Overview of the IEEE P1500 Standard

Francisco DaSilva [1], Yervant Zorian [2], Lee Whetsel [3], Karim Arabi [4], Rohit Kapur [5]

[1] Synopsys, Inc. Mountain View, CA. USA Francisco.DaSilva@synopsys.com
[2] Virage Logic. Fremont, CA. USA zorian@viragelogic.com
[3] Texas Instruments. Dallas, TX. USA l-whetsel@ti.com
[4] PMC Sierra. Burnaby, BC. Canada Karim_Arabi@pmc-sierra.com
[5] Synopsys, Inc. Mountain View, CA USA Rohit.Kapur@synopsys.com

Abstract

Design reuse has been a key enabler to efficient System-On-Chip creation, by allowing pre-designed functions to be leveraged, thereby reducing development cycles and time to market. The test of these pre-designed blocks, often referred to as cores, is a primordial factor to successful design reuse methodologies, and must be considered by anticipation with various degrees of challenges depending on the mergeable or non-mergeable nature of the core. This paper presents the state and accomplishments of the IEEE 1500 proposal for the test of non-mergeable cores.

1 Introduction

According to the ITRS (2001), the number of SoC logic transistors per designer-year, for a 10-person design team, will increase by 116% in the four year span ranging from 2001 to 2004. This increase is predicted to be around 392% by the year 2007. Another key statement from the same report is an accompanying exponential increase in design cost. These predictions make overall design cost productivity optimization methodologies, a vital economic requirement. One such methodology has been design reuse, where reuse methodologies allow pre-existing design functions (cores) to be efficiently integrated into new chips, thereby cutting design development cycle time and reducing time to market. This however assumes that certain guideline-steps are taken during the design of these cores, making them efficiently reusable. Design-for-Reuse therefore becomes an indispensable condition to the above-mentioned design productivity optimization. The same is valid for efficient test of non-mergeable cores, because this would require test reuse, which in turn assumes that certain test-specific guideline-steps are taken during the design of these cores. The test of these cores can become a challenge, with various types of difficulties including automation and test Plug-and-Play just to name a few. Many companies have recognized these issues for years and have developed home-grown solutions to cope with them. While these solutions are appropriate in scenarios where the core provider and the core user are the same entity, these solutions don't scale very well to cases where third-party core providers or core users are involved. This creates a need for a standard approach to the resolution of these challenges. To meet this requirement, the IEEE P1500 working group was formed and has developed a core-level solution to facilitate test integration and test reuse. This solution is based on the combination of a hardware architecture and an information model that provides a standard and automated approach to the test reuse challenge. After presenting the IEEE P1500 hardware and information model, the following sections will discuss compliance to P1500 and current limitations of the standard.

2 IEEE P1500 hardware architecture

The IEEE P1500 architecture is characterized by flexibility and scalability. Careful consideration was given to the Plug-and-Play aspects of heterogeneous cores integrated into the same chip. The P1500 hardware architecture comprises an Instruction Register (the Wrapper Instruction Register), and two data registers, the Wrapper Bypass Register (WBY) and the Wrapper Boundary Register (WBR). The use of Core Data Registers (CDRs) is also anticipated by the standard. Access to these registers is provided via a set of wrapper interface ports. Figure 1 displays the mandatory components of the IEEE P1500 architecture.

0-7803-8106-8/03 $17.00

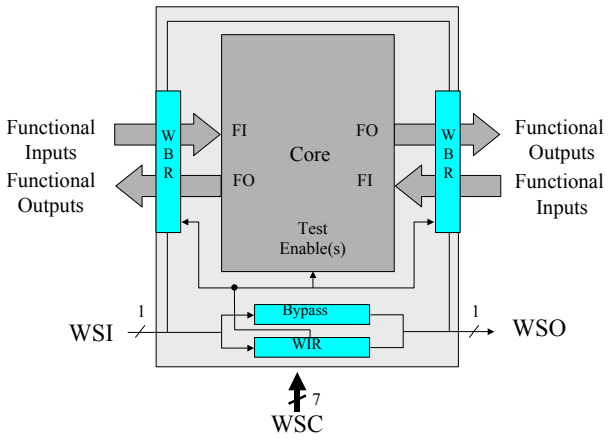


Figure 1: Mandatory components of the IEEE P1500 wrapper

2.1 The wrapper interface ports

IEEE std. P1500 defines a collection of wrapper interface ports that fall into two categories:

- Wrapper Serial Ports. (for serial access to the wrapper)
- Wrapper Parallel Ports (for parallel access to the wrapper)

Figure 2 represents the P1500 wrapper interface terminals.

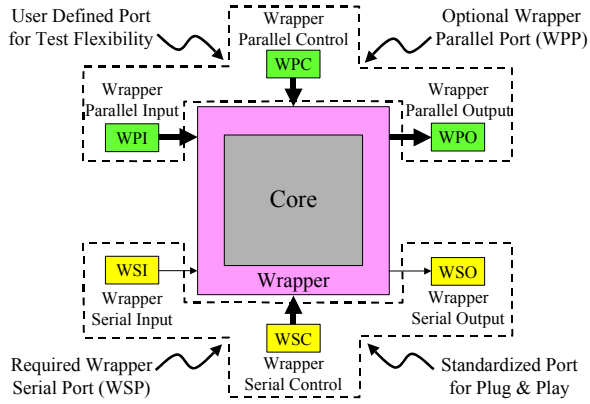


Figure 2: Wrapper interface terminals

2.1.1 Wrapper Serial Ports (WSP)

WSP terminals are mandatory signals made of:

- A Wrapper Serial Input (WSI) terminal
- A Wrapper Serial Output (WSO) terminal
- A Wrapper clock (WRCK) terminal
- A Wrapper Reset (WRSTN) terminal
- An Instruction Register selection (SelectWIR) terminal
- A Wrapper Capture (CaptureWR) terminal
- A Wrapper Shift (ShiftWR) terminal

Optionally, the WSPs can contain functional clocks referred to as Auxiliary Clocks.

The Wrapper Instruction Register exclusively relies on the state of WSP terminals for the configuration of the P1500 wrapper.

2.1.2 Wrapper Parallel Ports (WPP)

WPP terminals are optional signals provided for the anticipated use of high bandwidth test requiring more data terminals than the mandatory serial path connecting the WSI and WSO terminals. These parallel access terminals are user-defined.

2.2 The Wrapper Boundary Register (WBR)

The WBR provides access to core terminals from the wrapper interface ports. It is a collection of P1500 wrapper cells configurable in response to an instruction shifted into the WIR. While The IEEE P1500 architecture mandates use of storage elements for the creation of P1500 wrapper cells, this architecture also recognizes the need for exempting test-only core terminals from that requirement. The P1500 hardware anticipate the use of harnessing logic to provide control over these test-only terminals that are otherwise exempted from the need of a wrapper cell. This type of harnessing provides parallel access to the P1500 wrapper from a Test Access Mechanism (TAM). Whether accessed through the mandatory serial wrapper interface of the optional parallel wrapper interface, the WBR's behavior is categorized in terms of events.

2.2.1 WBR events

P1500 defines a set of events that control the operation of the WBR. These events are Shift, Capture, Update and Transfer. The WBR events are defined below.

2.2.1.1 The Shift event

Shift is an event whereby the data stored in the WBR shift path is advanced one storage position closer to the WBR's Test Output (TO). The data present at the WBR's Test Input (TI) is loaded into the shift path storage element closest to the WBR's TI.

2.2.1.2 The Capture event

Capture is an event whereby the data present on WBR cell functional inputs (CFI), or on WBR cell functional outputs (CFO), at a characteristic instant is stored in a sequential element within a WBR cell. P1500 requires that data captured in a cell is stored in the shift path storage element of the cell closest to the cell's test input (CTI), or in the shift path storage element of the cell closest to the cell's test output (CTO), or in the optional off-shift-path update storage element of the WBR cell.

2.2.1.3 The Update event

Update is an optional event whereby data stored in a WBR cell's shift path storage element closest to CTO is loaded into an off-shift-path storage element. The update event is only applicable to wrapper cells provided with an update storage element. The provision of an update storage element is an optional step.

2.2.1.4 The Transfer event

Transfer is an optional event that moves data to or within the shift-path of a WBR cell dependent on both or either:

- the case where data stored by the capture event is stored in any storage element other than the shift path storage element closest to CTI. In this case, the transfer event will cause the data present in the storage element used for the purpose of provisioning the capture event, to be stored into the shift path storage element closest to CTI;
- the case where more than one storage element exists in the shift path; in this case, the transfer event will cause data stored in the shift path to move one element closer to CTO.

2.2.2 P1500 wrapper cells

A P1500 wrapper cell is expected to contain at least one register and respond to the shift and capture events. The P1500 hardware architecture ensures that all the test data related to a particular core terminal is contained in the wrapper cell associated with that terminal. This provides a scalable architecture containing one or more registers that can be adapted to multiple-stimulus type of testing such as at-speed testing. The P1500 wrapper has a flexible WBR structure that supports any type of wrapper cells.

2.2.2.1 P1500 wrapper cell naming convention

In order to represent the scalable nature of P1500 wrapper cells, the following regular expression is used to name P1500 wrapper cells:

/WC_S[DF]\d+(_C([IOB][IOU])| N)?(_U[DF])?(_O)?(_G)?/

The above regex provides a descriptive, parse-able method for identifying P1500 wrapper cells. Each cell type name begins with "WC", meaning WBR Cell, followed by a sequence of characters that describes the capabilities and structure of the cell. The information will indicate whether a particular storage element is shared or dedicated to wrapper operation, how many shift-path storage elements exist, the existence and type of the optional update cell and the existence of a safe data. To support this, from one to five fields will exist in the name in a specified order. Each field begins with an underscore.

The first field is mandatory, and describes the nature and number of shift path storage elements. The first letter of this field is "_S", to represent the word "Shift". The second letter is either D or F indicating "Dedicated" or (shared) Functional, followed by an integer indicating the number of shift path storage elements. This first field has the following regular expression (regex) format: /_S[DF]\d+/.

The second field indicates the site where data is captured. The first letter of this field is "_C", to represent the word "Capture". The second letter of this field specifies the origin of captured data: I (CFI Input), O (CFO Output) or B (selectable). The last letter of this field indicates the capture site: the first element of the shift path (I), the last element of the shift path (O) or the update storage element (U). In the case where data is captured from CFI and the capture site is the first element of the shift path (i.e. "_CII"), the entire second field may be omitted. In case the wrapper cell being described does not perform the capture function, both second and third letters in this field should be replaced by "N" (None), to indicate that there are no origins for captured data, and that the capture site does not exist. The regex matching this second field is /(_C([IOB][IOU]) | N)?/.

The third field describes the nature of the update element. It is composed of two letters, the first of which is "_U" representing the word "Update". The second letter is either a D or an F indicating dedicated or (shared) Functional. Its regex format is /(_U[DF])?/. The absence of this field indicates the absence of an Update storage element.

The fourth field indicates by its presence or absence, the presence or absence of an observe-only characteristic. It is composed of an under-bar followed by an O. Its regex format is /_O/.

The final field indicates by its presence or absence, the presence or absence of safe data support in non-hazardous mode. It is composed of an under-score followed by a G (Guarded data). Its regex format is / _G/.

Figure 3, Figure 4 and Figure 5 show P1500 wrapper cells named with the above regular expression.

2.2.2.2 P1500 wrapper cell examples

The following two examples are provided to demonstrate the flexibility built into P1500 wrapper cells. Figure 3 depicts the simplest P1500 wrapper cell structure containing a single storage element (represented with the circle). The letters inside the circle describe WBR events that affect the storage element. The storage element shown in the below figure responds to the shift (S) and capture (C) events. In addition, the register is shared with functional logic (F). The wrapper cell pin names are:

- CTI: Cell Test Input
- CTO: Cell Test Output
- CFI: Cell Functional Input
- CFO: Cell Functional Output

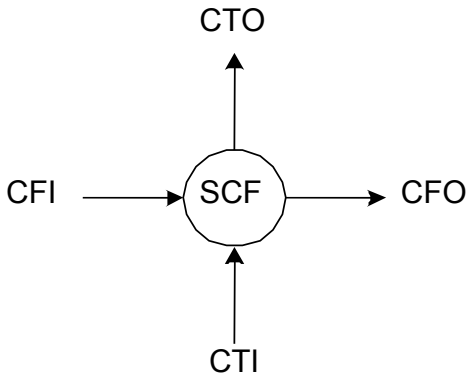


Figure 3: A WC_SF1_CII wrapper cell

The example in Figure 4 is a more complex case featuring the optional Transfer and Update events along with multiple shift-path storage elements.

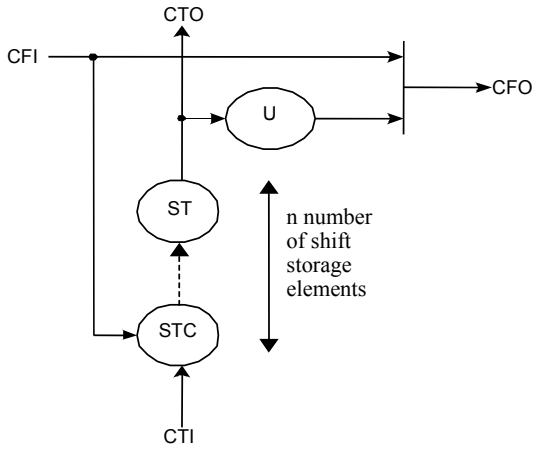


Figure 4: A WC_SDn_CII_UD wrapper cell

Figure 5 shows other flexibility characteristic of a P1500 wrapper cell. The cell in this figure selectively captures data from CFI, or CFO (to provide testability on the CFO terminal). In addition, this cell has provision for safe data output to prevent corruption of external logic data.

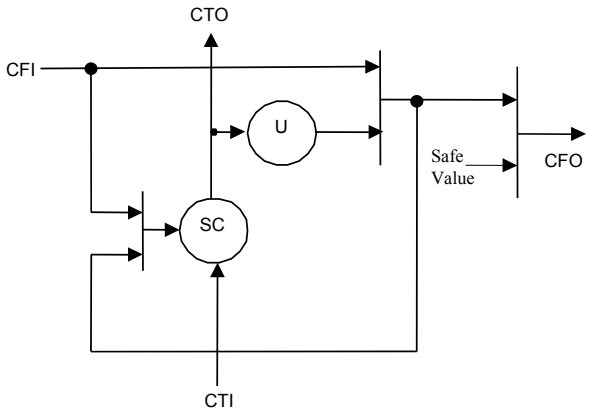


Figure 5: A WC_SD1_CBI_UD_G wrapper cell

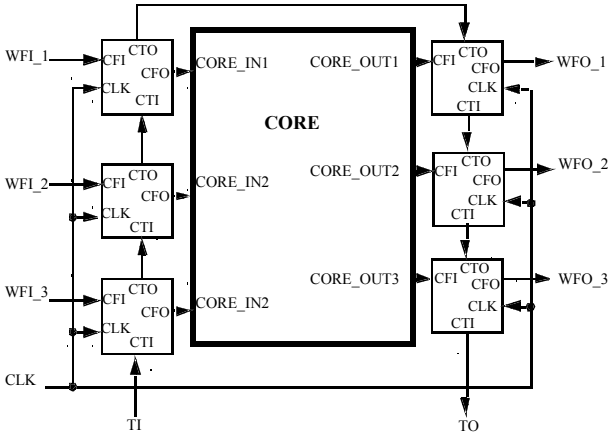
2.2.3 Configuration of the WBR

P1500 provides support for serial and parallel types of test. Each type (serial or parallel) of test corresponds to a serial or parallel configuration of the WBR

2.2.3.1 Serial configuration of the WBR

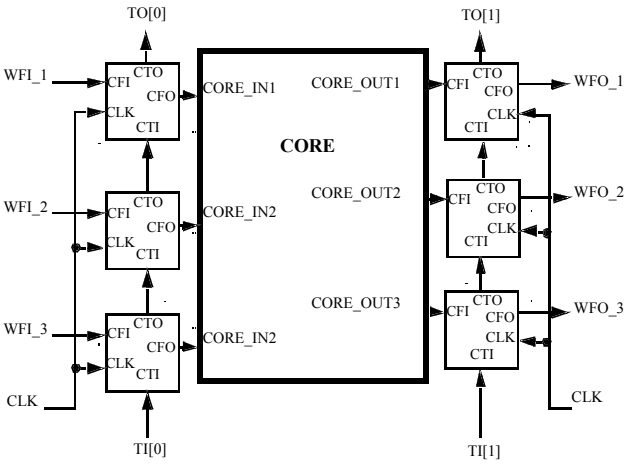
The serial configuration of the WBR is mandated by the P1500 standard. Through this configuration, the WBR is

accessed via a unique set of Test Input (TI) and Test Output (TO) signals.



2.2.3.2 Parallel configuration of the WBR

The parallel configuration of the WBR is optional. In this configuration the Wrapper Boundary Register becomes a set of chain segments, with each segment serially individually accessible via its own Test Input (TI) and Test Output signals.



2.3 The Wrapper Instruction Register

The wrapper instruction register configures the P1500 wrapper into a certain mode of operation determined by an instruction shifted into the WIR via the WSP set of terminals. The WIR circuitry decodes loaded instructions and provides individual controls to the WBR, WBY or CDRs, as depicted in Figure 6. In addition, the state of the WSP set of terminals is used to determine selection of the currently active data register.

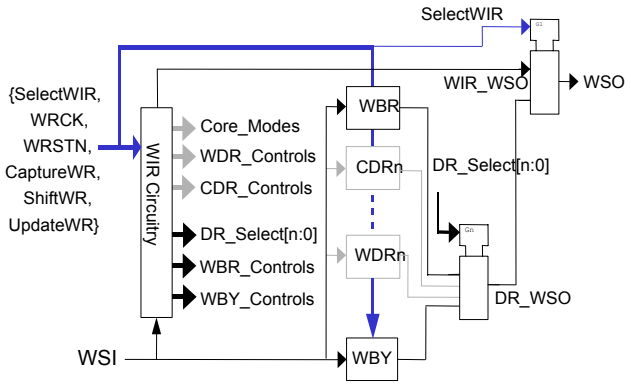


Figure 6: Data Register control from the WIR

2.3.1 IEEE P1500 instructions

IEEE std. P1500 wrapper instructions determine the state the wrapper. Two types (mandatory and non-mandatory) of instructions are defined by the proposed standard, with provision for optional user-defined instructions. The predefined P1500 instructions cover needs for inward-facing test, outward-facing test and also the serial, parallel or hybrid nature of the test. Serial instructions are required to use WSP terminals while parallel instructions use WPP terminals. Hybrid instructions are specific to tests that require the use of a combination of WSP and WPP terminals. A generic naming convention was defined for P1500 instructions as follows:

W<Parallel/Serial/Hybrid>
<Mode><Configuration>
Where:

W – stands for wrapper and is a prefix to all standard 1500 instructions
Parallel/Serial/Hybrid - an “S” denotes a serial mode instruction. A “P” represents a parallel mode instruction. An “H” stands for a hybrid instruction. Note that a standard parallel instruction (e.g. WP_EXTEST) must have the WBY between WSI and WSO – although this is not a hybrid instruction. An instruction by which any register other than WBY is configured between WSI and WSO and which also uses the parallel port, is classified as a hybrid instruction.
Mode – “Mode” is a shortened description of the instruction mode, such as Bypass, Preload, Clamp, Safe, Intest, Extest or User (for user-defined instructions).

Configuration – "Configuration" describes the wrapper configuration of a particular instruction. Possible configurations are SCAN to denote that internal core scan chain(s) participate in the test, RING to denote that only the WBR participate in the test (at the exclusion of any internal core chains), User for a user defined configuration not covered in the above configurations.
For instance, during the serial instructions WS_INTEST_SCAN and WS_INTEST_RING, SCAN denotes that internal scan chains are included in the single scan chain between WSI and WSO. RING indicates that only the wrapper chain is between the WSI and WSO.
The IEEE P1500 standard requires the following instructions:

- WS_Bypass
- WS_Extest
- One Serial, Parallel or Hybrid Intest instruction

The following figures depict the wrapper configuration during mandatory P1500 instructions.

Figure 7 depicts the P1500 wrapper configuration following the load and update of the WS_Bypass instruction into the WIR. In this, configuration the data path flows from WSI to WSO through the bypass register.

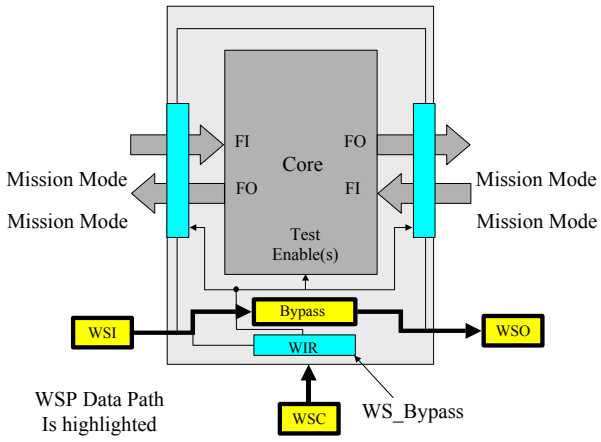


Figure 7: Wrapper configuration during the WS_Bypass instruction

In Figure 8 the P1500 wrapper is configured as a result of the WS_Intest_Scan instruction being active in the WIR. In this configuration, data flows from WSI to WSO via the WBR and the core (internal scan and functional I/Os).

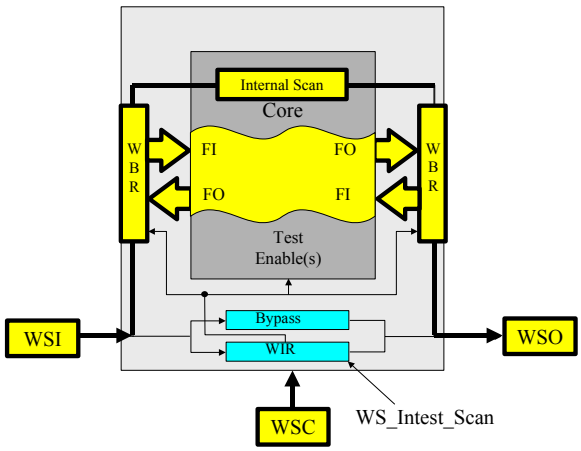


Figure 8: Wrapper configuration during the WS_Intest_Scan instruction

Figure 9 shows the P1500 wrapper configured following the load and update of the WS_Extest instruction. In this configuration, data flows from WSI to WSO through the WBR (which interacts with the external logic being tested).

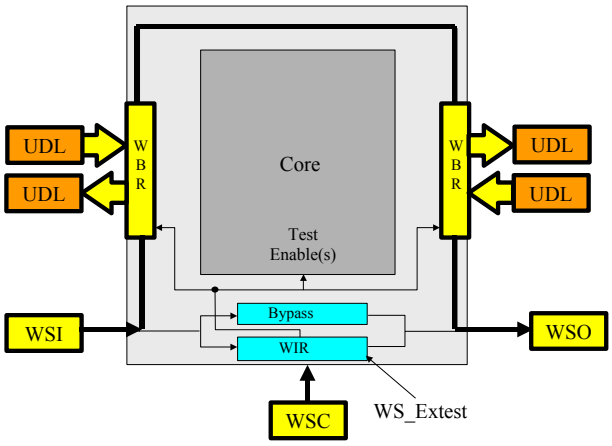


Figure 9: Wrapper configuration during the WS_Extest instruction

2.4 The Wrapper Bypass Register (WBY)

The WBY is a mandatory register selected by the WIR in response the WS_BYPASS instruction. This data register is then connected between the WSI and WSO terminals of the WSP and can be shifted using protocol applied to the signals of the WSC terminals. This register is the default

wrapper data Register and is selected when no other wrapper data register is selected. While the WBY preferably contains a single register to facilitate abbreviation of the scan path length through a wrapper, it may contain additional registers. The WBY does not have an update stage.

2.5 Timing and Plug-and-Play

The operation of a P1500 wrapper relies on well-defined protocols applied via the WSP terminals. These protocols insure plug-and-play interoperability between serially connected wrappers in an SoC.

2.5.1 Wrapper enabled and disabled states

The "wrapper enabled" state allows regular P1500 operation of the wrapper logic. Conversely the "wrapper disabled" state puts the wrapper logic into a transparent state that forces functional operation of the core. The "wrapper disabled" state is activated by loading the WS_Bypass instruction into the WIR or by asserting the WRSTN WSP terminal to its active (low) value. The "wrapper disabled" state can be achieved by gating the WIR outputs (with WRSTN = 0). In this case a synchronous reset operation is still needed to load the WS_BYPASS instruction into the WIR before the Wrapper is enabled by de-asserting the WRSTN signal. For this reason it is allowed to require an active WRCK clock during Wrapper Disabled state, prior to using the Wrapper in the Enabled state.

2.5.2 Wrapper operation timing

Figure 10 displays the timing relationship between WSP terminals, for a proper operation of the P1500 wrapper. At the exception of WRSTN, all WSP terminals are active high. Also noteworthy are the facts that update operations are effective on the falling edge of WRCK, and the WSO terminal is clocked on the falling edge of WRCK.

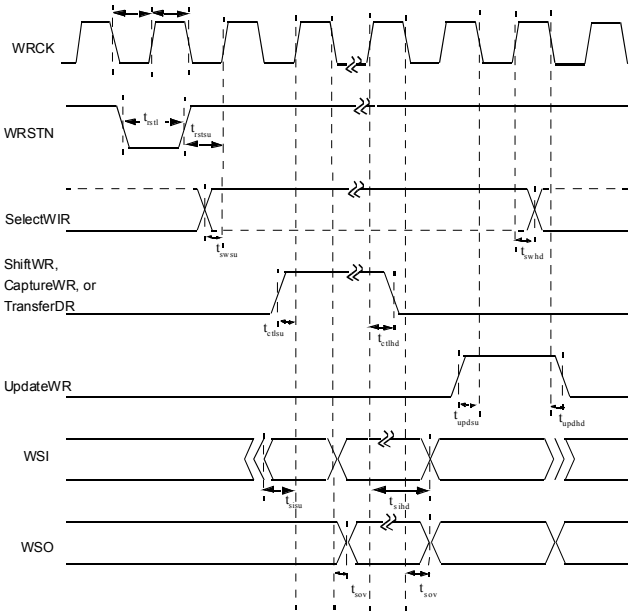


Figure 10: WSP timing diagram

The above-described P1500 architecture defines design-for-test-reuse considerations to be taken, specific to the reuse of non-mergeable cores. This however only addresses half of the problems related to test reuse. SoC-specific issues such-as TAM design and pattern mapping from core-level to SoC level must also be considered. P1500 does not address TAM-related challenges, but provides a solution for ease of pattern mapping. This is achieved via the use the IEEE P1450.6 Core Test Language to create core-level P1500 information model.

3 P1500 Information Model

One of the challenges specific to the core-provider / core-user business model is providing test-relevant information from the core-provider to the core-user in a consistent manner that is not company-specific. This is one of the areas where the P1500 standard innovates by tying the P1500 hardware architecture to a P1500 specific language. This language is the P1450.6 Core Test Language (CTL).

3.1 Introduction to CTL

The Core Test Language was developed to allow the description of design-specific data required for test purposes. CTL is an extension to the Standard Test Information Language (STIL), and as such carries protocol and data (patterns) required to integrate core-level test data into an SoC environment. CTL has a

modular approach that separates test protocol from test data, thereby simplifying the test data integration challenge at the SoC level. With the use of CTL, patterns provided with a core remain unchanged, while the protocol information is adapted to the SoC environment hosting the integrated core. This is achieved without requiring the netlist of the core, but by considering the core as a black-box. The black-box approach is particularly useful for test data integration since certain cores (hard cores) are not (by definition) provided with a full netlist. This approach also serves Intellectual Property protection purposes.

3.1.1 P1500 and CTL

One of the main P1500 achievements is the ability to automate the overall test reuse process from the core creation stage to the core integration stage. This was achieved by defining the P1500 hardware with a very structured approach wherever appropriate in order allow ease of software automation. The following two sub-sections show how P1500-specific information is described in CTL for automation purposes.

3.1.1.1 Core pin wrapping description in CTL

One example of this is the parse-able structure of the P1500 wrapper cell naming convention.

```
1  Environment (Env_name) {
2  CTL (CTL_name){
3   External {
4    (sigref_expr {
5     (connectTo {
6           Wrapper /IEEE1500/
7           (CellID cell_enum | PinID
8            user_defined);
9     })*
10   })+
11  }
12 }
13 }
14 cell_enum =
15  /(WC_S[DF]\d+(_C([IOB][IOU]) |
16  N)?(_U[DF])?(_O)?(_G)?) |
17     (User USER_DEFINED)/
```

The above CTL description shows P1500-specific CTL keywords used to identify P1500 hardware components. The first one on line 6 notes the existence of a P1500 wrapper. The following line (7) identifies the P1500 wrapper cell associated with a particular core pin (PINID). The wrapper cell is described via the cell_enum variable. The cell_enum variable represents the regular expression used to name P1500 wrapper cells in section 2.2.2.1 (P1500 wrapper cell naming convention). This is also shown on lines 14 to 17. The above CTL example therefore shows a mechanism for describing the association of a P1500 wrapper cell to a core terminal. The absence of a wrapper cell on a particular core terminal is also describable in CTL.

3.1.1.2 P1500 protocol specification in CTL

There are two types of protocols involved with the integration of P1500 compliant cores in an SoC. The first type of protocol relates to the operation of P1500-specific functions, while the second type of protocol relates to the mapping of core-level patterns into the SoC environment. Only the former type is addressed here, since the latter chip-specific.

The proper operation of P1500 core depends on the definition of predefined sequences of event referred to here as protocols. Therefore, protocols are associated with P1500 instructions, and are expected to be defined by the core provider for the core-user's benefit. The below section of CTL code is an example describing the protocol associated with the loading of an Intest instruction into the WIR.

```
1  MacroDefs {
2  setup_intest {
3    W default_WFT;
4    V { WRSTN=0; }
5    V { WSP[0..5] = 110100; }
6     Shift { V {
7     i_wir='instruction[0..3]';
8     WRCK=P;}}
9    V { si_wir=X; WRCK=0; ShiftWR=0;
10   UpdateWR=1;}
11   V { WRCK=P;}
12   V { WRCK = 0; SelectWIR=0;
      UpdateWR =0;}
13 }
14 Environment {
15  CTL {
16   PatternInformation {
17       Macro setup_intest {
18       Purpose Instruction
19       }
20    }
21  }
22 }
```

The above CTL code contains two sections, a macro definition section (lines 1 to 13) and an environment section (lines 14 to 22). The purpose of the above setup macro is to load the Intest instruction into the WIR. This is done by first putting the P1500 logic into wrapper-enabled mode (line 4), and then constraining Wrapper Serial Port terminals (line 5) to allow shifting into the WIR. The actual shifting of the WIR occurs in lines 6 through 8, followed by the update of the instruction with lines 9 through 11 (with UpdateWR active). The environment section of the CTL code puts the above setup macro in to the context of the overall test. In this particular example, the "Purpose" (line 18) indicates that the above macro is to be applied to the "Instruction" register.

The above examples are meant for describing how P1500 specific information is provided in CTL to allow test reuse automation at the SoC level. The P1500 standard requires that CTL is provided with P1500 cores as a condition for claiming P1500 compliance.

4 IEEE 1500 Compliance

Cores can be provided and used by third-party companies and because of this it may not always be possible for the core-provider to optimally select certain P1500 components such as the type of wrapper cell to be used to wrap the core, since, this selection could be TAM dependent and therefore chip-dependent. For this reason, the P1500 standard allows cores to exist in both unwrapped and wrapped forms and has defined unwrapped P1500 compliance requirements as well as wrapped P1500 compliance requirements. Both compliance levels require provision of the core information in CTL, by using CTL specific keywords identified by the P1500 standard in order to achieve CTL interoperability. The type of P1500 compliance achieved by a particular core is expected to be provided in the accompanying CTL code.

4.1 Unwrapped Compliance

The P1500-unwrapped compliance level is an intermediate level defined to ensure that appropriate test information is provided to the core user. A P1500-unwrapped compliant core information model enables automatic P1500 wrapper generation. For unwrapped-compliant cores, the core user is expected to complete compliance by meeting wrapped compliance requirements, without having to modify the original (unwrapped) core.

4.2 Wrapped Compliance

The P1500-wrapped compliance level requirements must be met in order to achieve test reuse. The information model for a wrapped-compliant core provides all data required by core integrators to successfully integrate the core and map test patterns to the top-level of the device. Wrapped compliance is achievable by the core provider or the core user. Core users can achieve wrapped compliance from unwrapped-compliant cores or directly from non-compliant cores.

5 Limitations

The P1500 standard does not provide support for bi-directional wrapper cells, since the use of bi-directional terminals is discouraged under design reuse guidelines.

6 Discussion

While P1500 does not provide direct support for bi-directional terminal wrapping, users of the standard can provide wrapper cells on the individual signals composing the bi-directional terminal. In the case where the bi-directional terminal exists at the core level, wrapper cells for the input, enable and output signals of the bi-directional can be provided inside the core being wrapped.

7 Conclusions

The presented IEEE P1500 standard overview provides a solution to test reuse challenges by defining core-level design-for-test-reuse requirements. These requirements are a combination of hardware rules and information model requirements that allow for automated Plug-and-Play test integration at the SoC level.

Acknowledgements

At the time this document was written the following people were P1500 task force contributors:

Alan Hales, Brion Keller, Bulent Dervisoglu, Douglas Kay, Erik Jan Marinissen, Fidel Muradali, Francisco da Silva, Grady Giles, Jason Doege, Jim Monzel, Jon Udell, Karim Arabi, Lee Whetsel, Luis Basto, Mike Collins, Mike Mateja, Mike Ricchetti, Paul Soong, Rohit Kapur, Saman Adham, Teresa McLaurin, Tom Waayers, Vivek Chickermane, Wu-Tung Cheng, Yervant Zorian

References

[1] Semiconductor Industry Association (SIA), International Roadmap for Semiconductors (ITRS), 2001

[2] IEEE P1500 Web Site, http://grouper.ieee.org/groups/1500/

[3] Erik Jan Marinissen, Yervant Zorian, Rohit Kapur, Tony Taylor, and Lee Whetsel, Towards a Standard for Embedded Core Test: An Example, in Proceedings IEEE International Test Conference (ITC), 1999, pages 616-627.

[4] Erik Jan Marinissen, Rohit Kapur and Yervant Zorian, On Using IEEE P1500 SECT for Test Plug-n-Play, in Proceedings of the International Test Conference (ITC), 2000, pages 770-777.

[5] Yervant Zorian and Erik Jan Marinissen, System Chip Test: How will it impact your design, Embedded Tutorial 8.1 at ACM/IEEE Design Automation Conference (DAC) 2000

[6] CTL Web Site, http://grouper.ieee.org/groups/ctl/

[7] IEEE Computer Society, IEEE Standard Test Interface Language (STIL) for Digital Test Vector Data Language Manual IEEE Std. 1450.0-1999. IEEE New York, September 1999.

[8] Rohit Kapur, Maurice, Lousberg, Tony Taylor, Brion Keller, Paul Reuter, Douglas Kay, "CTL the Language for Describing Core-Based Test," in Proceedings of the International Test Conference, 2001, pages 131-139.

[9] Michael Keating, Pierre Bricaud, Reuse Methodology Manual for System-on-a-Chip Designs, Kluwer Academic Publishers, Norwell, Massachusetts.

[10] Erik Jan Marinissen and Maurice Lousberg, Macro Test: A Liberal Test Approach for Embedded Reusable Cores, in Digest of Papers of IEEE International Workshop on Testing Embedded Core Based Systems (TECS), 1997, pages 1.2-1.9.

[11] V. Iyengar, K. Chakrabarty, and E.J. Marinissen, "Co-Optimization of Test Wrapper and Test Access Architecture for Embedded Cores," *Journal of Electronic Testing: Theory and Applications*, vol. 18, no. 2, pp. 213–230, April 2002.

[12] E.J. Marinissen and Y. Zorian, "Challenges in Testing Core-Based System ICs," IEEE Communications Magazine, vol. 37, no. 6, pp. 104–109, June 1999.

[13] M. Sugihara, H. Date, and H. Yasuura, "A Novel Test Methodology for Core-Based System LSIs and a Testing Time Minimization Problem," in Proceedings IEEE International Test Conference (ITC), Washington, DC, Oct. 1998, pp. 465–472.

[14] L. Whetsel and M. Ricchetti, "Tapping into IEEE P1500 Domains," in Digest of Papers of IEEE International Workshop on Testing Embedded Core-Based Systems (TECS), Marina del Rey, CA, May 2001, pp. 3.2-1–7.

[15] L. Whetsel, "An IEEE 1149.1 Based Test Access Architecture For ICs With Embedded Cores", IEEE International Test Conference 1997, pp. 69-78.

[16] L. Whetsel, "Addressable Test Ports - An Approach to Testing Embedded Cores", IEEE International Test Conference 1999, pp. 1055-1064