

IEEE P1500, a Standard for System on Chip DFT

Kim Petersén

HDC, Hardware Design Center

723 50 Västerås Sweden

Email: kim.petersen@hdc.se

key words: IP, DFT, SoC, BIST, BISR

ABSTRACT

This document briefly describes the upcoming standard IEEE 1500 [4], titled "Standard Testability method for Embedded Core-based Integrated Circuits".

IEEE P1500 defines a mechanism for the test of digital aspects of core designs within a System-on-Chip (SoC). This mechanism is a scaleable standard architecture for enabling test reuse and integration for embedded cores and associated circuitry.

1. INTRODUCTION

The rapid decrease of state of the art silicon process line width creates the possibility to design core based SoCs. The advantage of using these so called cores is that they speed up the design cycle of large complex system chips and enable import of external design expertise[1]. Example of cores are CPUs, DSPs, memories and communication modules of various kinds.

The change from transistor level development to core/IP level based development of SoCs make IC development work to become more similar to development of PCBs. But is the similarity also valid for manufacturing test?

At the IC-level is a core normally called IP and at the PCB level can a core be compared with a component. Both components and IPs are predeveloped. However: One important difference to keep in mind is that IPs are not manufactured blocks of logic, but components are. Hence: For a PCB it is normally enough to test only the interconnections between the components, but for a SoC it is necessary to test both all transistors and interconnections between the transistors.

This difference implies a huge difference in how manufacturing test must be carried out of IP-based SoCs. It must change from the traditionally more or less flat activity, into a hierarchically approach similar to the one used for PCBs. By doing so for IP-based ICs, not only reuse of design is achieved (through use of IP) but also reuse of testability (through P1500) is made possible.

This paper describes how a standard called IEEE P1500 can make test of core based SoCs become more similar to test of PCBs. It handles two main requirements that might seem contradicting at first sight: easy integration and interoperability (plug-n-play) on one hand, and

flexibility and scalability on the other. What also is shown is examples of how the standard can be used to implement BIST (Built-In Self Test) and BISR (Built-In Self Repair).

The paper is organised as follows. Section 2 arguments why SoCs must have embedded functionality as BIST and BISR. Section 3 motivates why a new standard as P1500 is needed inside SoCs. Section 4 gives a short introduction of the P1500 standard. Section 5 describes the hardware structure of P1500 in more detail. Section 6 shortly describes the instructions to use together with the P1500 hardware. Section 7 describes the different main states the hardware structure can be put into. Section 8 concludes the paper.

2. WHY BUILT-IN CAPABILITY IS DEMANDED FOR SoC's

One important force behind the development of more and more complex SoC's is to enable the creation of low price consumer products that are both portable and complex, and that also can be powered by batteries. The introduction of deep sub-micron SoC's has introduced a demand of change of production test from traditional static external test into embedded test at full speed. Example of motivations why BIST is needed is included as follows:

- The number of transistors behind each pin at the package level is increasing all the time, this situation introduces a growing bottleneck with the usage of external access of test patterns.
- Already today is the internal silicon frequency 10 times higher than the frequency used at the package pins, and the difference continues to increase. These also introduce a bottleneck with the usage of externally accessed test patterns.
- For silicon line widths below 0,35um: Static test do not give enough fault coverage, production test at full speed must be applied. External testers have major problem to supply frequencies of several 100's of MHz.
- The change from static test patterns to "at full speed" test patterns introduce an increase of the number of test patterns required with a factor of six. It is difficult to have this huge amount of test patterns to fit into the memory of testers.

- The extremely high level of integration enabled by the usage of SoC's introduce integration of structures such as uPrs, memories, analog and digital into one piece of silicon. It is very expensive and difficult to build a "super"-tester that effectively can handle this mixture of structures
- New complex network based products as for example a mobile phone system and Internet have very high demand on excellent availability. One important contribution to high availability is the ability for the system to indicate presence of a possible problem and also point out the source of the problem. This feature can be introduced by implementing production test as BIST.

BIST is not enough for future SoC's, also other built in capabilities must be applied. Example of such capability is BISR (Built-In-Self Repair). One driving force for BISR is the increasing usage of embedded memories at SoC's, today about 75% [6] of the total silicon area and expected within the next ten years to grow to 95%. Embedded memories have a much lower yield than other types of structures at silicon. By using repair, the yield in the production can be increased. Today is usually an external approach used, as part of the production test procedure, to repair faulty bit positions. This approach is time consuming and therefore only a limited amount of faults can be repaired. The number of faults possible to repair, can be significantly improved if the repair capability is embedded at the chip, since the speed of the repair procedure is significantly improved [6].

To enable efficient re-use of BIST and BISR methods (and much more), a well-defined interface at IP-level is needed that enables access to the internal functionality. IEEE P1500 is foreseen to create a well-defined interface, that facilitates the interoperability of IP-cores from different sources.

3. WHY 1149.1 NOT IS ENOUGH FOR SoC's

Several articles have proposed how to use IEEE 1149.1 for IP-based designs to create infrastructures to handle issues as BIST, BISR debug [9] a.s.o. All the proposals have shown (more or less) to have one or more limitation such as for example:

- Extra pin(s) introduced at package [7]
- Compliance problem with 1149.1 [7], [5]
- Each embedded TAP controller must be modified to work together at the chip level [8].
- Not possible to access both chip-level data registers and data registers embedded in an IP module, at the same time [9]. It is interesting, during debug, to be able to concurrently access both chip-level data registers and data registers embedded in an IP module.

The demands in the IEEE 1149.1 standard that contributes most to the difficulty to use the standard at the IP-level are

- 1149.1 assumed only one TAP for each IC.
- The Bypass register shall always have the length of exactly one bit.
- The ID register shall always have the length of exactly 32 bits.
- The instruction register shall always have a constant length, independent of the instruction asserted for the moment.

IEEE P1500 do not include these limitations, and therefore is more suited for usage at the SoC-level.

P1500 is introduced as a complement to the standard 1149.1, and not as a replacement for 1149.1.

4. OVERVIEW OF IEEE P1500

P1500 is standardising both a Core Test Language to transfer test knowledge about cores, as well as a scalable wrapper. It facilitates easy integration and interoperability with respect to (but not limited to) testing and for manufacturing defects, especially when various cores of different sources are brought together in one system chip.

P1500 only standardises the test wrapper around the core and its interface to one or more TAMs (Test Access Mechanism). The specification does not describe how to test individual cores, nor SoC test integration and optimisation, this is the responsibility of the core provider. But it is developed to help to solve various aspects of core-based testing as:

- Transfer of information about the core's test from the core provider to the core user.
- Access from test stimuli generator and test response evaluator (either ATE or BIST engine) to the core under test.
- Optimisation with respect to multiple criteria (for example: total test time, maximum power dissipation) of the multiple core tests at system chip level.

The standard is in the balloting phase since December 2003, and is in October 2004 very close to reach version 1.0.

4.1. The Core Test Language (CTL)

CTL is a standardised language for capturing and expressing test-related information for reusable IPs. CTL is built on IEEE 1450.6, which is part of the standardisation activity called STIL (Standard Test Interface Language), see ref. [2].

CTL is meant to standardise the core test knowledge transfer, and is intended to be used together with P1500

in the same way as BSDL (Boundary Scan Description Language) is used together with IEEE 1149.1.

4.2. The Wrapper Structure

The standard defines a number of ports and registers.

Two different ports are defined, the mandatory Wrapper Serial Port (WSP) and the optional Wrapper Parallel Port (WPP). The wrapper terminals in the WSP provide serial access to the IEEE P1500 wrapper.

Three different mandatory registers are defined, the Wrapper Instruction Register (WIR), the Wrapper Bypass Register (WBY) and the Wrapper Boundary Register (WBR). They are all more or less similar to the corresponding registers defined in the standard IEEE 1149.1 [3]. Any number of user specific registers can be added into the IEEE P1500 architecture, indicated by "WDR" (Wrapper Data Registers) in figure 1. Also, any number of user specific registers embedded in the core logic can be connected into the IEEE P1500 architecture, indicated by "CDR" (Core Data Registers) in figure 1.

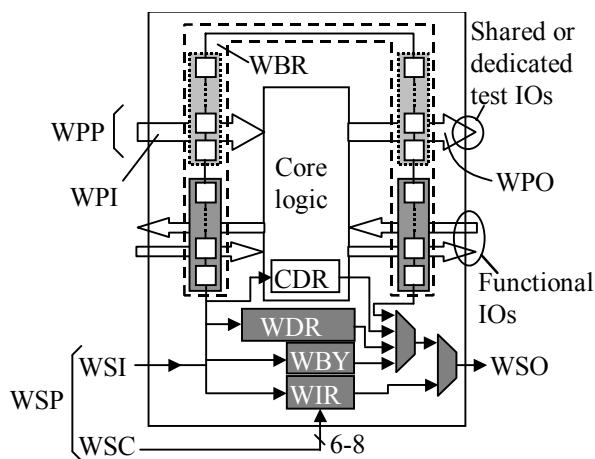


Fig. 1. Example of a IEEE P1500 wrapped core

The WBY provides a bypass path for the WSI-WSO terminals of the WSP.

The WIR enables all wrapper operations. The WIR can optionally be interfaced to the core logic for establishing test mode or functional operation. The WIR can also optionally be extended to generate controls for selection and operation of other data registers, for example CDRs.

The WBR is the register through which test data stimuli are applied and pattern responses are captured. This register allows internal testing of the core logic, as well as testing of external connectivity to other cores and SoC integration circuitry, in response to an instruction loaded into the WIR.

Wrapper register events are enabled

- by the rising edge of WRCK for Capture, Shift and Transfer,
- and falling edge of WRCK for Update.

WSO only change state at the falling edge of WRCK.

See [4] for further details about P1500.

4.3. Overview of How to Connect at The IC-Level

The standard does not define how to connect IEEE P1500 wrapped cores to primary terminals at the IC level.

One possible way of connecting wrapped cores to primary terminals at the IC level is shown in figure 2.

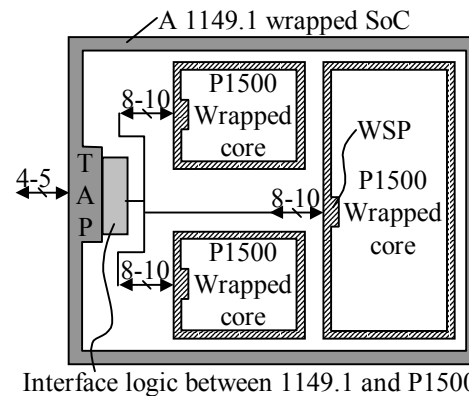


Fig. 2. Example of how connect WSPs at IC-level

The interface logic needed to connect IEEE P1500 wrapped cores, as shown in figure 2, into the IEEE 1149.1 architecture can be as small as four two-input OR-gates up to as complex structures as for example an embedded test processor. The final choice is the responsibility of the core integrator.

5. THE DIFFERENT PORTS AND REGISTERS

5.1. The Wrapper Serial Port (WSP)

The WSP comprise ten terminals, eight mandatory and two optional, as shown in figure 3. In reality, one of the optional terminals, AUXCK can exist in any number of times. But with the purpose to make the text easier to read, it is treated as one single terminal only in this paper.

The WSP terminals facilitate standard "plug and play" operation of the IEEE P1500 architecture. The mandatory part of the WSC part in the WSP is exactly the same outputs achieved from the 16 state TAP controller defined in the IEEE 1149.1 standard [3]. This is by purpose to make it easy to directly connect IEEE P1500 wrapped cores to the IEEE 1149.1 architecture.

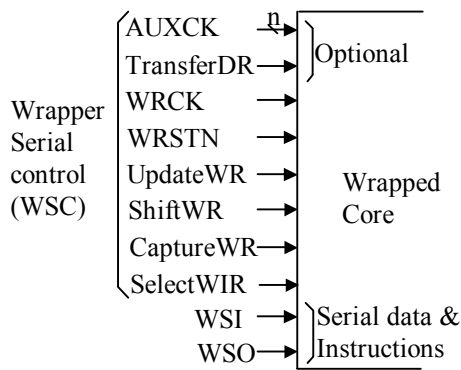


Fig. 3. Wrapper Serial Port

AUXCK, any number of user defined auxiliary clock(s) may be part of the serial port.

TransferDR is required when the WBR include cells with a transfer capability (see section 6 for more information).

WRCK is the IEEE P1500 wrapper clock.

WRSTN is the active-low wrapper reset dedicated for IEEE P1500. When asserted, it unconditionally resets the WIR and puts the wrapper into its normal operation mode, i.e. the wrapper is transparent and the WBY is connected between WSI and WSO. WRSTN may be used to reset other wrapper registers or wrapper circuitry as needed. WRSTN may be either synchronous or asynchronous. It is recommended that WRSTN only is de-asserted at the falling edge of WRCK, this sequence can be used for any implementation of reset. WRSTN is the only terminal at WSP that may be asynchronous to WRCK.

CaptureDR, ShiftDR and UpdateDR: These terminals control and enable wrapper register operations. A terminal is asserted when it is logic 1 and de-asserted when it is logical 0.

SelectWIR determines type of wrapper register selected. As long as SelectWIR is asserted to logic 1, the WIR is unconditionally selected and connected between WSI and WSO, and enabled to carry out operations such as “shift”, “update” or “capture” using the WSP terminals. SelectWIR must be de-asserted to logic 0 in order for any data register (i.e. WBY, WBR, WDR or CDR) to be selected and connected between WSI and WSO.

The WSI and WSI terminals are used to serially “scan-in” and “scan-out” wrapper data and instructions.

5.2. The Wrapper Instruction register (WIR)

The WIR (see figure 4) is an instruction register in which IEEE P1500 wrapper instructions are serially loaded through the standard WSP. The WIR contains a shift stage, instruction decode, and update stage. Only a single WIR is allowed in each IEEE P1500 wrapper.

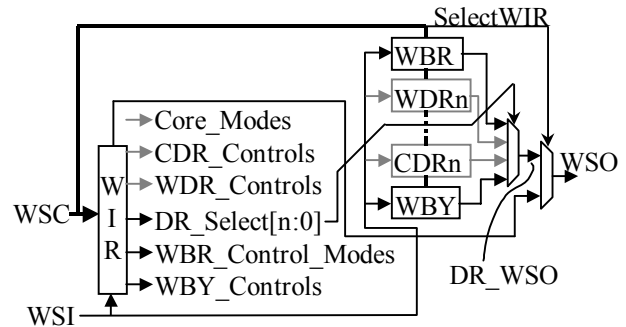


Fig. 4. Example of a WIR interface to wrapper and core

The WIR circuitry shall retain its current state indefinitely while the WRCK is stopped, provided that the WRSTN terminal is logical 1.

The WIR will reset to the WS_BYPASS wrapper instruction whenever the WRSTN terminal transition to a logic 0. The minimum length of the WIR is two storage elements.

5.3. The Wrapper Bypass register (WBY)

The WBY provides a minimum-length serial path between the wrapper's WSI and WSO. This allows more rapid movement of test data to and from other core wrappers, provided the wrappers are connected serially.

Each WBY can be configured as an n-bit bypass shift-register, as shown in figure 5, where $n \geq 1$. A one-bit WBY is the preferred length. But, the possibility to make a WBY longer than one bit makes it possible to connect IEEE P1500 wrapped cores together hierarchically, this without violating the IEEE P1500 compliance.

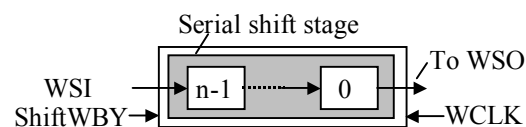


Fig. 5. Wrapper Bypass Register

The WBY is the default data register between WSI and WSO, after WRSTN is asserted, and should be selected by the current wrapper instruction when no other data register is selected.

5.4. The Wrapper Boundary register (WBR)

The WBR can either consists of dedicated (i.e. for test operation only) wrapper boundary cells or shared (i.e. shared between normal functional and test operation) wrapper boundary cells, or a mixture of dedicated and shared cells.

IEEE P1500 specifies a total of 26 different WBR-cells, ranging from lower complexity compared to the boundary scan cells defined by the standard IEEE 1149.1 [3] up to much more complex cells.

The shift path in each WBR-cell can contain any number of storage elements, from one and upward.

WRSTN may optionally be used to reset the WBR cells to a known state.

5.5. The Wrapper Parallel Port (WPP)

WPP is an optional user defined parallel interface. Its terminals are divided into the groups called Wrapper parallel Input (WPI), Wrapper parallel Output (WPO) and Wrapper parallel Control (WPC). These groups are used when the wrapper is configured into parallel mode.

The WPP can be used to enable test of the core logic by using for example Scan or any type of BIST (Built-In Self Test), when the BIST logic is placed outside the wrapped core.

6. WIR INSTRUCTIONS

The instruction loaded into the WIR, together with the IEEE P1500 wrapper signals, determine the mode of operation of the wrapper and possibly the core itself. A particular instruction may result in one or more wrapper or core data registers being serially connected between WSI and WSO or WPI and WPO. Further, the active instruction may select one or more other registers, separate from the register(s) between WSI and WSO or WPI and WPO.

A total of 11 instructions are defined, three of them are unconditionally mandatory:

- WS_BYPASS
- WS_EXTEST.
- Wx_INTEST.

One of the 11 instructions are conditionally mandatory, and must be included when the IEEE P1500 architecture comprise a WBR composed entirely of cells with a dedicated flip-flop on its output.

- WS_PRELOAD

The binary value for each instruction is not defined in the standard specification, and may be selected by the wrapper designer (normally the core provider).

It is possible to add any number of user defined instructions.

The WS_BYPASS instruction is selected when no test operation of that core is required, the instruction connects WBY between WSI and WSO.

The WS_EXTEST instruction allows testing of off-core circuitry and core-to-core interconnections. While the WS_EXTEST instruction is selected, only the WBR shall be connected for serial access between WSI and WSO, i.e. no other test data register may be connected in series with the WBR.

The WX_INTEST instruction allows testing of the core circuitry. While the Wx_INTEST instruction is selected, the WBR shall be in inward facing mode, and the operation of the core should not disturb circuitry external

to the core (the x in Wx is a place holder for an S, P or H to indicate whether the instruction is serial, parallel or hybrid). A hybrid instruction is a wrapper instruction which has mixed use of WSP and WPP terminals. A serial instruction is a wrapper instruction that exclusively uses WSP terminals. A parallel instruction is a wrapper instruction which uses WPP terminals and also configures the WBY between WSI and WSO.

The WS_PRELOAD instruction enables the wrapper to be functionally configured, and is used to allow shifting of the WBR via WSI and WSO, without causing interference to the operation of the core logic. This instruction would typically be utilised before other defined instructions are selected (e.g. WS_EXTEST).

7. WRAPPER STATES

A wrapper can be in one of two main states, disabled and enabled.

As long as WRSTN is equal logic 0, the wrapper is unconditionally in a disable state (i.e. the wrapper is transparent, with the instruction WS_BYPASS in the WIR and with WBY connected between WSI and WSO, and the core logic is operating in normal mode).

As long as WRSTN is equal to logic 1, the wrapper is enabled. This implies that the operation of the wrapper is fully defined by the combination of the contents in the WIR together with the status of the terminals at the WSP.

The terminals at the WSP sets each instruction into one of the possible pre-defined events called Shift, Update, Capture, Transfer and Apply. One or more of these events apply to each instruction implemented. The instruction WS_EXTEST will be used, in the text to follow, to describe the correlation between instruction and event. In figure 6 is the functionality described for four different events when the instruction in the WIR is WS_EXTEST. It is assumed that all inputs are at the left side and all outputs at the right side.

Shift: On each rising edge at WRCK data is advanced one storage position from WSI to WSO, as shown in figure 6.

Update: It is an optional event whereby data stored closest to its shift output is loaded into an off-shift storage element.

Capture: The value present on the functional input is stored into its WBR cell storage element closest to its shift output.

Transfer: It is an optional event whereby shift is carried out locally at each individual WBR cell, and not on the entire WBR as in the event Shift. On each cell data is shifted into, out from or within the cell, but not between cells in the WBR. This event is introduced to enable test at full clock speed.

Apply: It is inferred from the operation from the other four events.

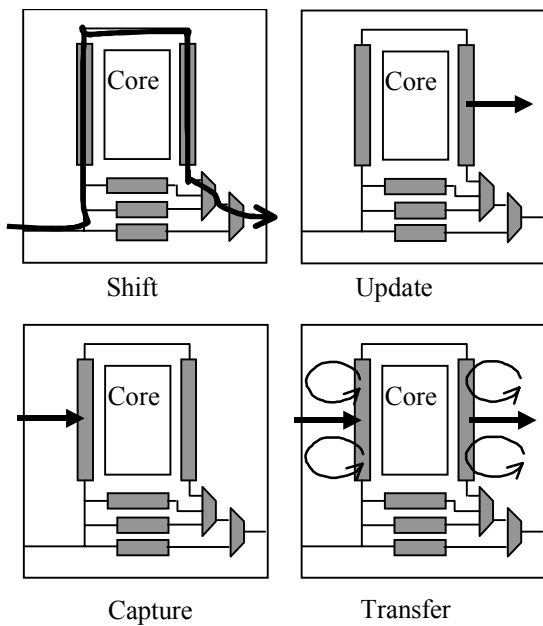


Fig. 6. Function of the four of the five predefined events

8. FUTURE WORK

This work has been created as a co-operation between HDC, Royal Institute of Technology and IRSED and is also economically supported by IRSED (the Industrial Research School in Electronic Design).

Investigate if IEEE P1500 is applicable also for designs of type Network-on-Chip (NoC).

9. REFERENCES

- [1] Erik Jan Marinissen, Rohit Kapur and Yervant Zorian. On Using IEEE P1500 SECT for Test Plug-n-Play. International Test Conference (ITC) 2000, pages 29.1 1-8.
- [2] IEEE Std. P1460, December 2003. Extensions to STIL for Core Test Language (CTL) Support. STIL is an abbreviation for Standard Test Interface Language.
- [3] IEEE Std 1149.1-1990; IEEE Standard Test Access Port and Boundary-Scan Architecture.
- [4] IEEE Std. P1500/D0.8, December 2003. Standard Testability Method for Embedded Core-based Integrated Circuits.
- [5] E-J. Marinissen et al: A Structured and Scalable Mechanism for Test Access to Embedded Reusable Cores. International Test Conference 1998, pages 284-293.
- [6] Y. Zorian. Embedded Memory Test and Repair: Infrastructure IP for SoC Yield. International Test Conference 2002, pages 340-349.
- [7] N. Jarwala: Designing "Dual Personality" IEEE 1149.1 Compliant Multi-Chip Modules. International Test Conference 1994, pages 446-455.

- [8] L. D. Whetsel: An IEEE 1149.1 based Test Access Architecture For ICs With Embedded Cores. International Test Conference 1997, pages 69-78.
- [9] B. Vermulen, T. Weyers and S. Bakker: IEEE 1149.1-compliant Access Architecture for Multiple Core Debug on Digital System Chip. International Test Conference 2002, pages 55-63.