Efficient Structured Scan Patterns Retargeting for Hierarchical IEEE 1687 Networks

Ahmed M. Y. Ibrahim and Hans G. Kerkhoff
Testable Design and Test of Integrated Systems Group (TDT)
University of Twente
Enschede, the Netherlands
{a.m.y.ibrahim, h.g.kerkhoff}@utwente.nl

Abrar Ibrahim, Mona Safar and M. Watheq El-Kharashi
Computer and Systems Engineering Department
Faculty of Engineering, Ain Shams University
Cairo, Egypt
{abrar.ibrahim, mona.safar, watheq.elkharashi}@eng.asu.edu.eg

***Abstract*—The IEEE 1687 standard introduced a large design space of compliant networks for accessing embedded instruments. Such networks could grow in their structural complexity and inter-component temporal dependencies. Scan pattern retargeting is defined as the procedure of translating an instrument-level pattern to several network-level ones. Pattern retargeting could become computationally intensive with the increase of structural and temporal dependencies. Structured pattern retargeting was previously introduced as a formal and light-weight pattern retargeting methodology for arbitrary IEEE 1687 networks. In this work, we present a dedicated structured retargeting method for hierarchical IEEE 1687 networks. The proposed method significantly reduces the retargeting time for pure hierarchical networks compared to the general one, while resulting in the same network access time. The retargeting time is of a special importance in the case of on-chip retargeting, which is used for on-line monitoring using IEEE 1687 networks.**

***Keywords*—Embedded instruments, IEEE 1687, IJTAG, structured retargeting.**

I. Introduction

The IEEE 1687 standard [1] presents standardized access networks for accessing embedded instruments that are utilized for testing, debugging, monitoring and other purposes. The standard follows a descriptive approach in defining its compliant instrument networks, which results in a very large design space, where such networks could significantly grow in their structural complexity.

Hierarchical IEEE 1687 networks are the most common architectural organization of compliant networks, since their construction can be carried out in a structured and scalable manner by using Segment Insertion Bits (SIBs). SIBs enable the insertion and exclusion of scan segments from the active scan path. Figure 1(a) shows an example of a hierarchical network, while Figure 1(b) shows the internal organization of the SIB.

Scan pattern retargeting is the process of translating an instrument-level pattern, to a set of network-level scan vectors. For example, Figure 1(c) shows the set of retargeted scan vectors that should be shifted to the network in order to write the pattern (10110) to register (A), starting from the network at the reset state (all ScanMux control registers (C) are '0's). Pattern retargeting could become computationally intensive with the growth of the inter-register structural and temporal dependencies [2]. Hence, pattern retargeting methods are required to be devised in a formal manner.

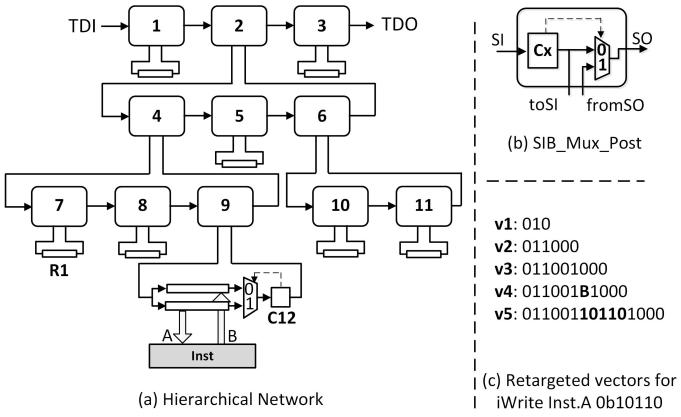

Figure 1: Hierarchical IEEE 1687 network.

A retargeting software tool [3] or hardware coprocessor [4] processes on a structural model of the network, and then generates the set of retargeted scan vectors to access a certain instrument. Dynamic pattern retargeting is performed if the retargeter is generating the scan vectors simultaneously while accessing the network. This occurs in the case of run-time debugging or dependability management [5] using IEEE 1687 networks. In this case, optimizing both the retargeting and network access times becomes important. Consequently, structured retargeting was introduced in [6] as a light-weight dynamic retargeting method for arbitrary networks.

In this paper, a novel structured retargeting method is introduced. It significantly reduces the retargeting time for hierarchical networks compared to the previously introduced one in [6], without affecting the access time. The method is still applicable to the less common non-hierarchical networks, enabling a comprehensive solution, however with a degraded access time. The retargeting time is especially important for on-chip IEEE 1687 network controllers, such as the ones that have been recently introduced by industrial EDA tools [7].

The remainder of the paper is organized as follows: section II discusses the terminologies used and related works. Temporal dependencies modelling and the temporal characteristics of hierarchical networks are presented in section III. In section IV the optimized structured retargeting method for hierarchical

networks is presented. Experimental results are discussed in section V. Finally conclusions are discussed in section VI.

II. Terminology and Related Work

The IEEE 1687 instrument network, connected between the Test Data In (TDI) and Test Data Out (TDO) ports, is primarily constructed using ScanRegisters, ScanMuxs and control signals. A ScanRegister is a sequential network component that, at least, is able to perform the shift function, and optionally the capture and/or the update functions. ScanRegisters are used for network configuration and for data delivery to/from the instruments; the latter is often referred to as a Test Data Register (TDR). A ScanMux is a combinatorial network component that enables selecting one of its (M) input scan paths according to the given address signal values. ScanMuxs are used as the mechanism for configuring the scan path.

The ScanMux address control signal (S) is the result of a combinatorial function in terms of the contents of the network ScanRegisters. A ScanMux Control Bit (SCB) is any ScanRegister in the network that contributes to the combinatorial function of an (S) signal. The current state of all SCBs represents the network state. Several network states can configure an active scan path that includes a certain register.

The selection of the register 'R', denoted as Sel(R), is defined as a Boolean function in terms of a minimum set of the SCBs, that when evaluated to True (according to the SCBs values) R becomes included in the active scan path. For example, Table I shows the selections of all ScanRegisters in the network shown in Figure 1(a). The selections represent the inter-registers structural dependencies. A formal methodology for extracting the register selections in arbitrary IEEE 1687 networks was introduced in [4].

Table I: Structural Dependencies for all the ScanRegisters in Figure 1(a).

Registers	Selection
C1 & C2 & C3	(True)
C4 & C5 & C6	(C2)
C7 & C8 & C9	(C2 $\wedge$ C4)
C10 & C11	(C2 $\wedge$ C6)
C12	(C2 $\wedge$ C4 $\wedge$ C9)
R1	(C2 $\wedge$ C4 $\wedge$ C7)
A	(C2 $\wedge$ C4 $\wedge$ C9 $\wedge$ C12)
B	(C2 $\wedge$ C4 $\wedge$ C9 $\wedge$($\neg$ C12))

Pattern retargeting to a register (R) is a computational process that tries to set the network to a state that evaluates the selection of the target register to True, by shifting one or more configuration scan vectors to the network, each within a Capture-Shift-Update cycle (CSU). The configuration vectors satisfy the inter-register temporal dependencies, which result in ultimately activating the target register.

Previous works discussing retargeting methodologies have approached it as a Boolean Satisfiability (SAT) modelling and computational problem, e.g. [2] and [8]. In SAT-based retargeting, the network is modelled to capture its structural organization, then the temporal behaviour is modelled by unrolling the SAT instances over a number of transitions. These

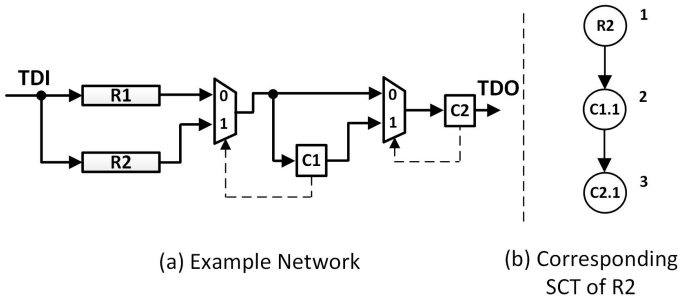

Figure 2: The corresponding SCT for the network in Figure 1(a).

transitions correspond to the application of configuration vectors in several CSUs, until the network state includes the target register in the active scan path. The resulting sequence of vectors becomes the output of the pattern retargeting process.

In [6] we introduced a different method for performing pattern retargeting based on resolving the inter-register temporal dependencies in a structured manner. The method analyzes the structural organization of the network and extracts the inter-register structural dependencies (i.e. the selections). The inter-register temporal dependencies can be further extracted from the structural ones, and using a dedicated tree traversal algorithm, one can generate the configuration vectors to access the target register. The temporal modelling of IEEE 1687 networks and structured retargeting will be briefly explained in the following sections since they form the basis of this work.

III. Temporal Modelling

A register is said to have a temporal dependency on another one when the latter is required to be set to a certain value in order for the former to become accessible after a certain number of subsequent CSUs. For example, in the network shown in Figure 2(a), in order to access R2, C1 is required to be set to '1', which in turn requires C2 to be set to '1' in order to make C1 accessible. While R2 can be shown not to have a "structural" dependency on C2, however, R2 is said to have a "temporal" dependency on C2.

A. The SCB Chaining Trees (SCTs)

The inter-register temporal dependencies can be modelled as a directed rooted tree data-structure $T = (V, E, v_r)$. The root (v_r) represents the target register, while a child node ($v \in V$) represents an SCB literal in the parent selection clause. The resulting tree is referred to as the SCB Chaining Tree (SCT).

Figure 2(b) shows the SCT for R2 in the network shown in Figure 2(a), where Sel(R2) = C1, Sel(C1) = C2 and Sel (C2) = True. Here the root node (v_1) representing the target register has a single child node (v_2) representing the single literal in Sel(R2), and similarly, v_2 has a single child node (v_3) representing the single literal in Sel(C1). Finally, v_3 has no children since Sel(C2) = True.

Except for the root, each node has a satisfying value ($sat\_val(v) \in \{0, 1\}$) according to whether the corresponding literal was negated or not. For instance, in Figure 2(b), $sat\_val(v_2)$ = $sat\_val(v_3)$ = 1 since both the corresponding literals in Sel(C1) and Sel(C2) were not negated.

A node in the SCT has two dynamic variables according to the network state: 1- *active* and 2- *satisfied*, which result in four different node states. A node is considered to be satisfied if the current state of the corresponding SCB is equal to the node's sat_val. A node is considered to be active if the network is configured to include the corresponding SCB in the active scan path, ***which occurs by satisfying all its children***.

B. Temporal properties of Hierarchical Networks

Hierarchical IEEE 1687 networks are a special type of networks that are constructed using a hierarchical organization of SIBs. The hierarchy could be viewed as a tree with the leaves being the instruments.

In general, the selection of a register "R" (either an SCB or a TDR) at level (l) of the network is given recursively using the selection of the SCB of its parent SIB as shown in equation (1). While all the SCBs of the first level SIBs have their selections equal to True as shown in equation (2). For example, in Figure 1(a), the selection of R1 is derived from the selection of its parent (i.e. C7) as (Sel(C7) $\wedge$ C7), which can be verified using Table I, as well for the remaining registers.

$$Sel(R_l) = Sel(C_{parent}) \wedge C_{parent}, \forall (l > 1) \quad (1)$$

$$Sel(C_1) = True \quad (2)$$

For instruments with a special TDR organization for optimizing the access time [9] such as instrument "inst" in Figure 1(a), equation (1) can be extended to accommodate for the instrument-level non-hierarchical organization as follows:

$$Sel(TDR) = Sel(C_{parent}) \wedge C_{parent} \wedge F \quad (3)$$

where F is a Boolean function that when evaluated to True, (TDR) becomes selected between the host interface of the parent SIB. For instance, in Figure 1(a), Sel(B) = (Sel(C9) $\wedge$ C9 $\wedge$ ($\neg$ C12)), here F = ($\neg$ C12).

Equations (1) and (3) present the selection of a register in a hierarchical network in terms of the selection of the SCB in the parent SIB. This recursive nature of the structural dependency in hierarchical networks results in a unique property of the corresponding SCTs, where many identical temporal dependencies exist.

For example, Figure 3 shows the corresponding SCT to the hierarchical network shown in Figure 1(a). It is clear that the subtree beneath v_5 (i.e. T2) is identical to the subtree beneath v_1 excluding the branch leading to v_5 (i.e. T1). Such redundancy will be exploited in our optimized structured retargeting for hierarchical networks that is presented in the next section.

Finally, the number of nodes (N) of an SCT of a TDR in a purely hierarchical network can be calculated as follows:

$$N = 2^L \quad (4)$$

where L is the hierarchical level at which this TDR is located. For instance, register R1 in Figure 1(a) is located at level 3, and hence its SCT has 8 nodes.

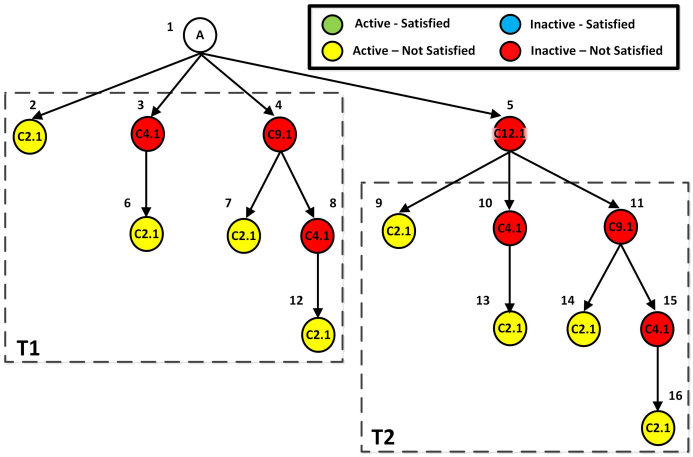

Figure 3: Post-reset SCT of the network shown in Figure 1(a).

IV. Structured Retargeting for Hierarchical Networks

In [6] we have presented a structured retargeting methodology for arbitrary SCTs. The main goal of this method is to activate the root node, which corresponds to configuring an active scan path that includes the target register represented by the root node. This is iteratively accomplished by satisfying the set of currently active nodes, which is done by shifting a corresponding scan vector, such that their parents become activated. Hence, the parents could be further satisfied, which ultimately leads to the root node being activated.

For example, in the SCT shown in Figure 2(b), and assuming the network at the reset state (all SCBs set to '0'), one starts by satisfying v_3, which is the only active node, by shifting a corresponding scan vector. Hence v_2 becomes activated and could be further satisfied, after which v_1 would become active. This is performed in [6] using a tree traversal referred to as the Modified Depth First Search (MDFS) traversal. In MDFS, the tree is completely traversed in a depth-first fashion, while assuming every node that is either satisfied or active as a leaf one. The resulting set of active nodes are further reduced to resolve any conflicting satisfying values for the same SCB.

A. Optimized SCT Traversal for Hierarchical Networks

In MDFS, it is attempted to traverse the entire tree before producing the set of active nodes, which is further reduced for conflicts resolution. Subsequently, one configuration scan vector is generated to satisfy the resulting set of active nodes.

While this traversal ensures producing satisfying values for all the currently active nodes which optimizes the number of required CSUs for retargeting, for hierarchical networks with deep hierarchies, traversing all the nodes in the SCT might lead to a long retargeting time due to the exponential node count as shown in Equation (4). In addition, it also increases the hardware requirements for maintaining the temporal dependencies in the case of on-chip retargeting [4].

As discussed earlier, SCTs of hierarchical networks incorporate several identical nodes. This property could be exploited such that a complete SCT traversal becomes unnecessary.

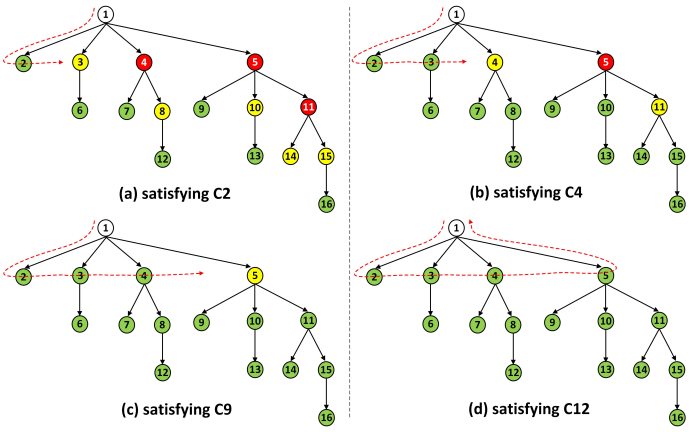

Figure 4: Applying MPO-DFS to the SCT in Figure 3.

Instead of traversing the entire SCT before satisfying the active nodes, one can start satisfying the active nodes once reached by the traversal algorithm, and hence, dynamically changing the network state while traversing it. Since the traversal does not attempt to further traverse a branch when an active or satisfied node is reached, dynamically satisfying the redundant nodes will reduce the tree traversal path.

This can be illustrated in Figure 4, where dynamically satisfying the active nodes once reached is applied to the SCT shown in Figure 3. In MDFS, the traversal would start with traversing the entire SCT (16 nodes) in order to generate the first vector which sets C2 to '1'. In our optimized approach, once v_2 is reached and found to be active, the algorithm would immediately generate a configuration vector that sets C2 to '1' and dynamically change the network state where all nodes corresponding to C2 become active (Figure 4(a)). This is further applied to nodes v_3, v_4 and v_5, and then the root node becomes active after only traversing the nodes at the first level.

For purely hierarchical networks, the traversal would simply traverse only the first level of the SCT. However, in order to support arbitrary networks as well, a generic traversal algorithm was developed as shown in Algorithm 1.

In Algorithm 1, a depth-first search traversal similar to MDFS is applied, however, with a post-order processing, referred to as the Modified Post-Order Depth First Search (MPO-DFS). This means that children nodes are attempted to be satisfied first before their parents during the traversal. The inputs to the algorithm are the target register (R), the registers selections (Sel) and the current network state (St). While the output is a set of Access Vectors (AV) that are applied to configure the network to access the target register (R).

It is shown in line 16 that the processing of a node (i.e satisfying it when it becomes activated) occurs after the return of the recursive MPO-DFS function calls on all the children. In this case, this node was initially not active at the start of the tree traversal; however, it was activated by satisfying its children during the recursive function call in line 15.

Algorithm 1 MPO-DFS.

Input: R, Sel, St
Output: AV

```
1:  Create the root node (r)
2:  r.id ← R
3:  while r is not active do
4:      MPO-DFS(r)
5:  procedure MPO-DFS(v)
6:      Create children nodes of v according to Sel(v.id)
7:      v.children ← created children nodes
8:      Mark v as visited
9:      if v is not active then
10:         if v is not satisfied then
11:             for all w ∈ v.children do
12:                 if w is not visited then
13:                     w.id ← corresponding id as in Sel(v.id)
14:                     w.depth ← v.depth + 1
15:                     MPO-DFS(w)
16:             if v is active then        ▷Function return. Post-order
17:                 Satisfy v by shifting a corresponding AV
18:     else
19:         if v is not satisfied then ▷Avoid generating an AV for a
    satisfied node
20:             Satisfy v by shifting a corresponding AV
```

B. Dynamic Conflict Resolution

For purely hierarchical networks, there exist no temporal conflicts in the corresponding registers SCTs. However, in order to ensure a generic solution for arbitrary networks, temporal conflicts should be considered.

In MDFS, temporal conflicts were resolved by selecting the active node that has the largest distance from the root among all nodes that represent a certain register. In MPO-DFS temporal conflicts are dynamically resolved by the traversal with no special selection of the nodes like in MDFS.

Figure 5(a) shows a network with conflicting temporal dependencies for register R2 as shown in Figure 5(b). Figures 5(c-h) show the dynamic resolution with the MPO-DFS traversal. The traversal starts from the left-most branch and satisfies every active and not satisfied node, until all the children of the root node are satisfied.

Considering the two cases where a node (v_2) has a conflict with a descendant of a right sibling, and another node (v_4) with a conflict with a descendant of a left sibling. One can see that going through the traversal, all the children of the root node will be eventually satisfied (Figure 5(h)). Although satisfying v_2 and v_4 resulted in de-satisfying v_5 and v_6, however, they became irrelevant to satisfying the root node since their parent v_3 was already satisfied.

It can be shown that if the order of the children nodes of the root was changed such that node v_3 is swapped with node v_2, pattern retargeting for R2 will require only 5 CSUs instead of the required 6 CSUs for the original SCT. In this case the sequence of traced and satisfied nodes will be ($v_5 \rightarrow v_6 \rightarrow v_3 \rightarrow v_2 \rightarrow v_4$). Therefore, reordering the children nodes for MPO-DFS such that the deepest branches are traversed first will reduce the network access time in case of conflicts.

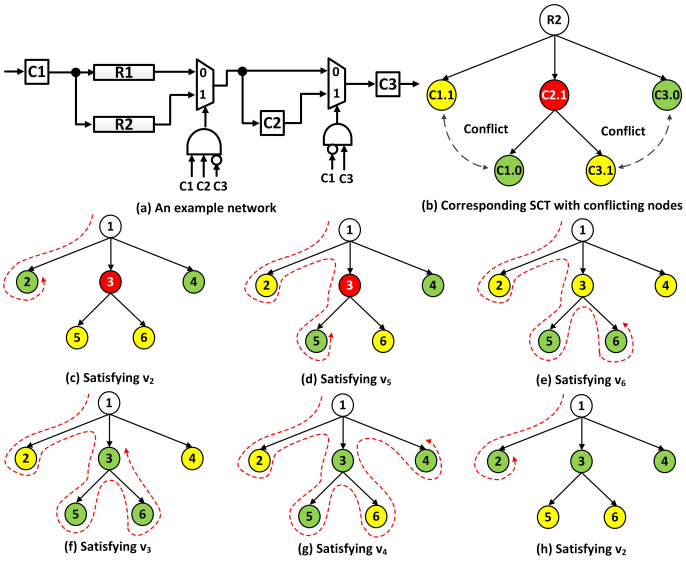

Figure 5: Dynamic conflict resolution in MPO-DFS.

V. Experimental Results

The developed retargeting method was verified on a subset of the IEEE 1687 benchmark suite [10] and further evaluated against the previous work presented in [6].

A. The Experimental Setup

We performed our experiments on the networks presented in the "Basic" category [10] since they provide a good diversification of the network organization. They range from pure hierarchical networks (TreeFlat and TreeUnbalanced), pure hierarchical with optimized TDR organization (TreeFlatEx and TreeBalanced), semi-hierarchical with ScanMuxs inserted within the hierarchy presenting a few temporal conflicts (Mingle) to finally an arbitrary network with many temporal conflicts (BasicSCB). The TDRs in all networks are not located at a single level of the hierarchy, but rather they are distributed in different branches and levels in the hierarchy, resulting in different SCT complexities for the TDRs in the same network.

Table II lists the structural statistics of the 6 networks presented in the Basic category. The second column shows the maximum hierarchical levels in the network (L), while the third shows the number of SIBs, and the fourth shows the number of TDRs. Finally the fifth column shows the maximum number of nodes (N) in a single SCT of a TDR in this network and the name of this TDR. Note that there might be several TDRs with the same maximum node count.

Table II: Network statistics of the Basic category in [10].

Network	Max L	No. of SIBs	No. of TDRs	Max N
BasicSCB	-	0	5	16 (WI2.SReg)
Mingle	3	10	8	16 (WI3.SReg)
TreeFlat	2	12	11	8 (All)
TreeFlatEx	5	49	78	32 (M4.SR4)
TreeBalanced	7	43	52	128 (M8.SR4)
TreeUnbalanced	11	28	40	2048 (M1.SR8)

A software implementation of our optimized retargeting method shown in Algorithm 1 was developed, along with the previously developed generic method in [6]. The selections of each TDR and SCB in the network were calculated and provided to the software for the SCT generation. Furthermore, a software structural model for each network was developed, for verifying the target register accessibility using the generated scan patterns from the retargeting module.

Single register access experiments were performed for each TDR in the network, starting from the network at the reset state using both MDFS and MPO-DFS SCT traversals.

We evaluated both methods regarding their execution times in terms of the number of traced nodes during the consecutive traversals that are required for activating the root node. The number of traced nodes is especially important for the hardware implementation of structured retargeting.

In addition, the quality of the generated patterns was evaluated in terms of both the number of required Capture-Shift-Update Cycles (CSUs) and the access time in clock cycles (CC) that are required to configure the network to include the target TDR in the active scan path. The access time can be calculated using equation (5), where N is the number of CSU cycles and l_i is the length of the shift vector in the i^{th} CSU cycle, and the constant 4 is the overhead of the CSU cycle being caused by the IEEE 1149.1 TAP FSM.

$$Access\ Time = \sum_{i=1}^{N} l_i + N \times 4 \tag{5}$$

B. Analysis of the Results

Table III shows the results of the single access experiments. For each network, the average and maximum values of the traced nodes are shown, as well as the number of CSUs and the access time after attempting to access each TDR in this network.

For the BasicSCB network, it is clear that MDFS outperforms MPO-DFS in both retargeting and access times. This occurs not only because the SCTs do not have much identical temporal dependencies such that they dynamically reduce the traversal path length, but also since there exist an abundance of temporal conflicts that increase the traversal path length.

Reordering the SCTs in the case of MPO-DFS leads to better results such that a reduction of the number of traced nodes of 50% of the average and 44% of the maximum was achieved. Also a reduction of the number of CSUs of 45% of the average and 40% of the maximum, and a reduction of the access time of 44% of the average and 39% of the maximum were achieved. However, these enhancements are still below the performance achieved with MDFS.3

Furthermore, it can be shown that for the remaining networks with hierarchical organizations, the number of traced nodes in case of MPO-DFS were significantly less than these of their MDFS counterparts due to the existence of many identical temporal dependencies. For instance, in the TreeUnbalanced network, the MDFS method was required to visit 6131 SCT nodes in order to perform pattern retargeting

Table III: Results of structured retargeting for single register access

Network	MDFS						MPO-DFS					
	Traced Nodes		CSUs/access		Access Time (CC)		Traced Nodes		CSUs/access		Access Time (CC)	
	av	max	av	max	av	max	av	max	av	max	av	max
BasicSCB	9.4	16	1.4	2	15.8	23	14.2	18	4.4	5	48.6	57
Mingle	27.4	42	3.3	4	26	35	5.9	8	3.4	4	27.3	35
TreeFlat	11	11	2	2	23	23	5	5	2	2	23	23
TreeFlatEx	10.2	89	2.1	5	50.7	428	4.1	7	2.1	5	50.7	428
TreeBalanced	167	375	5.4	7	235.6	2164	7.4	9	5.4	7	235.6	2164
TreeUnbalanced	341.1	6131	4.5	11	4570.5	60427	6.5	13	4.5	11	4570.5	60427

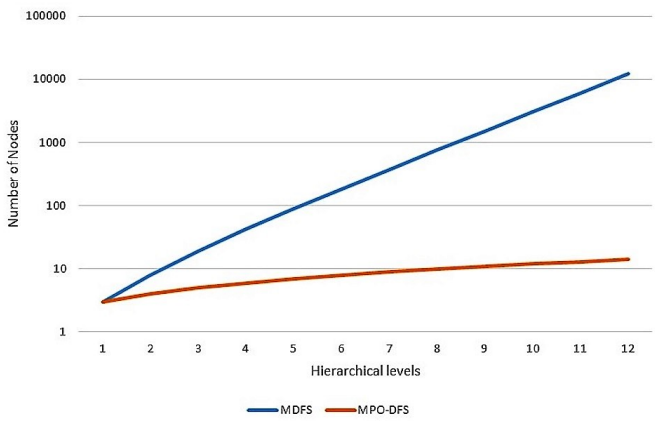

Figure 6: Effect of the increased hierarchical levels on the number of traced nodes for the two structured retargeting methods.

to register M1.SR8, while only 13 nodes were visited in the case of MPO-DFS. A large number of traced nodes might be tolerable in software, for instance in the case of debuggers, however this becomes a clear problem for a hardware retargeter, since all the temporal dependencies need to be resolved and stored for the retargeting processing.

Figure 6 shows the number of traced nodes for both MDFS and MPO-DFS that are required for a TDR access in balanced hierarchical networks with different hierarchical levels. It can be shown that the traced nodes exponentially increases in the case of MDFS with the increase of the TDR's depth in the hierarchy, while it linearly increases in the case of MPO-DFS.

Finally, the quality of the produced vectors by MPO-DFS in terms of the number of CSUs and the access time were slightly lower in case of the Mingle network than those produced by MDFS, since there exist a few of temporal conflicts as the network is not purely hierarchical. While for the remaining networks, the quality of the produced vectors in both methods were identical. The long access times in the last three networks occur due to the structural organization of the network where several long TDRs exist in the access path to other TDRs.

VI. Conclusions

Dynamic pattern retargeting is performed simultaneously while accessing IEEE 1687 networks; therefore reducing the retargeting time becomes important for reducing the overall test time. In this paper we presented a novel structured retargeting method that is optimized for hierarchical networks. The presented method results in generating an identical retargeted pattern set to the one generated by the previous work resulting in the same access times, while significantly reducing the retargeting time for hierarchical networks in terms of the number of traced nodes.

The number of traced nodes is an important parameter for a hardware implementation of structured retargeting, since the hardware retargeter is required to dynamically resolve the temporal dependencies (i.e. the SCT nodes), and perform the SCT traversal on those resolved dependencies. Therefore, reducing the number of traced nodes in structured retargeting will result in a less hardware complexity for maintaining the dynamically resolved dependencies.

The method is still applicable to the less common non-hierarchical networks although with a degraded performance, and consequently can be utilized as a generic retargeting solution for software debuggers or EDA tools which generate IEEE 1687 network controller hardware IPs.

VII. Acknowledgement

Parts of this research were carried out within the EU-PENTA project "HADES", financed by the European Commission (EC) and the Netherlands Enterprise Agency (RVO).

References

[1] IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device, IEEE Std 1687-2014, 2014.

[2] R. Baranowski, M. A. Kochte, and H.-J. Wunderlich, "Reconfigurable Scan Networks: Modeling, Verification, and Optimal Pattern Generation, ACM Trans. Design Automation of Electronic Systems (TODAES), vol. 20, no. 2, pp. 1-27, 2015.

[3] M. Portolan, "A novel test generation and application flow for functional access to IEEE 1687 instruments," European Test Symposium (ETS), 2016, pp. 1-6.

[4] A. Ibrahim and H. G. Kerkhoff, "Analysis and Design of an On-Chip Retargeting Engine for IEEE 1687 Networks," European Test Symposium (ETS), 2016, pp.1-6.

[5] A. Ibrahim and H. G. Kerkhoff, "A cost-efficient dependability management framework for self-aware system-on-chips based on IEEE 1687," International Symposium on On-Line Testing and Robust System Design (IOLTS), 2017, pp. 1-2.

[6] A. Ibrahim and H. G. Kerkhoff, "Structured scan patterns retargeting for dynamic instruments access," VLSI Test Symposium (VTS), 2017, pp.1-6.

[7] Steve Pateras and Mike Santarini, "Tessent MissionMode: New, Runtime DFT Technology Paves Way for Self-Correcting Automotive Electronics" [Online], Available: : http://go.mentor.com/4XrS8

[8] R. Krenz-Baath, F. G. Zadegan and E. Larsson, "Access time minimization in IEEE 1687 networks," International Test Conference (ITC), 2015, pp. 1-10.

[9] A. Ibrahim and H. G. Kerkhoff, "iJTAG integration of complex digital embedded instruments," International Design & Test Symposium (IDT), 2014, pp.18-23.

[10] A. Tertov et al., "A suite of IEEE 1687 benchmark networks," International Test Conference (ITC), 2016, pp. 1-10.