Access Time Analysis for IEEE P1687

Farrokh Ghani Zadegan, *Student Member, IEEE,* Urban Ingelsson, *Member, IEEE,*
Gunnar Carlsson, *Member, IEEE,* Erik Larsson, *Senior Member, IEEE*

Abstract—The IEEE P1687 (IJTAG) standard proposal aims at providing a standardized interface between the IEEE Standard 1149.1 test access port (TAP) and on-chip embedded test, debug and monitoring logic (instruments), such as scan-chains and temperature sensors. A key feature in P1687 is to include Segment Insertion Bits (SIBs) in the scan-path to allow flexibility both in designing the instrument access network and in scheduling the access to instruments. This paper presents algorithms to compute the overall access time (OAT) for a given P1687 network. The algorithms are based on analysis for flat and hierarchical network architectures, considering two access schedules, i.e. concurrent schedule and sequential schedule. In the analysis, two types of overhead are identified, i.e. network configuration data overhead and JTAG protocol overhead. The algorithms are implemented and employed in a parametric analysis and in experiments on realistic industrial designs.

Index Terms—Access time calculation, IEEE P1687 IJTAG, P1687 network architectures, instrument access schedules

1 Introduction

The complexity and reduced feature sizes in recent integrated circuit (IC) designs, necessitates the provision of on-chip infrastructures for test, debug and monitoring. The IEEE Standard 1149.1 (a.k.a. JTAG[1]) [1], originally intended for board test, has proved useful in ad hoc access to such on-chip infrastructure, as discussed in [2]. However, since JTAG has not been originally meant for accessing such on-chip test, debug and monitoring logic (collectively called *instruments*), there are some drawbacks associated with its use in this context [2]. Furthermore, due to the lack of a uniform method of using JTAG to access the on-chip instruments, electronic design automation (EDA) support in this context is limited. Therefore, there is a need to standardize how JTAG circuitry should connect to the embedded logic. The IEEE P1687 standard proposal [2], [3], [4], [5], [6] aims to address this need of standardization by describing a flexible data transport infrastructure (called *network*) to interface JTAG to the chip internal instruments. P1687 has therefore received the informal name of Internal JTAG (IJTAG). When ratified, P1687 will specify methods for access and control of embedded instruments [3]. Here, instrument refers to any device with a shift-register [7] that could be included in the JTAG scan-path. Examples of instruments include embedded sensors, internal scan-chains and IEEE standard 1500 wrapped cores [8].

Compared with similar efforts for interfacing the on-chip instruments with JTAG, e.g. the works presented in [9], [10], [11], [12], [13], P1687 is characterized by introduction of a single JTAG instruction called GateWay ENable (GWEN) and a hardware component called Segment Insertion Bit (SIB). The use of SIBs makes it possible to create a multitude of different networks for the same set of instruments, and to have the benefit of flexibility in scheduling the access to those instruments, as will be discussed in this paper. To setup the scan-path, P1687 proposes to transport SIB control data together with instrument data on a single wire (the JTAG scan-path), and this will affect overall access time (OAT).

Since IEEE P1687 has recently been proposed, only a few studies have considered it [4], [9], [14], [15]. In particular, no study has investigated the impact of IEEE P1687 on overall access time (OAT). Therefore, this paper analyzes (Sections 4, 5, and 7) the impact of IEEE P1687 on OAT and identifies two types of overhead associated with accessing embedded instruments using P1687, namely network configuration data (i.e. SIB control data) overhead and JTAG protocol overhead. Furthermore, it is shown that OAT depends on several parameters including the placement of instruments and SIBs. To make it possible to calculate OAT for large P1687 networks, a first OAT calculation tool, IJTAGcalc, is presented (Section 6) for both concurrent and sequential schedules. IJTAGcalc is used in a parametric analysis (Section 7) to investigate how OAT and the identified overhead types react to changes in the design variables. IJTAGcalc is also used to calculate OAT for a number of P1687 networks based on ITC'02 benchmark set (Sections 8 and 9).

2 Background and Prior Work

In this section, JTAG is briefly introduced and the drawbacks associated with its application in accessing the on-chip instruments are explained. It should be noted that such application is beyond the intended scope of JTAG. Furthermore, it will be discussed how P1687 addresses these drawbacks.

- *F. Ghani Zadegan, U. Ingelsson and E. Larsson are with the Department of Computer and Information Science, Linköping University, Sweden, SE-581 83, Linköping, Sweden e-mail: ghanizadegan@ieee.org, urban.ingelsson@liu.se, erik.larsson@liu.se.*
- *G. Carlsson is with Ericsson AB, Sweden e-mail:gunnar.carlsson@ericsson.com.*
- *The preliminary version of this work has been presented in the Asian Test Symposium, Shanghai, 2010.*

1. Joint Test Action Group

Digital Object Indentifier 10.1109/TC.2011.155

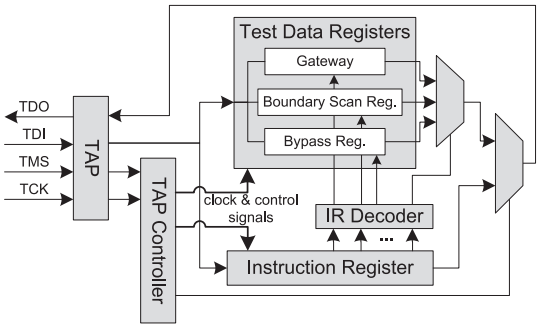


Fig. 1. A conceptual view of JTAG circuitry and how P1687 *Gateway* is interfaced to JTAG

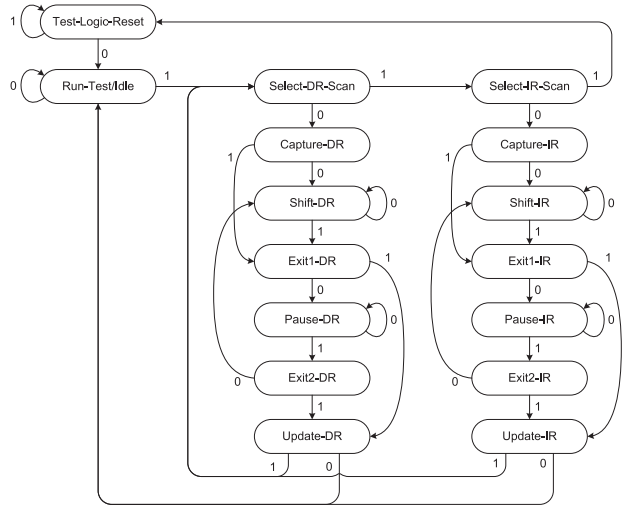


Fig. 2. JTAG TAP Controller state diagram

Fig. 1 shows a conceptual view of JTAG circuitry in a chip [1]. Accessing the on-chip JTAG circuitry is done through the test access port (TAP) which includes four mandatory signals, namely test data input (TDI), test data output (TDO), test mode select (TMS) and test clock (TCK). The TMS signal is decoded by a state diagram (see Fig. 2) to generate the control signals required for the *capture*, *shift* and *update* operations on instruction register (IR) and test data registers (TDRs). The capture operation is defined as parallel loading a value into the IR (or any of the TDRs), the update operation is defined as transferring logic values from the shift-register stage of the IR (or any of the TDRs) to their latched parallel outputs, and the *shift* operation is defined as shifting the data serially into and out of the IR (or any of the TDRs) one bit per TCK [1].

The state diagram (Fig. 2) is implemented in the TAP Controller, see Fig. 1. From Fig. 2 it can be seen that the state diagram has two similar branches, (1) the IR branch used for performing operations on the IR (IR-Scan) and (2) the DR branch used for performing operations on the current TDR (DR-Scan). The current TDR is selected by the instruction currently loaded into the IR, which is decoded by the IR Decoder, see Fig. 1. Input vectors are shifted into the selected TDR by shifting the data when the TAP Controller is in the Shift-DR state. By keeping TMS at logic '0' it is possible to shift in as many bits as required. Moving to the Update-DR state makes the shifted vector appear at the parallel outputs of the TDR. The data that should be parallel loaded into the TDR, i.e. the output vectors, are captured at the Capture-DR state and are shifted out by moving to the Shift-DR state. It is possible to shift in the next input vector while shifting out the output vector corresponding to the previous input vector. The following sequence of five states, Exit1-DR, Update-DR, Select-DR, Capture-DR and Shift-DR, will be assumed and called a CUC (Cycle of Update and Capture) in the rest of this paper for applying inputs and capturing outputs between two shift operations.

The on-chip instruments serve different purposes such as testing (e.g. internal scan-chains, IEEE Standard 1500 wrapped cores [8], and BIST engines), debugging and diagnosis (e.g. shadow capturing of key registers [2]), monitoring (e.g. temperature sensors), and configuration (e.g. SERDES characterization). From [7] it can be understood that in most cases the interface to the instruments is a shift (and update) register. As examples of efforts to connect the embedded instruments to JTAG TAP, the following can be mentioned. In [10], [11], [14] IEEE Standard 1500 wrapped cores are connected to the JTAG TAP, in [16] integrity loss sensors are accessed through JTAG TAP, and in [12], [13] internal scan-chains are connected to/combined with JTAG boundary scan register.

JTAG is a successful and widely used standard, and the JTAG TAP is available on most modern ICs [2]. But there are three drawbacks associated with the use of JTAG TAP to access the embedded instruments, namely (1) posing a trade-off between scalability of hardware and flexibility in scheduling the access to the instruments, (2) lack of a language suitable for describing all sorts of instruments, and (3) lack of a language related to the JTAG description that describes the operation of the instrument independent of the placement, configuration, or use of that instrument in the overall access mechanism.

The first drawback can be explained by assuming that each instrument is added to the JTAG circuitry as a separate TDR, to allow the individual access to each of the instruments. In this scenario, the JTAG circuitry does not scale well with the increase in the number of instruments, either because the instruction register (IR) becomes too long or the IR Decoder becomes too complex. To have an idea of the number of instruments in a modern SOC, consider an ASIC from Ericsson which contains 64 cores, each core having its dedicated data and instruction memories. This ASIC also contains a number of SERDESs and hardware accelerators. Therefore, there are more than 200 blocks of logic inside this ASIC where each block may contain MBIST, LBIST, sensors, etc., which can be regarded as on-chip instruments. It can be seen that the number of instruments in this ASIC would amount up to several hundreds. It should be noted that chaining all instruments into a single scan-path

helps to avoid the scalability problem—by requiring only one instruction for selecting the scan-path and accessing the instruments—at the cost of losing flexibility in scheduling the access to instruments. The loss of such flexibility might in turn increase the instrument access time. In addition, such an architecture has a high risk of failure, since any problem with the scan-path, e.g. stuck-at-fault, will render all instruments in that scan-path inaccessible. Section 8 and Section 9 will present experiments to elaborate on the drawback of using such *chain of instruments*.

The second drawback with using JTAG to access the instruments is that boundary scan definition language (BSDL) which is part of the JTAG standard and is used to describe the boundary scan devices, is neither efficient nor sufficient to describe all types of instruments [15].

The third drawback with using JTAG is that there is no portable vector or portable procedure language that can be used to describe the operations associated with the instrument regardless of where and how that instrument is used. The serial vector format (SVF) [17], which is used to describe the JTAG operations, is written at the chip-level, not the instrument level. Furthermore, SVF files must be recreated for any changes in the configuration, i.e. length of TAP IR, length of the instrument interface shift-registers and their placement order on the scan-path, etc.

To address the drawbacks associated with the use of JTAG in accessing the embedded instruments, the IEEE P1687 standard proposal standardizes the way the embedded instruments are accessed through JTAG TAP by (1) proposing an interface between JTAG TAP and on-chip instruments to introduce flexibility and scalability into the JTAG scan-path, (2) proposing an instrument connectivity language (ICL) to describe the characteristics of the instruments and the requirements for interfacing to them [3], and (3) a procedural description language (PDL) to describe the operation of an instrument independent of its placement, to facilitate the re-targeting of the vectors for that instrument to chip-level and board-level. The focus of this paper is only on the hardware aspects of P1687.

The interface proposed by P1687 for connecting JTAG TAP to instruments is implemented by adding a TDR called *Gateway* to the JTAG circuitry, see Fig. 1. The Gateway is selected by loading an instruction called GateWay ENable (GWEN) through IR-Scan, which makes the Gateway accessible from the TDI and TDO terminals [1], [3]. Once the GWEN instruction is set, any further access, configuration and control of instruments through P1687 will be done through DR-Scans [1], [3], as will be detailed in Section 3.

There has been previous work on P1687. In [4] a possible implementation of P1687 is demonstrated in a case study of characterization and test of high-speed serial I/O links, and a four-layer API is presented for developing the instrument access procedures which allows easy reuse and portability of the test algorithms. In [15] a scan description language is suggested and is evaluated in the context of P1687. It should be noted that although P1687 has attracted interest in the research community [9], [14], no study has so far considered the instrument access time in the context of P1687, which is the focus of this paper.

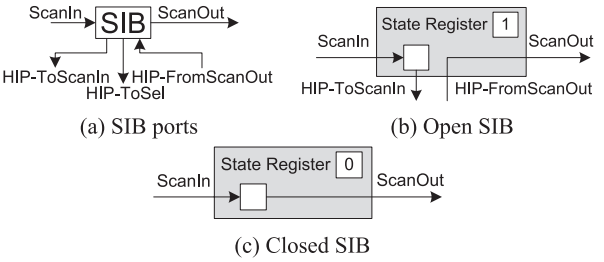


Fig. 3. Simplified view of the SIB component

The IEEE P1687 standard proposal is, at the time of writing of this paper, in the final stage of review but still subject to changes. However, even in case of changes in the hardware aspects of the final standard draft, the analysis approach presented in this work can still be applied to the changed draft, or basically to any similar situation where instruments on a multiplexed scan-path are accessed through a JTAG TAP.

3 P1687 Hardware

As was noted in Section 1, P1687 introduces a new hardware component called SIB, used to set up the scan-path for P1687 networks. The P1687 Gateway itself is made from one or several SIBs. Fig. 3(a) shows a simplified view of a SIB. Besides the ScanIn and ScanOut terminals, SIB has a hierarchical interface port (HIP) used to connect to a P1687 network segment. A segment can be either simply an instrument or composed of other SIBs. A SIB acts as a doorway with two states. It is either open (Fig. 3(b)) and includes the segment on its HIP the in scan-path (hence the name Segment Insertion Bit), or it is closed (Fig. 3(c)) and transfers the data from its ScanIn port to its ScanOut port, excluding the segment on its HIP.

The use of SIBs to include the shift-registers of instruments in the scan-path, or exclude them from the scan-path, can be compared to using multiplexers in daisychain architecture to include/exclude scan-chains [18]. As shown in Fig. 4, in the daisychain architecture, a long scan-path is formed out of the available scan-chains. By providing multiplexers and bypass registers, it is possible to shorten the path to a certain scan-chain by bypassing the others. Here, control signals are provided separately from scan-chain data. That is, the control signals should either be provided from the chip pins or—as shown for the a, b and c control signals in Fig. 4—through a shift-register accessible from a control pin. This is in contrast to P1687 which proposes to transport control data together with instrument data on a single wire, i.e. the TDI-TDO path of JTAG.

The hierarchical interface port (HIP) has three terminals: HIP-ToScanIn, HIP-FromScanOut and HIP-ToSel. HIP-ToScanIn and HIP-FromScanOut connect

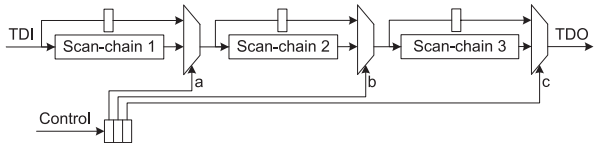
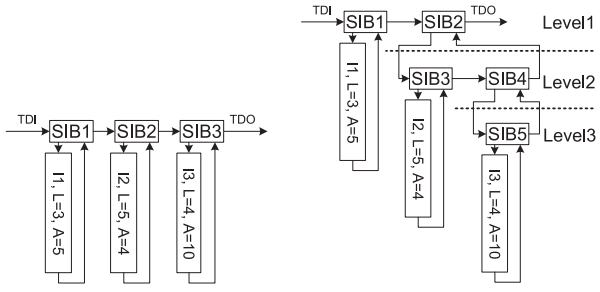


Fig. 4. The daisychain architecture



(a) Flat architecture (b) Hierarchical architecture

Fig. 5. Example P1687 instrument access networks

the scan-path to the network segment connected to the HIP when the SIB is open. The HIP-ToSel signal is activated when the SIB is open, to enable the segment connected to the HIP. To clarify the need for HIP-ToSel it should be noted that a SIB, in addition to the terminals shown in Fig. 3(a), has clock, select and three control (i.e. shift-enable, capture-enable and update-enable) input terminals. The clock and control signals are shared among all components in a P1687 network and the control signals should be gated separately for each SIB using its select input. For example, consider that SIB s_1 is to be accessed through the HIP of SIB s_2. In this case, the HIP-ToSel output of s_2 should be connected to the select input of s_1. In a similar way, the assumed interface for the instruments also requires the clock, select, and control inputs in addition to the serial data in and serial data out signals. Here again, the control signals are shared among all instruments (and SIBs) in the network and should be gated for each instrument by using its select input. That is, when an instrument is to be accessed through the HIP of a SIB, the HIP-ToSel output of the SIB should be connected to the select input of that instrument. However, to keep the figures in this paper simple, the HIP-ToSel signal is not shown, but it is assumed that whenever a SIB is open, the instrument or the network segment connected to its HIP is enabled.

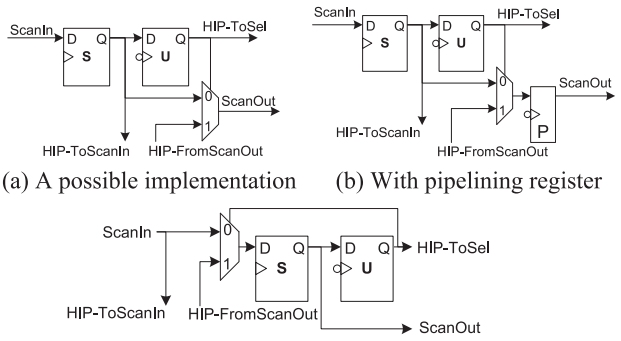
The state of SIBs in a P1687 network is set by embedding control bits in each input vector such that after being shifted in, when the TAP Controller is in the Shift-DR state, each control bit is placed into the register of its intended SIB. The control bit for each SIB is then transferred into the SIB's State Register, shown in Fig. 3(b) and Fig. 3(c), once the TAP Controller is in the Update-DR state. Moving to Update-DR and back to Shift-DR for shifting out the output vector and shifting in the next input vector is part of the cycle of update and capture (CUC) explained in Section 2.

Fig. 5(a) shows a P1687 network of three instruments (I1, I2 and I3) and three SIBs, one for each instrument. The control, select and clock signals are not shown. In Fig. 5(a), L stands for the length of the shift-register for each instrument and A stands for the number of instrument-specific *accesses*. In this paper, *access* is defined as (1) shifting input bits into the instrument's shift-register, (2) latching the contents of the shift-register to be applied to the internal circuitry of the instrument, (3) capturing the output of the instrument into the shift-register and (4) shifting the captured values out. The shifting-out of the instrument outputs can overlap in time with shifting-in the input command bits for the next access. Considering the relatively slow P1687 clock (i.e. TCK applied to JTAG TAP) [4], [5], it is assumed that the time it takes an instrument to process the applied inputs and make the outputs ready to be captured, is less than the time it takes to move from Update-DR to Capture-DR in the TAP Controller.

It is important to note that not all instrument types are accessed as described above. For example, a BIST engine might be selected (by opening its corresponding SIB) and activated (by launching the BIST) and then be de-selected (by closing its SIB) while still active and running. Later in the access schedule, the BIST can be selected again and its Done and Fail signals be polled. In this work, we only consider the instruments that are either selected and active, or de-selected and inactive.

The type of architecture that is implemented by the SIBs in Fig. 5(a), is called a flat architecture in the remainder of this paper. In the flat architecture no SIB is connected to the HIP of another SIB. Fig. 5(b) shows another network of the same three instruments, i.e. I1, I2, and I3. Here, there are five SIBs and three of these SIBs are connected to the TAP through the HIP of SIB2. This type of architecture is called hierarchical architecture in the remainder of this paper. Each SIB that has another SIB connected to its HIP, represents a doorway to another level of hierarchy, such as SIB2 and SIB4 in Fig. 5(b). In this paper, for the sake of terminology, a SIB having only an instrument on its HIP is referred to as an *instrument SIB* and a SIB having one or more SIBs on its HIP is called a *doorway SIB*. It should be noted that SIB1, SIB2 and SIB3 in Fig. 5(a) form the IJTAG Gateway, while in Fig. 5(b) SIB1 and SIB2 form the Gateway. The SIBs forming the Gateway receive their select signal directly from JTAG instruction decoder (IR Decoder in Fig. 1).

For the purpose of access time analysis, which is the focus of this paper, the simplified view of the SIB presented in Fig. 3 is sufficient, and the implementation details are out of the scope of this work. However, the implementation might have an impact on instrument access time, as will be briefly discussed here. One possible implementation of a SIB is presented in Fig. 6(a). As can be seen in Fig. 6(a), each SIB contains a *shift* register (S) and an *update* register (U). The U register serves as the State Register shown in Fig. 3(b) and Fig. 3(c). When the value in the U register is a logic 0, the scan-path is through the S register and the multiplexer (see Fig. 3(c)). But



(a) A possible implementation (b) With pipelining register

(c) Using a wide fan-out input and pipelining the ScanOut path using the S register

Fig. 6. A simplified RTL view of a SIB

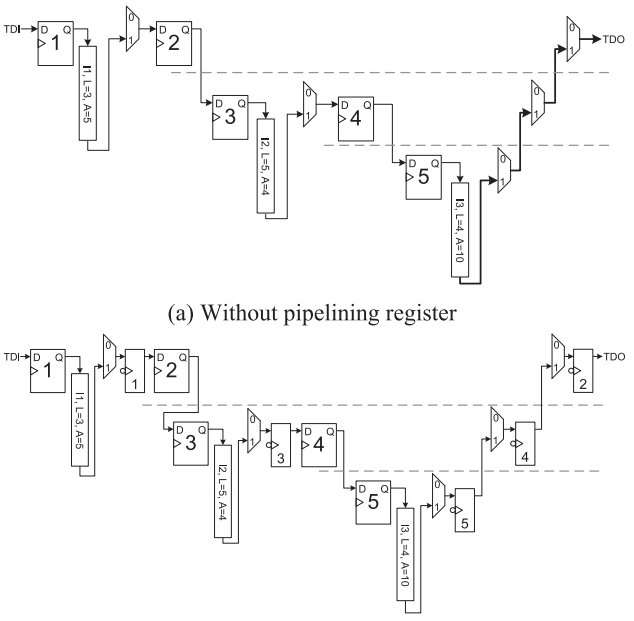
when U contains a logic 1, the scan-path will be through S to the HIP output, i.e. HIP-ToScanIn, and back from the HIP input, i.e. HIP-FromScanOut, to the multiplexer (see Fig. 3(b)). Fig. 7 shows the scan-path of the network presented in Fig. 5(b) at the RTL level, when all the SIBs are open. The thick lines in Fig. 7(a) denote a combinatorial path which can be a limiting factor for the P1687 clock when the depth of hierarchy increases, thus affecting the instrument access time. To avoid such long combinatorial paths in the P1687 networks, it is possible to add a register at the ScanOut output of the SIB, as shown in Fig. 6(b), to pipeline the combinatorial path [3].

The pipelining register acts on the negative edge of the clock. Therefore, its presence only affects the shifting of the data when two or more pipelining registers exist on the scan-path with no positive-edge triggered register in between. Dummy bits should be embedded in the input vectors such that after the vector is shifted in, the SIB control bits and the instrument data bits end up in the correct registers of the respective SIB or instrument's shift-register. The existence of two pipelining registers on the scan-path with no positive-edge triggered register in between only happens once per doorway SIB, as can be seen from Fig. 7(b). In Fig. 7(b), the pipelining register of SIB4, which is a doorway SIB, is on the scan-path immediately (i.e. with only combinational circuitry in between) after the pipelining register of SIB5, and the same is true for SIB2 (also a doorway SIB) and SIB4.

By using the implementation shown in Fig. 6(c), it is possible to pipeline the ScanOut path by using the S register itself. This, however, comes at the cost of applying a wide fan-out signal to the ScanIn inputs. Therefore, the SIB implementation in Fig. 6(c) makes it possible to avoid both long combinatorial paths and the need for scanning in dummy bits. In the rest of this paper it is assumed that the SIB is implemented as shown in either Fig. 6(a) or Fig. 6(c).

4 Analysis: Flat Architecture

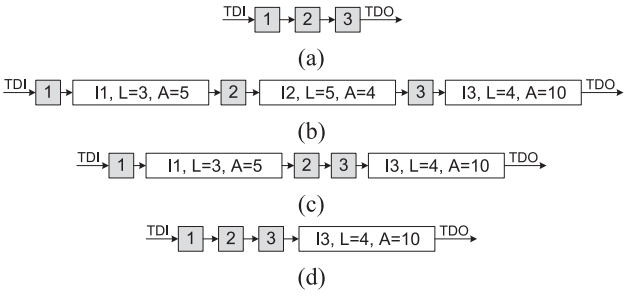
In this section, the flat architecture shown in Fig. 5(a) is analyzed with respect to overall access time (OAT). Two access schedules are considered, namely concurrent schedule (Section 4.1) and sequential schedule (Section 4.2).



(a) Without pipelining register

(b) With pipelining register

Fig. 7. The scan-path for the network in Fig. 5(b) drawn at the RTL level when all SIBs are open



(a)

(b)

(c)

(d)

Fig. 8. Scan-path configurations of the flat architecture example for the concurrent schedule

4.1 Concurrent schedule

In the concurrent schedule, accesses for all instruments start as soon as possible, which for the flat architecture means all accesses start at the same time. When an instrument is no more active (i.e. there are no more inputs to be applied to it) it is excluded from the network, by closing its corresponding instrument SIB. This makes the scan-path shorter for accessing the rest of the instruments. Various forms of concurrency can be considered, but the concurrent schedule as defined and used in this paper results in the minimum access time. This type of concurrency is not possible using original JTAG specifications and is unique to P1687 in this regard.

The input data for the instruments and the control bits for the SIBs on the current scan-path are concatenated appropriately to form the input vector. While an input vector is shifted in, the output vector corresponding to the previous inputs to the instruments is shifted out. Each output vector contains the output

TABLE 1
Flat architecture, concurrent schedule

Row No.	Scan-path	Scanned bits SIBs	I1	I2	I3	CUC	# of scan sequences	Sum for scan-path
1	Fig. 8(a)	3	0	0	0	5	1	$3+5$
2	Fig. 8(b)	3	3	5	4	5	5	$(15+5)\cdot 5$
3	Fig. 8(c)	3	3	0	4	5	1	$10+5$
4	Fig. 8(d)	3	0	0	4	5	5	$(7+5)\cdot 5$
OAT								$\sum$=183

TABLE 2
Flat architecture, sequential schedule

Row No.	Scan-path	Scanned bits SIBs	I1	I2	I3	CUC	# of scan sequences	Sum for scan-path
1	Fig. 9(a)	3	0	0	0	5	1	$3+5$
2	Fig. 9(b)	3	3	0	0	5	6	$(6+5)\cdot 6$
3	Fig. 9(c)	3	0	5	0	5	5	$(8+5)\cdot 5$
4	Fig. 9(d)	3	0	0	4	5	11	$(7+5)\cdot 11$
OAT								$\sum$=271

data from each of the instruments on the scan-path and the contents of the S registers of the SIBs on the scan-path.

In the following, it will be described how to calculate the OAT for the flat architecture shown in Fig. 5(a) and the concurrent schedule, with the help of Fig. 8 and Table 1. In Fig. 8, the gray boxes represent the S registers inside the correspondingly numbered SIBs. Table 1 will be explained as the OAT calculation is described.

Before accessing the instruments, the SIBs must be opened, since the scan-path initially only consists of the SIBs in the Gateway, as shown in Fig. 8(a). To open the SIBs, three bits with logic value of '1' are scanned in (one bit for each SIB) and subsequently a CUC is performed. The three bits each corresponds to the S register of a closed SIB, and they are accounted for on the row marked 1 in Table 1, column "SIBs". After the CUC, which takes five clock cycles (TCKs) as indicated in the column "CUC", all instruments are included in the scan-path, as shown in Fig. 8(b). At this point, input data can be applied to all three instruments, with a total scan-path length of 1_{SIB1}+3_{I1}+1_{SIB2}+5_{I2}+1_{SIB3}+4_{I3}=15 bits, where 1_{SIBx} represents the 1-bit S register inside SIB$_x$. The number of bits for the three instruments (called 3_{I1}, 5_{I2}, 4_{I3} above) are counted in the columns I1, I2 and I3 of Table 1. After four input vectors have been applied, accessing instrument I2 is complete and its shift-register should be excluded from the scan-path, which is done by setting the control bits such that SIB2 is closed, and SIB1 and SIB3 are kept open. This operation, to close SIB2, cannot occur until the output corresponding to the last input to I2 has been scanned out. Therefore, a fifth scan sequence is required during which the last output vector of I2 is scanned out and the SIB control bits to exclude I2 from the scan-path are scanned in. In total, five scan sequences are performed on the scan-path shown in Fig. 8(b), which is represented under column "# of scan sequences" in the row marked 2. After exclusion of I2 from the network, the scan-path has a total length of 1_{SIB1}+3_{I1}+1_{SIB2}+1_{SIB3}+4_{I3}=10 bits. The scan-path is now as shown in Fig. 8(c). After one scan sequence which is shifting out the last outputs of I1, represented by the row marked 3, the access to instrument I1 is complete and SIB1 is closed. The scan-path becomes as shown in Fig. 8(d). Four input vectors, hence four scan sequences, remain for instrument I3 and one more scan sequence is used to scan out the last of the outputs for instrument I3, while closing SIB3. For these last five scan sequences the total scan-path

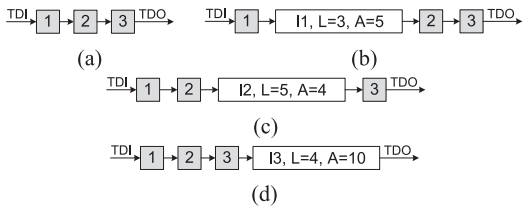


Fig. 9. Scan-path configurations of the flat architecture example for the sequential schedule

length is 1_{SIB1}+1_{SIB2}+1_{SIB3}+4_{I3}=7 bits, as shown in the row marked 4.

Table 1 shows the number of bits of different types that are scanned in for each scan sequence and the number of sequences performed on each scan-path configuration. The scan-path configuration corresponding to each row is specified under the column "Scan-path". The last column (i.e. "Sum for scan-path") shows the total number of bits that are scanned in for each scan-path. OAT is the sum of the values in this last column, as shown on the last row, which for this example is 183 clock cycles.

It should be noted that the SIB control bits contribute to OAT by $3+3\cdot 5+3+3\cdot 5=36$ clock cycles. Furthermore, the number of clock cycles spent performing CUC is $5+5\cdot 5+5+5\cdot 5=60$. These 36+60=96 clock cycles spent scanning SIB control bits and performing CUC are considered overhead, because no actual instrument data is transported during this time, and in the rest of this paper will be referred to as *SIB programming overhead* and *CUC overhead*, respectively. It can be seen that OAT consists of three components, namely instrument data, SIB programming overhead, and CUC overhead. An *overhead ratio* can be defined as total overhead divided by OAT, which for the above example is calculated as $96/183 \approx 0.52$. In general, many number of accesses and short scan-chains lead to a high overhead ratio. Of course, the length of the instrument scan-chains does not impact the overhead (the number of scanned SIB control bits or the number of CUCs), but long scan-chains effectively limit the overhead ratio.

4.2 Sequential schedule

In this section, OAT will be calculated for the flat architecture considering the sequential schedule. In the sequential schedule, the instruments are accessed one at a time, and the assumed order of access is the order that the instruments appear on the scan-path when all SIBs are open. The order of access can affect the overall access time in hierarchical networks, if it causes closing the already opened doorway SIBs

and reopening them again to access instruments in segments connected to the HIPs of those SIBs. It is also assumed that the access for each instrument is completed before accessing any other instrument.

Similar to how Fig. 8 and Table 1 described the access for the concurrent schedule, Fig. 9 and Table 2 will be used to explain the steps of sequential access to the instruments in the network shown in Fig. 5(a). Initially, the scan-path is as shown in Fig. 9(a). Three bits are used in the first scan sequence (see the row marked 1 in Table 2) to open SIB1 so that for the six following scan sequences the scan-path is as shown in Fig. 9(b). The row marked 2 in Table 2 shows that the three bits of I1 are included in scan-path. After six scan sequences (see the row marked 2), all the five input vectors for I1 have been applied and the corresponding outputs have been scanned out, while closing SIB1 and opening SIB2 so that the scan-path becomes as shown in Fig. 9(c). For this configuration of the scan-path, four input vectors for I2 are applied followed by a scan sequence to scan out the last outputs (see the row marked 3). Fig. 9(d) shows the scan-path as it is after completing the access for I2. Finally, 11 scan sequences (see the row marked 4) are applied to complete the access for I3 and scan out the last outputs, while closing SIB3. As can be seen from Table 2, OAT for the sequential schedule is 271 clock cycles, which should be compared to 183 clock cycles for the concurrent schedule discussed in Table 1. The difference in OAT can be explained by a larger number of scan sequences performed in the sequential schedule, which leads to more SIB and CUC overheads. Similar to how the overhead ratio was calculated in Section 4.1, the overhead ratio for the flat architecture and the sequential schedule can be calculated as $(69 + 115)/271 \approx 0.68$.

5 Analysis: Hierarchical Architecture

This section discusses the overall access time (OAT) for the hierarchical architecture shown in Fig. 5(b).

Table 3 and Table 4 show the steps to calculate OAT for the concurrent and sequential schedules, respectively. These tables are similar to Tables 1 and 2 in structure. The possible configurations referred to by the column "Scan-path", are presented in Fig. 10.

The access according to the concurrent schedule was explained for the flat architecture in Section 4.1. For the hierarchical architecture, in contrast to the flat architecture, when all instruments in a network segment have become inactive, the doorway SIB whose HIP is connected to that segment will be closed to exclude all the instruments and SIBs on that segment from the scan-path. As can be seen from Table 3, OAT for the hierarchical architecture and the concurrent schedule is 223 clock cycles, which should be compared to 183 clock cycles for the concurrent schedule and the flat architecture. In this example, the hierarchical architecture leads to a longer OAT

TABLE 3
Hierarchical architecture, concurrent schedule

Row No.	Scan-path	Scanned bits SIBs	I1	I2	I3	CUC	# of scan sequences	Sum for scan-path
1	Fig. 10(a)	2	0	0	0	5	1	$2+5$
2	Fig. 10(f)	4	3	0	0	5	1	$7+5$
3	Fig. 10(h)	5	3	5	0	5	1	$13+5$
4	Fig. 10(i)	5	3	5	4	5	4	$(17+5)\cdot 4$
5	Fig. 10(g)	5	0	0	4	5	7	$(9+5)\cdot 7$
OAT								$\sum$=223

TABLE 4
Hierarchical architecture, sequential schedule

Row No.	Scan-path	Scanned bits SIBs	I1	I2	I3	CUC	# of scan sequences	Sum for scan-path
1	Fig. 10(a)	2	0	0	0	5	1	$2+5$
2	Fig. 10(b)	2	3	0	0	5	6	$(5+5)\cdot 6$
3	Fig. 10(c)	4	0	0	0	5	1	$4+5$
4	Fig. 10(d)	4	0	5	0	5	5	$(9+5)\cdot 5$
5	Fig. 10(e)	5	0	0	0	5	1	$5+5$
6	Fig. 10(g)	5	0	0	4	5	11	$(9+5)\cdot 11$
OAT								$\sum$=310

because of two factors. Firstly, the overhead from the additional SIBs affects OAT. Secondly, the overhead in terms of capture-and-update cycles (CUC) is higher, due to opening the doorway SIBs to access the other levels of hierarchy.

Accessing instruments according to the sequential schedule was discussed in Section 4.2. For the hierarchical architectures, it is additionally assumed that only those doorway SIBs are open which are on the shortest scan-path to the instrument being accessed. Table 4 shows that for the sequential schedule, OAT is 310 clock cycles, which should be compared with 271 clock cycles for the sequential schedule and the flat architecture. The reason for the higher OAT with the hierarchical architecture is more SIB programming overhead and more CUCs.

The overhead ratio for the hierarchical architecture can be calculated as $(66 + 70)/223 \approx 0.61$ for the concurrent schedule and $(98 + 125)/310 \approx 0.72$ for the sequential schedule.

It should be noted that in the example discussed in Section 4 and this section, the flat architecture and the concurrent schedule led to the lowest overall access time (OAT). This is not a general conclusion, since other examples may show lower OAT on other architectures and schedules. For example, if for the networks shown in Fig. 5 the number of accesses for instruments were 20, 5 and 2 for I1, I2 and I3 respectively, the hierarchical network resulted in a lower OAT for both concurrent and sequential schedules.

6 Test Time Calculation Tool: IJTAGcalc

This section will describe a tool called IJTAGcalc consisting of two algorithms for calculating OAT for a given P1687 network. Algorithm 1 is for concurrent schedule and Algorithm 2 is for sequential schedule.

The terminology used in the algorithms, when defining variable names, is from a tree structure. The JTAG TAP is the root of the tree and the SIBs define the nodes. For each SIB s, there is a subtree of SIBs

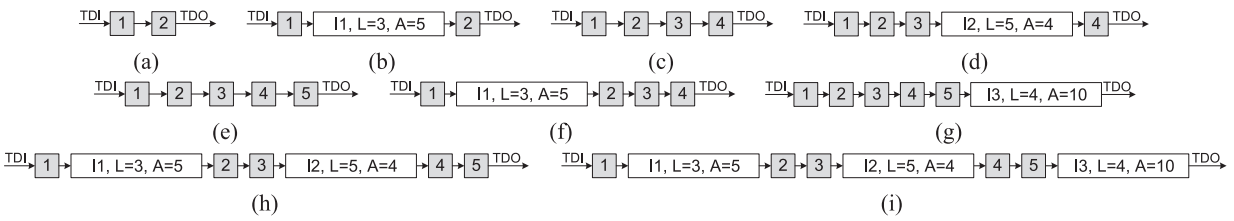


Fig. 10. Scan-path configurations of the hierarchical architecture example

that is accessed through the HIP of s. This subtree is empty and the SIB is a leaf node in case the SIB is an instrument SIB. SIBs that are in the subtree of s and on the next hierarchy level are referred to as children of s. In the example shown in Fig. 5(b), SIB2 has the subtree consisting of SIB3, SIB4 and SIB5. SIB2, which is on Level1, is the parent of SIB3 and SIB4, since they are on Level2.

To describe the placement of instruments, it is considered that each SIB can have at most one instrument connected to its HIP. The properties of this instrument, i.e. L and A, are associated with the connected SIB. If an instrument is connected to a SIB s, then $s.ILength$, $s.IAccesses$ and $s.IRemaining$ define the properties of the instrument. Initially, $s.IAccesses$ and $s.IRemaining$ are set to the number of accesses (A) for the instrument, and $s.ILength$ is set to the length of the instrument's shift-register (L). For each input vector that is applied, $s.IRemaining$ is decremented. When $s.IRemaining$ has reached 0, the final output vector of the instrument is to be shifted out. Therefore, a negative number in $s.IRemaining$ represents that the instrument is not to be accessed anymore. If SIB s is a doorway SIB, $s.ILength$ and $s.IAccesses$ are set to 0, and $s.IRemaining$ is set to -1, which has the effect that no instrument is connected to s. The variable $s.SRemaining$ for a SIB s will at all times hold the maximum of the value of the $IRemaining$-variables over all the SIBs in the subtree of s. In practice, this means that when $s.SRemaining$ reaches a negative value, SIB s can be closed.

Algorithm 1 for the Concurrent Schedule

```
1: function IJTAGcalcConcurrent()
2: for each SIB s do
3:     s.SRemaining := max{IAccesses found in subtree of s}
4: end for
5: while TAP.SRemaining > −1 do
6:     SSLength :=0                          // Scan sequence length
7:     TAP.SRemaining := Traverse(TAP)       // SSLength is
       updated by the Traverse function.
8:     OAT := OAT + SSLength + CUC
9: end while
10: end function
11:
12: function Traverse(node)
13: subtreeSAccessList := {−1}
14: for each child ∈ node.children do
15:     SSLength := SSLength + 1
16:     if child.SRemaining > −1 or child.IRemaining > −1
        then
17:         if child.IsOpen=False then
18:             child.IsOpen := True
19:         else
20:             child.SRemaining := Traverse(child)
21:             SSLength := SSLength + child.ILength
22:             child.IRemaining := child.IRemaining − 1
23:         end if
24:     else
25:         child.IsOpen := False          // might already be closed
26:     end if
27:     append max{child.SRemaining,child.IRemaining} to
        subtreeSAccessList
28: end for
29: return max{subtreeSAccessList}
30: end function
```

6.1 Concurrent Schedule

This section describes the algorithm for the concurrent schedule, presented in Algorithm 1. The algorithm consists of two functions, namely IJTAGcalcConcurrent and Traverse. In the Lines 2-4 of IJTAGcalcConcurrent, the $SRemaining$ variable is initialized for all the SIBs. Lines 5-9 describe a loop where each iteration contains a call to function Traverse. Each iteration corresponds to a scan sequence (see Table 1) and by summing the number of bits ($SSLength$) with the CUC for each scan sequence, OAT is added up (line 8). The iterations finish when there are no more input vectors to apply, as given by the $SRemaining$ variable. At this point, OAT will have been found.

In Algorithm 1, Traverse is a key function which returns the value for $SRemaining$. It also updates the global variable $SSLength$ which keeps track of the number of bits that have been scanned in during each scan sequence. The basic operation of the Traverse function is to inspect the child nodes of the node that was passed to Traverse as parameter. For these child nodes, the number of remaining accesses is calculated as the return value of Traverse. Since each child is a SIB, the $SSLength$ variable is incremented by one to represent the time it takes to scan in a control bit for the SIB (line 15). If the SIB is in the closed state but there are still input vectors to be applied to any instrument in its subtree (as indicated by the $SRemaining$ and $IRemaining$ variables, line 16), the SIB is opened (line 18). In the opposite situation, when there are no more input vectors to be applied for the subtree of a SIB, that SIB is closed (line 25). For an open SIB with remaining input vectors, a recursive call to Traverse is performed (line 20). The $SSLength$ variable is incremented by $ILength$ which signifies the shifting of the bits of one input vector (line 21). Furthermore, the number of remaining accesses is reduced by one (line 22). The number of remaining accesses for the subtree for which Traverse was called is calculated by taking the maximum number of accesses remaining for any of the child nodes (line 27 and line 29).

Algorithm 2 for the Sequential Schedule

```
1: function IJTAGcalcSequential(node)
2: if size(node.Children)>0 then
3:     SIBs := SIBs+size(node.Children)
4:     OAT := OAT + SIBs + CUC
5:     for each child ∈ node.Children do
6:         IJTAGcalcSequential(child)
7:     end for
8:     SIBs := SIBs−size(node.Children)
9: else
10:    OAT := OAT + (node.ILength + SIBs + CUC) ·
       (node.IAccesses + 1)
11: end if
12: end function
```

6.2 Sequential Schedule

This section describes IJTAGcalcSequential (Algorithm 2) for the sequential schedule. IJTAGcalcSequential considers the same type of tree representation as was discussed in Section 6. The basic idea behind the OAT calculation is that there are $A_i + 1$ scan sequences for each instrument i, for which the number of shifted bits per scan sequence is constant. This can be seen in the examples of Table 2 and Table 4. The number of shifted bits during the instrument access, depends on the length of that instrument's shift-register and the number of SIBs on the scan-path to that instrument. To calculate OAT, IJTAGcalcSequential should be called with TAP as parameter (the root node of the tree). Before the call to IJTAGcalcSequential, the global variables $SIBs$ and OAT should be set to 0. Here, $SIBs$ is a variable that counts the number of SIBs on the scan-path, and OAT is the variable that will contain the OAT when IJTAGcalcSequential terminates. The number of SIBs on the scan-path will vary according to the location of the instrument that is being accessed within the P1687 network. Therefore, IJTAGcalcSequential keeps track of the SIBs that must be traversed to reach the level of hierarchy on which the accessed instrument is located. Each level of hierarchy is marked by a recursive call (line 6).

When IJTAGcalcSequential is called, it checks whether the current $node$ (which is a SIB) has any child SIBs (Line 2). If $node$ has children, the $SIBs$ variable should be incremented with the number of children (Line 3) and OAT should be increased to represent the initial SIB programming required for the newly opened level of hierarchy (Line 4). Similarly when the function is leaving this level, $SIBs$ is reduced to the previous value, corresponding to the previous level of hierarchy (Line 8). IJTAGcalcSequential should be called recursively for the children of the current $node$ (Lines 5- 7). If current $node$ has no child (Line 9), then OAT will be incremented with the access time required for applying all the instrument's input vectors and the scan-out of the last output vector (line 10).

7 Parametric Analysis

It is possible for P1687 adopters to adjust design variables to balance the trade-offs among design goals such as time overhead reduction, hardware overhead reduction, etc. Among such design variables are access schedule, network architecture, number of accesses (A), length of shift-registers (L), and number of instruments. For example, it might be possible for a designer to reduce the number of accesses required for an instrument by increasing the length of the shift-register for that instrument, and it is important to know how this decision affects the design goals. In this section, analysis is presented to demonstrate the effect of changes in design variables on OAT and its components, i.e. instrument data, SIB programming overhead, and CUC overhead. For the analysis, the algorithms presented in Section 6 are implemented and slightly modified to report the SIB programming and the CUC overheads in addition to OAT.

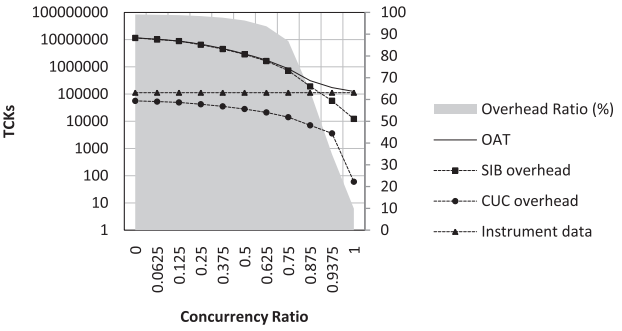
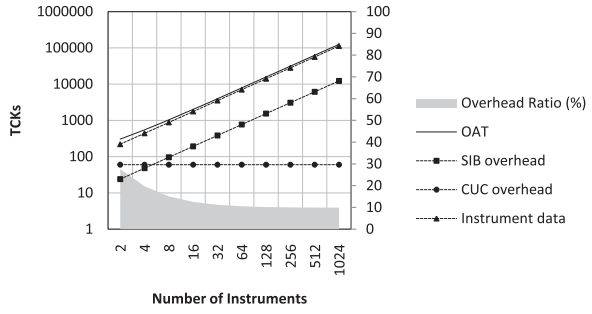


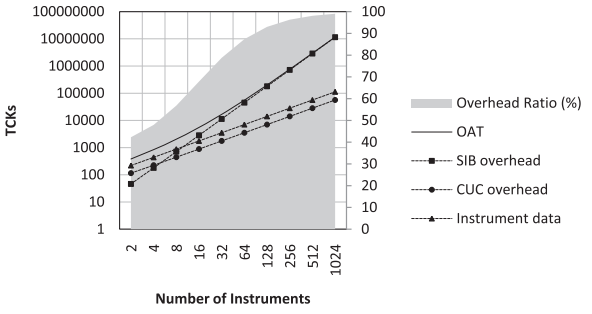
Fig. 11. How OAT changes with increase in *concurrency ratio*

As for the choice of schedule, it is possible to perform some accesses concurrently and some accesses sequentially. It should be considered that due to resource constraints, power constraints, or thermal constraints, it might not be always possible to use maximum concurrency. To study the effect of access schedule on OAT, a *concurrency ratio* is defined as the number of instruments being accessed concurrently divided by the total number of instruments. For example a concurrency ratio of 2/5 means that two instruments are accessed concurrently, followed by the sequential access to the remaining three instruments. For this analysis 1024 similar instruments ($L = 10$ and $A = 10$) are considered in a flat network. The concurrency ratio is increased from zero to one, where in each step 64 additional instruments are accessed concurrently, and OAT and its components are calculated. Fig. 11 presents the results for this experiment. In Fig. 11, the primary vertical axis is scaled logarithmically to present the OAT and its components in time units (TCKs). To present the overhead ratio in percentage, a secondary vertical axis is added to Fig. 11 and is scaled from 0 to 100. Using the secondary vertical axis, the overhead ratio is presented as a shaded area. Fig. 11 shows that as concurrency increases, the instrument data remains constant and both SIB programming and CUC overhead types decrease, leading to the instrument data dominating OAT.

As the second variable, the number of instruments



(a) OAT and its components in TCKs (concurrent schedule)



(b) OAT and its components in TCKs (sequential schedule)

Fig. 12. How OAT changes as the number of instruments increases

is considered. To study the effect of an increase in number of instruments on OAT and its components, two similar instruments with $L = 10$ and $A = 10$ in a flat network are considered. The number of instruments is increased in powers of two from 2 to 1024 and the instrument data, SIB programming overhead, and CUC overhead are calculated for each of these cases, using both concurrent and sequential schedules. Fig. 12 shows the results for this experiment. It can be seen from Fig. 12(a) that for the concurrent schedule, as the number of instruments increases, the instrument data becomes the dominating component of OAT, the CUC overhead remains the same since all of the instruments in this experiment have equal number of accesses, and the SIB overhead increases and remains a significant portion of OAT. But for sequential schedule, the CUC overhead increases and it is the SIB programming overhead which becomes extremely high and dominates the OAT (see Fig. 12(b)).

Another variable to consider is the number of accesses. Assuming 10 similar instruments each having $L = 20$ in a flat network, the OAT is calculated as the number of accesses (A) is increased from 2 to 1024 for all of the instruments, and for the concurrent and sequential schedules. Fig. 13 presents the results. It is interesting how the overhead ratio remains almost the same as the number of accesses increases (see Fig. 13(a) and Fig. 13(b)). As Fig. 13(a) shows, when using the concurrent schedule, the instrument data dominates OAT.

An experiment similar to the one above can be performed for the length of instrument shift-registers (L). Assuming 10 similar instruments each having $A = 10$ in a flat network, OAT is calculated as the length of shift-registers (L) is increased from 2 to 1024 for all of the instruments, and for the concurrent and sequential schedules. Fig. 14 presents the results from which it can be seen that for both concurrent and sequential schedules, both overhead types remain the same and OAT is dominated by the instrument data as L increases.

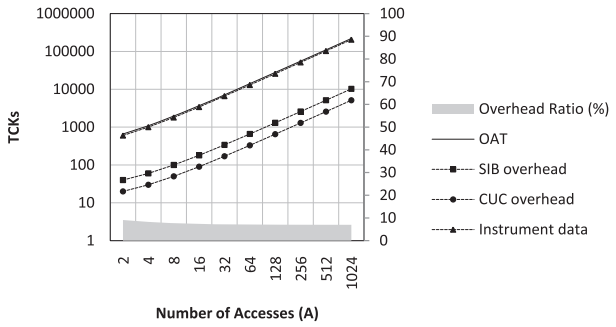
The last variable to consider is the network architecture. Given the flexibility in design of P1687 networks provided by SIBs, it is possible to design a large number of networks for the same set of instruments, by using hierarchy. However, to keep this study simple, only a very small subset of all possible networks could be considered. Therefore, we chose to start with a flat network (i.e. only a single level of hierarchy) and observe how OAT changes with increasing the number of hierarchical levels. In our experiment, to add a level of hierarchy, the instrument SIBs directly connected to the HIP of doorway SIBd (see Fig. 16) are divided into two groups, each group is connected to the HIP of an additional doorway SIB, and the two additional doorway SIBs are connected to the HIP of doorway SIBd. This idea is shown in Fig. 16 for eight instruments, but this experiment considers 1024 instruments ($L = 10$ and $A = 10$). For these 1024 instruments, 10 levels of hierarchy are considered where at the 10^{th} level each doorway SIB has two instrument SIBs connected to its HIP. The OAT calculation results are presented in Fig. 15. From Fig. 15(a) (for the concurrent access schedule) it is seen that an increase in the number of levels of hierarchy increases OAT, and from Fig. 15(b) an opposite trend is observed for the sequential access. Here, an important observation is that for both concurrent and sequential schedules, CUC overhead is not affected significantly by increase in hierarchical levels. Therefore, since instrument data is independent of network architecture, for both schedules, the increase and decrease in OAT can be attributed to the contribution of SIB programming overhead.

One important observation regarding this parametric analysis is that in all experiments and for almost all cases in each experiment, the CUC overhead has the least contribution to OAT. Table 5 summarizes the parametric analysis presented in this section, where the effect of increasing any of the design variables (Column 1) on OAT, its components, and the overhead ratio is presented. The effect is shown as increasing ($\uparrow$), decreasing ($\downarrow$) or unaffected ($-$).

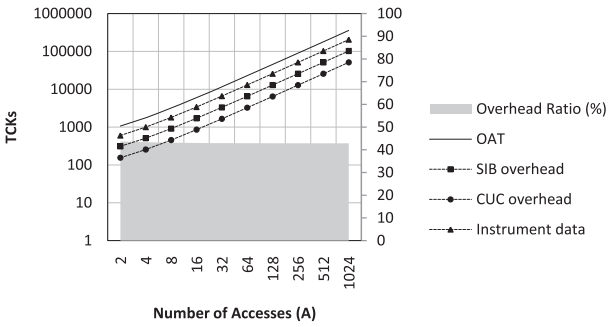
TABLE 5
Summary for the parametric analysis

Variable $\uparrow$	Instrument data	CUC overhead	SIB Prog. overhead	OAT	Overhead ratio
Number of instruments	$\uparrow$	(1)	$\uparrow$	$\uparrow$	(1)
Number of accesses (A)	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	(2)
Shift-register length (L)	$\uparrow$	$-$	$-$	$\uparrow$	$\downarrow$
Concurrency ratio	$-$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
Hierarchical levels	$-$	$\uparrow$	(1)	(1)	(1)

(1) Depends on schedule and properties of instruments (i.e. L and A).
(2) Remains almost the same.

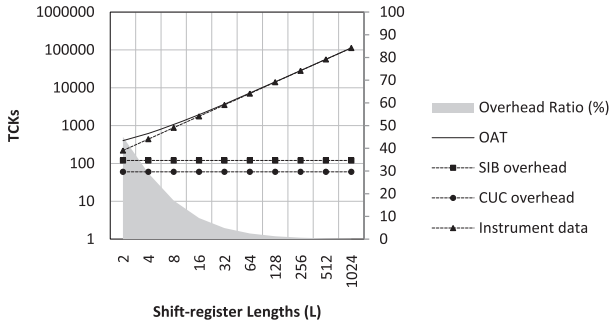


(a) OAT and its components in TCKs (concurrent schedule)

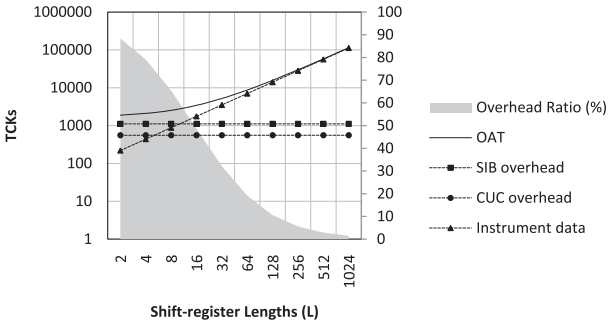


(b) OAT and its components in TCKs (sequential schedule)

Fig. 13. How OAT changes as the number of accesses (A) increases

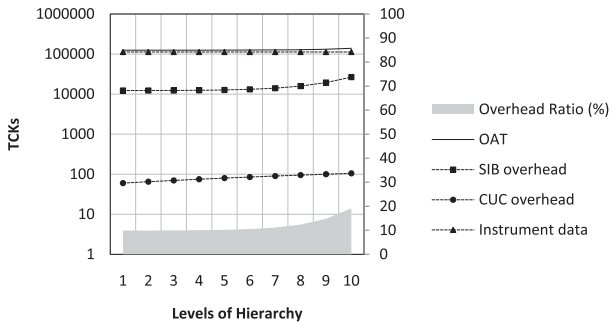


(a) OAT and its components in TCKs (concurrent schedule)

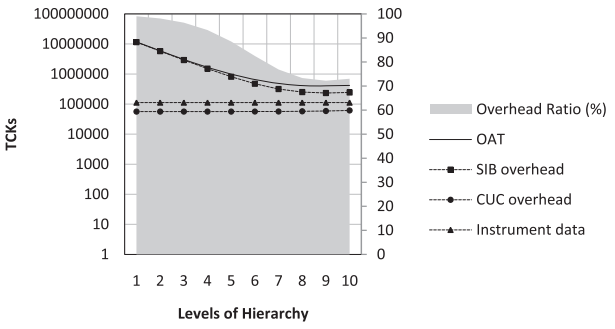


(b) OAT and its components in TCKs (sequential schedule)

Fig. 14. How OAT changes as the length of shift-registers (L) increases

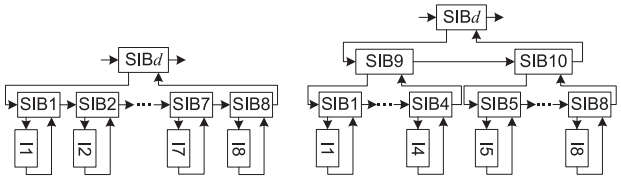


(a) OAT and its components in TCKs (concurrent schedule)



(b) OAT and its components in TCKs (sequential schedule)

Fig. 15. How OAT changes by increase in levels of hierarchy



(a) A P1687 network segment comprising eight instruments

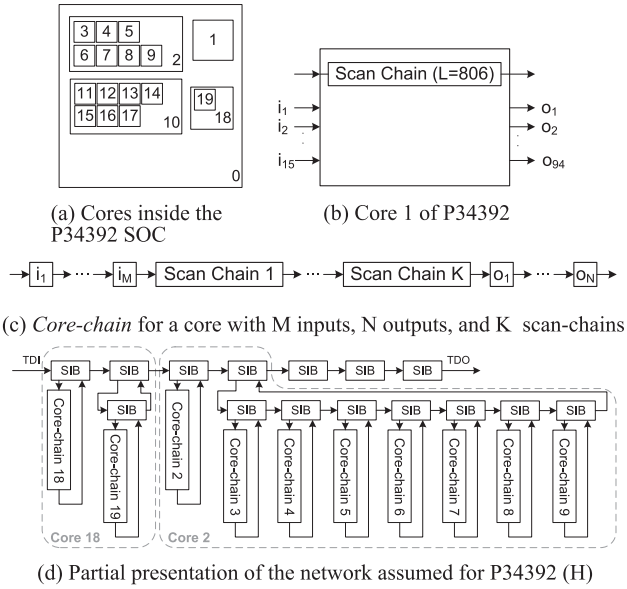
(b) Adding another hierarchical level to the segment in Fig. 16(a)

Fig. 16. Adding a level of hierarchy

8 Experimental Setup

We have performed experiments to study the connection of instruments to the JTAG TAP in respect to OAT, both without and with the use of P1687.

To perform the experiments, a set of instruments was required. Therefore, we have chosen to view the cores inside the SOCs of ITC'02 benchmark set [19], as design-for-test instruments. This way, considering how *access* is defined in this paper, the calculated OAT is actually the test application time. Each SOC from the ITC'02 set contains a number of cores, as shown in Fig. 17(a) for the P34392 SOC. Each core has a number of I/O pins as well as some internal scan-chains, as shown in Fig. 17(b) for Core 1 from P34392. Since in the context of P1687 all data are transported through a single wire, to access all of these I/O pins and internal scan-chains, boundary cells are assumed for each I/O pin and a *core-chain* is formed by concatenating the boundary cells and the internal scan-chains, as shown in Fig. 17(c) for a core with M inputs, N outputs and K scan-chains. To calculate the number of the boundary scan cells, each input/output terminal is counted as one cell and every bi-directional terminal is counted as three cells [1], [20]. For example, the length of the core-chain for Core 1, shown in Fig. 17(b), is calculated as $15 + 806 + 94 = 915$. It should be noted that in



(a) Cores inside the P34392 SOC

(b) Core 1 of P34392

(c) *Core-chain* for a core with M inputs, N outputs, and K scan-chains

(d) Partial presentation of the network assumed for P34392 (H)

Fig. 17. Using the ITC'02 benchmark SOCs for the experiments

case a core is hierarchical, e.g. core 10 in Fig. 17(a), the outputs of the child cores, e.g. core 11, should be considered as inputs to the parent core (i.e. the hierarchical core). Similarly, the inputs of the child cores should be considered as outputs of the parent core. Therefore, concerning the hierarchical cores, the boundary cells of each embedded child core are appropriately included in the set of boundary cells of its direct parent core [20].

For each of the SOCs, the total amount of instrument data that should be shifted in is calculated as $\sum_{i=1}^{N} L_i \cdot (A_i + 1)$, where N is the number of cores, A_i is the number of accesses for the core-chain of core i, and L_i is the length of the core-chain for core i. In this relation, $+1$ denotes the shift out of the last output. It should be noted that the chip-level module (Module 0), specified in the ITC'02 benchmarks, and modules containing BIST-engines are not included in the experiments and are also not accounted for in presentation of the results.

We have performed two sets of experiments to calculate and compare OAT when (1) instruments are connected in series as a single TDR to the JTAG TAP, and (2) when instruments are connected to JTAG TAP through P1687.

As for Experiment 1, all the shift-registers of instruments (here core-chains) are assumed to be connected in series as a single TDR to be accessed according to the concurrent schedule. This idea was referred to as *chain of instruments* in Section 2. The length of this TDR for each SOC, is the sum of the lengths of that SOC's *core-chains*. Of course, in this scenario it is not possible to exclude an instrument from the scan-path when there are no more input vectors to be applied to it. Therefore, the number of accesses for the TDR is the maximum number of accesses found among the core-chains that form the TDR.

For Experiment 2, P1687 networks are assumed for

TABLE 6
Results for Experiment 1

SOC	TDR*		Instrument	Dummy bits	CUC	OAT
	Length	Accesses†	data	overhead	overhead	
F2126	15205	422	5330439	1101276	2115	6433830
T512505	76554	3370	165400967	92662567	16855	258080389
U226	1288	2666	252929	3182167	13335	3448431
P22810	30885	12324	8172047	372485578	61625	380719250
A586710	41966	1914433	843806808	79497330436	9572170	80350709414
P34392	23456	12336	16728056	272648616	61685	289438357

*TDR is formed by concatenation of *core-chains* in the SOC.
†The number of accesses for a TDR is the maximum number of accesses found among the *core-chains* that form the TDR.

the same instruments (i.e. cores) used in Experiment 1. All six SOCs were included in an experiment with a flat network (denoted with "F") and three of the SOCs (P22810, P34392 and A586710 which have hierarchical cores) were also used in an experiment with a hierarchical network (denoted with "H"). In the hierarchical networks the child cores are assigned to the next hierarchy level in relation to their parent core, see Fig. 17(d). This is to make a one-to-one correspondence to the hierarchy in the SOC itself, for the sake of experiment. It should be noted that hierarchy in terms of P1687 can be implemented in a huge variety of ways, but to calculate and analyze OAT, we require one such implementation and we have chosen to take inspiration from the hierarchy of the cores. Fig. 17(d) shows part of the network assumed for P34392 (H) in which core 2 and core 18 are at the first level of the hierarchy and their child cores are at the second level. To perform Experiment 2, the algorithms presented in Section 6 are implemented and slightly modified to report the SIB programming and the CUC overheads in addition to OAT.

9 Experimental Results

The results of the performed experiments are reported and discussed in this section.

Regarding Experiment 1, chaining the instruments into a long scan-path, as was mentioned in Section 2, has the drawback of increased instrument access time. Table 6 shows OAT and its components for each SOC. Here, the OAT components are the instrument data, the number of dummy bits shifted during the access, and the CUC overhead. It can be seen that the amount of overhead due to the dummy bits can be prohibitively high.

Regarding Experiment 2, Table 7 and Table 8 present the results. Table 7 lists the instrument data and the network types considered for each SOC (columns 2 and 3) as well as the detailed results for accessing the instruments inside each network using the concurrent schedule (columns 4-6) and sequential schedule (columns 7-9). For each schedule, the two overhead components, i.e. SIB programming overhead and CUC overhead, are reported. OAT for each network and each schedule is the sum of the instrument data and the overhead components for the respective network and schedule.

Table 8 presents the same information presented in Table 7 in percentage of OAT for each network.

TABLE 7
Detailed experimental results (in TCKs) for Experiment 2

SOC†	Instrument data	Network‡	Concurrent schedule			Sequential schedule		
			Overhead types		OAT	Overhead types		OAT
			SIB Prog.	CUC		SIB Prog.	CUC	
F2126 (4)	5330439	F	1696	2120	5334255	3868	4835	5339142
T512505 (31)	165400967	F	104532	16860	165522359	325841	52555	165779363
U226 (5)	252929	F	13340	13340	279609	40475	40475	333879
P22810 (28)	8172047	F	345128	61630	8578805	701176	125210	8998433
		H	340728	61635	8574410	665898	125220	8963165
A586710 (5)	843806808	F	9572175	9572175	862951158	10709500	10709500	865225808
		H	9753317	9572175	863132300	10890647	10709505	865406960
P34392 (19)	16728056	F	234422	61690	17024168	1260498	331710	18320264
		H	250086	61695	17039837	841306	331725	17901087

†The numbers inside parentheses, denote the number of cores in the corresponding SOC.
‡F denotes a flat architecture and H denotes a hierarchical architecture.

TABLE 8
Detailed experimental results (in Percentage of OAT) for Experiment 2

SOC†	Network‡	Concurrent schedule			Sequential schedule		
		Overhead types		Instrument data	Overhead types		Instrument data
		SIB Prog.	CUC		SIB Prog.	CUC	
F2126 (4)	F	0.03	0.04	99.93	0.07	0.09	99.84
T512505 (31)	F	0.06	0.01	99.93	0.20	0.03	99.77
U226 (5)	F	4.77	4.77	90.46	12.12	12.12	75.75
P22810 (28)	F	4.02	0.72	95.26	7.79	1.39	90.82
	H	3.97	0.72	95.26	7.43	1.40	91.17
A586710 (5)	F	1.11	1.11	97.78	1.24	1.24	97.52
	H	1.13	1.11	97.76	1.26	1.24	97.50
P34392 (19)	F	1.38	0.36	98.26	6.88	1.81	91.31
	H	1.47	0.36	98.17	4.70	1.85	93.45

†The numbers inside parentheses, denote the number of cores in the corresponding SOC.
‡F denotes a flat architecture and H denotes a hierarchical architecture.

Therefore, the column "OAT" (which is always 100%) is removed from the table and the column "Instrument data" is repeated for each of the concurrent and sequential schedules. It should be noted that although the instrument data is independant of the network and the access schedule, when shown as a percent of OAT it might be different for each network and schedule.

Regarding the concurrent schedule, Table 8 shows that F2126 and T512505 both have relatively low overhead. Long instrument shift-registers and a small number of accesses should result in relatively low overhead. This is the case for all the cores of F2126. For T512505, there are some cores that have short core-chains but in those cases the number of accesses is also low. On the other hand, there is one core with a very long core-chain and the instrument data for this core corresponds to about 90% of the overall instrument data. Therefore, this core made such impact on OAT that the overhead from the other cores became negligible. U226 contains some cores that have short core-chains and a large number of accesses. Therefore, the overhead percentage of OAT for U226, which is about 10% (4.77% for the SIB programming overhead plus 4.77% for the CUC overhead), is larger than that of the other networks. In A586710 (F), there is a core with about two million accesses and a core-chain length of 326 cells. However, this large number of accesses has not resulted in a large SIB programming overhead ratio. The reason is that the 326-cell core-chain is significantly longer than the number of SIBs in the scan-path in A586710 (F), which effectively limits the SIB programming overhead ratio corresponding to this core. As for P22810 (F) and P34392 (F), the maximum number of accesses among the cores are similar, resulting in similar CUC overheads for both networks. However, the ratios of CUC overhead are not similar in these two networks due to the differences in their amounts of instrument data.

The observations regarding the networks with flat architecture typically applies also to the corresponding networks (from the same SOC) with hierarchical architecture, as can be seen in Table 8. It should be noted that even though there was a noticeable difference between OAT of the flat network and that of the hierarchical network for the small example of Fig. 5, see Section 5, the experiments with SOC benchmarks show little difference in overhead ratio. P22810 (H) and P22810 (F) show similar results because there are few hierarchical levels relative to the number of cores.

Therefore, P22810 (H) has characteristics similar to those of its flat counterpart.

Comparing the results for the concurrent schedule with those of the sequential schedule, it can be seen that the overhead ratio is larger when using a sequential schedule. For example, in case of U226 (F), the overhead ratio is about 24% (12.12% + 12.12%) for the sequential schedule compared to 10% for the concurrent schedule. The main reason for this is that the total number of scan sequences increase, leading to an increase in both CUC and SIB programming overhead, while the amount of instrument data stays the same in both schedules. P34392 shows a noticeable difference in overhead ratio between the flat network, marked by P34392 (F), and the hierarchical network, marked by P34392 (H), in the case of the sequential schedule. For the flat network the overall overhead is about 9% and for the hierarchical network the overhead is about 7% with the difference mainly due to a lesser SIB programming overhead. In P34392 (F), every scan sequence, independent of the core, includes 19 SIBs (the same number as the number of cores). However, in P34392 (H), the scan-sequences contain on average 13 SIBs. Since this is significantly less than 19 SIBs, this explains the noticeable difference in SIB programming overhead.

10 Conclusion

In expectation of ratification of IEEE P1687 IJTAG, this paper has presented an overview of this standard proposal and an analysis of how overall access time (OAT) is to be calculated in this context.

The analysis explored the configuration possibilities for P1687 networks and considered accessing the instruments in those networks using two schedules, namely concurrent schedule and sequential schedule. The analysis shows that overhead in terms of clock cycles spent performing other operations than shifting instrument data can be put into two categories, namely time spent on shifting control data (for P1687 network configuration) and time spent in the JTAG TAP controller for applying inputs to instruments and capturing their outputs.

This paper presents an OAT calculation tool for P1687 networks called IJTAGcalc, which supports both concurrent and sequential access schedules. IJTAGcalc is implemented and employed in a parametric analysis to study how changes in the design variables, e.g. number of instruments, number of accesses for instruments and lengths of instrument shift-registers, will affect OAT. From the key observations of this analysis the following can be mentioned: (1) length of an instrument's shift-register has no effect on overhead, (2) the time spent shifting instrument data is independent of the access schedule and the network architecture, (3) using hierarchical networks may reduce OAT, depending on the used access schedule, and (4) JTAG TAP controller overhead often has the least contribution to OAT. These observations can be utilized to optimize P1687 networks with respect to OAT. Furthermore, IJTAGcalc can be used to evaluate and compare the optimized networks, or be used in future optimization techniques where selecting among multiple alternative networks is required.

IJTAGcalc was also employed in performing experiments on ITC'02 benchmark set. The results show a possible overhead ratio of up to 24%. The results can be well explained by the observations made in the analysis. For a particular benchmark, it was seen that the network architecture has a noticeable impact on the overhead.

References

[1] IEEE association, "IEEE Std 1149.1-2001, IEEE Standard Test Access Port and Boundary-Scan Architecture," 2001.

[2] J. Rearick, B. Eklow, K. Posse, A. Crouch, and B. Bennetts, "IJTAG (Internal JTAG): A Step Toward a DFT Standard," in *Proc. ITC*, 2005.

[3] IJTAG, "IJTAG - IEEE P1687," 2010, http://grouper.ieee.org/groups/1687.

[4] J. Rearick and A. Volz, "A Case Study of Using IEEE P1687 (IJTAG) for High-Speed Serial I/O Characterization and Testing," in *Proc. ITC*, 2006, pp. 1–8.

[5] K. Posse, A. Crouch, J. Rearick, B. Eklow, M. Laisne, B. Bennetts, J. Doege, M. Ricchetti, and J.-F. Cote, "IEEE P1687: Toward Standardized Access of Embedded Instrumentation," in *Proc. ITC*, 2006, pp. 1 –8.

[6] A. L. Crouch, "IJTAG: The Path to Organized Instrument Connectivity," in *Proc. ITC*, 2007, pp. 1–10.

[7] J. Doege and A. Crouch, "The advantages of limiting p1687 to a restricted subset," in *Proc. ITC*, 2008, pp. 1 –8.

[8] IEEE association, "IEEE Std 1500-2005, IEEE Standard Testability Method for Embedded Core-Based Integrated Circuits," 2005.

[9] L.-T. Wang, R. Apte, S. Wu, B. Sheu, K.-J. Lee, X. Wen, W.-B. Jone, C.-H. Yeh, W.-S. Wang, H.-J. Chao, J. Guo, J. Liu, Y. Niu, Y.-C. Sung, C.-C. Wang, and F. Li, "Turbo1500: Toward Core-Based Design for Test and Diagnosis Using the IEEE 1500 Standard," in *Proc. ITC*, 2008, pp. 1–9.

[10] E. J. Marinissen and T. Waayers, "Infrastructure for modular SOC testing," in *Proc. CICC*, 2004, pp. 671–678.

[11] Y. Zorian and A. Yessayan, "IEEE 1500 utilization in SOC design and test," in *Proc. ITC*, 2005, pp. 1–10.

[12] G. Vranken, T. Waayers, H. Fleury, and D. Lelouvier, "Enhanced reduced pin-count test for full-scan design," in *Proc. ITC*, 2001, pp. 738 –747.

[13] Z. Stamenkovic, M. Giles, and F. Russi, "Combining internal scan chains and boundary scan register: A case study," in *proc. EUROCON*, May 2009, pp. 2064 –2069.

[14] M. Higgins, C. MacNamee, and B. Mullane, "SoCECT: System on Chip Embedded Core Test," in *Proc. DDECS*, 2008, pp. 326–331.

[15] M. Portolan, S. Goyal, B. Van Treuren, C.-H. Chiang, T. Chakraborty, and T. Cook, "A common language framework for next-generation embedded testing," *Design & Test of Computers, IEEE*, vol. 27, no. 5, pp. 36 –49, 2010.

[16] M. Tehranipour, N. Ahmed, and M. Nourani, "Testing soc interconnects for signal integrity using boundary scan," in *proc. VTS*, 2003.

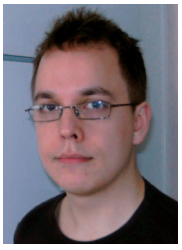
[17] ASSET InterTech, Inc., "Serial Vector Format Specification," 1999, http://www.asset-intertech.com/support/svf.pdf.

[18] J. Aerts and E. J. Marinissen, "Scan chain design for test time reduction in core-based ICs," in *Proc. ITC*, 1998, pp. 448–457.

[19] E. J. Marinissen, V. Iyengar, and K. Chakrabarty, "A set of benchmarks for modular testing of SOCs," in *Proc. ITC*, 2002, pp. 519–528.

[20] S. K. Goel, "Test-access planning and test scheduling for embedded core-based system chips," Ph.D. dissertation, University of Twente, 2005, http://doc.utwente.nl/48260/.

Farrokh Ghani Zadegan received his B.S. degree in Electrical Engineering from Ferdowsi University of Mashhad, Mashhad, Iran in 2001, and his M.S. degree in Electrical Engineering from Linköping University, Linköping, Sweden in 2010. He is currently a research assistant in Embedded Systems Laboratory, Linköping University, Linköping, Sweden.



Urban Ingelsson received the M.Sc. degree from Linköping University, Sweden, in 2005 and the Ph.D. degree from University of Southampton, UK, in 2009. Currently he is a post-doc at Linköping University. His research interests include test, diagnosis and fault-tolerance of digital circuits. Urban is a member of the IEEE.

Gunnar Carlsson Gunnar Carlsson is a Test and DFT Strategist with Ericsson. He has 30+ years of experience in the Test and DFT areas. Gunnar has been involved in test and DFT spanning from ASIC over boards to network nodes, and over the product life cycle from design verification over production test to testing in the field.

Erik Larsson is Associate Professor at the Department of Computer and Information Science at Linköping University (LiU). He received his M.Sc., Tech. Lic and Ph.D from Linköping University in 1994, 1998, 2000, respectively. He did his Post Doc at the Computer Design and Test Laboratory at Nara Institute of Science and Technology (NAIST), Japan, and was at NXP Semiconductors, Eindhoven, The Netherlands from October 2008 until May 2010.

His current research interests include test planning for manufacturing test, test during operation (in-situ), scan-chain diagnosis, silicon debug and validation, IJTAG/SJTAG, stacked 3D chip test, fault-tolerance for MPSoCs (Multi-Processor System-on-Chip), and property checking in distributed systems (MPSOcS with Network-on-Chip (NoC)). He has more than 120 publications in these areas. He authored the book Introduction to Advanced System-on-Chip Test Design and Optimization

He received the Institution of Engineering and Technology (IET) Premium Award, 2009, and the best paper award at IEEE Asian Test Symposium (ATS), 2002. His paper “An Integrated System-on-Chip Test Framework” has been selected to be included in Design, Automation, and Test in Europe, The Most Influential Papers of 10 Years DATE, 2008. He supervised the thesis that won prize as best Master thesis in Engineering in Sweden (“Lilla Polhemspriset” 2008).

Erik Larsson is in the Steering committee of Workshop on RTL ATPG & DFT (WRTLT) and International Workshop on Reliability Aware System Design and Test (RASDAT), and was program chair of European Test Symposium (2011), vice program chair European Test Symposium (2010), topic chair of European Test Symposium, 2007-2009, 2012, Design Automation and Test in Europe, 2007(vice), and 2008,2009 Great Lake Symposium on VLSI, 201. He is member of the technical committee of a number of conferences. Erik Larsson is Senior member of IEEE.