

# IEEE P1500, Boundary Scan for SoCs

Kim Petersén

Department of Microelectronics and Information technology  
 Royal Institute of Technology  
 164 40 Kista Sweden  
 Email: [kim.petersen@hdc.se](mailto:kim.petersen@hdc.se)

key words: IP, DFT , SoC, NoC.

## ABSTRACT

This document briefly describes the upcoming standard IEEE 1500 [1], 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 scalable 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 SoCs with tens of million of gates, and in a near future 100 million of gates and above. Such huge designs requires a change of development strategy to be able to handle Time to Market (TTM) demands. One part of this change is visible as a change from the traditional transistor level development to IP level based development of SoCs. IP-level based design makes IC development become more similar to PCB development. One important difference to keep in mind is that IPs are pre-developed, but not manufactured, blocks of logic.

Increased usage of IP blocks force a change of how DFT (Design for Test) must be carried out at IC-level. Traditionally DFT has been carried out as a more or less a flat activity at IC level. Such strategy could be used in the absence of IP blocks, since more or less all logic was developed from scratch in each project. But, with increased usage of IP blocks the DFT work must change to a hierarchical activity.

The IEEE P1500 specification does not describe how to test individual cores, this is the responsibility of the core provider.

## 2. OVERVIEW OF HOW TO USE

IEEE P1500 shall be used in conjunction with the standard Core Test Language (CTL) IEEE 1450.6, as shown in figure 1. IEEE 1450.6 is part of the standardisation activity called STIL (Standard Test Interface Language), se ref. [3] for further information.

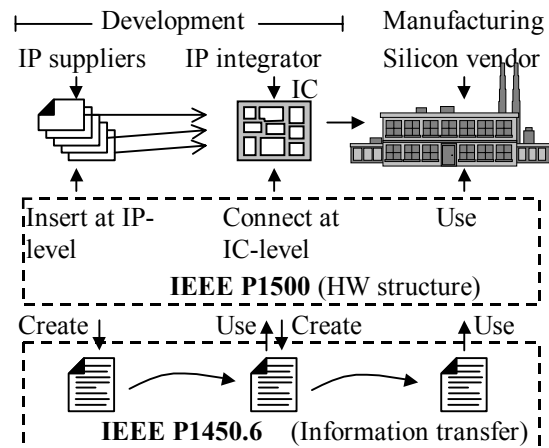


Fig. 1. How to use IEEE P1500

IEEE P1500 defines a serial and a parallel test access mechanism a rich instruction set for testing cores, i.e. reusable megacells, and SoC interconnect and features for core isolation and protection.

The description of the IEEE P1500 in this document is focused on the serial access mechanism.

## 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.

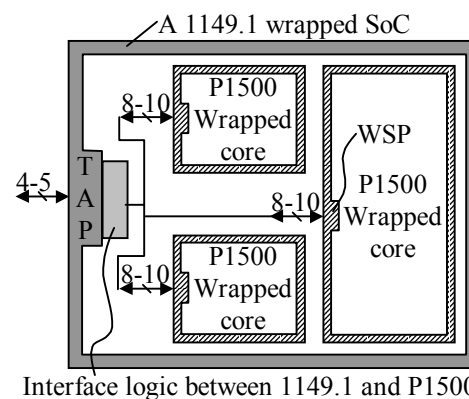


Fig. 2. Example of how connect WSPs at IC-level

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 processors. The final choice is the responsibility of the core integrator.

**4. THE DIFFERENT PORTS AND REGISTERS**

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 [2]. Any number of user specific registers can be added into the IEEE P1500 architecture, indicated by "WDR" (Wrapper Data Registers) in figure 3. 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 3.

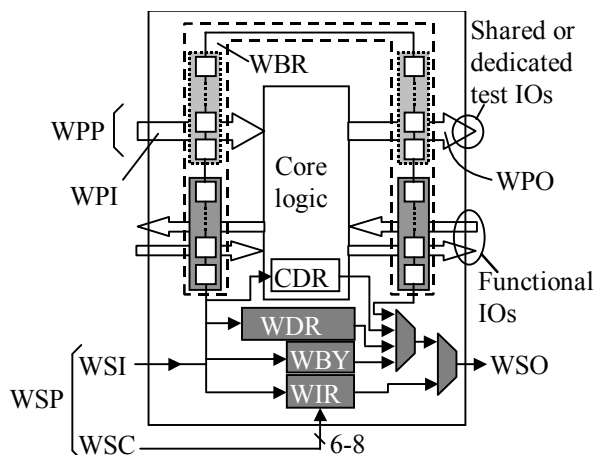


Fig. 3. Example of a IEEE P1500 wrapped core

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.

**4.1. The Wrapper Serial Port (WSP)**

The WSP comprise ten terminals, eight mandatory and two optional, as shown in figure 4. 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 [2]. This is by purpose to make it easy to directly connect IEEE P1500 wrapped cores to the IEEE 1149.1 architecture.

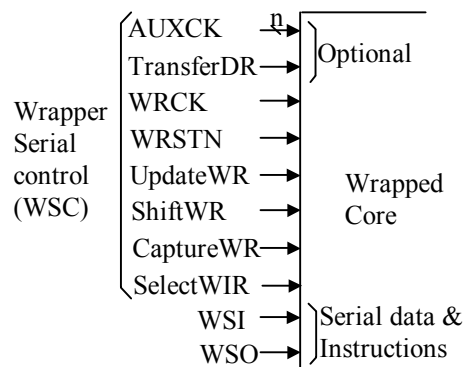


Fig. 4. Wrapper Serial Port

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. SelectWIR is asserted to logic 1, the WIR is unconditionally selected and connected between WSI and WSO, and enabled to shift, update of 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 WSO terminals are used for serial scan-in and scan-out of wrapper serial data and instructions.

**4.2. The Wrapper Instruction register (WIR)**

The WIR (see figure 5) 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.

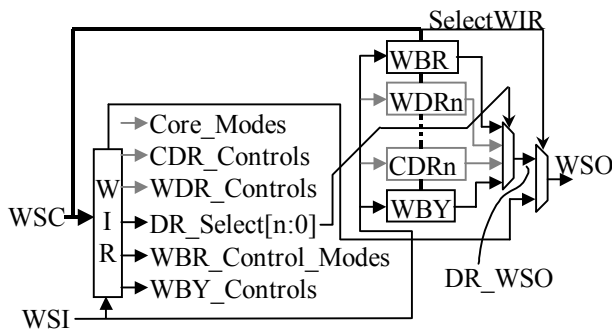


Fig. 5. Example of a WIR interface to wrapper and core

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.

**4.3. 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 6, 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.

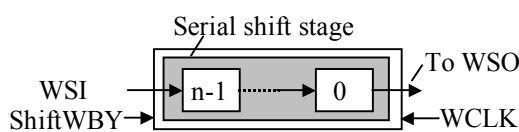


Fig. 6. Wrapper Bypass Register

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.

**4.4. 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 26 different WBR-cells, ranging from lower complexity compared to the boundary scan cells defined by the standard IEEE 1149.1 [2] 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.

**4.5. 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.

**5. 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\_EXTTEST.
- 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 weather 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).

## 6. 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 evens 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 7 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 7.

**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.

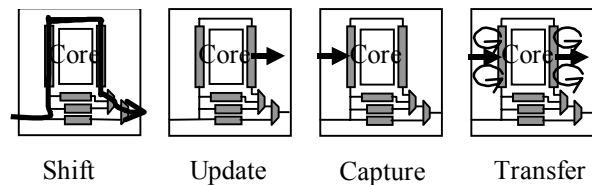


Fig. 7. Function of the four of the five predefined events

## 7. FUTURE WORK

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

## 8. REFERENCES

- [1] IEEE Std. P1500/D0.8, December 2003. Standard Testability Method for Embedded Core-based Integrated Circuits.
- [2] IEEE Std 1149.1-1990; IEEE Standard Test Access Port and Boundary-Scan Architecture.
- [3] IEEE Std. P1460, December 2003. Extensions to STIL for Core Test Language (CTL) Support. STIL is an abbreviation for Standard Test Interface Language.