVs into multiple chains is elaborated, along with the presentation of an efficient clustering algorithm.

3.1 Hard fault detection

This subsection outlines our proposed method for detecting hard faults in ILVs. In our approach to testing ILVs, we connect them in a serial chain and transmit a binary value 0/1 from the first ILV. We then analyze the output bit from the final ILV in the chain to determine whether the ILVs have passed or failed the test.

To enhance clarity, we demonstrate this method using four ILVs and establish that the approach remains the same when analyzing more ILVs. The proposed method for detecting SAFs, bridging faults, and multiple faults is detailed in the following subsections.

To clarify the models of faults we detect, stuck-at faults are identified by an ILV consistently maintaining logic 0 (S/0) or logic 1 (S/1). Bridging faults, on the other hand, occur when two ILVs are shorted. If both ILVs have the same values, their states remain unchanged. Conversely, if they have different values, one ILV dominates the other. Therefore, our methodology inherently considers and addresses all potential hard fault models for bridging faults, including wired-AND and wired-OR scenarios, by leveraging the behavior of dominant ILVs and accounting for various fault conditions. By considering all scenarios of dominant behavior for shorted ILVs, we implicitly cover all fault models of bridging faults.

Multiple faults arise when at least two faults occur simultaneously within an ILV array. Our method comprehensively detects multiple faults across all possible scenarios, ensuring implicit detection of all hard fault models. For instance, in scenarios where a stuck-at fault is bypassed by a bridging fault, we consider S/0 or S/1 for stuck-at faults and all scenarios explained earlier for bridging faults, ensuring that we detect them all.

3.1.1 Bridging faults

First, we explain the suggested approach for detecting bridging faults, as we will subsequently demonstrate that detecting bridging faults can effectively identify all SAFs and multiple faults. In Fig. 1a, we present a chain of four ILVs connected in a serial fashion. Each ILV is linked to a switch box (SB), which controls the system's transition from normal mode to test mode. The output from each ILV can either directly connect to the next ILV in the chain or pass through a NOT gate using another SB.

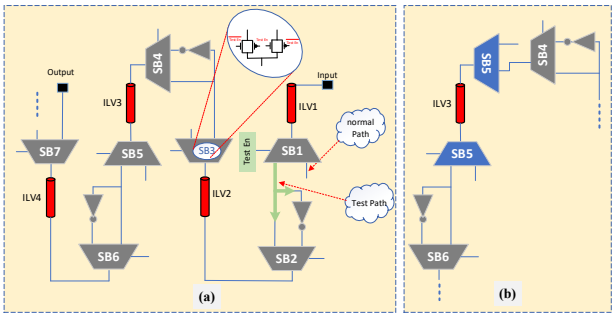


Fig. 1 (a) Proposed BIST structure for testing ILVs in a chain. (b) Signal path can flow either upward or downward through the ILV

The final ILV in the chain is connected to the output port, which indicates whether the ILVs pass or fail the test. In this structure, the SB functions as a transmission gate to accommodate the bidirectional flow of signals in an ILV. This is because the signal

Unedited

path can flow both upward and downward through the ILV (Fig. 1b). To maintain simplicity and clarity, we have not shown this in Fig. 1a. Note that the ILVs are positioned irregularly within the tier, resulting in different distances between ILV_i and ILV_j compared to ILV_i and ILV_k (Fig. 2).

Considering Fig. 1, assume that each ILV is connected to the next through a NOT gate, and the IC is in test mode. By applying a bit 0 from ILV_1, bridging faults between ILV_i and ILV_{i+1} can be detected. For instance, if there is a resistive fault between ILV_2 and ILV_3, then the value of ILV_3 (0) is influenced by the value of ILV_2 (1), causing the output to be 0, which is distinct from the fault-free IC. Similarly, it can be easily demonstrated that bridging faults between ILV_i and ILV_{i+3} can also be detected in this scenario. For example, in the absence of faults, if a bit 0 is transmitted through the chain, ILV_1 takes the value 0, and ILV_4 takes the opposite value. However, if there is a bridging fault between these ILVs, then the value of ILV_4 will be 1, influenced by ILV_1, causing the output to be 0.

However, detecting potential bridging faults, such as those between ILV_i and ILV_{i+2} in this scenario, is not possible as they share the same value in the chain. Consequently, detecting a large number of these types of faults is not feasible using this structure. To detect all potential bridging faults in a chain, it is necessary to have multiple configurations with variable numbers and locations of NOT gates.

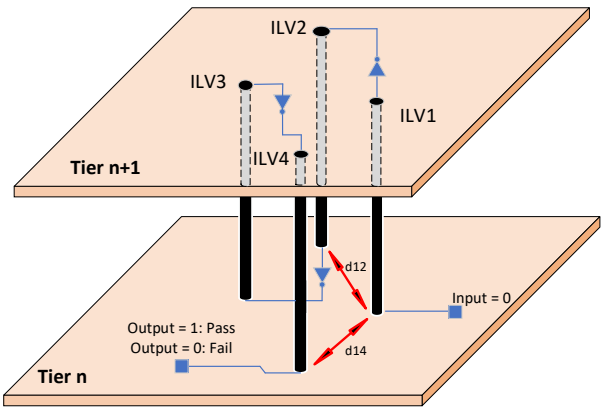


Fig. 2 3D illustration of ILVs organized in a chain between two tiers

Theorem I: the minimum number of configurations required to detect all potential bridging faults in a chain with N ILVs is $\lceil \log_2 N \rceil$.

Proof: if the constructed chain contains N ILVs, we represent it using the chain set based on Eq. (1). An arbitrary test configuration for the chain is defined in Eq. (2), where g_i denotes an inversion occurring ($g_i = 1$) or not ($g_i = 0$) in the output of ILV_i with respect to its input.

$$Chain = \{ILV_{s_0}, ILV_{s_0}, ..., ILV_{s_{N-1}}\} \quad (1)$$
$$s_i \in \{1, 2, ..., N_{ILVs}\}, s_i \neq s_j$$
$$Config_t = \{g_0(t), g_1(t), ..., g_{N-1}(t)\} \quad (2)$$
$$g_i(t) \in \{0, 1\}$$

When a bridging fault occurs between a pair of ILVs, they become equipotential. To detect each bridging fault in the chain, we need to ensure the existence of at least one test configuration where the configurations of the linked ILV pair differ. In other words, there is a test configuration ($Config_z$) for any ILV pair (ILV_i, ILV_j) where $g_i \neq g_j$. Without loss of generality, we assume the chain length is a power of 2 (2^w). To create the fewest test configurations covering all potential bridging fault pairs, we use a binary approach. At each step, the configurations of targeted ILVs are divided into two subsets. In the right subset, all g_m values are set to 1 (inverted versions of the chain input), while the left subset is set to 0 (like the chain input). If the ILV pair elements belong to different subsets in step k ($config_k$), this specific test configuration can identify the associated bridging fault. Conversely, if the pair belongs to a single left or right subset, we proceed to the next step ($Config_{k+1}$) to divide the current part into two new subsets. Note that the length of the targeted subset in $Config_k$ is equal to $2^w/2^k$. In the worst-case scenario, the pair linked to the bridging fault is resolved when the number of test configurations reaches w. Therefore, the number of test configurations that ensure detection of every bridging fault would be $w=\log_2 N$.

In cases where $N \neq 2^w$, adding dummy ILVs theoretically aligns the problem with the $N = 2^w$ scenario to determine necessary test configurations. However, incorporating dummy ILVs is not essential for our testing process, as the main difference lies in some

Unedited

bridging faults occurring in $N=2^w$ but not in this case. For instance, consider a scenario with five clusters. We already know that the fault-free output pattern for these clusters should be 10010. Thus, we can directly compare the actual output bitstream of the clusters to this expected pattern to identify any faults. Hence, the number of required test configurations is calculated using Eq. (3) to ensure detection of all potential bridging faults in this scenario.

$$w = \lceil \log_2 N \rceil \tag{3}$$

Fig. 3 illustrates the number of configurations needed to detect all potential bridging faults in a chain consisting of 8 ILVs. As depicted, each pair of different ILVs exhibits at least one distinct value across three configurations, allowing for the detection of any bridging fault between them. Note that constructing the first configuration in the chain requires only one NOT gate between ILV_3 and ILV_4, while, for the subsequent configurations, three and seven NOT gates, respectively, are needed to construct the desired chains.

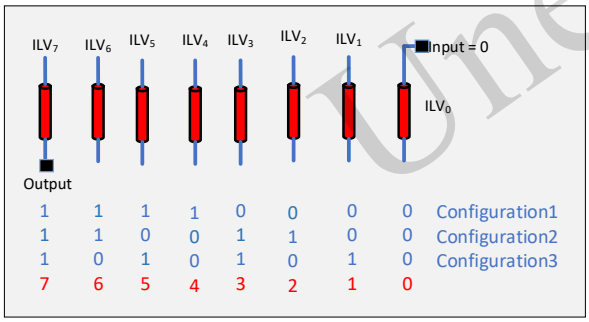


Fig. 3 Number of configurations needed to detect all potential bridging faults in a chain consisting of 8 ILVs

3.1.2 SAFs and multiple faults

In this subsection, we provide evidence that the proposed structure can effectively detect all single stuck-at faults (SAFs) as well as nearly all total multiple faults. This is achieved by applying two distinct bits (01 or 10) while considering a NOT gate between each two subsequent ILVs. For example, when there is a NOT gate between each two subsequent ILVs, if all ILVs are fault-free, the output should be 1 when a bit 0 is applied to the input ILV (Fig. 1). However, if there is an S/1 fault at ILV_1 or ILV_3, or an S/0 fault at ILV_2 or ILV_4, the output will be 0. As a result, these faults can be detected. Similarly, by applying a bit 1 to the input ILV, we can detect S/0 faults at ILV_1 and ILV_3, and S/1 faults at ILV_2 and ILV_4. Therefore, every single stuck-at fault is detected by applying a pair of different bits.

One of the significant advantages of the proposed approach is its capability to detect almost all multiple faults in a small number of test configurations and a short test time. These multiple faults can be of the same type, such as some SAFs or some bridging faults, or different types. Note that the method proposed by Chaudhuri *et al*. (2023) detects all multiple faults but a large number of test configurations are used to complete the test.

To demonstrate the multiple fault detection capabilities of the proposed approach, we explore various scenarios. Assuming the presence of multiple bridge and SAF faults in a chain, we define the fault set according to Eq. (4), where M represents the number of single faults within the chain and SF_i denotes the i'th single fault, which could be a bridging fault between ILV pairs (BRF_{kl}) or a stuck-at fault within a single ILV (SAF_r). These indices are arranged so that the last faulty ILV (ILV_{last}) is closer to the chain's output. Therefore, the last faulty ILV belongs to the SF_{M-1} within the *MulFaultSet* set.

$$MulFaultSet = \{SF_0, SF_1, ..., SF_{M-1}\} \tag{4}$$

$$SF_i \in \{BRF_{kl}, SAF_r\} \tag{5}$$

$$BRF_{kl} := bridgingfault(ILV_k, ILV_l) \tag{6}$$

$$SAF_r := StuckAtFault(ILV_r) \tag{7}$$

In the first scenario, at least one ILV (ILV_s) has a stuck-at fault ($SF_S = SAF_S$) without any bypass from bridging faults. Consequently, the output of the ILVs remains permanently at 0 (or 1), and applying a 0-1 sequence to the chain's input results in a fixed value at the output due to the propagation of the faulty ILV value. Therefore, all multiple faults containing a SAF without any bypass from a bridging fault will be detected. Notably, scenarios involving multiple SAFs fall within this category and are detectable through our approach.

Because of the arrangement of test configuration sets in a chain of length $N=2^w$, the number of available NOT gates is an odd number in each test configuration. Consequently, in a fault-free case, applying a 0-1 sequence generates a 1-0 sequence in the chain's output. As shown in Theorem I, every BRF_{kl} removes an odd number of NOT gates in the chain, resulting in at least one test configuration producing

Unedited

non-inverting logic between the input and output of the chain. For instance, in a test configuration, if the number of NOT gates is 2m+1 and the bridge fault removes 2m'+1 of them, it leads to the subtraction of NOT gates (2m+1-2m'-1= 2m-2m'), resulting in 2(m-m'), which is an even number. This even count of remaining NOT gates in the chain produces non-inverting logic, facilitating fault detection.

Suppose two bridge faults (BRF_{ij} and BRF_{kl}) exist in the chain. Three situations arise: firstly, one fault bypasses the other, meaning BRF_{kl} falls within the range of BRF_{ij} (or vice versa). The range of an SF_{kl} refers to the ILVs lying between ILV_{first} and ILV_{last} of the bridging fault pair described by Eq. (10).

$$ILV_{first} = ILV_k \tag{8}$$

$$ILV_{Last} = ILV_l \tag{9}$$

$$Range_SF_{kl} = \{ILV_{k+1}, ILV_{k+2}, ..., ILV_{l-2}, ILV_{l-1}\} \tag{10}$$

Therefore, we can describe the first situation using Eq. (11). In this scenario, the number of bypassed NOT gates is determined by BRF_{ij}, and the test configuration detecting BRF_{ij} also detects the double fault (BRF_{kl} and BRF_{ij}).

$$Range_SF_{kl} \subset Range_SF_{ij} \tag{11}$$

In the second scenario, ILV_{first} of BRF_{kl} lies within the range defined by BRF_{ij}, and ILV_{last} of BRF_{kl} lies after the ILV_{last} of BRF_{ij}. In this case, the chain's output is influenced by the output value of ILV_{last} of BRF_{kl}, rendering BRF_{ij} ineffective. Therefore, the test configurations detecting BRF_{kl} would also be capable of detecting the double fault.

In the third scenario, two bridging faults (BRF_{kl} and BRF_{ij}) are disjoint, a representation of which can be illustrated according to Eq. (12).

$$Range_SF_{kl} \cap Range_SF_{ij} = \phi \tag{12}$$

In this situation, two cases can be identified. In the first case, suppose that the test configuration sets capable of detecting BRF_{kl} and BRF_{ij} are defined as Eq. (13) and Eq. (14), respectively.

$$TCS_{ij} = \{Config_h \mid \text{Test configuration h detects } BrF_{ij}\} \tag{13}$$

$$TCS_{kl} = \{Config_h \mid \text{Test configuration h detects } BrF_{kl}\} \tag{14}$$

We derive the distinct test set according to Eq. (15), representing the test configurations specifically detecting either BRF_{kl} or BRF_{ij}.

$$DTS_{ij,kl} = (TCS_{ij} - TCS_{kl}) \cup (TCS_{kl} - TCS_{ij}) \tag{15}$$

In each test configuration within $DTS_{ij,kl}$, one of the single bridge faults bypasses an odd number of NOT gates, while the other bypasses an even number of NOT gates. Consequently, their summation results in an odd number, enabling the test configuration to detect the double fault.

In the second case, $DTS_{ij,kl}$ equals an empty set (or $TCS_{ij} = TCS_{kl}$), signifying that similar test configurations can detect disjoint BRF_{kl} and BRF_{ij}. The total number of removed NOT gates includes those removed by BRF_{kl} (2m'+1) and BRF_{ij} (2m+1). Consequently, as an even number of NOT gates is removed (2(m+m'+1)), the respective test configuration cannot detect the double fault. As we prove in Appendix A, the number of undetectable double faults is a small ratio of all double bridging faults (DBFs). So, our proposed approach detects almost all DBFs.

Given the probability P of a single bridge fault occurring between any two ILVs among N ILVs, according to the multiplication rule of the probability of independent events, the probability of k independent bridge faults occurring simultaneously can be expressed as P^k. Our method's effectiveness is validated by nearly detecting all potential DBFs with a probability of occurrence at P^2. Consequently, considering the rapidly diminishing probability with increasing k, and acknowledging our comprehensive detection of events up to P^2, we prioritize our detection efforts to address cases involving single and double bridging faults (k <= 2). In section 3.3.2, we will use an algorithm to connect ILVs in a chain in a way that significantly reduces the likelihood of undetectable DBFs occurring.

Our approach can detect triple faults (involving three bridged ILVs), quadruple faults (involving four bridged ILVs), and so on. This ability stems from considering these faults as a single bridge fault encompassing the initial and final ILVs connected.

As previously proved, when a bridge fault bypasses different types of faults, only the detection of the single bridge fault results in detecting multiple faults. For example, a triple fault behaves like a situation where a SAF is encompassed by a bridging fault. Similarly, a quadruple fault is like a bridge fault that falls within the range of another fault. Therefore, our proposed approach can detect almost all multiple faults within ILVs.

Unedited

3.2 Weak fault detection

The Elmore delay model for a chain comprising N ILVs, where a NOT gate is placed between each pair of subsequent ILVs, is illustrated in Fig. 4. The total delay from the input to the output of the chain is calculated using Eq. (16) (the proof of the formula can be found in Appendix B). In this equation, it is assumed that the K'th ILV in the chain (counting from the output end, where the last ILV is number 1) suffers a resistive open fault and R_{open} is the related resistance of the resistive open fault.

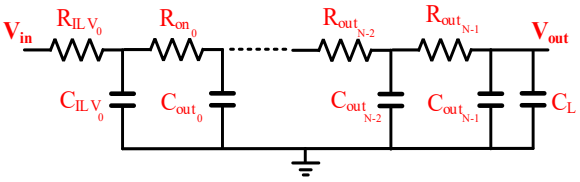


Fig. 4 Elmore delay model for a chain comprising N ILVs, where a NOT gate is placed between each pair of subsequent ILVs

$$Delay_open = (\ln 2)[KR_{open}C_{ILV} \quad (K-1)R_{open}C_{out} + R_{open}C_L] \tag{16}$$

For the clock period of T_{clk}, the open fault is detected if:

$$Delay+Delay_open>,T\text{-}clk. \tag{17}$$

The amount of *Delay_open* is directly dependent on the location of open (K). For instance, in a chain comprising 16 ILVs, the *Delay_open* is equal to $[\ln 2(R_{open}C_{ILV} + R_{open}C_L)]$ when the last ($K=1$) is open, and $[\ln 2\,(16\,R_{open}C_{ILV} + 15\,R_{open}C_{out} + R_{open}C_L)]$ when the first (K=16) ILV is open. If $C_{ILV} = C_{out} = C_L$, the detectable range of resistive for the last ILV is $[R_{open_min}, \infty)$, while the detectable range for the first ILV is $[R_{open_min}/16, \infty)$.

The *Delay_open* of Chaudhuri *et al.* (2019) is equal to $[\ln 2\,R_{open}\,(C_{ILV} + C_L)]$. On the other hand, it is expressed as $[\ln 2\,R_{open}\,(KC_{ILV} + (K-1)\,C_{out} + C_L)]$ in our proposed method, where $K \geq 1$. Therefore, in the worst-case scenario when $K=1$, the *Delay_open* of our proposed method matches the *Delay_open* of Chaudhuri *et al.* (2019).

Consequently, in the worst-case scenario, R_{open_min} for the proposed method is equal to R_{open_min} of Chaudhuri *et al.* (2019). However, for all resistive opens in the chain, except the last one, our proposed method can detect lower R_{open_min} than the method of Chaudhuri *et al.* (2019).

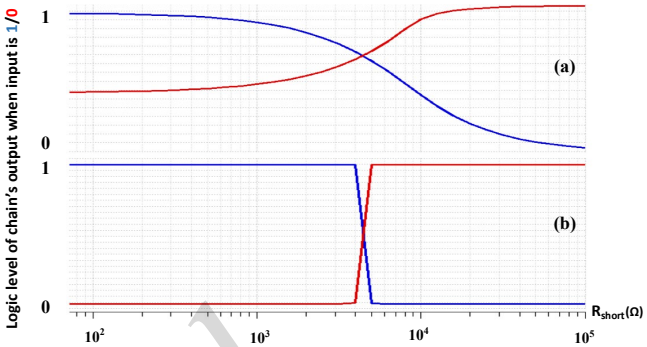


Fig. 5 Logic level of a chain's output comprising 8 ILVs when ILV$_6$ and ILV$_7$ are shorted (a) without a buffer at the end of chain and (b) with a buffer at the end of chain when input is 1 (blue)/ 0 (red)

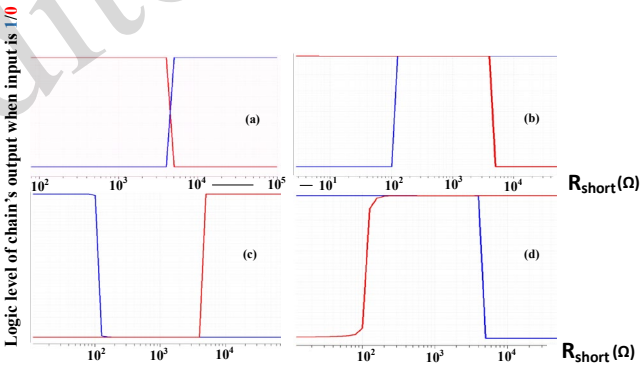


Fig. 6 Logic level of a chain's output comprising 8 ILVs when short occur between (a) ILV$_0$ and ILV$_1$, (b) ILV$_0$ and ILV$_3$, (c) ILV$_3$ and ILV$_6$, and (d) ILV$_4$ and ILV$_7$ when input is 1 (blue)/ 0 (red)

3.2.1 Resistive short faults

In a chain containing N ILVs, the resistive short between any two ILVs is denoted by R_{short}. To analyze the impact of R_{short} at various locations along the chain, the third configuration illustrated in Fig. 3, where there is a NOT gate between each two subsequent ILVs, was simulated using CADENCE virtuoso. Fig. 5a shows the logic level at the chain's output when ILV$_6$ and ILV$_7$ are shorted. Resistive shorts are undetectable when R_{short} exceeds $10^{K\Omega}$, as the output is equal to the fault-free chain's output. Additionally, there is a transition in the output chain between 0 and

Unedited

V_{dd} when R_{short} falls within the range of $5^{K\Omega} < R_{short} < 10^{K\Omega}$. We mitigate this transition by adding a buffer at the end of the chain to ensure stable and reliable operation of the circuit (Fig. 5b).

The results of this simulation for four different scenarios are shown in Fig. 6, where shorts occur between ILV_0 and ILV_1, ILV_0 and ILV_8, ILV_0 and ILV_3, and ILV_7 and ILV_5. Across all cases, the R_{short} is about $5^{K\Omega}$, which is lower than the maximum detectable R_{short} reported by Chaudhuri *et al*. (2019).

3.3 Overall clustering-BIST algorithm

When testing all ILVs in a chain, the test time becomes impractical and fault localization would be impossible, especially for M3D ICs with a large number of ILVs. To address this issue, we cluster the ILVs into groups and test them in multiple parallel chains.

The overall flow of our proposed BIST algorithm is shown in Fig. 7. The algorithm starts with a default number of possible ILV clusters. Then, based on the procedure of 3.6, the locations of ILVs and the logic netlist of every tier are fed into the clustering algorithm. In the next step, the ILVs are divided into K clusters according to the clustering algorithm described in sub-section 3.3.2. Then, the boundaries of adjacent clusters are investigated in a refinement stage to minimize the number of ILVs in the vicinity of the expandable cluster. This is because such ILVs should be added to the chain of the expandable cluster to cover possible borderline bridging faults. The relevant mechanism is described in sub-section 3.3.1.

In the next step, chains are formed for every cluster and the overlap handling routing (according to sub-section 3.3.1) is applied, followed by the construction of the BIST configuration generator and BIST controller (sub-section 3.5). At the end of this step, the proposed BIST structure is completed and then the clusters are evaluated in parallel to calculate the SAF/BF coverage and fault localization granularity (FLG). If the calculated value is satisfactory then the algorithm will be terminated. If it is unsatisfactory, the number of clusters (K) is modified (increased or decreased based on the results) and the algorithm is repeated from the clustering algorithm step.

3.3.1 Detection of high-likelihood bridging faults through overlapping area analysis

If ILVs are clustered based only on the tier area or ILV count in each cluster, there may be many ILVs located at the borders of different clusters that are near each other, increasing the likelihood of a bridging fault. To detect bridging faults occurring at the borders of the clustered ILVs, the clustering approach shown in Fig. 8 is used, where each cluster has an overlap with its adjacent clusters.

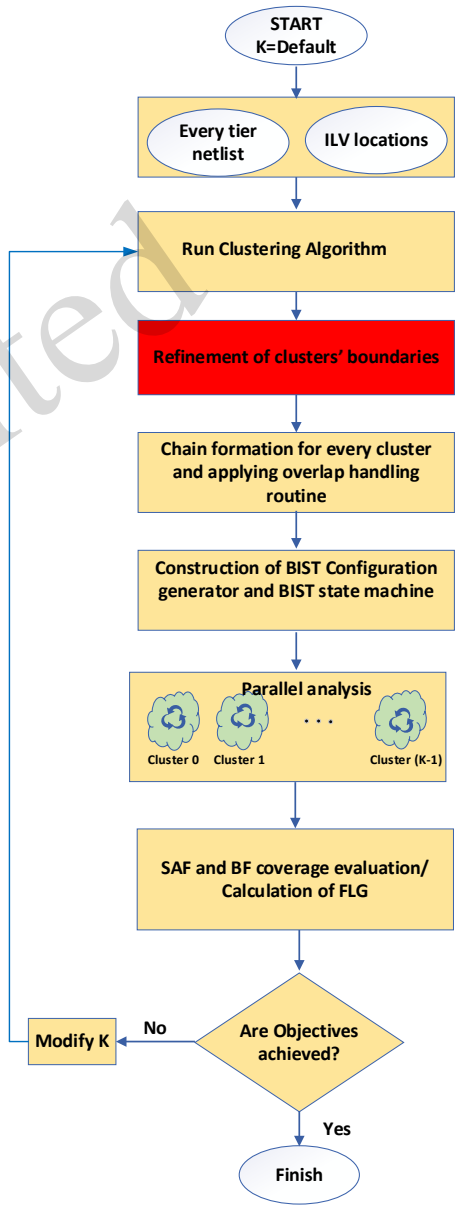


Fig. 7 Proposed clustering-BIST algorithm

Unedited

Therefore, ILVs located at the border of two clusters are included in both clusters to detect any potential bridging faults. As an example, ILVs located in the orange area are clustered together as a group, and the cluster on its right-hand side may include several ILVs located in the orange area to ensure that any potential bridging faults are detected.

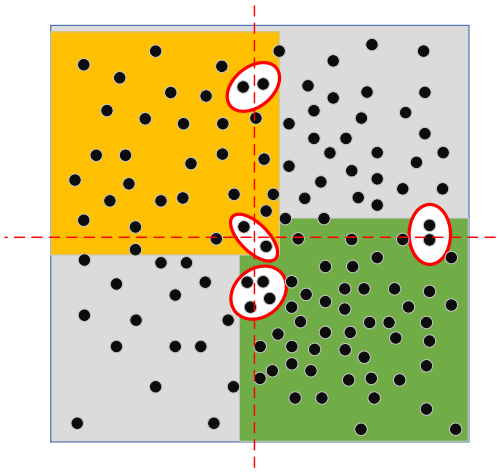


Fig. 8 Efficient clustering method to detect bridging faults in the boarders of the different clusters

Assume that there are N ILVs in a chain. The number of all potential bridging faults in the chain is calculated as $\binom{N}{2} \approx \frac{N^2}{2}$. If the chain is divided into k clusters, each consisting of N/k ILVs, the total number of bridging faults that can be detected using the proposed approach is calculated using Eq. (17).

$$\frac{(\frac{N}{k})^2}{2} \times k = \frac{N^2}{2k^2} \times k = \frac{N^2}{2k} \tag{18}$$

Therefore, there are $(\frac{N^2}{2}) - (\frac{N^2}{2k})$ bridging faults remaining undetected. However, they include faults that are very unlikely to occur. Assume that after placing ILVs in the tier, the distance between the two closest ILVs is d_0 and the distance between two ILVs is d. Note that in our experiments, the parameter d_0 is the pitch size (i.e., the minimum allowable space between two adjacent ILVs in the related M3D IC technology). The pitch size is generally greater than the diameter of the ILV (Koneru and Chakrabarty, 2020). The probability of a bridging fault occurring between two ILVs is represented by $P(d)$ and is greater for ILVs that are closer together. To find an accurate model for P(d) we need to analyze experimental results in the laboratory. However, we can intuitively suppose that there is an exponential relationship between the probability of a bridging fault occurring and the physical separation of ILVs. This assumption was already considered by Chaudhuri *et al.* (2023). According to the related proof in Appendix C, the probability formula would be Eq. (18), wherein d_0 is the minimum distance between adjacent ILVs, the distance between two arbitrary ILVs equals hd_0, and a is the dimension of the chip area.

$$P(hd_0) = \frac{e^{(\frac{\sqrt{2}a}{d_0}+1)}}{d_0(e^{\frac{\sqrt{2}a}{d_0}} - e)} e^{-h} \tag{18}$$

Fig. 9 shows that the likelihood of a bridging fault ($P(hd_0)$) occurring between two ILVs separated by a distance greater than $8d_0$ is nearly negligible, for d_0=1 and a=100.

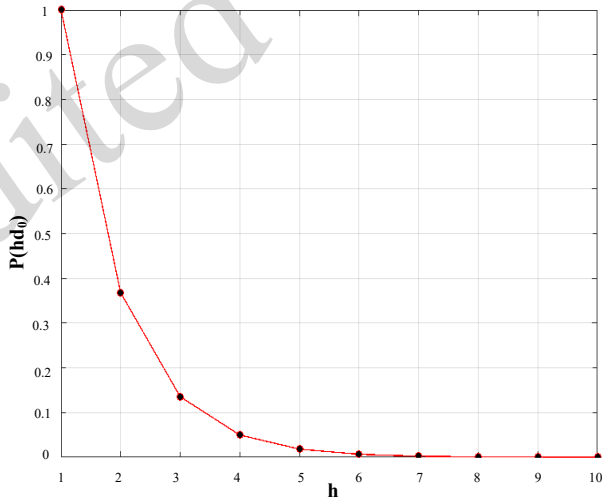


Fig. 9 Probability of occurring a bridging fault $P(hd_0)$ between two ILVs for different values of h, for d0=1 and a=100

Therefore, although our proposed approach is unable to detect all possible bridging faults when ILVs are grouped into several clusters, it still detects all those with a high likelihood of occurring, given the overlap distance we consider.

This structure reduces the probability of failing to detect bridging faults, and results in an increase in design complexity.

To clarify the overlap mechanism, we have illustrated four adjacent clusters, each of which includes 8 ILVs (Fig. 10a). Based on our previous discussion, the clusters are configured in the chain format to be tested for SAFs and other BFs. Note that in this figure, we

Unedited

have used the test configuration in which the NOT gates have been used between subsequent ILVs in the chain. Suppose that we extend the first cluster (Cluster 1) to include a single ILV from the other three clusters which have a minimum distance to the ILVs of Cluster 1 (Fig. 10a, highlighted box). To extend the cluster we connect its last ILV to the extra ILV of Cluster 2 and then this ILV is connected to the extra ILV of Cluster 3 and then this ILV is connected to the extra ILV of Cluster 4 and, finally the last ILV is connected to the Output of Cluster 1. The necessary circuit is shown in Fig. 10b, wherein the added SBs are shown that can be configured in two modes. The first mode is related to the conventional testing of the clusters independently and the second is related to testing of the extended version of Cluster 1. Expanding the cluster requires more hardware overhead (the extra SBs) and more test configurations. So, the refinement step in Fig. 7 is needed to modify the clusters' borderline so that a minimum number of ILVs are included in the expanding mechanism.

In our proposed BIST framework, the expansion of the clusters is managed so that number of clusters involved is minimized. For example, in Fig. 8, only two clusters (orange and green) among the four original clusters need to be expanded to cover all possible inter-cluster bridging faults. We can test these four clusters in three phases: the first phase tests every cluster independently. The second phase involves the expansion of the orange cluster and tests the extended version of the cluster to seek possible inter-cluster DBFs. In the last phase, a similar approach is applied to the extended version of the green cluster. Note that some ILVs are included in the expansion of multiple clusters and some pairs are tested in multiple phases of the proposed approach, which could be considered as redundancy.

3.3.2 Clustering algorithm

This subsection provides a comprehensive description of the algorithm used for clustering ILVs (the highlighted portion of Fig. 7) and the algorithm used to connect ILVs inside a chain in a way that the likelihood of undetectable DBFs occurring reduces significantly.

To begin, we will explain the procedure for generating the dataset. Each CAD tool performs chip routing using its distinct cost function. Consequently, the location of ILVs cannot be predicted and needs to be estimated using specific distributions. To address this, we create a synthetic dataset with a random distribution for ILV locations.

Algorithm 1 clusters a set of ILV locations using the K-means algorithm, which requires four inputs: the number of clusters, a flag indicating whether to allow overlap between clusters, a tolerance coefficient, and an extension value.

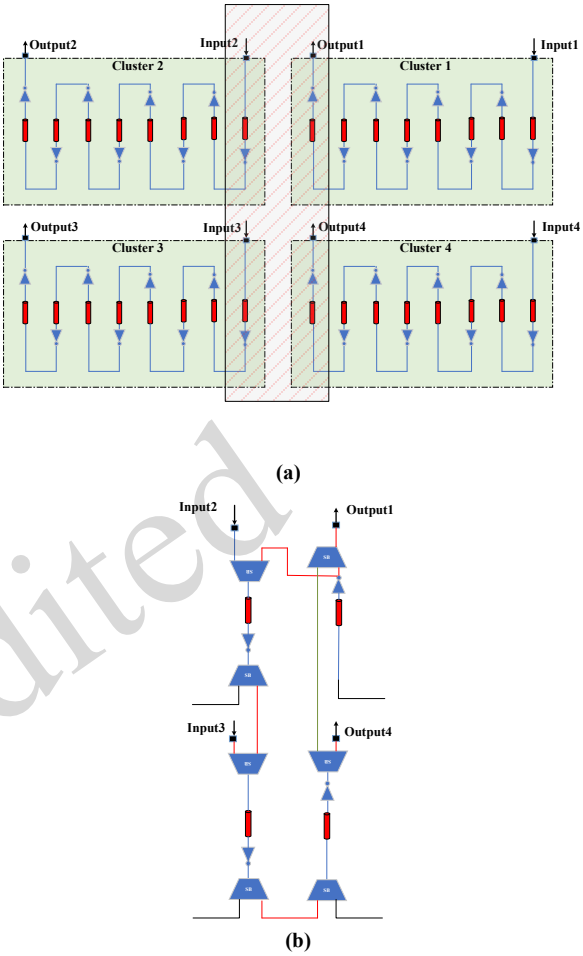


Fig. 10 Solving signal conflicts in overlap regions while maintaining practical test time

In this K-means clustering, if the ILV locations follow a random distribution, the algorithm tends to balance the number of ILVs across clusters. However, this balancing behavior can reduce the algorithm's performance. To mitigate this, we introduce a tolerance coefficient *t,* which allows the algorithm to deviate from perfect balance by ±t percent, improving overall efficiency. Additionally, the extension value *h* is used to expand clusters based on Eq. (30), enabling the algorithm to accommodate bridging faults. The effect of this parameter is shown in Fig. 9. The algorithm ultimately produces a list of cluster assignments for each ILV location. As an example, in Fig. 11 we have clustered

Unedited

1000 ILVs across four different randomly generated datasets with k = 4. The figure highlights certain ILVs located at the cluster boundaries, which are considered potential candidates for bridging faults.

Algorithm 1 Clustering algorithm

Input:
DATASET: ILV locations
K: number of clusters
t: tolerance coefficient
Ov: overlap enable
h: extension value
Output:
A list of cluster assignments for each ILV location

```
G = create_graph (dataset)
AdjMat = adjacency_matrix (G)
d0 = min (AdjMat)
if : ov = "enabled"
   org_cluster = K-Means (DATASET, t, K)
   cluster_assignments = boundary_extenstion (org_cluster, h,d0)
else          // ov = "disabled"
   cluster_assignments = K-Means (DATASET, t, K)
return cluster_assignments
```

In section 3.1.2, we showed that most DBFs can be identified using the suggested method. However, a few specific symmetrical DBFs cannot be detected. Algorithm 2 connects ILVs within chains in such a manner that it significantly decreases the likelihood of symmetrical DBFs occurring.

The algorithm consists of the following steps: first, a graph and its adjacency matrix are created for the ILVs inside a chain, and the distances between each pair of ILVs are calculated. Then, the distances are arranged in descending order. Next, the function DBF is used to find all symmetrical DBFs. Then, the first two longest distances are assigned to the first symmetrical DBF. For example, suppose that in a chain with 8 ILVs, BRF_{01} (ILV_0, ILV_1) and BRF_{45} (ILV_4, ILV_5) are symmetrical bridging faults. Then, ILV_0 is connected to the ILV with the longest possible distance in the chain, to reduce the probability of its occurrence. The same is done for BRF_{45}. After assigning all longer distances to symmetrical DBFs, the rest of the ILVs are connected randomly, as they are all detectable.

3.4 Fault localization

Suppose we are conducting a test on an IC comprising 1024 ILVs, divided into 16 clusters of 64 ILVs each. These cluster outputs are denoted as output 1 through output 16. Six configurations are used to test a cluster that contains 64 ILVs. Across all these configurations, if the ILVs are fault-free, sending a binary 0 to the cluster inputs should produce an output of 1. As a result, the fault's location can be determined once the cluster outputs are shifted out. For example, if output 8 registers a value of 0, it implies the presence of at least one fault within cluster 8. In cases where the ILV count in a cluster is not a power of 2, we still have a clear understanding of the expected 0/1 cluster outputs, allowing us to easily distinguish between faulty and fault-free clusters.

Algorithm 2 Connecting ILVs inside chains

Input:
ILV_list: ILV locations in each cluster
Output:
Chain: connected ILVs

```
G = create_graph (ILV_list)
AdjMat = adjacency_matrix (G)
D = Sort_descending (AdjMat, G)
DBF = find_dbf(G)
for i = 1 : length(D)
Chain= assign_distances_to_dbf(Di, DBFi)
  // Di -> (di, di-1) ,  DBFi -> (DBF1,k, DBF2,k)
end
Chain = rand_assign_rest_distances (Di, G)
Return Chain
```

Our approach is unable to diagnose the type of fault. However, by dividing the ILVs into more clusters, we can attain a higher level of insight into specific locations of faults. For instance, in a scenario where a cluster comprising 64 ILVs is tested and multiple faults are detected within it, our approach confirms the presence of at least one fault in the cluster. However, if we divide the 64 ILVs into 8 clusters of 8 each, we can gain a more comprehensive understanding of both the number of faults and their specific locations.

3.5 Test configurations generator and BIST controller

Whenever we need to test the ILVs in the M3D IC, we can easily put the circuit into the test mode by setting specific switches or control signals. For example, in Fig. 1, we can set switches SB_1, SB_3, SB_5, and SB_7 to 1 to activate the test mode.

Unedited

In test mode, two types of SBs require control. The first type switches overlap areas between adjacent clusters. For instance, in Fig. 10 there are seven SBs to ensure all four clusters can be tested. By setting all SBs to 1, we can test the extended version of cluster 1, then the other group of SBs (which have a similar architecture to Fig. 10 b) are used to test the extended version of cluster 3. Finally, the overlap related SBs are bypassed and the intra-cluster chain is formed to test clusters 2 and 4 independently.

The second type of SB connects an ILV or a NOT gate to the subsequent ILV in the chain, as seen with SB_2, SB_4, and SB_6 in Fig. 1. This introduces complexity, as there can be multiple ILVs within a cluster and multiple configurations (Fig. 3). Therefore, we need a module to generate different configurations. If there is a NOT gate between two ILVs in a chain, the SB between them must be set to 1. By XORing adjacent ILVs, we set the SBs appropriately.

A schematic of the configuration generator for a chain of 8 ILVs is shown in Fig. 12. The required configurations to test 8 ILVs are shown in Fig. 3. In test mode, there is an SB between each two adjacent ILVs to connect them directly (SB selector = 0) or via a NOT gate (SB selector = 1). The first configuration in Fig. 3 is generated by setting only the SB selector between ILV_3 and ILV_4 to 1 and the others to 0. By XORing the bitstreams of adjacent ILVs and applying their outputs to SB selectors, we generate configurations for 8 ILVs. Generally, to generate the configurations needed for N ILVs requires a $\lceil \log_2 N \rceil$-bit counter, a MUX with $\lceil \log_2 N \rceil$ inputs, an N-bit shift register, an *N*-bit register, and *N-1* 2-bit XORs.

Fig. 13 shows the state machine of the on-chip controller for controlling signals to execute the proposed BIST. When the test_en signal is set to 1, test initiation occurs. This signal guides the ILVs from their normal path to the test path and prepares each ILV to connect directly to the next ILV in a chain or through a NOT gate, forming the initial configuration. Within the N-bit counter of the configuration generation, upon reaching a count from 0 to 2^N-1, the cnt signal is triggered, advancing the controller to the next state. Initially, a subset of the SBs responsible for toggling overlap areas is activated. Then, test patterns 010 or 101 are introduced from the inputs of the chain for each configuration.

It takes *K* clock cycles for a single bit to transmit from the input to the output of the chain and *K′* clock cycles for the input to transmit from the overlap SB to the output. We need a total of $3\ (K + K')$ cycles to apply three test patterns in one configuration for scanning out. A counter is set up, and when it reaches $3\ (K + K') - 1$, the clk_chain signal turns to 1, causing the controller to move to the next step. In this state,

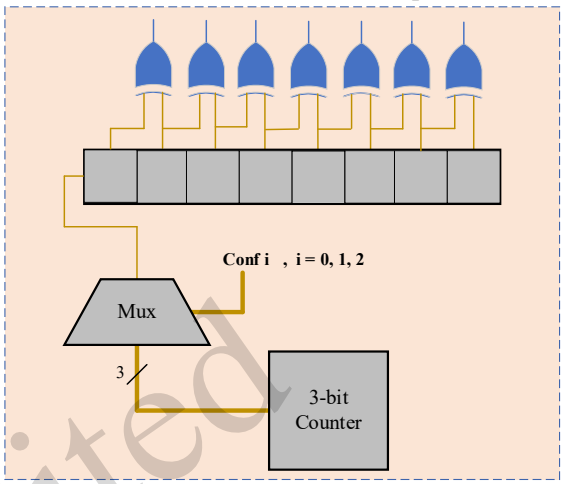


Fig. 12 Schematic of configuration generator for a chain of 8 ILVs and 3 test configurations

the output of the third single bit of the test pattern is scanned out. If several configurations remain untested, the conf signal remains at 0, indicating that the ILVs should be tested in the next configuration.

The test_en signal, which initiates the BIST mechanism, can be controlled via the JTAG interface, and needs one input pin. The outputs of the chains are captured and stored in a shift register that serves as temporary storage for the output signals from all the chains.

Once stored, the output data are serially scanned out from the shift register through a single pin. When dealing with a large number of clusters, using a single pin for scanning out the cluster outputs can increase test time. To mitigate this, we can use more output pins.

To calculate the number of clock cycles required for the proposed BIST approach, consider an example with 4 clusters, each containing 128 ILVs with a NOT gate between each two subsequent ILVs.

First, we calculate the time required to set SBs in the chains and overlap areas to configure the chain, referred to as setup_time. Activating the test_en signal takes one cycle. A 7-bit counter in the configura-

Unedited

tion generator counts to $2^7 = 128$ clocks, setting the SBs to connect ILVs inside the chain and setting overlap areas to 1.

Next, the input pattern is transmitted from input to chain output, denoted as chain_time. For a frequency of 1^{GHz}, it takes 5 clock cycles for the input to transmit to the output of the chain containing 128 ILVs and 1 clock cycle to transmit from the overlap

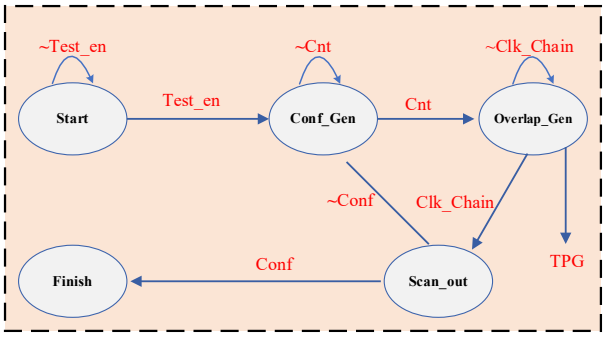
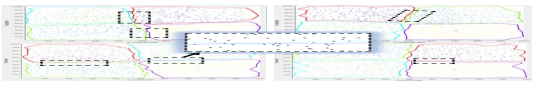


Fig. 13 State machine of the on-chip controller utilized for controlling signals to execute the proposed BIST

SBs to the output if the number of overlapped ILVs between two clusters is 24, as determined through simulation results using CADENCE Virtuoso. Considering that there are 3 single bits to apply to the chains, a total of 3 (5 + 1) = 18 clock cycles is required to receive the input signal at the chain output. For configurations with fewer NOT gates, the chain_time is lower than 18 clock cycles, but this time is reported for all configurations due to negligible impact on total test time. The cluster outputs are scanned out from one output pin in 4 + 1 clock cycles, denoted as scan_out_time. Each configuration tests 128 ILVs in a chain, and there are 7 configurations. The total number of clock cycles needed to complete the test is calculated as follows:



(19)

As explained earlier, the clusters are tested in three phases (Fig. 10). The clock cycles required for the third phase can be calculated using Eq. (19). For the first and second phases (expanding version of clusters 1 and 3), we require 9 extra clock cycles: one to connect the SBs to 1, three to transmit three inputs from the overlap SBs to the chain output, and five to scan out the outputs. Therefore, the test time for these phases is 1064 + 9 = 1073.

3.6 Mechanism of highly accurate estimation of ILVs' locations

Note that the process of our proposed method is based on the estimated location of ILVs before BIST insertion, which may be slightly changed in the final implementation. To achieve a precise estimation of ILV locations, we consider the following actions. First (Step 1), circuit partitioning and implementation is performed using the method of Rahimi and Jahanirad (2021). Note that a fixed portion of the bottom tier is blocked in the placement stage to be reserved for the BIST configuration generator and BIST state machine. The gates and modules in the tiers as well as the source and sink of the required ILVs are determined in this step. In the second step (Step 2), the benchmark circuit is modified by adding the required SBs and NOT gates in the path between the source and sink of every connection, which are interpreted as ILVs in the first step, according to Fig. 1. Note that in this step, the ILV chains are not formed and only the SBs and NOT gates are added to the related source and sinks. In the third step (Step 3), the modified version of the benchmark circuit is implemented again. In this step the partitioning stage of the Rahimi and Jahanirad (2021) flow is omitted because the ILV assigned connections determined in the first step remain unchanged. The added SBs and NOT gates would contribute to the placement and routing process of the related tiers. So, the modified version of the layout including these modules is generated in every tier in this step. In the fourth step (Step 4), the proposed BIST method is performed wherein the ILV clusters and required extra SBs which contribute to the overlap handling routine, as well as the details of BIST configuration generator and BIST state machine, are extracted (Fig. 7). This modified version of the benchmark circuit undertakes another round of implementation (the Place and Route stages of Rahimi and Jahanira, 2021) in the fifth step (Step 5). Note that in this step, we restrict the implementation of the BIST configuration generator and BIST state machine to the dedicated area in Step 1.

In the fifth step, the ILV locations are vulnerable to deviation from the pre-assumed location of step 4. This issue may lead to a change in the boundaries of ILV clusters which should be handled appropriately.

Unedited

According to the precautions mentioned through steps 1-4, the locations of the ILVs experience a small change after the Step 5. Consequently, the deviation of cluster boundaries did not affect the results in our simulations. To evaluate the change in ILV location deviation, we measured the center-to-center distance of the ILV location before and after Step 5 for all ILVs of the benchmark circuit. The values of the center-to-center deviations normalized to the diameter of an ILV indicate a maximum deviation of 0.57. This cannot change the boundary of an adjacent cluster because d_0 (the pitch size) is at least an ILV diameter in the current M3D IC fabrication technology (Koneru and Chakrabarty, 2020).

However, if the boundary change (e.g. when an ILV moves from a cluster to an adjacent cluster) happens for an unseen benchmark circuit, then the affected clusters should be modified and the related chains and the extra modules for overlap handling should be re-considered. In this situation, the last two steps of the proposed BIST methodology should be repeated for new clusters, except that the clustering stage of the BIST methodology should be bypassed in Step 4.

4 Experimental results

In this section, we present the evaluation of our proposed method. To conduct this assessment, we used simulations on large circuits from the VTR benchmark suite (Rose et al., 2012). To ascertain the ILV count, each benchmark circuit underwent a Min-cut partitioning process, following a approach similar to that described by Rahimi and Jahanirad (2021). In the clustering of ILVs, we assumed that these ILVs are distributed randomly on the chip.

For the synthesis of the benchmark circuits, involving the conversion from Register-Transfer Level (RTL) to a netlist, we used Synopsys Design Compiler (DC) and leveraged the Nangate 45-nm Standard-Cell Library. The reported power and area metrics were derived from the outputs of DC, and cell counts were standardized by normalizing all cell types to the equivalent of NAND2X1 gates. Additionally, all other necessary components (e.g., the ILV clustering stage and fault simulation) were simulated using C++ programming language.

To offer a comprehensive insight into the analysis, we will provide results for different numbers of clusters (specifically, k=2, k=16, and k=64). This approach allows us to assess the outcomes for different cluster sizes.

In Table 1, we present a comprehensive analysis of the total dynamic power, cell area, and gate count associated with the proposed BIST architecture across various numbers of ILVs, ranging from 16 to 1024. The results highlight a direct connection between the number of ILVs and the metrics. As the ILV count increases, these measures show a noticeable upward trend. Grouping the same number of ILVs into a greater number of clusters results in a decrease in the parameters.

For instance, consider the case of 512 ILVs, divided into 2, 16, and 64 clusters, resulting in 3441, 1634, and 1557 number of gates, respectively. When all 512 ILVs are grouped into one cluster, the required configuration generator includes modules like a 9-bit counter, a 512-bit shift register, a 512-bit register, a 9-input multiplexer, and 511 2-input XOR gates. In contrast, grouping 512 ILVs into 64 clusters each with 8 ILVs, permits a single shared configuration generator, comprising a 3-bit counter, an 8-bit shift register, an 8-bit register, a 4-input multiplexer, and 7 2-input XOR gates.

Although dividing ILVs into more clusters means we need some extra SBs to switch between the clusters that overlap, these extra SBs have a negligible impact on power, area, and the number of gates as they are much smaller than the SBs inside the clusters themselves. Furthermore, when ILVs are divided into 64 clusters a 64-bit shift register is required to serially scan out the outputs. However, this is still much smaller than the shift register used in the configuration generator for a cluster including 512 ILVs.

In Table 2, we present the estimated overhead introduced by the overlapped areas between clusters in terms of the number of switch boxes (SB-OV), controller configuration complexity (# config-OV), and overall hardware overhead. For configurations with only two clusters, two SBs are added to support the overlap. However, as the number of clusters increases (e.g., clustering 1024 ILVs into 16 or 64 groups), more ILVs are shared between neighboring clusters, leading to an increase in the number of SBs required. The column labeled "# config-OV" repre-

Unedited

sents the number of unique SB configurations needed by the controller, which depends on the maximum number of overlaps each cluster has with adjacent clusters. Even in the worst-case scenario, this number does not exceed six, indicating that the controller logic remains relatively simple and scalable. The last column shows the estimated hardware overhead in terms of additional gate count due to the SBs in overlap areas. When the overhead was less than 1%, we marked it as "NI" (negligible impact). Even in the most demanding case (N=256, 64 clusters), the total gate overhead remains modest at just 6%, confirming that the proposed overlap-handling mechanism introduces minimal hardware complexity.

Three critical parameters presented in Table 3 include the number of test configurations, test time, which indicates the cycles required for fault detection and localization, and fault localization granularity (FLG). FLG is defined as the count of ILVs within which we can confirm the presence of detected faults. Table 3 provides a comparison between the proposed approach that we refer to as Clustering-BIST and ILV-BIST (Chaudhuri et al., 2023). We exclude reporting results for clusters containing fewer than 4 ILVs. For instance, we do not present results for scenarios where 75 ILVs are divided into 25 clusters, each consisting of 3 ILVs.

The first column of Table 3 shows the number of ILVs, ranging from 25 to 750. The test configurations, test time, and FLG values presented for the ILV-BIST approach are extracted from the published paper. Note that the test time reported for ILV-BIST corresponds to the fault detection time and is represented as 3t, where 't' represents the number of required test iterations to detect short faults.

The results from Clustering-BIST are reported for four cases when ILVs are divided into 1, 5, 25, and 125 clusters. Two output pins are used to scan out the outputs, as discussed in Fig. 10. The number of test configurations is notably smaller in all cases for the proposed method compared to ILV-BIST. As ILVs are divided into a larger number of clusters, the number of test configurations decreases. This is because testing a chain of ILVs with a smaller ILV count requires fewer test configurations, and the ability to test clusters in parallel contributes to this efficiency.

In terms of test time, the ILV-BIST approach shows greater efficiency than Clustering-BIST when all ILVs are in a single cluster. However, when they are divided into more clusters, Clustering-BIST exhibits significantly lower test times than ILV-BIST. For instance, with 750 ILVs organized in a single cluster, the test time is 8455 cycles for Clustering-BIST and 2385 cycles for ILV-BIST. In contrast, when those 750 ILVs are partitioned into 25 clusters, each containing 30 ILVs, the test time for Clustering-BIST drops to just 335 cycles. This is because all 25 clusters are tested in parallel. More clock cycles are needed to test 750 ILVs in 125 clusters than to test them in 25 clusters in clustering-BIST, because more clock cycles (125) are needed to serially scan out the outputs when there are 125 clusters. Note that the number of clock cycles required to complete the test can be reduced by adding output pins to scan out the outputs in parallel when there are a large number of clusters.

For example, the number of clock cycles required to test 500 ILVs divided into 125 clusters can be reduced from 393 to 20 and 76 cycles when using 125 and 20 output pins instead of one output pin, respectively.

ILV-BIST consistently outperforms Clustering-BIST in FLG across all cases and benchmarks, maintaining a value of 3. However, as ILVs are divided into more clusters in Clustering-BIST, this value gradually converges toward the FLG achieved by ILV-BIST. For instance, when 500 ILVs are divided into 5, 25, and 125 clusters, the corresponding FLG values are 100, 20, and 4, respectively. The best achievable FLG value for Clustering-BIST is 4, given that clusters with fewer than 4 ILVs are not considered.

The initial column of Table 4 presents the benchmark circuits included in this study. Subsequent columns detail the number of ILVs, total power consumption, cell area utilization, and gate count corresponding to each benchmark. Table 5 shows the outcomes attained through the implementation of the proposed Clustering-BIST approach, based on various metrics. The results are reported for scenarios wherein the ILVs are divided into 2, 16, and 64 clusters.

We define the metrics of power overhead, area overhead, and hardware overhead as the ratios of their respective values in the Clustering-BIST to those in the benchmark design. In general, the parameters of

Unedited

area, power consumption, and hardware overhead show consistently low values of under 10%, across all cases examined in this study. These parameters show a clear correlation with the number of ILVs within the benchmark circuits (Table 1). Furthermore, these parameters show even lower values when ILVs are grouped into more clusters. For instance, in the case of Stereovision2, the power overhead measured at 5.76% for 2 clusters reduces to 2.31% when using 64 clusters. Similarly, for LU32PEENG and LU64PEENG, when ILVs are grouped into 64 clusters, all metrics exhibit low values of about 1%. Test time, FLG and critical path delay overhead values are shown for different benchmarks in Table 5. Note that the increment of critical path delay (CPD) values for the benchmark circuits is below 2%. Such a low overhead is due mainly to the proposed BIST architecture adding two transmission gates (TGs) per ILV to the critical path of the circuit. On the other hand, according to Fig. 10, the overlap mechanism adds a few extra SBs to adjacent clusters, which makes series TGs (i.e. two-level SBs). The critical path (especially in the large circuits) contains 10s logic gates, so addition of a few numbers of the low delay TG modules will not significantly increase the overall delay of the CPD.

As previously clarified, these parameters are exclusively dictated by the number of ILVs and clusters, as elaborated in detail in Table 3. Consequently, we provide the results in Table 4 without redundant explanations.

In Table 6, we compare the proposed method against ILV-BIST (Chaudhuri et al., 2023) in terms of power, area, and hardware overheads across several benchmarks. To ensure a fair comparison, we regenerated the ILV-BIST method and reported its optimal performance results. For our method, we used the scenario when the number of clusters is 64.

The results indicate that ILV-BIST slightly outperforms our method in terms of the power overhead. However, our proposed method demonstrates advantages over ILV-BIST in terms of the area and hardware overhead. Specifically, the area and power overhead for our method were 1.29 and 1.48, respectively, while for ILV-BIST, these values were 1.7 and 2.29, respectively.

5 Conclusions

In this paper, we have introduced an innovative BIST design that focuses on the detection and localization of SAFs, multiple faults, and bridging faults in irregularly placed ILVs in M3D ICs.

The introduced BIST method has proven to be highly effective in detecting various types of faults, including SAFs, bridging faults, and multiple faults. All multiple and bridging faults in a single chain are detected using the proposed approach. Furthermore, bridging faults with a high likelihood of occurring are detected in multiple chains using the proposed method.

One noteworthy achievement of this research is the significant reduction in the number of test configurations required and the test time. By dividing ILVs into a larger number of clusters and testing them as such, the proposed approach has minimized the number of ILVs tested in each cluster, resulting in a mere 20 test cycles for efficient fault detection. This, in turn, contributes to improved efficiency in terms of power consumption, area utilization, and hardware overhead, particularly for large-scale benchmarks.

Unedited

Table 1 Total power, cell area, and gate count for the proposed BIST architecture across varying numbers of ILVs (2, 16, 64)

N	Power (mw)			Area ((um)^2)			#of Gates		
	2	16	64	2	16	64	2	16	64
16	0.02	-	-	257.12	-	-	137	-	-
32	0.05	-	-	470.26	-	-	252	-	-
64	0.091	0.05	-	882.45	469.32	-	452	251	-
128	0.17	0.08	-	1678.18	856.93	-	898	458	-
256	0.33	0.15	0.16	3274.5	1595.54	1625.28	1751	85	869
512	0.65	0.31	0.29	6433.96	3.58.69	2913.85	3441	1634	1557
1024	1.27	0.6	0.54	12747.85	5953.56	5449.02	6816	3183	2913

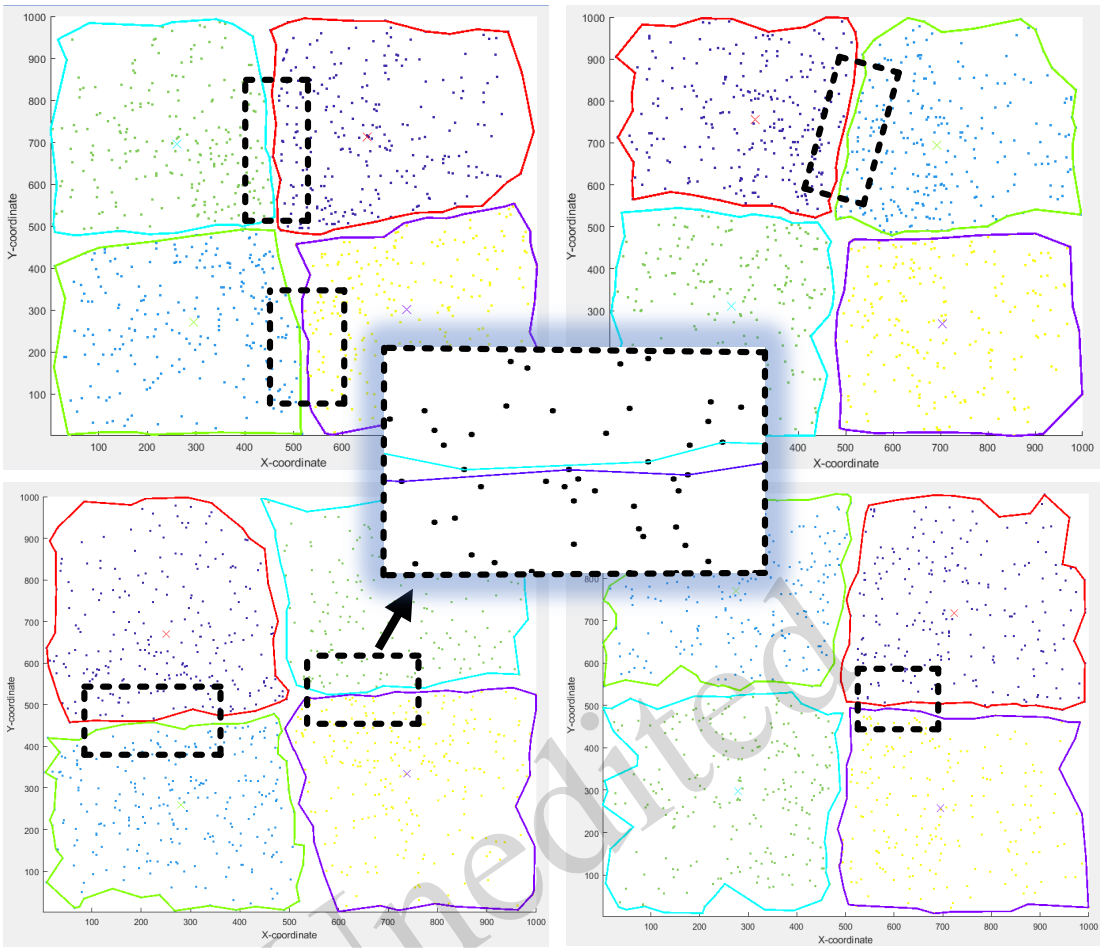


Fig. 11 Clustering 1000 ILVs across four different datasets in four clusters (k = 4). Zoomed area illustrates some potential bridging faults occurring at the borders of the clusters

Unedited

Table 2 Overhead analysis of overlapped ILVs: SB count, controller configurations to handle overlaps, and Hardware Impact

N	#SB-OV			# config-OV (controller overhead)			Hardware overhead (%)		
	2	16	64	2	16	64	2	16	64
16	2	-	-	2	-	-	NI	-	-
32	2	-	-	2	-	-	NI	-	-
64	2	14	-	2	3	-	NI	5	-
128	2	32	-	2	4	-	NI	3	-
256	2	40	98	2	4	3	NI	2	6
512	2	52	144	2	5	4	NI	2	5
1024	2	68	194	2	6	5	NI	1	2

Table 3 Comparison of clustering-BIST and ILV-BIST in terms of test configurations, test time, and FLG

N	#Test configurations (iterations)					Test time (cycles)					FLG (#of ILVs)				
	Clustering-BIST				ILV-BIST	Clustering-BIST				ILV-BIST	Clustering -BIST				ILV-BIST
	#Clusters					#Clusters					#Clusters				
	1	5	25	125		1	5	25	125		1	5	25	125	
25	5	3	-	-	10	164	55	-	-	30	25	5	-	-	
50	6	4	-	-	31	430	108	-	-	93	50	10	-	-	
75	7	4	-	-	75	585	119	-	-	225	75	15	-	-	3
100	7	5	2	-	59	829	195	92	-	177	100	20	4	-	
200	8	6	3	-	319	1851	366	136	-	957	200	40	8	-	
500	9	7	5	2	199	4755	1070	285	393	597	500	100	20	4	
750	10	8	5	3	795	8455	1216	335	531	2385	750	150	30	6	

Table 4 Different parameters associated with benchmark circuits

Benchmark	#of ILVs	Total power (mW)	#of Gates
Stereovision0	948	16.97	273,276
Stereovision1	987	10.95	278,469
Stereovision2	1106	23.78	447,575
LU32PEENG	1543	96.89	745,003
LU64PEENG	1980	135.51	1,455,275

Unedited

Table 5 Results of the clustering-BIST for different benchmark circuits

Benchmark	Power overhead (%)			Area overhead (%)			Hardware overhead (%)			#Test time (cycles)			FLG (ILV count)			Critical path delay overhead (%)
	#Clusters			#Clusters			#Clusters			#Clusters			#Clusters			#Clusters
	2	16	64	2	16	64	2	16	64	2	16	64	2	16	64	64
Stereovision0	7.09	3.27	2.95	4.35	2.01	1.74	4.88	2.45	1.92	4881	530	417	474	60	15	2.04
Stereovision1	9.85	5.28	4.57	4.44	2.21	1.82	4.97	2.69	2.03	4933	554	432	494	62	16	1.95
Stereovision2	5.76	2.67	2.31	3.07	1.41	1.23	3.45	1.8	1.46	6275	696	507	553	70	18	1.71
LU32PEENG	1.97	0.91	0.82	2.57	1.18	1.03	3.11	1.31	1.14	8765	914	545	772	97	25	1.36
LU64PEENG	1.81	0.84	0.73	1.69	0.77	0.67	1.99	0.92	0.85	11155	1153	572	990	124	30	1.16

Table 6 Comparative analysis of the proposed method against ILV-BIST in terms of power, area, and hardware overhead

Benchmark	Power overhead (%)		Area overhead (%)		Hardware overhead (%)	
	Clustering-BIST	ILV-BIST	Clustering-BIST	ILV-BIST	Clustering-BIST	ILV-BIST
Stereovision0	2.95	2.65	1.74	2.27	1.92	3.01
Stereovision1	4.57	4.28	1.82	2.35	2.03	3.21
Stereovision2	2.31	2.2	1.23	1.64	1.46	2.2
LU32PEENG	0.82	0.75	1.03	1.37	1.14	1.84
LU64PEENG	0.73	0.69	0.67	0.9	0.85	1.21
Average	**2.27**	**2.11**	**1.29**	**1.7**	**1.48**	**2.29**

Contributors

Hadi JAHANIRAD conceived and supervised the project, contributed to the methodology and software, and participated in writing review editing. Ahmad MENBARI and Hemin RAHIMI contributed to the conceptualization, methodology, software, and writing. Daniel ZIENER assisted with the software development and writing.

Conflict of interest

All the authors declare that they have no conflict of interest.

References

Batude P, Fenouillet-Beranger C, Pasini L, et al., 2015. 3DVLSI with CoolCube process: an alternative path to scaling. Proc Symp VLSI Technology, p.T48-T49. https://doi.org/10.1109/VLSIT.2015.7223698

Campregher N, Cheung PYK, Constantinides GA, et al., 2005. Analysis of yield loss due to random photolithographic defects in the interconnect structure of FPGAs. Proc ACM/SIGDA 13th Int Symp Field-Programmable Gate Arrays, p.138-148. https://doi.org/10.1145/1046192.10462

Chang K, Koneru A, Chakrabarty K, et al., 2017. Design automation and testing of monolithic 3D ICs: opportunities, challenges, and solutions: (Invited paper). Proc IEEE/ACM Int Conf on Computer-Aided Design, p.805-810. https://doi.org/10.1109/ICCAD.2017.8203860

Chaudhuri A, Banerjee S, Park H, et al., 2019. Built-in self-test for inter-layer vias in monolithic 3D ICs. Proc IEEE European Test Symp, p.1-6. https://doi.org/10.1109/ETS.2019.8791515

Chaudhuri A, Banerjee S, Park H, et al., 2020. Advances in design and test of monolithic 3-D ICs. *IEEE Des Test*, 37(4):92-100. https://doi.org/10.1109/MDAT.2020.2988657

Chaudhuri A, Banerjee S, Kim J, et al., 2021. Built-in self-test and fault localization for inter-layer vias in monolithic 3D ICs. *ACM J Emerg Technol Comput Syst*, 18(1):22. https://doi.org/10.1145/3464430

Chaudhuri A, Banerjee S, Kim J, et al., 2023. Built-in self-test of high-density and realistic ILV layouts in monolithic 3-D ICs. *IEEE Trans Very Large Scale Integr (VLSI) Syst*, 31(3):296-309. https://doi.org/10.1109/TVLSI.2022.3228850

Chen T, Ding RY, Liu J, et al., 2024. A scan-chain-based built-in self-test for ILV in monolithic 3-D ICs. *IEEE Trans Instrum Meas*, 73:3536013. https://doi.org/10.1109/TIM.2024.3472787

Cron A, Marinissen EJ, 2021. IEEE standard 1838 is on the move. *Computer*, 54(11):88-94. https://doi.org/10.1109/MC.2021.3106415

Dang K, Ahmed AB, Abdallah AB, et al., 2019. TSV-IaS: analytic analysis and low-cost non-preemptive on-line detection and correction method for TSV defects. Proc IEEE Computer Society Annual Symp on VLSI, p.501-506. https://doi.org/10.1109/ISVLSI.2019.00096

Garcia-Buendia N, Muñoz-Montoro AJ, Cortina R, et al., 2024. Mapping the landscape of quantum computing and high performance computing research over the last decade. *IEEE Access*, 12:106107-106120. https://doi.org/10.1109/ACCESS.2024.3411307

Geffken RM, et al., 1999. Method of Forming a Self-Aligned Copper Diffusion Barrier in Vias. US Patent 5, 985, 762.

Huang LH, Cheng YY, Wu TL, 2025. Analysis and optimization of HBM3 PPA for TSV model with micro-bump and hybrid bonding. *IEEE Trans Compon, Packag Manuf Technol*, 15(1):22-29. https://doi.org/10.1109/TCPMT.2024.3483445.

Jutman A, 2004. Shift register based TPG for at-speed interconnect BIST. Proc 24th Int Conf on Microelectronics, p.751-754.https://doi.org/10.1109/ICMEL.2004.1314941

im J, Murali G, Vanna-Iampikul P, et al., 2020. RTL-to-GDS design tools for monolithic 3D ICs. Proc IEEE/ACM Int Conf on Computer Aided Design, p.1-8.

Kim S, Park H, 2024. Comprehensive physical design flow incorporating 3-D connections for monolithic 3-D ICs. *IEEE Trans Comput-Aided Des Integr Circuits Syst*,

Unedited

43(7):1944-1956.
https://doi.org/10.1109/TCAD.2024.3357600

Koneru A, Chakrabarty K, 2020. An interlayer interconnect BIST and diagnosis solution for monolithic 3-D ICs. *IEEE Trans Comput Aided Des Integr Circuits Syst*, 39(10):3056-3066.
https://doi.org/10.1109/TCAD.2019.2935410

Koneru A, Kannan S, Chakrabarty K, 2016. Impact of wafer-bonding defects on monolithic 3D integrated circuits. Proc IEEE 25th Conf on Electrical Performance of Electronic Packaging and Systems, p.91-94.
https://doi.org/10.1109/EPEPS.2016.7835425

Lee YW, Lim H, Seo S, et al., 2019. A low-cost concurrent TSV test architecture with lossless test output compression scheme. *PLoS One*, 14(8):e0221043.
https://doi.org/10.1371/journal.pone.0221043

Mok J, Lim H, Kang S, 2021. Enhanced postbond test architecture for bridge defects between the TSVs. *IEEE Trans Very Large Scale Integr Syst*, 29(6):1164-1177.
https://doi.org/10.1109/TVLSI.2021.3063651

Park H, Ku BW, Chang K, et al., 2020. Pseudo-3D approaches for commercial-grade RTL-to-GDS tool flow targeting monolithic 3D ICs. Proc Int Symp on Physical Design, p.47-54. https://doi.org/10.1145/3372780.3375567

Pendurkar R, Chatterjee A, Zorian Y, 2001. Switching activity generation with automated BIST synthesis for performance testing of interconnects. *IEEE Trans Comput-Aided Des Integr Circuits Syst*, 20(9):1143-1158.
https://doi.org/10.1109/43.945309

Pentapati S, Lim SK, 2024. Heterogeneous monolithic 3-D IC designs: challenges, EDA solutions, and power, performance, cost tradeoffs. *IEEE Trans Very Large Scale Integr (VLSI) Syst*, 32(3):413-421.
https://doi.org/10.1109/TVLSI.2023.3347372

Rahimi H, Jahanirad H, 2021. An evolutionary approach to implement logic circuits on three dimensional FPGAs. *Expert Syst Appl*, 174:114780.
https://doi.org/10.1016/j.eswa.2021.114780

Rajski J, Tyszer J, 2013. Fault diagnosis of TSV-based interconnects in 3-D stacked designs. Proc IEEE Int Test Conf, p.1-10. https://doi.org/10.1109/TEST.2013.6651894

Rose J, Luu J, Yu CW, et al., 2012. The VTR project: architecture and CAD for FPGAs from Verilog to routing. Proc ACM/SIGDA Int Symp on Field Programmable Gate Arrays, p.77-86.
https://doi.org/10.1145/2145694.2145708

huries S, Billoint O, Choisnet S, et al., 2020. M3D-ADTCO: monolithic 3D architecture, design and technology co-optimization for high energy efficient 3D IC. Proc Design, Automation & Test in Europe Conf & Exhibition, p.1740-1745.
https://doi.org/10.23919/DATE48585.2020.9116293

Yonehara T, 2015. Epitaxial layer transfer technology and application. Proc IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conf, p.1-3.
https://doi.org/10.1109/S3S.2015.7333529

Zhou YL, Zhou ZX, Wei YM, et al., 2024. A CFMB STT-MRAM-based computing-in-memory proposal with cascade computing unit for edge AI devices. *IEEE Trans Circuits Syst I: Regul Pap*, 71(1):187-200.
https://doi.org/10.1109/TCSI.2023.3327461

Appendix:

A. Elaboration of the number of undetectable double faults:

We claim that the number of undetectable double faults is a small portion of all potential double bridging faults (DBF). Let us elaborate into the situation concerning the last test configuration ($Config_{last}$). In this configuration, there is a NOT gate between every adjacent ILVs (refer to Fig. 2). Hence, if the range of a bridging fault contains an even number of ILVs, then $Config_{last}$ can detect it (due to the inclusion of an odd number of NOT gates). To fulfill this condition, the difference between the indices of ILV_{first} and ILV_{last} must be an odd number. If ILV_{first} is ILV_0, then ILV_{last} could be 1,3,…,2x-1,N-1 (total N/2 = $2^{(w-1)}$ bridge faults). Similarly, for $ILV_{first} = ILV_1$, ILV_{last} could be 2,4,6,…,2y,…,N-2 (total $(2^{(w-1)}-1)$ bridge faults). Therefore, the total number of detectable single bridging faults using this test configuration is calculated according to (1-A). In the first line of (1-A), the first summation deals with the SBF (single bridging fault) where ILV_{first} has an even index, while the second summation involves the SBFs starting at an odd-indexed ILV.

$$\begin{aligned} Total_Det_SBF &= \sum_{i=0}^{\frac{N}{2}-1}\left(\frac{N}{2}-i\right)+\sum_{i=0}^{\frac{N}{2}-1}\left(\frac{N}{2}-1-i\right)= \\ &2\times\sum_{i=0}^{\frac{N}{2}-1}\left(\frac{N}{2}\right)-\sum_{i=0}^{\frac{N}{2}-1}(1)-2\times\sum_{i=0}^{\frac{N}{2}-1} i = \\ &N\times\left(\frac{N}{2}\right)-\left(\frac{N}{2}\right)-\left(\frac{N}{2}-1\right)\times\left(\frac{N}{2}\right)=\frac{N^2}{4} \end{aligned} \tag{1-A}$$

Among all these detectable SBFs by $Config_{last}$, only the bridge faults between two adjacent ILVs with an even ILV_{first} index are exclusively detected by $Config_{last}$ (meaning the other test configurations cannot detect these specific SBFs). The rest of the SBFs can be detected by at least one of the other test

Unedited

configurations. The number of these exclusive SBFs is N/2. Consequently, every double SBF that consists of these N/2 single bridging faults cannot be detected by $Config_{last}$. To construct such a double fault, we need to select two of these exclusive SBFs. Hence, the number of these undetectable DBFs is calculated by the first term of (2-A). Additionally, aside from these undetectable DBFs, all symmetrical DBFs cannot be detected, as represented in (2-A). When we refer to symmetrical DBFs, we are describing DBFs consisting of two SBFs where the positions of their ILVs are symmetrically placed if we divide the chain into two parts from its midpoint. The second term in (2-A) represents all possible SBFs in the first section of the chain that form a DBF when paired with their symmetrical SBFs in the second section of the chain. Meanwhile, the last term in (2-A) eliminates the adjacent SBFs in the first part of the chain since they are already accounted for in the first term.

On the other hand, total number of double bridging faults in the chain is calculated using (3-A). Ultimately, the ratio of the undetectable double bridging faults to the total number of double bridging faults is calculated using (4-A).

For instance, if the chain length is 8 (N=8), the ratio would be 0.0238 (or 2.38%). This ratio diminishes rapidly with larger N values. Consequently, our approach can detect almost all DBFs.

B. Delay calculation for resistive open fault:

The total delay from input to the output of the chain is calculated using (1-B).

Assume that $R_{ILV0}= R_{ILV1}=\ldots= R_{ILVN-1}= R_{ILV}$, $C_{ILV0}= C_{ILV1}=\ldots= C_{ILVN-1}= C_{ILV}$, $R_{on0}= R_{on1}=\ldots= R_{onN-2}= R_{on}$ and, $C_{out0}= C_{out1}=\ldots= C_{outN-2}= C_{out}$. Consequently, the Delay is calculated using (2-B) and (3-B).

If the *K*'th ILV in the chain (counting from the output end, where the last ILV is number 1) as a resistive open fault represented as R_{open}, the delay associated with R_{open} is added to the total delay of the chain, calculated using (4-B).

C. Modeling of bridging fault occurrence probability:

Here we consider the model for *P(d)* according to (1-C) which indicates a decreasing trend with increasing the distance among an ILV pair. Assume that the area of tier is $a\times a$ and its diameter as the longest possible distance between two ILVs is $\sqrt{a}$. Then we can deduce from (2-C) and (3-C), the more specific formula in (4-C).

If we consider the overlap area between two clusters equal to hd_0, all bridging faults with the probability of occurring greater than (5-C) are detected.

Unedited

$$\#undetecDBF=\binom{N/2}{2}+\binom{N/2}{2}-\frac{N}{4}=2\times\frac{\frac{N}{2}\times(\frac{N}{2}-1)}{2}-\frac{N}{4}$$

$$=\frac{N^2}{4}-\frac{N}{2}-\frac{N}{4}=\frac{N^2-3N}{4} \tag{2-A}$$

$$\#totDBFs=\binom{N}{2}\binom{N-2}{2}=\frac{N\times(N-1)}{2}\times\frac{(N-2)\times(N-3)}{2}= \tag{3-A}$$

$$(N^4-6N^3+11N^2-6N)/4$$

$$Ratio=\#undetecDBF/\#totDBF=$$

$$(\frac{N^2-2N}{2})\Big/(N^4-6N^3+11N^2-6N) \tag{4-A}$$

$$Delay = (\ln 2)[\sum_{i=0}^{N-1} R_{ILV_i} \sum_{j=i}^{N-1} C_{ILV_j} + \sum_{i=0}^{N-2} R_{ILV_i} \sum_{j=i}^{N-2} C_{out_j} + \sum_{i=0}^{N-2} R_{on_i} \sum_{j=i+1}^{N-1} C_{ILV_j} + \sum_{i=0}^{N-2} R_{on_i} \sum_{j=i}^{N-2} C_{out_j} + (R_{ILV_0} + R_{on_0} + \ldots + R_{ILV_{N-1}})C_L] \quad (1\text{-B})$$

$$Delay = (\ln 2)[\sum_{i=1}^{N} i.R_{ILV}C_{ILV} + \sum_{i=1}^{N-1} i.R_{ILV}C_{out} + \sum_{i=1}^{N-1} i.R_{on}C_{ILV} + \sum_{i=1}^{N-1} i.R_{on}C_{out} + (NR_{ILV} + (N-1)R_{on})C_L] \quad (2\text{-B})$$

$$Delay = (\ln 2)[(\frac{N(N+1)}{2})R_{ILV}C_{ILV} + (\frac{N(N-1)}{2})R_{ILV}C_{out} + (\frac{N(N-1)}{2})R_{on}C_{ILV} + (\frac{N(N-1)}{2})R_{on}C_{out} + (NR_{ILV} + (N-1)R_{on})C_L] \quad (3\text{-B})$$

$$Delay_open = (\ln 2)[KR_{open}C_{ILV} + (K-1)R_{open}C_{out} + R_{open}C_L] \quad (4\text{-B})$$

$$P(d) = ce^{-\frac{d}{d_0}} \quad (1\text{-C})$$

$$\int_{d_0}^{\sqrt{2}a} P(d)\,dx = \int_{d_0}^{\sqrt{2}a} c\left(e^{-\frac{x}{d_0}}\right)dx = 1 \quad (2\text{-C})$$

$$\rightarrow cd_0\left(\frac{1}{e} - \frac{1}{e^{\frac{\sqrt{2}a}{d_0}}}\right) = 1 \quad (3\text{-C})$$

$$\rightarrow P(d) = \frac{e^{(\frac{\sqrt{2}a}{d_0}+1)}}{d_0(e^{\frac{\sqrt{2}a}{d_0}} - e)} e^{-\frac{d}{d_0}} \quad (4\text{-C})$$

$$\rightarrow P(hd_0) = \frac{e^{(\frac{\sqrt{2}a}{d_0}+1)}}{d_0(e^{\frac{\sqrt{2}a}{d_0}} - e)} e^{-h} \quad (5\text{-C})$$

Unedited