Design Automation for IEEE P1687

Farrokh Ghani Zadegan[1], Urban Ingelsson[1], Gunnar Carlsson[2] and Erik Larsson[1]
[1] Linköping University, Linköping, Sweden
ghanizadegan@ieee.org, urban.ingelsson@liu.se, erik.larsson@liu.se
[2] Ericsson AB, Stockholm, Sweden
gunnar.carlsson@ericsson.com

***Abstract*—The IEEE P1687 (IJTAG) standard proposal aims at standardizing the access to embedded test and debug logic (instruments) via the JTAG TAP. P1687 specifies a component called Segment Insertion Bit (SIB) which makes it possible to construct a multitude of alternative P1687 instrument access networks for a given set of instruments. Finding the best access network with respect to instrument access time and the number of SIBs is a time-consuming task in the absence of EDA support. This paper is the first to describe a P1687 design automation tool which constructs and optimizes P1687 networks. Our EDA tool, called PACT, considers the concurrent and sequential access schedule types, and is demonstrated in experiments on industrial SOCs, reporting total access time and average access time.**

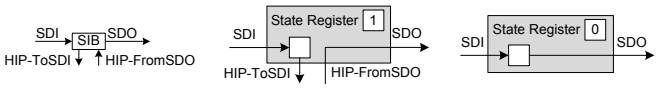
***Keywords*-IEEE P1687 IJTAG, Design Automation, Instrument Access, Access Time Optimization**

I. Introduction

Integrated circuits (ICs) are becoming increasingly advanced. For example, an ASIC from Ericsson contains 64 processors where each processor has its dedicated data memory and instruction memory, and a number of SERDESs and hardware accelerators; hence more than 200 blocks of logic. To ensure testability and reliability most ICs have embedded test, debug and monitoring logic (referred to as instruments). A typical IC contains several hundreds of such instruments. In the Ericsson ASIC mentioned above, each block of logic contains one or more instruments. Examples of such instruments include Memory BIST, Logic BIST, scan-chains and temperature sensors. It can be seen that the number of instruments in this ASIC amounts up to several hundreds.

There is no standard method (and thus no EDA support) for accessing on-chip instruments. Therefore, IEEE P1687 [1] is proposed to provide a uniform access method for connecting to instruments, and to facilitate test reuse in different stages of a chip's life cycle, i.e. prototyping, wafer test, board test, system test and in-field test. Such standardization makes provision of EDA tools possible. Without EDA support, manual design of instrument access networks will be extremely time consuming, particularly when there are many instruments, such as in the above mentioned ASIC.

This paper contributes towards the provision of EDA support for instrument access by design of P1687 networks. The P1687 standard proposal introduces a programmable component called Segment Insertion Bit (SIB) that is used to configure the scan-path by including/excluding P1687 network *segments*. A network segment can be an instrument or itself a smaller network of SIBs and instruments. Given a set of instruments, the use of SIBs makes it possible to create a multitude of alternative P1687 networks, each leading to a different instrument access time. Optimizing for low instrument access time makes P1687 network design a complicated and time-consuming task. To eliminate time consuming manual design of P1687 networks, this paper presents novel algorithms for automated design of optimized P1687 networks. The algorithms are implemented in a tool named P1687 Automatic Construction Tool (PACT).



(a) SIB ports (b) Open SIB (c) Closed SIB

Fig. 1. Simplified view of SIB component

The next section gives an overview of P1687 and reviews prior work. Section III defines the P1687 network design problem. Section IV and Section V present design automation algorithms for two instrument access schedule types. Sections VI and VII present experimental setup and results.

II. Overview of P1687 and Review of Prior Work

P1687 proposes to use the IEEE 1149.1 (JTAG) TAP for accessing on-chip instruments from outside the chip. This is in line with the widespread use of 1149.1 in ad hoc access to on-chip test and debug features [2]. Therefore, P1687 has received the informal name of IJTAG (Internal JTAG). To interface the on-chip P1687 network to the JTAG TAP, a special Test Data Register is added to the JTAG circuitry, which will form a flexible scan-path including arbitrary subsets of instruments, between the Test-Data-Input (TDI) and Test-Data-Output (TDO) terminals of TAP. The special Test Data Register is called Gateway and is selected by loading a JTAG instruction called Gateway Enable (GWEN). The Gateway is composed of one or more SIBs.

Fig. 1(a) shows a simplified view of a SIB. Besides Serial-Data-In (SDI) and Serial-Data-Out (SDO) ports, the SIB has a Hierarchical Interface Port (HIP) which connects to a P1687 network segment. A SIB has two states. It is either open (Fig. 1(b)) and includes the segment on the HIP in the scan-path, or it is closed (Fig. 1(c)) and transfers the data from its SDI port to its SDO port, excluding the segment on the HIP. Whether the SIB is open or closed, it corresponds to a 1-bit data register on the scan-path. The state of the SIB is set by scanning in a control bit into its register which is transferred to its state register (shown in Fig. 1(b) and Fig. 1(c)) by an update signal from the JTAG TAP.

978-3-9810801-7-9/DATE11/©2011 EDAA

Since IEEE P1687 has recently been proposed, only a few studies have considered it [3], [4], [5], [6]. However, no study has considered automated design of optimized P1687 networks. In [3] and [4], the authors proposed techniques for testing IEEE 1500 wrapped cores and have considered future integration of those techniques with P1687. In [5], a case study for test and configuration of high-speed serial I/O (HSSIO) links using P1687 is presented. There, it is mentioned that due to the need for high-volume manufacturing test of HSSIO links and difficulties associated with external test equipments, using on-chip test instruments will be an attractive solution. However, in [5] accessing P1687 instruments through JTAG TAP is regarded as a bottleneck, from which it can be inferred that instrument access time is an important parameter for optimization. According to [5] accessing the on-chip instruments can be done individually or in unison. In [6], overall instrument access time calculation methods are presented for P1687 networks having scan-chains as instruments, while making use of sequential and concurrent access schedules (similar to the *individual* and *in unison* access methods in [5]). The overall access time (OAT) consists of time transporting instrument data and two types of overhead, i.e. SIB programming overhead and JTAG protocol overhead (CUC overhead). The SIB programming overhead, which is the time spent transporting the total number of required SIB control bits, arises from the fact that SIB control data (1 bit per SIB) are transported along with instrument data on P1687 networks. CUC (Capture and Update Cycle) is the progression of five states (Exit1-DR, Update-DR, Select-DR-Scan, Capture-DR and Shift-DR) in the TAP controller state machine. Every write and read operation on an instrument requires a CUC to apply the inputs and capture the outputs.

In [6], it is pointed out that time spent transporting instrument data is independent from the P1687 network structure and the access schedule. In contrast, network structure and access schedule affect both SIB programming overhead and CUC overhead. Furthermore, it is shown that the length of the scan-chain instruments has no impact on the overhead. It should be noted that while in [6] the effect of the network structure and the access schedule on overhead is observed, no method for reduction of overhead is proposed. In [6], the P1687 network is considered to be given and OAT is calculated. In this paper, we develop the design automation of P1687 networks.

From the prior work, it can be seen that this paper is the first to address the automated design of P1687 networks and instrument access time reduction. From [5], we infer that P1687 networks should be optimized with regard to low instrument access time. Therefore this paper presents design automation results (Section VII) in terms of overall access time for a set of instruments and both sequential and concurrent access schedule types, as well as the corresponding average access time.

III. Scope and Problem Definition

The following describes instrument *access* as used in this paper. From [1], it is assumed in this paper that each instrument contains a shift-and-update register. In this context, an access to an instrument is defined as (1) shifting input bits into the instrument's shift-register, (2) latching the contents of the shift-register to be applied as inputs to 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.

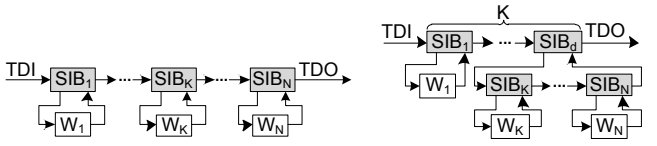
For the notation in this paper, a SIB having a single instrument connected to its HIP is referred to as *instrument SIB* and if the segment connected to the HIP is a network of SIBs and instruments, the SIB is called a *doorway SIB*. It is assumed that there is a fixed number of instrument SIBs, one for each instrument, regardless of the network structure. This ensures that for each instrument, access can be independently scheduled. In contrast, the number of doorway SIBs can vary with the network structure. Since doorway SIBs effectively change the length of the scan-path, by including/excluding network segments that include other SIBs, the impact of the number and placement of doorway SIBs on the SIB programming overhead will be significant. Compared to the SIB programming overhead, CUC overhead varies to a lesser degree with the number and placement of doorway SIBs [6]. Therefore, an effective way to reduce the instrument access time by P1687 network design, as is the focus of this paper, is reduction of the SIB programming overhead by appropriate placement of doorway SIBs. Since the SIB programming overhead depends on the access schedule (see Section II), the access schedule should be considered in P1687 network design, as is discussed in Section IV and Section V.

In this paper, to prioritize the access time for different instruments, it will be assumed that each instrument has a *weight*. The weight is the number of accesses to the instrument. Each access requires SIB control bits which add to the SIB programming overhead. Instruments with higher weights could have a larger contribution to SIB programming overhead than those with lower weights. Design of P1687 networks should minimize the number of SIBs on the scan-path of the instruments with high weights to reduce SIB programming overhead.

Above it was seen that effective reduction of access time is possible by reduction of SIB programming overhead. Therefore, the P1687 network design problem is defined as follows: Given a set S of instruments, where W_i is the weight of the instrument i ($i \in S$), and a schedule which can be either concurrent or sequential, a P1687 network should be found, such that the SIB programming overhead is minimized and the number of SIBs is kept low.

IV. Method for Concurrent Schedules

Fig. 2(a) shows N instruments (represented by the white boxes) in a single-level design, i.e. no hierarchy, which is referred to as the flat architecture in the rest of this paper. Fig. 2(b) shows the same instruments in a two-level design. W_i is the weight for instrument i. The instruments are ordered so that $W_1 > \ldots > W_K > \ldots > W_N$. In the concurrent schedule, all instruments are accessed at the same time and all accesses are performed as soon as possible in the schedule. Some instruments have fewer accesses than the others. By



(a) In a single-level design (b) In a two-level design

Fig. 2. N instruments in single-level and two-level designs

closing the instrument SIBs whose corresponding instruments are not accessed anymore (say instruments K through N) the scan-path will become shorter for the instruments that are still accessed (say instruments 1 through $K-1$). For the flat architecture, this leaves the closed instrument SIBs themselves on the scan-path, contributing to the SIB programming overhead for each subsequent access. By using multi-level (hierarchical) designs, such as the two-level design shown in Fig. 2(b), it is possible to reduce the SIB programming overhead due to the instrument SIBs (for instruments K through N) by excluding them from the scan-path.

Before accessing instruments in the network shown in Fig. 2(a), all the SIBs should be opened. This is done by shifting N bits to program the SIBs followed by a CUC. These N bits are considered overhead since they are not part of the input/output data for the instruments. Furthermore, each of the N SIBs that are on the active scan-path must be programmed for every access. Since W_1 is the maximum number of accesses among the instruments, a total of W_1 accesses will be performed in the concurrent schedule and $(W_1+1)\cdot N$ clock cycles are spent in total on shifting these SIB control bits. Therefore, the SIB programming overhead for the design shown in Fig. 2(a) is calculated as $O = N+(W_1+1)\cdot N$.

To access the instruments in the network shown in Fig. 2(b), K bits should be shifted in to open the SIBs at the first level of hierarchy, marked 1 through $K-1$ and d, followed by a CUC. Subsequently, SIB control bits to open SIB_K through SIB_N are shifted in, together with the first input commands for instruments corresponding SIB_1 through $\text{SIB}_{\text{K-1}}$. Therefore, $N+1$ control bits are shifted in besides the instrument data. Now that SIBs at the second level are open, W_K more accesses are performed to all the instruments. At this point, no more input data exists for the instruments for SIB_K through SIB_N and SIB_d should be closed to shorten the scan-path for the rest of instruments. Accessing the instruments W_K times, requires shifting $(W_K+1)\cdot(N+1)$ control bits. Once SIB_d is closed, the rest of input data (i.e. those left from W_1) are to be applied. This requires $(W_1 - W_K - 1)\cdot K$ more control bits to be shifted in. Therefore, the total SIB programming overhead for the design in Fig. 2(b) is calculated as $O = K + (N+1) + (W_K+1)\cdot(N+1) + (W_1 - W_K - 1)\cdot K$. Based on these calculations, it can be concluded that if (1) is satisfied for the set of N instruments shown in Fig. 2, the design in Fig. 2(b) will result in less SIB programming overhead, at the cost of the additional SIB_d. Based on this observation, Algorithm C (C for concurrent) is presented for the construction of P1687 networks, optimized for the concurrent schedule.

$$K + (N+1) + (W_K+1)\cdot(N+1) + (W_1 - W_K - 1)\cdot K < N + (W_1+1)\cdot N \quad (1)$$

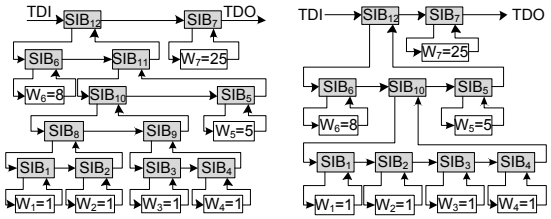
Algorithm C Method for Concurrent Schedule

```
1:  L := 1                                   //Initially the design has one level
2:  S := {W_1, W_2, ..., W_N}                //Initially S contains all the instruments
3:  while |S| > 2 do
4:      Starting from W_2, find K that satisfies (1) for the instruments in S
5:      if there is no such K then
6:          break                            //No reduction is possible
7:      end if
8:      I_L := First K − 1 instruments       //Current level gets the first K-1 instruments in S
9:      S = S − I_L                          //The used instruments are removed from S
10:     L := L + 1                           //A new level is added for the rest of the instruments
11: end while
12: I_L := S                                 //The last level contains the remainder of the instruments
```

In Algorithm C, L is the hierarchical level number. It will start at 1 (line 1) and be incremented (line 10) for each successful introduction of a new hierarchy level (lines 3-11). Initially, S contains N instruments that are represented by their weights and sorted in descending order based on their weights (line 2). If the observation regarding (1) can be applied (line 4), some instruments remain on the hierarchy level specified by L (this corresponds to moving instruments from S to I_L on line 8 and line 9) and the rest are moved to the next level of hierarchy (they remain in S for further processing). This continues until there are only two instruments in S or the observation regarding (1) cannot be applied. The outcome of Algorithm C is a list of instrument sets, named $I_1, I_2, \ldots, I_L$, where I_1 contains the instruments on the first level, I_2 contains the instruments for the second level, and so on. It should be noted that when the observation regarding (1) is applied on line 4, W_1 in (1) refers to the first element in the current set of instruments stored in S. Furthermore, adding hierarchy levels is done by adding a doorway SIB such as SIB_d in Fig. 2(b). There will be at most one doorway SIB at each level of hierarchy in the network.

V. Method for Sequential Schedules

This section studies the design of P1687 networks with the objective of access time reduction for sequential access scheduling. In sequential schedules, instruments are accessed one at a time. Therefore, the total SIB programming overhead will be the sum of the SIB programming overheads for all the instruments. The SIB programming overhead for Instrument 6, which is connected to SIB_6 in Fig. 3(a) is taken as an example. Before accessing Instrument 6, two levels of hierarchy should be opened. On the first level of hierarchy, two SIB control bits are required to open SIB_{12} and to program SIB_7 to remain closed. Subsequently, four SIB control bits are required to keep SIB_{12} open, to open SIB_6 and to program SIB_{11} and SIB_7 to remain closed. While Instrument 6 is accessed, these four SIBs will be on the scan-path. So far, six (2+4) bits are shifted to open the SIBs before the first access to Instrument 6. To complete all eight ($W_6 = 8$) required accesses to the Instrument 6, nine repetitions of the programming of the four SIBs on the scan-path are required. After eight repetitions, all data to the instrument has been shifted in and one more repetition is required to shift out the output data for the eighth access. For accessing Instrument 6 eight times, $(8+1)\times 4$ SIB control bits are required because of the four SIBs on the scan-path. In total,



(a) Output of Algorithm H (b) Output of Algorithm HO

Fig. 3. Example P1687 networks

the SIB programming overhead due to accessing Instrument 6 is 42 $(2 + 4 + (8 + 1) \times 4)$ clock cycles.

As mentioned in Section III, the instrument with the largest weight could have the largest contribution to the SIB programming overhead. Such instruments should be on a short scan-path. In terms of a multi-level network, instruments with large weight should be placed on a level close to the JTAG TAP to avoid many SIBs on their scan-paths. Also, instruments with lesser weight should be placed on a level further away from the TAP so that their instrument SIBs do not add to the scan-paths of the instruments with larger weight. To develop an algorithm for constructing a P1687 network with the above mentioned placement of instruments according to their weights, we have taken inspiration from Huffman Construction, which is a method for constructing labeled trees of symbols, used in variable length coding [7]. The basic idea in Huffman Construction is that symbols with higher frequency of occurrence (weight) are assigned shorter length code words. To construct such a tree, symbols with larger weights are placed closer to the root of the tree.

In construction of a P1687 network, an analogy can be made between weight of a symbol in Huffman Construction, and the weight of an instrument. That is, since instruments with larger weights are accessed more frequently, they should be placed in the P1687 network such that the number of SIBs on their scan-path (which is analogous to the length of the code word for the symbol) becomes relatively low. Algorithm H (H for Huffman) shows the steps to construct a P1687 network out of a given set of instruments, such that the access time is optimized for the sequential schedule. On line 1, Algorithm H receives a set of weights for the instruments. The algorithm applies a key idea of Huffman Construction, which is to *combine* a set X of instruments (lines 4-6) and treat them as one instrument, where $W_X = \sum_{i \in X} W_i$. To combine a set X of instruments, a doorway SIB is added and the set X of instruments are connected to its HIP. In Algorithm H, two instruments are combined at a time. By starting with the instruments with the smallest weight (line 3), they will end up in the hierarchy levels further away from the JTAG TAP. This means that instruments with high weights end up with a short scan-path. The procedure of combining instruments continues until all instruments have been combined on the HIP of a single doorway SIB (lines 2-7) which is replaced by JTAG TAP (TDI-TDO) afterwards.

Fig. 3(a) shows the P1687 network that was designed using Algorithm H for a set of instruments with the weights 1, 1, 1, 1, 5, 8 and 25. It should be noted how the instruments are placed in the network. The weights determine the hierarchy level and

Algorithm H Construction for Sequential Schedule

1: $S := \{W_1, W_2, \ldots, W_N\}$
2: **while** $|S| > 1$ **do**
3: Find W_i and W_j that are smaller than all other items in S
4: Combine the two instruments i and j to form X
5: Remove W_i and W_j from S
6: Add W_X to S
7: **end while**

Algorithm HO Method for Sequential Schedule

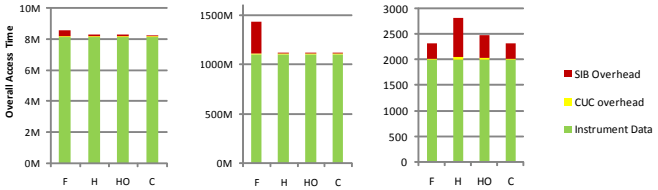
1: run Algorithm H
2: **for** each SIB_d **do**
3: $SIBOverhead$:= SIB programming overhead of the network
4: Remove SIB_d
5: $NewSIBOverhead$:= SIB programming overhead of the network
6: **if** $NewSIBOverhead > SIBOverhead$ **then**
7: Restore SIB_d
8: **end if**
9: **end for**

the instrument with the highest weight (25) is placed so that it can be accessed with only two SIBs on the scan-path. If the instruments in Fig. 3(a) were arranged in flat architecture, the SIB programming overhead would be 350 clock cycles with the sequential schedule, while the SIB programming overhead for the design in Fig. 3(a) is 244 clock cycles. Therefore, reduction of SIB programming overhead is achieved at the cost of five additional doorway SIBs (SIB_8 through SIB_{12}).

It can be possible to further reduce the SIB programming overhead in the network constructed by Algorithm H. From the design shown in Fig. 3(a), SIB_8, SIB_9 and SIB_{11} can be removed, as shown in Fig. 3(b), to reduce the SIB programming overhead to 215 clock cycles. The reason for this possibility of further SIB programming overhead reduction is that in the analogy to Huffman Construction, there is no counterpart for the SIB programming overhead coming from opening the SIBs before the first access to a given instrument. An optimization step should therefore follow the construction, to analyze a P1687 network and find the doorway SIBs that should be removed to further reduce the SIB programming overhead. The complete method for the sequential schedule is thus as suggested in Algorithm HO (HO for Huffman Optimized). The basic idea in Algorithm HO is to construct an initial network, using Algorithm H, and examine the effect of removal of each of the doorway SIBs in that network (line 4) on the total SIB programming overhead. Removal of a doorway SIB is done by replacing the doorway SIB by the network segment on its HIP. To this end, Algorithm HO compares the SIB programming overhead before (line 3) and after (line 5) removal of each of the doorway SIBs, and restores the removed SIB (line 7) if the SIB programming overhead increases after removal of the SIB (line 6).

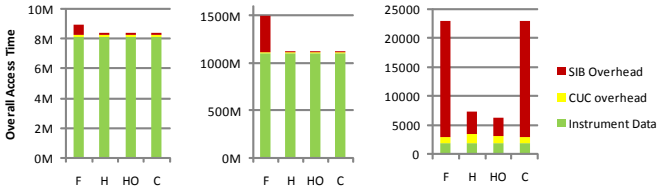
VI. Experimental Setup

A design automation tool, P1687 Automatic Construction Tool (PACT), has been implemented. As inputs PACT accepts a schedule type (either concurrent or sequential) and a set of instruments S, specified by a weight W_i (see Section III) which represents the number of accesses that are required for instrument i ($i \in S$). The output of PACT is a description of a P1687 network (a tree representation with SIBs for nodes,



(a) P22810 (b) Merge12 (c) S100

Fig. 4. OAT of the designs when accessed using concurrent schedule



(a) P22810 (b) Merge12 (c) S100

Fig. 5. OAT of the designs when accessed using sequential schedule

where leaf nodes are instrument SIBs, associated with the corresponding instrument) for which PACT has endeavored to achieve a low instrument access time while attempting to keep the number of doorway SIBs low. When the concurrent scheduling type is given as input, PACT performs Algorithm C. In Section VII this is called the C approach. Otherwise, if the schedule type is sequential, PACT performs Algorithm HO which leads to an initial P1687 network (approach H) and a final network (approach HO). Besides the C, H and HO approaches from PACT we define the F approach representing the flat architecture, for comparison. Although some of the above-mentioned approaches are optimized for a certain schedule, all four approaches are used in all of the experiments presented in Section VII, again for comparison.

In experiments with PACT, as input a set of instruments is required. We have, without loss of generality, chosen to view the cores of the ITC'02 [8] Benchmark SOCs as instruments. These can represent many types of instruments because of the variety in the length of shift-registers and the number of accesses found among the instruments. Consequently, in the context of the experiments, the instruments are cores and the shift-register of an instrument is the *core-chain* for each core. In this case the instrument data consist of test stimuli (applied as inputs) and test responses (captured as outputs), and an *access* is application of one test pattern. Because of how access is defined in Section III test application time is identical to overall access time. Since in the context of P1687, all data are transported through a single wire, the internal scan-chains and boundary cells corresponding to the core inputs and outputs are concatenated to form a *core-chain*. The length of a *core-chain* is calculated as described in [6]. Besides the ITC'02 Benchmarks, we experimented with two SOCs, Merge12 and S100. Merge12 is the full set of instruments from all 12 SOCs available in the ITC'02 Benchmark Set. Merge12 has 167 instruments and is investigated to evaluate PACT for a large set of instruments. S100 contains 100 instruments, each with a 10-bit shift-register and each requiring one access. S100 is investigated to consider a circuit with many simple instruments which require few accesses and have short shift-registers. For space reasons, this paper only reports results on the ITC'02 benchmark P22810, Merge12 and S100.

To evaluate the P1687 networks that resulted from the experiments, we report SIB programming overhead and CUC overhead, as well as OAT. Besides reporting OAT, Section VII gives the average access time which is the OAT divided by the total number of accesses, and the average number of SIBs on the scan-path, which is the total SIB programming overhead divided by the number of accesses.

VII. Experimental Results

Fig. 4 and Fig. 5 show overall access time (OAT) for P22810, Merge12 and S100 for the concurrent and sequential access schedules, respectively. The bars show the fractions of OAT that correspond to transport of instrument data, transport of SIB control bits (SIB programming overhead) and performing CUC (CUC overhead). The results are presented in detail in Table I. For each SOC, Column 1 shows the number of instruments (cores) and Column 2 presents the amount of instrument data for each SOC. Column 3 indicates the design approach considered on each row and Column 4 indicates the number of doorway SIBs in the resulting design. For all four approaches, the number of instrument SIBs is equal to the number of instruments and not included in Table I. The instrument data is calculated as $\sum_{i=1}^{N} L_i \cdot (W_i + 1)$, where N is the number of instruments. W_i and L_i are the number of accesses and the length of the shift-register for instrument i, respectively. Columns 5-9 and Columns 10-14 show results for schedules of the sequential and concurrent types respectively. Within both blocks, CUC overhead and SIB programming overhead are presented along with OAT. Furthermore, Column 7 and Column 12 show the average number of SIBs on the scan-path considering all accesses (see Section VI). Similarly, Column 9 and Column 14 show the average instrument access time (see Section VI).

The primary aim of PACT is to reduce instrument access time by reducing SIB programming overhead compared to the flat architecture. Such reduction can be seen in Fig. 4 and Fig. 5. From Table I it can be seen that, the impact of instrument data on OAT remains constant for different P1687 networks (the results of the F, H, HO and C approaches) and different access schedule types. In contrast, Fig. 4 and Fig. 5 show that SIB programming overhead and CUC overhead vary with both network and schedule type. The variation in SIB programming overhead is considerable and is the main parameter that can be adjusted to reduce OAT, while CUC overhead varies only slightly, which is why PACT is developed to reduce SIB programming overhead.

For P22810 and Merge12, it can be seen that the resulting networks corresponding to H, HO and C result in similar OAT. In such cases, the secondary aim of PACT, to keep the number of SIBs low without increasing OAT, is considered in Column 4 of Table I. In the context of the primary and secondary aims, the following shows that PACT operates correctly. Fig. 4 shows that for the concurrent schedule type, the C approach result in the lowest OAT. Therefore, PACT correctly recommends the C approach when instructed to optimize for the concurrent

TABLE I

SOC	Instrument Data	P1687 Design Approach	# Doorway SIBs	Sequential Schedule					Concurrent Schedule				
				CUC Overhead	SIB Prog. Overhead Total	SIB Prog. Overhead Average	Access Time Total	Access Time Average	CUC Overhead	SIB Prog. Overhead Total	SIB Prog. Overhead Average	Access Time Total	Access Time Average
P22810 (28 cores)	8172047	F	0	125210	701176	28.00	8998433	359.35	61630	345128	13.78	8578805	342.59
		H	26	125340	133734	5.34	8431121	336.69	61630	57526	2.30	8291203	331.10
		HO	15	125285	131715	5.26	8429047	336.61	61630	62741	2.50	8296418	331.31
		C	17	125295	148635	5.93	8445977	337.28	61630	46886	1.87	8280563	330.68
Merge12 (167 cores)	1105716046	F	0	11427410	381675494	167.00	1498818950	655.80	9572175	319710645	139.89	1434998866	627.88
		H	165	11428235	7253344	3.17	1124397625	491.97	9572175	4772942	2.09	1120061163	490.08
		HO	94	11427860	7225621	3.16	1124369527	491.96	9572175	4836595	2.12	1120124816	490.10
		C	101	11427915	10521078	4.60	1127665039	493.40	9572175	4520598	1.98	1119808819	489.97
S100 (100 cores)	2000	F	0	1005	20100	100.50	23105	115.52	15	300	1.50	2315	11.57
		H	98	1495	3834	19.17	7329	36.64	45	784	3.92	2829	14.14
		HO	21	1175	3083	15.41	6258	31.29	30	447	2.23	2477	12.38
		C	0	1005	20100	100.50	23105	115.52	15	300	1.50	2315	11.57

schedule. Similarly, Fig. 5 for the sequential schedule, shows that the HO approach leads to the lowest OAT. It should be noted, that while the H approach achieves a reasonably low OAT, it results in a higher number of doorway SIBs than the HO approach (Table I). When the H and C approaches are comparable to the HO approach in terms of OAT, HO results in a lower number of doorway SIBs. Therefore, PACT correctly recommends the HO approach when instructed to optimize for the sequential schedule.

For P22810, it can be seen that PACT reduces OAT by a small fraction compared to the result of the flat architecture, at the cost of 15 and 17 additional SIBs (see HO and C for P22810 in Table I). A more considerable reduction (25% of OAT and 25% of average access time for sequential schedules) is seen for circuit Merge12, where the reduction is achieved at the cost of 94 additional SIBs (see HO for Merge12 in Table I). More dramatic reduction in OAT is achieved for S100. From the results for the three SOCs it can be seen that the benefit of applying PACT to a circuit depends on the set of instruments in the SOC. In this context, PACT is useful for evaluating a SOC in terms of the size of the possible reduction in OAT.

For Merge12, the SIB programming overhead ratio is very large for the F approach and becomes significantly smaller for all other approaches which have hierarchical architecture. This can be explained by the fact that Merge12 contains an instrument with $W = 1914433$. Considering a flat architecture with all 167 cores, this instrument causes a SIB programming overhead of $167 \times 1914433 = 319710311$ clock cycles. This alone, constitutes 84% of the SIB programming overhead for the F approach. The number of SIBs on the scan-path to this instrument is 167 whereas the same number for the H, HO and C approaches is 2, which is reflected by the large reduction in average SIB programming overhead shown in Table I. The drastic amounts of SIB programming overhead for circuit S100 and the sequential schedule (Fig. 5) is due to the fact that the instrument shift-registers are short compared to the scan-path length, especially for the F and C designs in which each instrument has 100 SIBs on its scan-path. However, in Table I, the average SIB programming overhead is 100.50 (and not 100) because this number includes the overhead invested in opening the first SIB (see Section V). For the concurrent schedule type, the average SIB programming overhead is less because it is amortized over more than one instrument.

From the above, it is seen that PACT can reduce instrument access time and keep the cost in terms of additional doorway SIBs low, which is a contribution to the development of design automation tools for circuits incorporating P1687. For all of the experiments, including those with >100 instruments, PACT produced the recommended P1687 network within <10 seconds, on a 1.83 GHz Intel® Core™2 Duo based computer with 3 GB of RAM.

VIII. Conclusion

IEEE P1687 standard proposal aims at standardizing the access to the on-chip test, debug and monitoring logic (called instruments) through JTAG TAP. To construct the access network, P1687 proposes a component called SIB to be used to connect to instruments or other SIBs. By using SIBs, it is possible to design a multitude of access networks for the same set of instruments. This paper contributes to the development of EDA tools by presenting algorithms for the automated design of optimized P1687 networks. The algorithms are implemented in a tool called PACT (P1687 Automatic Construction Tool). Given a set of instruments and an access schedule which can be either sequential or concurrent, PACT designs a P1687 network which is optimized with respect to instrument access time while the cost in terms of number of SIBs is kept low. It was shown that reducing control data overhead (for programming SIBs) is the key to reduce the overall access time. Therefore, this paper focused on reduction of SIB programming overhead. To this end, hierarchical structures, that provide a shorter scan-path for the instruments which are more frequently accessed, proved effective. PACT is employed in experiments on industrial SOCs and two designs with >100 instruments. The results showed that in a matter of seconds PACT helped reduce access time by up to 25%, compared with straight-forward single-level structures without hierarchy for the same set of instruments.

References

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

[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] L.-T. Wang *et al.*, "Turbo1500: Toward Core-Based Design for Test and Diagnosis Using the IEEE 1500 Standard," in *Proc. ITC*, 2008, pp. 1–9.

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

[5] 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.

[6] F. Ghani Zadegan, U. Ingelsson, G. Carlsson, and E. Larsson, "Test Time Analysis for IEEE P1687," in *Proc. ATS*, 2010.

[7] R. P. Grimaldi, *Discrete and Combinatorial Mathematics*. Pearson Education, 2004, ch. 12, pp. 609–614.

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