Institutionen för datavetenskap
Department of Computer and Information Science

Final Thesis

Test Scheduling with Power and Resource Constraints for IEEE P1687

by

Golnaz Asani

LIU-IDA/LITH-EX-A--12/004--SE

2012-05-15
Final Thesis

Test Scheduling with Power and Resource Constraints for IEEE P1687

by

Golnaz Asani

Supervisor: Urban Ingelsson and Farrokh Ghani Zadegan
Dept. of Computer and Information Science at Linköpings Universitet

Examiner: Erik Larsson
Dept. of Computer and Information Science at Linköpings Universitet
Abstract

IEEE P1687 (IJTAG) is proposed to add more flexibility—compared with IEEE 1149.1 JTAG—for accessing on-chip embedded test features called instruments. This flexibility makes it possible to include and exclude instruments from the scan path. To reach a minimal test time, all instruments should be accessed concurrently. However, constraints such as power and resource constraints might limit concurrency. There is a need to consider power and resource constraints while developing the test schedule.

This thesis consists of two parts. In the first part, three test time calculation approaches, namely session-based test schedule with a fixed scan path, session-based test schedule with a reconfigurable scan path, and session-less test schedule with a reconfigurable scan path are proposed. In the second part, three test scheduling approaches, namely session-based test scheduling, optimized session-based test scheduling, and optimized session-less test scheduling are studied and three algorithms are presented for each of the test scheduling approaches. Experiments are carried out using the test scheduling approaches and the results show that optimized session-less test scheduling can significantly reduce the test time compared with session-based test scheduling.

Keywords: IJTAG, IEEE P1687, Test Time Calculation, Test Scheduling, Resource Constraints, Power Constraints
Acknowledgements

I am grateful to my examiner Dr. Erik Larsson for generous support and patient guidance throughout the work with this thesis.

I would like to thank both of my supervisors; Farrokh Ghani Zadegan and Urban Ingelsson for all of their help and support throughout my master thesis work. None of this would’ve been possible without their guidance and formidable knowledge in the domain area. Thank you! My deepest appreciation also goes out to my family and friends who have supported, comforted and aided me in my times of need and made me feel warm and welcome during times of stress and worry throughout this process.

I also would like to thank all the people who have helped me with administration and other things also involved in the thesis work, as well as my opposition Hoda Ghaeini, for taking her time to read and comment on my work.
## Contents

<table>
<thead>
<tr>
<th>List of Figures</th>
<th>ix</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Tables</td>
<td>xi</td>
</tr>
<tr>
<td><strong>1 Introduction and Background</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Organization of the Report</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Prior Work</td>
<td>3</td>
</tr>
<tr>
<td>1.4 Introduction to IEEE P1687 (IJTAG)</td>
<td>5</td>
</tr>
<tr>
<td>1.5 Concurrent Access and Sequential Access in P1687 Environment</td>
<td>6</td>
</tr>
<tr>
<td>1.6 Problem Definition</td>
<td>9</td>
</tr>
<tr>
<td><strong>2 Test Time Calculation</strong></td>
<td>13</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>13</td>
</tr>
<tr>
<td>2.2 Session-Based Test Schedule with a Fixed Scan path</td>
<td>14</td>
</tr>
<tr>
<td>2.3 Session-Based Test Schedule with a Reconfigurable Scan Path</td>
<td>17</td>
</tr>
<tr>
<td>2.4 Session-Less Test Schedule with a Reconfigurable Scan Path</td>
<td>20</td>
</tr>
<tr>
<td>2.5 Summary</td>
<td>23</td>
</tr>
<tr>
<td><strong>3 Test Scheduling Approaches and Algorithms</strong></td>
<td>25</td>
</tr>
<tr>
<td>3.1 Mapping Test Scheduling to Strip Packing</td>
<td>25</td>
</tr>
<tr>
<td>3.2 Test Scheduling Approaches in P1687</td>
<td>28</td>
</tr>
<tr>
<td>3.2.1 Session-Based Test Scheduling</td>
<td>29</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>3.2.2 Optimized Session-Based Test Scheduling</td>
<td>30</td>
</tr>
<tr>
<td>3.2.3 Optimized Session-Less Test Scheduling</td>
<td>31</td>
</tr>
<tr>
<td>3.3 Test Scheduling Algorithms</td>
<td>32</td>
</tr>
<tr>
<td>3.3.1 Session-Based and Optimized Session-Based Test Scheduling</td>
<td>32</td>
</tr>
<tr>
<td>3.3.2 Optimized Session-Less Test Scheduling</td>
<td>33</td>
</tr>
<tr>
<td>3.4 Summary</td>
<td>34</td>
</tr>
<tr>
<td>4 Experiments</td>
<td>37</td>
</tr>
<tr>
<td>4.1 Experimental Setup</td>
<td>37</td>
</tr>
<tr>
<td>4.2 Experimental Results</td>
<td>38</td>
</tr>
<tr>
<td>5 Conclusion and Future Work</td>
<td>41</td>
</tr>
<tr>
<td>5.1 Conclusion</td>
<td>41</td>
</tr>
<tr>
<td>5.2 Future Work</td>
<td>42</td>
</tr>
<tr>
<td>Bibliography</td>
<td>43</td>
</tr>
</tbody>
</table>
# List of Figures

1.1 Resource graph for the example system .................................. 3
1.2 Schedule types ........................................................................ 4
1.3 JTAG circuitry with a gateway register for IEEE P1687 access ........ 6
1.4 Instruments $I$ and $I'$ accessed sequentially ............................... 7
1.5 Instruments $I$ and $I'$ accessed concurrently ................................. 8
1.6 Impact of concurrency on test duration in P1687 ......................... 11

2.1 P1687 network used in this chapter .......................................... 14
2.2 Session-based test schedule with fixed scan path ......................... 15
2.3 Network configurations for session S1 in the session-based test schedule with a fixed scan path, shown in Figure 2.2 ......................... 16
2.4 Session-based test schedule with a reconfigurable scan path .......... 18
2.5 Scan path configurations used for session S1 in the session-based test schedule with a reconfigurable network shown in Figure 2.4 .................................................. 19
2.6 Session-less test schedule with a reconfigurable scan path .......... 20
2.7 A P1687-specific test schedule ................................................. 22
2.8 Scan path configurations ......................................................... 23

3.1 Test schedule represented by the individual test time for each test .................................................................................. 26
3.2 Test scheduling based on the number of test patterns .................... 28
3.3 Optimized session-less test schedule .......................................... 28
3.4 Session-based test schedule ..................................................... 29
3.5 Optimized session-based test schedule . . . . . . . . . . . . 30
3.6 Optimized session-less test schedule . . . . . . . . . . . . 31
# List of Tables

2.1 Properties for the instruments used in the examples throughout this chapter ........................................ 14
2.2 Test time calculation steps for the schedule in Fig 2.2 ................................................................. 17
2.3 Test time calculation steps for the schedule in Fig 2.4 ................................................................. 18
2.4 Representation of a session-less schedule using a succession of virtual sessions .......................... 21
2.5 Test time calculation steps for the schedule in Fig 2.7 ................................................................. 23
3.1 Properties for the instruments ................................................................. 26
4.1 Experimental Results ................................................................. 40
instruments is required, a scan path should be formed out of shift registers interfaced to all of those instruments. In [3] it is shown that in a JTAG scenario, such concurrent testing of instruments might lead to a significant test time overhead. Compared with JTAG, IEEE P1687 (IJTAG) [4] proposes a standard that enables a more flexible access to embedded instruments [5]. This flexibility allows to include only those instruments into the scan path which are needed to be tested in a given test scenario, and to exclude instruments from the scan path as soon as their test is completed, thus helping to avoid unnecessary time overhead. Avoiding time overhead in turn leads to reduced test time, which is highly related to cost, and needs to be minimized. In [3] it is shown that in IJTAG, fully concurrent test of instruments leads to the lowest test time. However, power and resource constraints may limit the use of concurrent tests. As an example of resource constraint, two instruments which use the same hardware resource cannot be tested simultaneously. As an example of power constraint, the power consumption of instruments tested at the same time should not exceed the maximum power limit which is defined for the IC. Therefore, there is a need for a scheduling method that considers constraints and minimizes test time. This thesis analyzes and proposes a solution to the resource and power-aware test scheduling problem in an IJTAG environment.

1.2 Organization of the Report

To describe the concepts discussed in this thesis, this chapter continues by presenting related work regarding scheduling (namely session-based and session-less schedules), describing IJTAG and presenting the problem statement. In Chapter 2, a test time calculation method is developed for general session-based and session-less schedules. Chapter 3 discusses test scheduling analysis and describes resource and power-aware test scheduling in an IJTAG environment. Further, Chapter 3 presents three test scheduling algorithms, namely session-based (SB), optimized session-based (OSB) and optimized session-less (OSL) test scheduling. Chapter 4 describes the experiments and the achieved results. The last chapter (Chapter 5) summarizes the thesis and discusses future work.
1.3 Prior Work

The goal of test scheduling is to decrease test time while considering constraints. Research in power- and resource-constrained test scheduling on ICs with cores that are equipped with built-in self test (BIST) has been done by Zorian [6], Chou, Saluja and Agrawal [7], and Muresan [8]. It was recognized that testing causes considerably higher circuit activity compared with normal operation. Higher circuit activity during test becomes an issue in IC manufacturing test. To keep test time at a minimum, several BISTs are activated concurrently, leading to high power consumption which may exceed the maximum power limit for the IC. So, appropriate scheduling of BIST-based tests is needed to satisfy the power consumption limit. The aims of [6] are on finding a test schedule that meets power constraints, and optimizing test time.

In [7], to address the test scheduling problem for tests sharing a resource, the power-constrained test scheduling problem is extended to also include resource constraints. For example, Figure 1.1 shows a model of a system. The test of each block $C_i$ is represented by $T_j$, and the resources are represented by $R_k$. For example, to test block $C_1$, test $T_1$ uses resource $R_1$. Tests $T_2$ and $T_3$ use the same resource $R_2$, and tests $T_3$ and $T_4$ use the same resource $R_3$. Therefore, tests $T_2$ and $T_3$, and tests $T_3$ and $T_4$ cannot be activated together. Tests $T_1$ and $T_3$ do not have any resource conflict, therefore they can be activated at the same time.

![Figure 1.1: Resource graph for the example system](image-url)
After finding the sets of tests that can be activated at the same time, a maximum power limit is considered for each of the tests. If tests $T_1$, $T_2$, $T_3$, and $T_4$ consume 400, 250, 200 and 200 power units, respectively, and given is a maximum power limit of 625 units, tests $T_1$ and $T_2$ cannot be activated at the same time because sum of the power consumption becomes higher than the maximum power limit, therefore tests $T_1$ and $T_3$ can be activated simultaneously. Tests that can be activated concurrently considering power and resource constraints are called time compatible tests. A session consists of a group of compatible tests that are performed concurrently. Each test has a test time and the duration of a test session is determined by the longest test in the session. The overall test time is the sum of the test time of each session. Chou [7] proposed a session-based schedule as shown in Figure 1.2(a) for the problem stated above. As Figure 1.2(a) represents, tests $T_1$ and $T_3$ can be activated concurrently in session 1, and Tests $T_2$ and $T_4$ can be activated concurrently in session 2. Muresan proposed a method for generating test schedules, which is called session-less test scheduling [8]. Session-less test schedules can decrease test time [9] compared with session-based test schedules, because each test is activated as soon as the previous test is finished. The concept of a session-less schedule is represented in Figure 1.2(b). The test access mechanism to implement a session-less schedule is not detailed in [8].

![Figure 1.2: Schedule types](image-url)

In [3] algorithms are presented to calculate the overall test time for two test schedules, namely fully concurrent and fully sequential test schedules. According to [3], in fully concurrent schedules all tests start at the same
time and the instruments are excluded from scan path as soon as their test is completed, and in fully sequential schedules tests are performed one by one and the next test starts as soon as the previous test is completed. However, no study has considered test time calculation for session-based and session-less schedules, which is the focus of this thesis work. Moreover, no previous study has been carried out on scheduling for an IJTAG environment. This thesis work presents power- and resource-constrained test scheduling with the aim of test time reduction.

1.4 Introduction to IEEE P1687 (IJTAG)

To add flexibility to the scan path, IEEE P1687 specifies a component called Segment Insertion Bit (SIB). SIBs can be programmed to include (exclude) instrument shift registers in (from) the scan path. P1687 has an informal name Internal JTAG (IJTAG) because it proposes to use the IEEE 1149.1 JTAG test access port (TAP) for accessing on-chip instruments from outside the chip [10]. A special Test Data Register (TDR) called Gateway is added in the JTAG circuitry to interface on-chip IJTAG circuitry with the JTAG TAP. A JTAG instruction called Gateway Enable (GWEN) selects the Gateway TDR. Gateway may be a single SIB or a number of SIBs connected in series. Figure 1.3 shows a partial JTAG circuitry with a Gateway register made of four SIBs. When the Gateway TDR is enabled, it connects Test-Data-Input (TDI) and Test-Data-Output (TDO) to an IJTAG network that consists of SIBs and instruments. As Figure 1.3 presents, the Gateway TDR forms a flexible scan path between the TDI and TDO terminals of TAP [11] by the SIBs which can be used to include the instruments in or exclude them from the scan path. The SIB in JTAG acts as a 1-bit register on the scan path during shifting. If a SIB is open, it includes the instrument in the scan path and the data is shifted into the instrument via a shift register (represented by SHR boxes in Figure 1.3). Otherwise, if the SIB is closed, data is shifted straight through. In Figure 1.3 assuming that only SIB1 and SIB3 are open, the bold lines represent the scan path including the DFT instrument and the PLL. Since SIB2 and SIB4 are closed, the corresponding instruments are excluded from the scan path. In the next section, for the network in Figure 1.3, it will be
1.5. Concurrent Access and Sequential Access in P1687 Environment

explained how instruments can be accessed sequentially and concurrently. Moreover, it will be discussed how the type of access impacts the test time.

![Diagram of JTAG circuitry with a gateway register for IEEE P1687 access](image)

**Figure 1.3:** JTAG circuitry with a gateway register for IEEE P1687 access

### 1.5 Concurrent Access and Sequential Access in P1687 Environment

In the following discussion, test application consists of applying test patterns. Each test pattern consists of test stimuli and the expected responses. To apply a test pattern, the test stimuli is shifted in, applied, and the responses are captured and shifted out. The shifted out responses, the produced responses, are compared against the expected responses. In a
sequential test schedule, tests are applied one by one. The next test starts when the current test is completed. Figure 1.4 represents the steps in sequential testing of instruments $I_1$ (DFT instrument) and $I_3$ (PLL) in Figure 1.3. In the first step Figure 1.4(a), SIB1 is programmed to be opened and instrument $I_1$ is tested. After finishing the testing of instrument $I_1$, SIB1 is closed and SIB2 is programmed to be opened and instrument $I_2$ is tested Figure 1.4(b). Figure 1.5 illustrates concurrent test scheduling. The SIBs of instruments tested concurrently (i.e. instruments $I_1$ and $I_3$) are programmed to be opened at the same time, and instruments $I_1$ and $I_3$ are tested simultaneously.

![Figure 1.4: Instruments $I_1$ and $I_3$ accessed sequentially](image)

To illustrate the concept of concurrency, consider $I_1$ (shift register length $l_1 = 2$ and the number of test patterns $(tp_1) = 2$) and $I_3$ (shift...
1.5. Concurrent Access and Sequential Access in P1687 Environment

Figure 1.5: Instruments $I_1$ and $I_3$ accessed concurrently

register length $l_3 = 1$ and the number of test patterns ($tp_3$) = 2). Figure 1.6 represents the clock cycles of applying tests to instruments $I_1$ and $I_3$ concurrently and sequentially. Test time in P1687 environment consists of time transporting data and overhead. There are two types of overhead in P1687 environment, namely SIB programming overhead and JTAG protocol overhead. **SIB programming overhead** is the time spent transporting SIB control bits which are represented by s boxes in Figure 1.6. **JTAG protocol overhead** is a progression of five states in the TAP controller state machine during apply and capture which are Exit-DR, Update-DR, Select-DR-Scan, Capture-DR, and Shift-DR. These progression is called CUC (Cycle of Update and Capture) [3] which is represented by five c boxes.

Figure 1.6(a) represents testing $I_1$ and $I_3$ sequentially. Initially all SIBs are closed. To test instrument $I_1$, SIB1 should be opened and all other SIBs should remain closed. This initial programming is called setup sequence. The setup sequence involves programming SIBs and performing CUC as shown in Figure 1.6(a) by the five leftmost s blocks followed by a CUC (five c blocks).

After opening SIB1, as Figure 1.6(a) shows, a scan sequence is applied. The scan sequence involves two parts. The first part is shifting test data for all active instruments and SIB control bits—represented by boxes containing the ID number of the instruments and s boxes, on the light gray line in Figure 1.6. In the example, this part consists of six bits, two bits for shifting data into instrument $I_1$ and four bits for programming SIBs.
The second part is applying test stimuli and capturing the corresponding responses (CUC).

Subsequently, the captured test responses can be shifted out as represented by the dark gray line. The shift-out of the test responses can be done at the same time as shifting in of the next test stimuli. After shifting out the responses for the second test stimuli, testing of instrument $I_1$ is completed.

To test instrument $I_3$ the same procedure should be followed.

Figure 1.6(b) represents testing $I_1$ and $I_3$ concurrently. In the initial SIB programming both SIB1 and SIB3 are opened. So, the shift registers of $I_1$ and $I_3$ are in the scan-path. Test stimuli of $I_1$ and $I_3$ can be applied at the same time, and the test responses of $I_1$ and $I_3$ can be shifted out at the same time. As can be seen from the example, in each scan sequence there is an overhead. Reducing the number of scan sequences through concurrent testing leads to less SIB programming overhead and less CUC overhead and consequently lower test time. Therefore, in this thesis the key idea for reducing test time is to schedule tests to maximize concurrency.

1.6 Problem Definition

**Problem** [Test Scheduling for Optimizing Test Time in a P1687 environment (TSOTT)]

Given is a P1687 network consisting of a set of instruments where each instrument is connected to the Gateway through a dedicated SIB; And that for each instrument (I), there is a unique ID ($i$), a number of test patterns ($tp_i$), a shift register length ($l_i$), and a peak power consumption value ($p_i$). Also given are power and resource constraints for the system. The power constraint is the maximum allowed peak power at any time during testing. The resource constraints are represented as a set of elements with the form of $(I_j, I_k)$ which specifies that instrument $I_j$ cannot be active at the same time with instrument $I_k$. The problem is to find a test schedule such that test time is minimized while the power and resource constraints are satisfied.

**Problem** [Test Time for General Schedules (TTGS)]

Given a P1687 network and a set of instruments as stated in Problem
1.6. Problem Definition

TSOTT, and a general test schedule which can be either session-based or session-less, calculate test time.
(a) Testing $I_1$ and $I_3$ sequentially

(b) Testing $I_1$ and $I_3$ concurrently

Figure 1.6: Impact of concurrency on test duration in P1687
1.6. Problem Definition
Chapter 2

Test Time Calculation

2.1 Introduction

In this chapter, to calculate test time for general test schedules (Problem TTGS), three approaches are proposed for session-based test schedules with a fixed scan path, for session-based test schedules with a reconfigurable scan path, and for session-less test schedules with a reconfigurable scan path. In a fixed scan path, no change in the P1687 network configuration (i.e. the opened/closed status of the SIBs) is made within a session, and therefore, instruments employed in the test remain on the scan path until the end of the session. However, in a reconfigurable scan path, instruments are excluded from the scan path as soon as their testing is completed, by programming the SIBs accordingly.

For the examples in this chapter, five instruments are considered whose properties (namely number of test patterns, shift register length, and peak power consumption) are listed in Table 2.1, and are assumed to be connected to a P1687 network illustrated in Figure 2.1.
2.2 Session-Based Test Schedule with a Fixed Scan path

Table 2.1: Properties for the instruments used in the examples throughout this chapter

<table>
<thead>
<tr>
<th>Instrument</th>
<th>(I_1)</th>
<th>(I_2)</th>
<th>(I_3)</th>
<th>(I_4)</th>
<th>(I_5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of test patterns ((tp))</td>
<td>12</td>
<td>6</td>
<td>12</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>Shift register length ((l))</td>
<td>9</td>
<td>8</td>
<td>1</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>Peak Power consumption ((p))</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

Figure 2.1: P1687 network used in this chapter

2.2 Session-Based Test Schedule with a Fixed Scan path

In session-based test schedules with a fixed scan path, during a session, the scan path does not change and all active instruments remain in the scan path until the end of the session. Figure 2.2 shows a test schedule from data in Table 2.1 with a fixed scan path. In Figure 2.2 each box \(T_i\) represents testing instrument \((I_i)\) where width of the box represents the number of test patterns \((tp_i)\) and the height of the box represents the power consumption \((p_i)\) for that instrument \((I_i)\). There are three sessions (which are represented by \(S\) in Figure 2.2) in the schedule. Session \(S1\) contains tests for instruments \((I_1)\) and \((I_2)\). The number of test patterns in the test of \((I_2)\) is lower than the number of test patterns in the test for instrument \(i1\); however, when the test for \(i2\) is completed, the SIB of instrument \((I_2)\)
is not closed and instrument \((I_2)\) remains on the scan path. Therefore, dummy bits (illustrated with black boxes in Figure 2.2) are transported until the test of instrument \((I_1)\) is completed.

![Figure 2.2: Session-based test schedule with fixed scan path](image)

Calculation of test time for a session-based test schedules with a fixed scan path will be explained by an example. Consider the five instruments in Figure 2.1, which are described in Table 2.1, and a given schedule as shown in Figure 2.2.

To calculate the test time, the duration of each session is calculated individually, and the test time is the sum of the session durations. In this chapter, all calculations are performed using tables similar to Table 2.2, where

- **Session** enumerates sessions
- **SIB** present the number of SIBs on the scan path for each scan path configuration
- **Scanned bits** represent the number of bits scanned for active instruments
- **CUC** represents the number of test clock cycles spent on performing the Cycle of Update and Capture (Section 1.5) for each scan sequence
- **Scan sequence** represents the number of subsequent scan sequences that have the same scan path configuration
• *Sum for scan path configuration* represents the total number of clock cycles spent at each scan path configuration.

![Diagram](image-url)

(a) Scan path configuration for the first scan sequence

![Diagram](image-url)

(b) Scan path configuration for the second scan sequence

Figure 2.3: Network configurations for session S1 in the session-based test schedule with a fixed scan path, shown in Figure 2.2

In the following the steps to calculate test time for the schedule in Figure 2.2 will be explained using Table 2.2. The test for instruments ($I_1$) and ($I_2$) are performed concurrently in session S1. In session S1, initially the scan path contains only SIBs. To apply test patterns to instruments ($I_1$) and ($I_2$), the corresponding SIBs should be opened. So, in the first scan sequence, which is a setup sequence, five bits are shifted to program the SIBs such that the SIBs corresponding to instruments ($I_1$) and ($I_2$) are opened. After applying CUC, the instruments in the session are included in the scan path as shown in Figure 2.3(a). Subsequently, test stimuli can...
be applied to instruments \((I_1)\) and \((I_2)\) within the scan path as shown in Figure 2.3(b). The length of the scan path is 22 bits and consists of 5 bits for the number of SIBs, 9 bits for the shift register length of instrument \(I_1\) and 8 bits for the shift register length of instrument \((I_2)\), as accounted for in the second row of Table 2.2. This sequence should be repeated for 13 times (12 times for the patterns for instrument \((I_1)\) and for shifting-out the last responses) to complete the test of instrument \((I_1)\). The same process is applied for session S2 and session S3.

Thus, as shown in the last column and bottom row of Table 2.2, test time can be calculated as 810 test time units.

<table>
<thead>
<tr>
<th>Session</th>
<th>SIBs</th>
<th>Scanned bits</th>
<th>CUC</th>
<th>Scan sequences</th>
<th>Sum for scan path configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1(Fig 2.3(a))</td>
<td>5</td>
<td>0 0 0 0 0 5</td>
<td>5</td>
<td>13</td>
<td>((5 + 22) \cdot 13)</td>
</tr>
<tr>
<td>1(Fig 2.3(b))</td>
<td>5</td>
<td>9 8 0 0 0 22</td>
<td>5</td>
<td>13</td>
<td>((5 + 22) \cdot 13)</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>0 0 0 0 0 5</td>
<td>5</td>
<td>13</td>
<td>((5 + 22) \cdot 13)</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>0 0 0 0 0 5</td>
<td>5</td>
<td>13</td>
<td>((5 + 22) \cdot 13)</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>0 0 0 0 1 6</td>
<td>5</td>
<td>13</td>
<td>((5 + 6) \cdot 13)</td>
</tr>
</tbody>
</table>

Sending dummy bits during the sessions is a shortcoming of session-based schedules with a fixed scan path. To improve this, session-based schedules with a reconfigurable scan path are used.

### 2.3 Session-Based Test Schedule with a Reconfigurable Scan Path

In session-based (SB) test schedules with reconfigurable scan paths, in contrast to a fixed scan path, the instruments used in a session are excluded from the scan path once the tests are completed by closing the corresponding SIBs. Figure 2.4 shows an example of a session-based test schedule with
a reconfigurable scan path. Figure 2.5 represents the scan path configurations for session S1 in Figure 2.4. The first two scan path configurations (Figures 2.5(a) and 2.5(b)) are the same as the first two configurations used in Figure 2.3(a). However, as shown in Figure 2.5(c) as soon as test of instrument \( i_2 \) is completed, instrument \( i_2 \) is excluded from the scan path.

To demonstrate the calculation of test time, consider the five instruments in Figure 2.1, which are described in Table 2.1, and a given schedule as shown in Figure 2.4. As Figure 2.4 shows, there are no black boxes (in Figure 2.2) corresponding to dummy bits. This is due to that the SIBs are closed as soon as tests are complete.

Figure 2.4: Session-based test schedule with a reconfigurable scan path

Table 2.3 describes the steps, i.e. setup sequence and scan sequences (defined in Section 1.5), which are applied according to Figure 2.4.

Table 2.3: Test time calculation steps for the schedule in Fig 2.4

<table>
<thead>
<tr>
<th>Session</th>
<th>SIBs</th>
<th>Scanned bits</th>
<th>( \Sigma )</th>
<th>CUC</th>
<th>Scan sequences</th>
<th>Sum for scan path configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1(Fig 2.5(a))</td>
<td>5</td>
<td>0 0 0 0 0</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>5 + 5</td>
</tr>
<tr>
<td>1(Fig 2.5(b))</td>
<td>5</td>
<td>9 8 0 0 0</td>
<td>22</td>
<td>5</td>
<td>7</td>
<td>(5 + 22) · 7</td>
</tr>
<tr>
<td>1(Fig 2.5(c))</td>
<td>5</td>
<td>9 0 0 0 0</td>
<td>14</td>
<td>5</td>
<td>6</td>
<td>(5 + 14) · 6</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>0 0 0 0 0</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>5 + 5</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>0 0 1 1 1</td>
<td>11</td>
<td>3</td>
<td>5</td>
<td>(5 + 17) · 3</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>0 0 1 0 0</td>
<td>6</td>
<td>5</td>
<td>8</td>
<td>(5 + 6) · 8</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>0 0 0 0 0</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>5 + 5</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>0 0 0 1 0</td>
<td>6</td>
<td>5</td>
<td>13</td>
<td>(5 + 6) · 13</td>
</tr>
<tr>
<td>Test time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>652</td>
</tr>
</tbody>
</table>
Figure 2.5: Scan path configurations used for session S1 in the session-based test schedule with a reconfigurable network shown in Figure 2.4
2.4. Session-Less Test Schedule with a Reconfigurable Scan Path

The test time in session-based test schedule with the fixed scan path is 810 test time units as calculated in Table 2.2, and the test time in the session-based schedules with the reconfigurable scan path is 652 as calculated in Table 2.3. The test time is higher with the fixed scan path. Scan path reconfiguration can be used to remove the need of dummy bits.

2.4 Session-Less Test Schedule with a Reconfigurable Scan Path

To show how to calculate the test time for a session-less schedule, we use the example with five instruments in Figure 2.1, which are described in Table 2.1, and a given schedule as shown in Figure 2.6.

Figure 2.6: Session-less test schedule with a reconfigurable scan path

Session-less schedules are defined as a succession of virtual sessions (VS), as will be defined shortly, and a set of rules for how to practically apply the schedule in a P1687 environment.

Figure 2.6 represents four VSs. Virtual sessions act as sessions, however, the new virtual session begin when any of the tests is completed. In other words, the difference between a session and a virtual session is that, in a session all tests are started at the same time, but in a virtual session, tests can start independently of each other as soon as a test is completed. Table 2.4, represents the succession of the VSs for the schedule shown in Figure 2.6. Each VS consists of a fixed set of active instruments and a number of test patterns to apply to those instruments. VS1 consists of
Test Time Calculation

Instruments \((I_1)\) and \((I_3)\) with twelve test patterns. VS2 consists of instruments \((I_5)\) and \((I_2)\) with six test patterns. VS3 consists of instruments \((I_5)\) and \((I_4)\) with two test patterns and VS4 consists of instrument \((I_5)\) with four test patterns. Some instruments are involved in two or more virtual sessions, such as instrument \((I_5)\). The test for such an instrument is not completed in one virtual session but continues in the next virtual session.

Before accessing the instruments, the P1687 network has to be configured. The following rules are used to obtain a P1687-specific schedule which includes the required configuration steps.

1) If any instrument, from the set of instruments in a virtual session, has not been activated in the previous virtual sessions (i.e. its corresponding SIB is still closed), the required configuration scan sequences are added to the schedule.

2) If in the beginning of a virtual session, the remaining number of patterns for an instrument mentioned for that virtual session, is equal to the number of patterns specified for the virtual session, i.e. test is completed by the end of this virtual session, one sequence is added to the schedule to complete the test for that instrument by performing the last shift-out.

Table 2.4: Representation of a session-less schedule using a succession of virtual sessions

<table>
<thead>
<tr>
<th>Virtual Session ID</th>
<th>Instruments</th>
<th>Number of test patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>VS1:</td>
<td>((I_1)) and ((I_3))</td>
<td>12 test patterns</td>
</tr>
<tr>
<td>VS2:</td>
<td>((I_5)) and ((I_2))</td>
<td>6 test patterns</td>
</tr>
<tr>
<td>VS3:</td>
<td>((I_5)) and ((I_4))</td>
<td>2 test patterns</td>
</tr>
<tr>
<td>VS4:</td>
<td>((I_5))</td>
<td>4 test patterns</td>
</tr>
</tbody>
</table>
2.4. Session-Less Test Schedule with a Reconfigurable Scan

In session-based test scheduling, to calculate test time for each session, we consider a setup sequence at the beginning of the session. However, in session-less test scheduling, if one test completes while other tests are running, new tests can be started and a setup sequence for the new test should be considered. As Figure 2.6 represents, in VS3, test of instrument \(I_4\) should be started while the test of instrument \(I_5\) is running. According to Rule 1, setup time for the test of instrument \(I_4\) is considered in VS3. Also, according to Rule 2 that states if a test is completed at the end of a virtual session, the shift-out of the last responses is also included in that virtual session, a shift-out step is also added to VS3 (represented by the black rectangle). Therefore, both Rule 1 and Rule 2 are used in VS3. The white boxes represent setup, and the black boxes represent the shift-out. Table 2.5 describes the scan sequences required to apply the test patterns for each virtual session according to the P1687-specific schedule (Figure 2.7). The columns of this table are similar to the previous tables in this chapter with the exception of one additional column “scan path”. The “scan path” column refers to the sub-figures in Figure 2.8 which show the scan path configurations corresponding to the scan sequences. Moreover, VS represents the virtual sessions, compared with sessions in the previous tables.
Table 2.5: Test time calculation steps for the schedule in Fig 2.7

<table>
<thead>
<tr>
<th>Scan path</th>
<th>VS</th>
<th>SIBs</th>
<th>Scanned bits</th>
<th>Σ</th>
<th>CUC</th>
<th>Scan sequences</th>
<th>Sum for scan path configurations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fig 2.8(a)</td>
<td>VS1</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>5 + 5</td>
</tr>
<tr>
<td>Fig 2.8(b)</td>
<td>VS1</td>
<td>5</td>
<td>10(I1+I3)</td>
<td>15</td>
<td>5</td>
<td>13</td>
<td>(5 + 15) · 13</td>
</tr>
<tr>
<td>Fig 2.8(c)</td>
<td>VS2</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>(5 + 5) · 1</td>
</tr>
<tr>
<td>Fig 2.8(d)</td>
<td>VS2</td>
<td>5</td>
<td>9(I5+I2)</td>
<td>14</td>
<td>5</td>
<td>7</td>
<td>(5 + 14) · 7</td>
</tr>
<tr>
<td>Fig 2.8(e)</td>
<td>VS3</td>
<td>5</td>
<td>12(I5+I4)</td>
<td>17</td>
<td>5</td>
<td>3</td>
<td>(5 + 17) · 3</td>
</tr>
<tr>
<td>Fig 2.8(f)</td>
<td>VS4</td>
<td>5</td>
<td>1(I5)</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>(5 + 6) · 3</td>
</tr>
</tbody>
</table>

Test time: 512 s

Figure 2.8: Scan path configurations

2.5 Summary

In this chapter, test time calculation for a given test schedule is studied. Three approaches are proposed to calculate test time, namely session-based test schedule with a fixed scan path, session-based test schedule with a reconfigurable scan path, and session-less test schedule with a reconfigurable scan path. The difference between the first two approaches is described in the following. In a fixed scan path, the instruments in a session remain on the scan path until the test of all instruments in the session are finished. To decrease test time a reconfigurable scan path is used instead. In a reconfigurable scan path, instruments in a session are excluded from scan path as soon as their tests are completed. Furthermore, the third approach is proposed for session-less test schedules.
Chapter 3

Test Scheduling Approaches and Algorithms

In this chapter, Problem TSOTT is solved. A test scheduling method with three approaches, namely session-based (SB), optimized session-based (OSB), and optimized session-less (OSL) are proposed for Problem TSOTT. For each approach, an algorithm will be presented.

3.1 Mapping Test Scheduling to Strip Packing

The goal of test scheduling methods is to define the order of the tests. It is common that scheduling is guided by a cost function, which can be to decrease test time. During the scheduling constraints must be considered while the cost function is optimized. The test scheduling problem for IEEE P1687 is similar to the strip packing problem [12] which is known to be NP-hard. In strip packing, rectangles are to be fit inside a strip which is limited on three sides, with the aim to minimize the length of the strip. In the test
3.1. Mapping Test Scheduling to Strip Packing

Figure 3.1: Test schedule represented by the individual test time for each test

scheduling problem (see Figure 3.1), the strip is limited on three sides by the time axis, power axis, and the maximum power limit. Moreover, each test is represented by a rectangle. The rectangle’s width shows the time required to complete the test and the rectangle’s height shows the peak power consumption required for the test. It can be seen from Figure 3.1, that minimizing the length of strip is equivalent to minimizing the test time. The approaches in [7, 8] can be seen as heuristics for solving the strip packing problem. If the approaches from [7, 8] are followed in a P1687 environment, it causes unnecessarily long test time as is shown in the following example. Consider five instruments which are described in Table 3.1 with a given flat network.

Table 3.1: Properties for the instruments

<table>
<thead>
<tr>
<th>Instrument</th>
<th>$I_1$</th>
<th>$I_2$</th>
<th>$I_3$</th>
<th>$I_4$</th>
<th>$I_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of test patterns ($tp$)</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Shift register length ($l$)</td>
<td>3</td>
<td>7</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Peak power consumption ($p$)</td>
<td>8</td>
<td>6</td>
<td>8</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

We introduce $\tau_i$ as the test duration of each individual instrument $i$, and by assuming the minimal P1687 network with only instrument $i$ and no SIBs, $\tau_i$ can be calculated as:

$$\tau_i = tp_i \cdot (l_i + CUC) + l_i$$

(3.1)
For example, $\tau_1$ is calculated as below:

$$
\tau_1 = t_p_1 \cdot (l_1 + CUC) + l_1
$$

$$
\tau_1 = 2 \cdot (3 + 5) + 3 = 19
$$

(3.2)

Similar to how $\tau_1$ was calculated for instrument $i_1$, the individual test duration for all other instrument can be calculated to be 19 time units. The schedule shown in Figure 3.1 is a schedule obtained according to the methods in [7, 8].

The schedule in Figure 3.1 is also shown in Figure 3.2, however, here the horizontal axis shows the number of patterns. The test time for this approach [7, 8] is 175 time units, calculated using the “session-based test schedules with a reconfigurable scan path” approach presented in Chapter 2. Figure 3.3 shows a test schedule generated by optimized session-less test scheduling approach, which will be explained later. The test time for this session-less schedule is 155 time units, calculated using the “session-less test schedules with a reconfigurable scan path” approach presented in Chapter 2. Chapter 1, explained how concurrency results in lower test time. The benefit of concurrency in P1687 is on the fact that test patterns applied together share the same SIB programming and CUC overhead. Despite the fact that the tests in Figure 3.1 are applied using the maximum possible concurrency given the power constraint, in practice in a P1687 environment, maximum concurrency is obtained using the schedule in Figure 3.3, and not the schedule in Figure 3.2 which is the equivalent of the schedule in Figure 3.1.

From the above, it can be seen that representing tests as rectangles defined by their peak power consumption and the number of patterns is beneficial for representing test schedules such that the benefit of concurrency can be visualized.
3.2 Test Scheduling Approaches in P1687

In this section, three test scheduling approaches will be proposed, namely session-based (SB), optimized session-based (OSB), and optimized session-less test scheduling (OSL).

In the following, three scheduling approaches will be proposed for solving Problem TSOTT. All of the examples in this chapter are based on the instruments that are detailed in Table 3.1 with a given network similar to Figure 2.1, a maximum power limit of 16 units, and a resource conflict between Test T3 and Test T4.

![Figure 3.2: Test scheduling based on the number of test patterns](image)

![Figure 3.3: Optimized session-less test schedule](image)
3.2.1 Session-Based Test Scheduling

The SB test scheduling approach is designed to solve Problem TSOTT by satisfying power and resource constraints. Each test is represented by $T_i$, where $I$ is a unique ID for each instrument. There is a Testlist containing all instruments. The first test given in the Testlist is moved to the first session. The other tests that can be run concurrently with the first test, considering power and resource constraints, are moved to the first session considering their order in the list. The length of the session (in number of test patterns) is determined by the test with the most number of test patterns in the session. When no more tests can be added to current session, a new session is created and the remaining tests in Testlist. The procedure finishes when the Testlist is empty.

Figure 3.4 shows an example SB test schedule for the Testlist (T1 T2 T3 T4 T5). In SB test scheduling the procedure starts by moving T1, which is the first test in the Testlist, to the first session. After moving T2 to S1 which does not have resource conflict with T1, no more tests can fit within the maximum power limit in the first session. So, S1 consisting of T1 and T2 has three patterns but since the only pattern of T2 is run concurrently with the first pattern in T1, the length of session S1 is two patterns. Subsequently, Testlist becomes (T3 T4 T5). Next, T3 is moved to the new session S2. T4 cannot be scheduled with T3 in S2, because of a resource conflict. Therefore, T5 is moved to S2 and the length of S2 becomes three patterns. Finally, T4 is scheduled in the last session S3. Consequently, Testlist becomes empty and SB test scheduling is completed.
3.2. Test Scheduling Approaches in P1687

3.2.2 Optimized Session-Based Test Scheduling

A test scheduling approach called OSB test scheduling is proposed, which solves Problem TSOTT. OSB test scheduling optimizes test time by firstly prioritizing tests that have a higher number of resource conflicts, and secondly prioritize tests that have a high number of patterns. The idea is to separate tests having resource conflict from those having no conflicts, and sort each group of tests based on the number of test pattern in a descending order.

In the following, to illustrate the OSB test scheduling approach is applied to the example in Table 3.1. Tests that have resource conflicts are separated from those that have no conflict. The first group contains tests with resource conflicts (T3 T4) and the second group contains tests without resource conflict (T1 T2 T5). Tests in each group are sorted based on the number of test patterns. After performing sorting, the first part of Testlist becomes (T3 T4) and the second part becomes (T5 T1 T2). Tests with resource conflicts are prioritized over the other tests, and Testlist becomes (T3 T4 T5 T1 T2). Subsequently the SB test scheduling approach is applied to the tests. T3 is placed in to the first session, T4 cannot be placed in the first session due to resource conflict (T1 T2 T5). Consequently, Testlist becomes (T4 T1 T2). T4 is moved to a new session S2, after moving T1 to S2 no other tests
can be moved to this session. The last session will consist of T2. Figure 3.5 represents the result of the OSB test scheduling on the example. The test time for this schedule is 165 time units.

### 3.2.3 Optimized Session-Less Test Scheduling

The approach in OSL scheduling is similar to OSB scheduling with the following differences:

- Instead of the session concept (used in session-based schedules), session-less test scheduling operates on virtual sessions (VSs).
- After finishing a test, a new VS is started

Testlist (T3 T4 T5 T1 T2) is considered as resulting from the sorting and prioritizing of OSB’s test scheduling. First, T3 is moved to the first VS (VS1). T4 cannot be moved to VS1, due to a resource conflict. T5 is moved to VS1. After moving T5 to VS1, no other tests can be added. As soon as T3, which has less test patterns than T5, has completed the next VS (VS2) containing the remainder of T5 is created (see Figure 3.6). Subsequently, Testlist consists of (T4 T1 T2). VS1 is completed and has the length of two patterns. Next, T4 is moved to VS2. As soon as T5 is completed the next VS (VS3) is created with the remainder of T4. This process is continues until Testlist becomes empty. Figure 3.6 shows the result of OSL test scheduling on the example in Table 3.1. The test time for this schedule is 155 time units.
3.3 Test Scheduling Algorithms

In this section, based on the test scheduling approaches presented in 3.2, three test scheduling algorithms will be presented.

3.3.1 Session-Based and Optimized Session-Based Test Scheduling

Algorithm 1 represents the OSB test scheduling algorithm. Line 1 is the sorting and prioritizing of tests with respect to the number of test patterns and the resource conflicts, respectively. If Line 1 is ignored, Algorithm 1 represents the SB test scheduling algorithm. There are three inputs:

- Testlist: A set of instruments to be tested. Testlist is a set, where each element is an instrument specified by a tuple as \((i, tp, p, rc)\). Here, \(i\) represents the unique ID of the instrument \(I\), \(tp\) represents the number of test patterns for the instrument, \(p\) is the peak power consumption when the instrument is active, and \(rc\) represents if the instrument has a resource conflict with any other instrument \((rc = 1)\) or not \((rc = 0)\).

- Resource Constraint: A list of resource conflicts: The resource conflicts are given as a set of \((I_j, I_k)\); which specifies that instrument \(I_j\) cannot be activated with instrument \(I_k\).

- Power Limit: Maximum power limit. The total power of active instruments must at no time exceed the maximum power limit.

The output of the algorithm is a set of sessions, stored in the Sessions set. Each session is represented with \((tp, I_l, I_m, ...)\); where \(tp\) specified the number of test patterns that are applied in the session and \(I_l, I_m, ...\) specifies the instruments that are active in the session.

In Algorithm 1, initially the Sessions set is emptied (Line 2). In each iteration of the main loop (Lines 3-18), a new session \(s\) is created (Line 4) and the the total power consumption \((p_s)\) and maximum number of patterns \((tp_s)\) for session \(s\) are set to 0 (Line 5-6). For each test in Testlist (Line 7-16) power and resource constraints are checked (Line 8 and Line...
Algorithm 1: Optimized session-based (OSB) scheduling

Input: Testlist as \{(I_i, tp_i, p_i, rc_i)\}, ...
Input: ResourceConstraint as \{(I_m, I_n), (I_m, I_n), \ldots\}
Input: PowerLimit

Output: Sessions as \{(tp_{S1}, \{I_n, I_o, \ldots\}), (tp_{S2}, \{I_m, \ldots\}), \ldots\}

1 Sort Testlist on rc then on tp, both in descending order;
2 Sessions := {};
3 while Size(Testlist) > 0 do
5 s := {};
6 p_s := 0;
7 tps := 0;
8 foreach (I_i, tp_i, p_i, rc_i) ∈ testlist do
9 if I_i has no constraints with any instrument in s then
10 if p_s + p_i ≤ PowerLimit then
11 s := s ∪ \{I_i\};
12 tps := max(tps, tp_i);
13 p_s := p_s + p_i;
14 Remove (I_i, tp_i, p_i, rc_i) from Testlist;
15 end
16 end
17 Sessions := Sessions ∪ \{(tp_s, s)\};
18 end

9) to assign a test to the session (Line 10). The maximum number of test patterns among the tests in Session s is found and recorded in tps (Line 11). The total summed up peak power consumption of tests in Session s is calculated and stored in p_s (Line 12). Subsequently, the tests that are assigned to the session, are removed from Testlist (Line 13). Finally, the created Session s is added to the Sessions set (Line 17).

3.3.2 Optimized Session-Less Test Scheduling

The OSL test scheduling algorithm is similar to the OSB test scheduling algorithm. The sorting of instruments in Line 1 is the same as the sorting of instruments in the OSB test scheduling algorithm. The input of the OSL test scheduling algorithm is the same input for the OSB test scheduling algorithm. However, in the OSL test scheduling algorithm, instead of ses-
3.4 Summary

The analysis on test scheduling in a P1687 environment shows that tests which are applied concurrently share the overhead, resulting in decreased test time. Based on this analysis, three test scheduling approaches have been proposed. For each of the approaches, one algorithm is presented. The SB and OSB algorithms produce session-based test schedules. The OSB algorithm improves the SB algorithm by organizing the order of the tests. The OSL algorithm produces session-less test schedules where VSs are used instead of sessions.
Algorithm 2: Optimized session-less (OSL) scheduling

Input: Testlist as \{\(I_i, t_{p_i}, p_i, r_{c_i}\),...\}
Input: ResourceConstraint as \{(I_m, I_n), (I_m, I_o),\ldots\}
Input: PowerLimit

Output: VirtualSessions as \{(t_{p_1}, \{I_m, I_o,\ldots\}), (t_{p_2}, \{I_m,\ldots\}),\ldots\}

1. Sort Testlist on rc then on tp, both in descending order;
2. VirtualSessions := \{
3. while Size(Testlist) > 0 do
4.     vs := \{
5.     p_{vs} := 0 ;
6.     t_{p_{vs}} := \infty ;
7.     foreach \(I_i, t_{p_i}, p_i, r_{c_i}\) \in testlist do
8.         if \(I_i\) has no constraints with any instrument in s then
9.             if \(p_{vs} + p_i \leq\) PowerLimit then
10.                vs := vs \cup \{I_i\};
11.                p_{vs} := p_{vs} + p_i;
12.                t_{p_{vs}} := \min(t_{p_{vs}}, t_{p_i});
13.            end
14.        end
15.    end
16.    VirtualSessions := VirtualSessions \cup \{(t_{p_{vs}}, vs)\};
17.    foreach \(I_i, t_{p_i}, p_i, r_{c_i}\) \in Testlist where \(I_i\) \in vs do
18.        t_{p_i} := t_{p_i} - t_{p_{vs}};
19.    end
20.    Remove all elements having \(t_p = 0\) from Testlist;
21. end
Chapter 4

Experiments

In this chapter, we report results from experiments evaluating the proposed algorithms, namely Algorithm SB, Algorithm OSB, and Algorithm OSL, in reducing test time.

4.1 Experimental Setup

The instruments and their associated tests are based on the ITC’02 benchmark set [13]. Each System On Chip (SOC) from the ITC’02 benchmark set contains a number of modules. Each module has the following information, the number of inputs, outputs, and bidirectional terminals, and also the number of scan chains and their lengths [14]. For our experiments, the I/O pins for each module and its internal scan chains are each considered as an instrument. The d695, p22810, p34392, and p93791 SOCs from the ITC’02 benchmark are considered for our experiments. The sets of instruments that are extracted from these SOCs, are called A, B, C, and D, respectively. A peak power value is needed for each instrument. The peak power value for each module of the d695, p22810, and p9379 SOCs are taken from [15], and it is assumed that all instruments from the same module consume the same amount of power. As for the instruments in p34392, the peak power for each instrument is assumed as a number proportional
to the length of the shift register for that instrument.

Two resource constraint sets are considered for each design. Each resource constraint set is created randomly, except for the second resource constraint set for D, the reason being to show the performance of our approaches in decreasing test time. Furthermore, in each set of instruments, the second resource constraint set contains all conflicts in the first resource constraint set. The number of conflicts in each set of resource constraints are presented in Table 4.1 in parenthesis for each of the resource constraint sets. Four maximum power limits are considered. The minimum power constraint for each set of instruments should be larger than the peak power consumption of each instrument in that set. Also, infinity is selected as the maximum power constraint.

4.2 Experimental Results

In this section, experimental results on the A, B, C, and D designs will be presented. The columns of Table 4.1 are organized as follows. The first column details the set of the instruments, and the number of instruments in each design. The second column, details the scheduling. The third column presents the test time for the corresponding schedule under different maximum power limits (Column “PC”), and under either no resource constraint or one of the sets of resource constraints described in Section 4.1. For the SB schedule, test time is reported both for a given fixed scan-path (marked SB) and reconfigurable scan-path (marked SB) as described in Chapter 2.

For PC=∞ and no resource constraints, the generated schedules are fully concurrent independent of the algorithm. Consequently, the schedules SB, OSB, and OSL are the same and have the same test time.

In the following a discussion on the obtained result will be presented. For all sets of instruments and for each set of constraints, the SB schedule has the largest test time for PC=∞. The reason is that for a SB schedule, the P1687 network configuration is fixed within a session. Therefore, dummy bits are scanned for the instruments whose tests are finished before the end of the session. Since for PC=∞ the concurrency is maximal for each set of constraints, the number of shifted dummy bits is more, leading
to the largest test time. In the SB schedule, instruments in the session are excluded from the scan path as soon as their testing is finished. Therefore, it can be seen that employing the flexible P1687 scan path helps achieve lower test time.

For designs A and D, in the SB, OSB, and OSL test schedules, an increase in the number of conflicts, increases test time. Similarly, in each resource constraint set, a decrease in the maximum power limit leads to an increase in test time, which is expected.

For the B and C designs, there is a result that diverges from the prevailing trend. As can be seen for design B, resource constraint set 1, and the OSB test schedule, by decreasing the maximum power limit from 650 to 450, the test time becomes lower. Furthermore, by adding more resource conflicts in the set of resource constraints, test time is decreased. The reason can be explained as follows. Our proposed scheduling algorithms are heuristics and do not guarantee an optimal solution. When using heuristics to solve the strip packing problem (see Section 3.1), reduction of the width of the strip (analogous to reduction of maximum power limit) and increasing the number of constraints, might lead to a better fitting of the rectangles (analogous to the individual test durations) inside the strip, hence a reduction in the strip length (analogous to the test time reduction).
### Table 4.1: Experimental Results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>No resource constraint</th>
<th>Resource constraints set 1 (≈ 50 ×)</th>
<th>Resource constraints set 2 (≈ 190 ×)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PC</strong></td>
<td>∞</td>
<td>850</td>
<td>680</td>
</tr>
<tr>
<td><strong>SB</strong></td>
<td>1,97</td>
<td>1,15</td>
<td>1,36</td>
</tr>
<tr>
<td><strong>A</strong></td>
<td>(147)</td>
<td>0,74</td>
<td>0,83</td>
</tr>
<tr>
<td><strong>OSB</strong></td>
<td>0,74</td>
<td>0,81</td>
<td>0,82</td>
</tr>
<tr>
<td><strong>OSL</strong></td>
<td>0,74</td>
<td>0,80</td>
<td>0,81</td>
</tr>
</tbody>
</table>

**Notes:***
- Size of the resource constraint set, i.e., number of resource conflicts.
- The numbers inside parentheses, denote the number of instruments in the corresponding set.
- TAT is calculated assuming that network configuration is not changed within a session, and therefore dummy bits should be shift in
Chapter 5

Conclusion and Future Work

5.1 Conclusion

IEEE P1687 (IJTAG) standardizes access to embedded instruments from the chip boundary, and provides the possibility to design a flexible scan path, compared with the fixed scan path length in JTAG. The flexibility that is enabled in a P1687 environment makes it possible to implement both session-based and session-less test schedules. Test scheduling in a P1687 environment is different from previously addressed test scheduling problems. If the test scheduling approaches which have been applied previously, were applied in a P1687 environment, that would result in unnecessarily long test time. The benefit of concurrency in P1687 is the fact that test patterns that are applied together share the same SIB programming and CUC overhead. Based on the analysis of the impact of P1687 on test scheduling, three scheduling algorithms are proposed. These algorithms are developed to be suitable for a P1687 environment and are called session-based (SB), optimized session-based (OSB) and optimized session-less (OSL). To compare the test time from the test scheduling approaches, three test time calculation approaches for session-based and session-less schedules in a P1687
environment have been proposed as well. Results on implemented algorithms for the approaches on benchmarks adapted from the ITC’02 SOC benchmark set show that with the SB test scheduling algorithm as a baseline, the OSB and OSL test schedules reduce test time. Furthermore, the OSL test scheduling approach performed consistently better than OSB in terms of test time reduction.

5.2 Future Work

In this work, it was assumed that each instrument is connected to the P1687 gateway through a dedicated SIB. However, it is possible to assume other network topologies. It would be interesting to adapt our proposed scheduling approaches to (or to develop new scheduling techniques for) more generic P1687 network topologies.
Bibliography


[8] Valentin Mureșan, Xiaojun Wang, Valentina Mureșan, and Mircea Vlăduțiu. Greedy Tree Growing Heuristics on Block-Test Schedul-


IEEE P1687 (IJTAG) is proposed to add more flexibility—compared with IEEE 1149.1 JTAG—for accessing on-chip embedded test features called instruments. This flexibility makes it possible to include and exclude instruments from the scan path. To reach a minimal test time, all instruments should be accessed concurrently. However, constraints such as power and resource constraints might limit concurrency. There is a need to consider power and resource constraints while developing the test schedule.

This thesis consists of two parts. In the first part, three test time calculation approaches, namely session-based test schedule with a fixed scan path, session-based test schedule with a reconfigurable scan path, and session-less test schedule with a reconfigurable scan path are proposed. In the second part, three test scheduling approaches, namely session-based test scheduling, optimized session-based test scheduling, and optimized session-less test scheduling are studied and three algorithms are presented for each of the test scheduling approaches. Experiments are carried out using the test scheduling approaches and the results show that optimized session-less test scheduling can significantly reduce the test time compared with session-based test scheduling.
Copyright

Svenska

Detta dokument hålls tillgängligt på Internet - eller dess framtida ersättare - under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oforändrat för ickekommersiell forskning och för undervisning. Overföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande.

För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsmann i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådan sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

English

The publishers will keep this document online on the Internet - or its possible replacement - for a period of 25 years from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© Golnaz Asani
Linköping, September 16, 2012