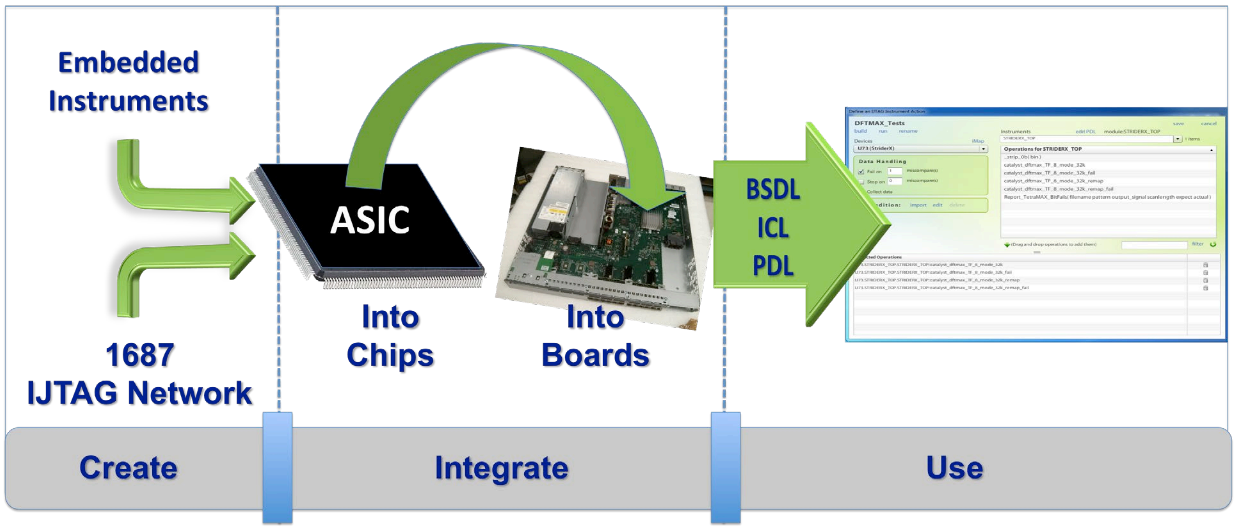
IEEE 1687 IJTAG Tutorial

- Third Edition -



BY

AL CROUCH

LARRY OSBORN

AND JOHN AKIN

ScanWorks®
More visibility. More tools. One Platform.

Al Crouch – Director of Hardware Engineering, AMIDA Technology Solutions

Al investigates the use of embedded instruments for IC test and debug, board test and debug, and software debug. He is a Senior Member of the IEEE and served as the vice chairman of the IEEE 1687 IJTAG working group that developed this standard for embedded instruments. He has contributed significantly to its hardware architecture definition. Al is also a member of the 1838 Working Group on 3D test and debug. Al's previous experience includes design-for-test and debug at various semiconductor companies, including TI, DEC and Motorola, as well as chief scientist at startup companies DAFCA and INOVYS. Al is the author of the popular "DFT for Digital ICs and Embedded Core Systems". He's an inventor and holder of 18 patents.

Larry Osborn – IJTAG and SourcePoint Product Manager at ASSET InterTech

Larry has over 30 years of experience in product management, hardware/software product design and development, product delivery to the marketplace and user support. Over the years, Larry has established a proven track record for identifying user needs and opportunities in the marketplace, providing innovative solutions and exceeding the expectations of users. At ASSET, Larry is responsible for the profit and loss for a product group. Prior to ASSET, he has held positions with Lockheed Martin, Wind River, Hewlett-Packard, Ford Aerospace, and Intel Corporation. He holds a bachelor's degree in Computer Science from the University of Kansas and various technical and marketing training certifications.

John Akin – Senior Configurable Logic Design Engineer at ASSET InterTech

John has over 27 years of experience in FPGA and ASIC design and verification. While working at ASSET, John has worked as both a configurable logic designer and software developer to produce hardware and software tools targeted at chip-, board- and system-level validation. Prior to ASSET he has also worked in telecom, infrared technology, and avionics. He graduated with honors receiving a Bachelor of Science degree in Electrical Engineering from the University of Oklahoma.

Table of Contents

Executive Summary 5
Introduction and History 5
Overview 6
  Industry Drivers 6
  Synergy of Standards 7
  Use Cases for IEEE 1687 IJTAG 8
  The IJTAG Ecosystem 8
  Three Deliverables 10
The Basic IEEE 1687 IJTAG On-Chip Architecture 10
  Controller (AccessLink) 11
  IJTAG Embedded Instruments 12
  Network 14
  Management (SIBs) 16
  Tradeoffs 18
  Instrument Portability and Re-Use 19
Advanced 1687 On-Chip Architectures 21
  Local Reset 22
  Multiple TAPs 22
  Alternative Controllers 26
IJTAG Description Languages 27
  Instrument Connectivity Language (ICL) 27
  Procedure Description Language (PDL) 30
Coordinating IEEE 1687 IJTAG at the Board Level 33
Summary and Conclusions 36

Table of Figures

Figure 1: The IEEE 1687 IJTAG Ecosystem 10
Figure 2: IEEE 1687 IJTAG basic architecture 11
Figure 3: An example embedded instrument with its instrument interface 13
Figure 4: An example embedded instrument showing location of PDL which enables IP portability 13
Figure 5: An example 1687 IJTAG network with separable interface highlighted 15
Figure 6: A scan path can be added before (pre-) or after (post-) the Shift Bit. 17
Figure 7: Distributing TDR control to SIBs 19
Figure 8: A portable IJTAG instrument with TDR and SIB and Local Reset 20
Figure 9: A single eTAP enabled by an IJTAG AccessLink instruction 24
Figure 10: A multi-TAP architecture showing how a ScanMux (SIB) can include eTAPs into the scan path. 25
Figure 11: An eTAP can be parked in either RTI or TLR 25
Figure 12: Chip-centric embedded instruments for chip provider purposes 34
Figure 13: Embedded instruments requested by board test organizations 35

List of Tables

Table 1: The basic IJTAG separable signal interface 14
Table 2: ICL language elements with proper syntax. 28
Table 3: PDL Level-0 commands by group 31
Table 4: PDL Level-1 commands defined in 1687. 32

Executive Summary

This eBook provides a tutorial on the approved IEEE 1687 Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device, which is commonly referred to as the Internal JTAG or IJTAG standard. The IJTAG standard specifies an efficient management methodology for embedded instruments and enables instrument portability and re-use from one integrated circuit (IC) design to another. At the device level, these instruments embedded in ICs, simply called embedded instruments, may perform IC test and debug, on-going or operational monitoring of the device, functional configuration, and other tasks. At the level of the circuit board where ICs have been deployed, embedded instruments are becoming essential for many board and system test and debug functions.

The IEEE 1687 IJTAG standard is made up of three major sections:

- Hardware architecture with a serial data transport mechanism that provides access to embedded instruments;
- An access network description language;
- An instrument procedure description language.

Introduction and History

The IEEE 1687 IJTAG study group first met to discuss a new embedded instrumentation standard at the International Test Conference (ITC) in 2005, following the ratification of the IEEE 1500 Embedded Core Test (ECT) standard. The initial study group was made up of representatives from semiconductor companies, test system suppliers, system manufacturers, JTAG tool providers and other interested parties.

The impetus behind the study group's first face-to-face meeting stemmed from the sudden increase in the number of embedded instruments. The group wanted to ensure the portability of these instruments by relieving them of the overhead associated with the IEEE 1500 ECT standard. In addition, the IJTAG working group intended to develop a standard so that embedded instruments would not need to be documented in the IEEE 1149.1 boundary-scan (JTAG) standard's Boundary Scan Description Language (BSDL), because such a requirement could have slowed the proliferation of embedded instruments. Following this first meeting, the working group submitted a Project Authorization Request (PAR) with the IEEE to begin defining the IJTAG standard.

Overview

The goal of IEEE 1687 Internal JTAG (IJTAG) is to streamline the use of instruments that have been embedded in chips. The intent is to facilitate the deployment of these embedded instruments in a wider array of chip-, board- and system-level validation, test and debug applications. Over the last decade, semiconductor manufacturers have designed embedded instruments into their chips to simplify the characterization, testing and debugging of these devices. Given an optimal standards-based tools environment, these same instruments can perform a much broader spectrum of chip-, board- and system-level validation, test and debug applications.

Industry Drivers

Several conditions in the electronics industry are motivating a trend toward embedded instruments and, thereby, created a need for the IEEE 1687 IJTAG standard. For circuit boards, the progress of advanced technologies such as complex microprocessors and very high speed buses has outstripped the capabilities of the older legacy validation and test equipment. By and large, this legacy equipment is intrusive in that it is external to the board being tested and it relies upon placing a physical probe on some sort of an access point on the board or in a chip on the board. For a number of reasons, the effective availability of these access points is rapidly diminishing, and this is reducing the validation and test coverage that can be achieved with legacy intrusive testers, such as oscilloscopes and logic analyzers for validation, and in-circuit test (ICT) and manufacturing defect analyzers (MDA) for production test. In addition, the higher speeds associated with on-board communications channels has been reducing the effectiveness of intrusive test equipment for many years. As a result of these trends, the electronics industry has turned to test, debug, and characterization methods based on non-intrusive software-driven embedded instruments, which can be accessed through a chip's JTAG

Several Types of Embedded Instruments

-- Built-In Self-Test (BIST) instruments
-- Environmental monitors
-- Process monitors
-- Debug capabilities (chip and board)
-- Functional configuration controllers

port and the circuit board's JTAG connector. Because of the non-intrusive nature of embedded instrumentation, testing, debug, and characterization is not limited by the restrictions associated with the physical probes of legacy test equipment.

At the chip level, several other factors have been driving the industry toward the adoption of embedded instruments. Keeping pace with Moore's Law has meant that chips have become much denser in terms of the number of transistors per square millimeter. In addition, chip frequencies have gone up significantly and devices are much more complex. All of this means that chip characterization times are longer and more sophisticated test equipment is needed. Advanced chip packaging concepts, such as stacking multiple die in three-dimensional packages, also complicate chip-level characterization and debug. The time-to-market for electronic products is rapidly shrinking, and this affects all aspects of a product's development cycle, including the validation and test portions. For example, the average production life of a cell phone IC today is approximately eight months. In the past, new test routines were developed separately for each phase of product development and manufacturing cycle. Now, the industry cannot afford the luxury of the extra time that is needed to re-develop tests for a product as it transitions from development to manufacturing to deployment in the marketplace. Portable tests and other routines that accompany chips and which can be re-applied in every phase of a product's life cycle are becoming a necessity because of the accelerated time-to-market. To achieve a level of portability, test methodologies must capitalize on embedded instrumentation. One way to do so is to take advantage of the capabilities of the IEEE 1687 IJTAG standard.

Synergy of Standards

The IEEE 1687 IJTAG standard is not meant to work in isolation from any other standard or to replace any standard that has already been adopted by the industry. Many of the activities associated with both IC and board test and debug already involve the family of IEEE 1149.1 Boundary-Scan Standards for board test, the IEEE 1500 ECT Standard for core test and the IEEE-ISTO 5001 Nexus Standard for software debug. Now the 1687 embedded instrumentation standard has been added to optimize and manage the use of embedded instruments.

Use Cases for IEEE 1687 IJTAG

The IEEE 1687 IJTAG standard will be applied at the chip and board levels. Chip designers, for example, will find IJTAG useful during design verification when it will be used in conjunction with a simulator or emulator. IJTAG will also be deployed in ATE test and in system test environments where it will become part of chip test, chip debug/diagnostics, chip characterization, and yield-analysis.

At the circuit board level, the IEEE 1687 IJTAG standard will be used to access instruments that are embedded in chips to perform board-level test, debug/diagnostics, characterization and ongoing system monitoring.

Finally, when a system is failing in the field, maintenance personnel can utilize the same tests based on embedded instruments to extract failure data from a system. This data, along with other environmental data, such as voltage and temperature, can be fed back to the organization's failure analysis and sustaining engineering teams where they can analyze the root causes of failures so that corrective actions can be taken. This will allow the duplication of the failure conditions and as a consequence will reduce the amount of 'no trouble found' (NTF) cases.

The IJTAG Ecosystem

Adoption by the electronics industry of the 1687 IJTAG standard will depend on a viable IJTAG ecosystem of developers and vendors who will supply IP and tools, and otherwise support the development, integration and use of IJTAG instruments. Such an ecosystem implies support throughout the semiconductor-to-system cycle of creating IJTAG embedded instruments and on-chip instrument networks, integrating both of these into chips and onto boards, and operating the IJTAG-accessible instruments that comprise the on-chip instrument networks. This ecosystem (Figure 1) is characterized by three distinct functional areas:

- Creation
- Integration
- Use

The ’creation’ phase generally involves IP providers and IC logic designers who will design or use EDA tools, such as Verilog generators. ‘Integration’ will encompass collecting the IJTAG Register-Transfer Level (RTL) data and intellectual property (IP) from IP creators and placing this IP into chips. IC integrators will then perform verification, power analysis, timing closure, layout optimization, and other tasks. Integration will also include bringing together all of the IJTAG Instrument Connectivity Language (ICL) and Procedure Description Language (PDL) files that will complete the bundle of embedded instrumentation IP and ensure its portability into multiple chip designs. Once a chip including IJTAG IP is fabricated, IC test, debug and yield-analysis processes will begin the ‘use’ phase at the chip level. In addition, a board-level use phase will begin soon thereafter. IJTAG chips will be integrated into board designs where some of the capabilities of the IJTAG embedded IP can be employed to accelerate board test and debug processes. Accessing and operating embedded IJTAG instruments at the board level is also a part of the IJTAG ecosystem’s use phase.

In summary, the IJTAG ecosystem must include a wide range of different types of organizations across the electronics industry, including chip and circuit board designers; EDA tools companies which generate embeddable IEEE 1149.1 JTAG, IEEE 1500 ECT, and IEEE 1687 IJTAG constructs and IP; and suppliers of IC and board testers and debuggers. An effective ecosystem will require that all members are able to work together to ensure information and files flow smoothly back and forth throughout the ecosystem. In other words, the tools that comprise the ecosystem must be interoperable. At the time of the IEEE 1687 Standard’s ratification, several cases of IJTAG interoperability including all elements of an ecosystem had been demonstrated publicly.

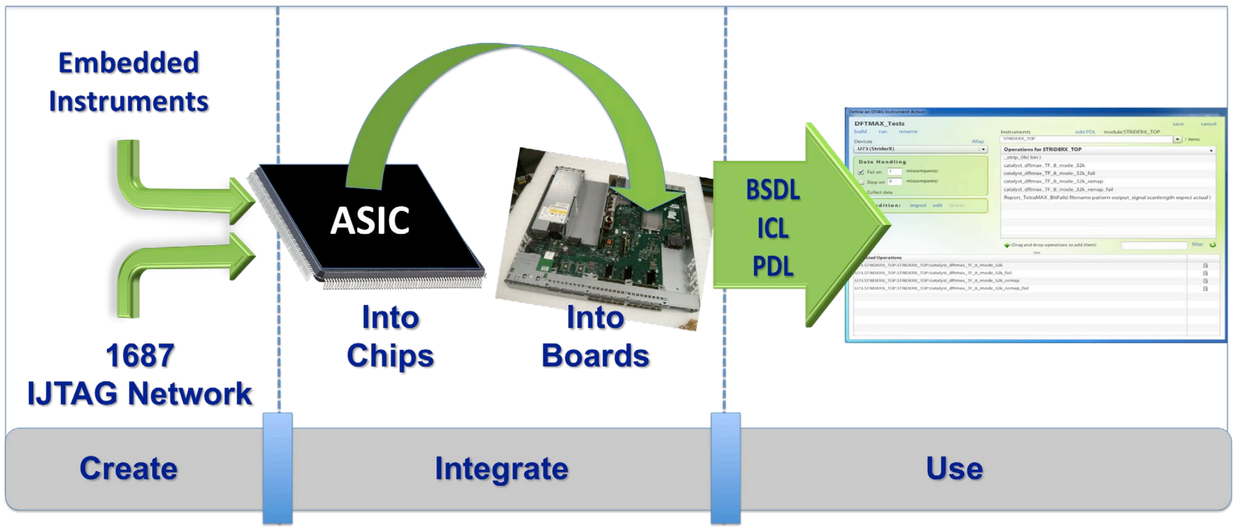


Figure 1: The IEEE 1687 IJTAG Ecosystem

Three Deliverables

The IEEE 1687 IJTAG standard states that an IJTAG-compatible chip, die or core must have three basic deliverables. First, the most significant deliverable is access from the package pins of the chip to the instruments and the instrument network that have been designed into the chip. Second, the IJTAG ICL file must describe the instrument access architecture in hardware so that a vector retargeting tool can operate the embedded instruments from the package pins of the part. And third, an IJTAG PDL file must exist and should represent or describe each embedded instrument's operational vectors at its instrument interface.

The Basic IEEE 1687 IJTAG On-Chip Architecture

Figure 2 below illustrates an IEEE 1687 IJTAG architecture at the chip level. The right side of the drawing shows the IJTAG network interfacing to IJTAG-compliant embedded instruments. The IEEE 1149.1 boundary-scan (JTAG) standard's Test Access Port (TAP) is on the left. The TAP functions as the interface for the embedded 1687 IJTAG architecture to the world outside of the chip.

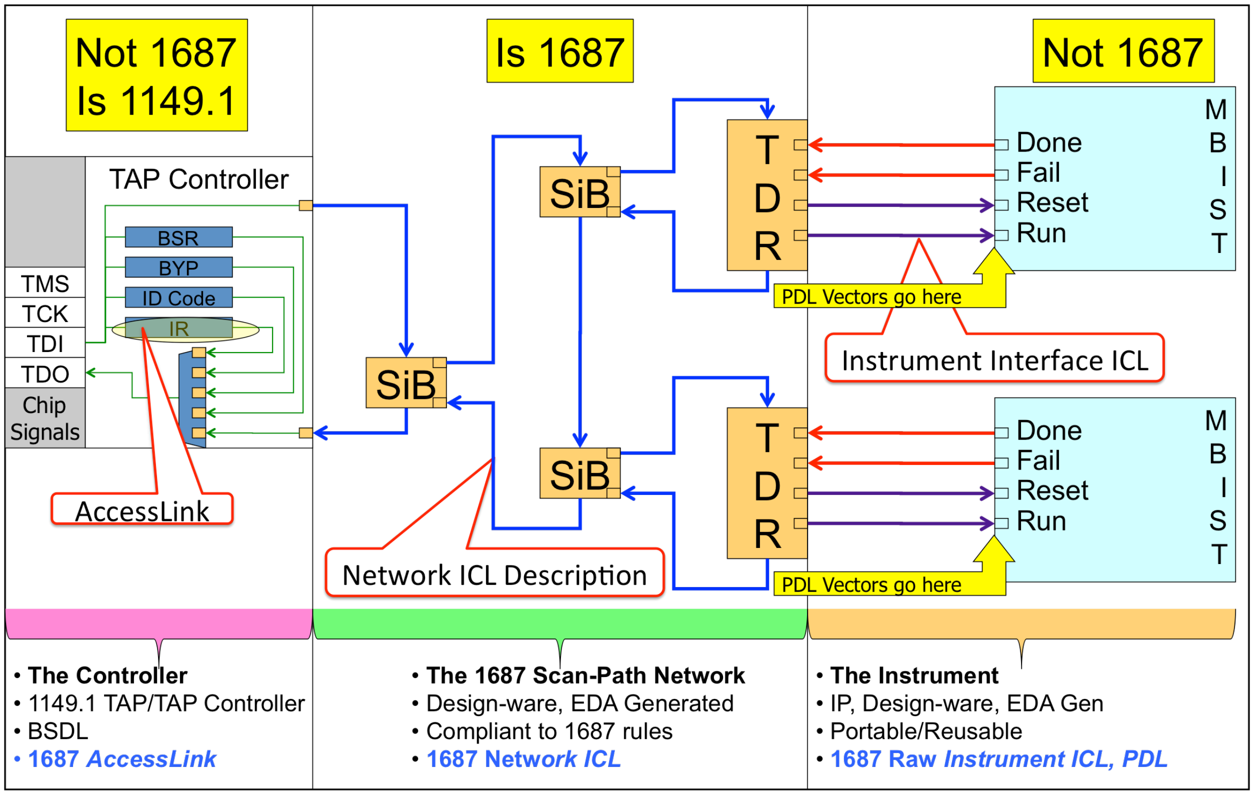


Figure 2: IEEE 1687 IJTAG basic architecture

Essentially, the boundary-scan TAP and its TAP Controller can access the embedded IJTAG instruments by accessing and operating the IJTAG network that connects the controller to the instruments. Several other IJTAG concepts are shown in this illustration, including the Segment Insertion Bit (SIB), ICL and PDL. These are described in more detail below.

Controller (AccessLink)

The left side of Figure 2 above shows a JTAG controller that interfaces the IJTAG network with embedded instruments to the chip's pins. This controller generates the operating protocol for the embedded instrument access network. The IEEE 1687 IJTAG standard does not define this controller, but it does describe how the IJTAG network is connected to any controller through a mechanism known as an AccessLink, which is defined in the standard. Currently, the only controller referenced in the 1687 IJTAG standard is the IEEE 1149.1 boundary-scan (JTAG) standard's Test Access Port (TAP) and TAP Controller. However, the IEEE 1687 IJTAG standard's AccessLink allows for other controllers, such as a direct pin interface or other controllers besides the JTAG. Other controllers in addition to the JTAG controller could be defined in future extensions to the IEEE 1687 IJTAG standard.

Regardless of controller type, such as JTAG, direct chip pins, SPI, I2C or others, the controller must provide the driving control and data interface signals to enable the IEEE 1149.1 JTAG operations of Shift, Capture, Update, and Reset for the JTAG-like Test Data Registers (TDR) which comprise the IJTAG on-chip network.

IJTAG Embedded Instruments

On the right side of Figure 2 is an embedded instrument. The composition of these embedded instruments is not defined in the IEEE 1687 IJTAG standard because the framers of the standard did not want to limit how instruments should be made by incorporating instrument restrictions into the standard itself. The only portion of an instrument that is defined in the IEEE 1687 IJTAG standard is the description of the signal interface connected to the instrument's TDR on the IJTAG instrument access network.

IJTAG embedded instruments are self-contained blocks of functionality (Figure 3). In addition to the interface to the IJTAG network, some instruments may also include the targets of their functionality. For example, a temperature monitoring instrument may also include a temperature sensor. The instrument controls its target, which in this case is the temperature sensor. The other alternative (Figure 3) is for an IJTAG instrument to be composed of its interface to the instrument network (TDR) and the operational protocol for the target, but the target remains a separate standalone entity. An example of such an instrument might be a memory test instrument (memory BIST in Figure 4) that is separate and independent from the memory itself. In addition to the memory test instrument, other instruments or other functionalities within the system could perform operations on the same memory. So, the IJTAG embedded instrument might be one of many units that would access and operate on the target.

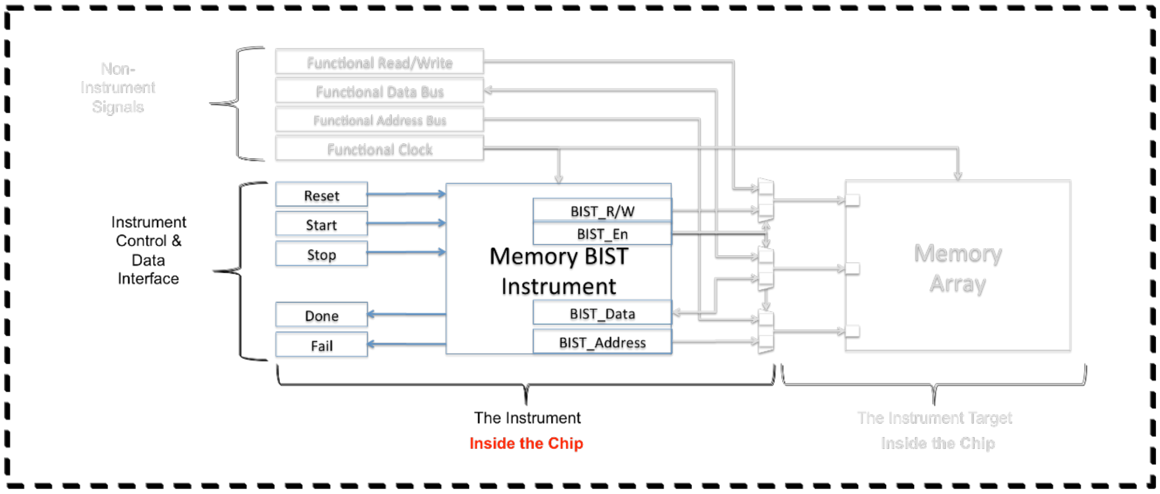


Figure 3: An example embedded instrument with its instrument interface

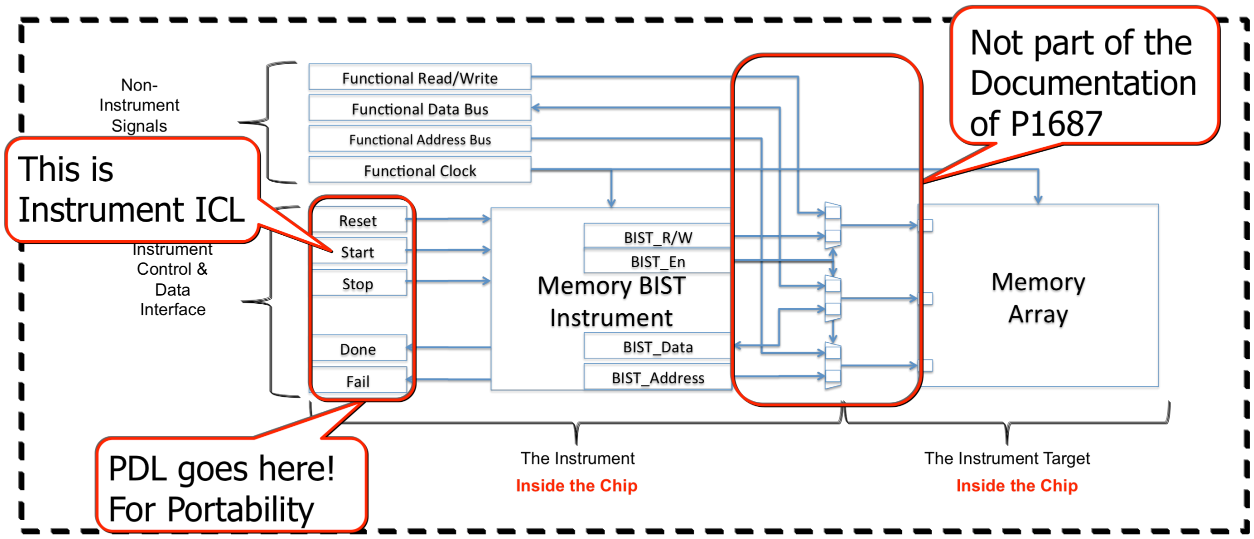


Figure 4: An example embedded instrument showing location of PDL which enables IP portability

To conform to the IEEE 1687 IJTAG standard, some portion of an embedded instrument's signal interface must be described as defined in the IJTAG standard and must be accessible to an IJTAG network through a TDR. The operation of a particular instrument's interface may support one of many possible protocols that are compatible with TDR operation and the IJTAG standard.

Network

An IJTAG network connects embedded instruments to the network controller. The goal of the architecture description in the IJTAG standard is to allow options, tradeoffs, and optimizations to be applied to the IJTAG network so the network may support operation and engineering tradeoffs, and so that network segments may have a measure of plug-and-play portability. The IJTAG network is made of serial scan path bits that can be organized as either of two different types of objects: 1) Test Data Registers (TDR); or 2) Segment Insertion Bit (SIB).

Although the IEEE 1149.1 boundary-scan JTAG standard refers to an entire scan path within a chip as one TDR, within the context of the IEEE 1687 IJTAG standard the scan path connecting embedded instruments is more accurately described as a segment of the chip's overall JTAG scan path which is made up of the individual TDRs that interface to each embedded instrument on the scan path. As a result, this IEEE 1687 on-chip IJTAG network is usually described as being comprised of tens or even hundreds of TDRs. TDRs are viewed as data bits associated with embedded instruments.

An entire network or subsections of an IJTAG network of embedded instruments should always maintain a compliant separable signal interface. If a network is divided for some reason, the same separable interface should provide access to both subdivided segments of the original network. This separable 1687 IJTAG interface is what makes an IJTAG network portable. For example, an embeddable IP core might contain a whole and complete IEEE 1687 IJTAG embedded instrument access network with multiple embedded instruments. When this core is integrated into a chip, the core's IJTAG network and connected instruments can be easily integrated within a larger IJTAG access network that already exists on the chip by connecting the core's IJTAG separable network interface to the control signals provided by the chip's TAP that operates the IJTAG network on the chip. The minimal defined separable 1687 signal interface is:

Table 1: The basic IJTAG separable signal interface

Signals	Definition	Type of Signal
TCK	The serial clock	Clock
ScanIn	The serial test data input	Data
ScanOut	The serial test data output	Data

ShiftEn	The TDR shift enable control signal	Control
CaptureEn	The TDR capture enable control signal	Control
UpdateEn	The TDR update enable control signal	Control
Reset	The TDR reset assert control signal	Control
Select	The TDR activate control signal	Control

This separable network interface that connects directly to the TAP Controller is referred to as IJTAG's AccessLink. As mentioned previously, the 1687 IJTAG standard supports the portable transfer of IEEE 1687 IJTAG networks as IP. Note that this is very similar to the IEEE 1500 ECT interface, except that IJTAG employs a 'Select' while 1500 defines a 'SelectWIR' because 1687 IJTAG does not require a formal Instruction Register (IR), but allows distributed network bits as pseudo-instructions.

Figure 5 below illustrates how the IJTAG separable hardware interface (AccessLink) to the on-chip network of instruments can interface to a standard IEEE 1149.1 boundary-scan TDR. Isolating the IEEE 1687 IJTAG architecture from the requirements of the chip controller and its interface leading off the chip ensures the portability of embedded instrument IP as well as any vector IP that may be associated with each instrument. In fact, an off-chip interface for an IJTAG network other than the IEEE 1149.1 boundary-scan TDR could emerge in the future, and this would not affect the portability of IEEE 1687 IJTAG instruments, vectors or networks.

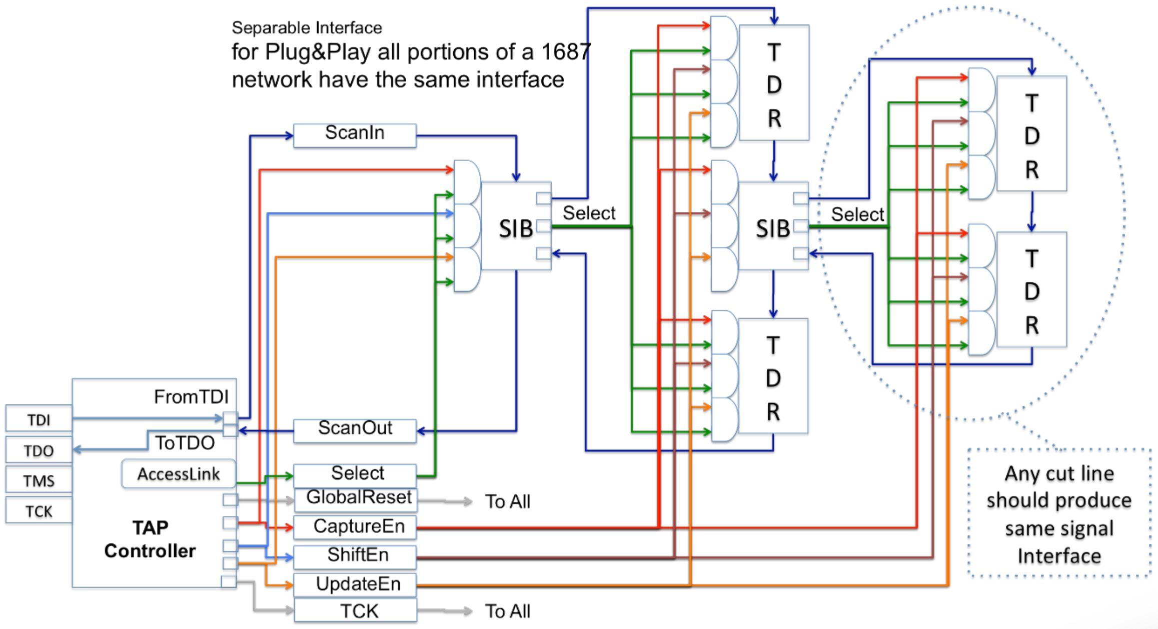


Figure 5: An example 1687 IJTAG network with separable interface highlighted

Figure 5 shows the IJTAG separable interface, which enables plug-and-play functionality by making networks, partial networks and wrapped instruments portable. Note that the operation of the TDR using CaptureEn, ShiftEn and UpdateEn is locally gated by a Select generated by either the IJTAG AccessLink Instruction or an IJTAG SIB.

Management (SIBs)

One of the main purposes for IEEE 1687 IJTAG is to provide effective management processes and procedures for the wealth of embedded instruments that are being integrated into today's semiconductor devices. This can mean enabling a variety of capabilities, including the following:

- Modifying the length of the scan path
- Flexibly scheduling instrument operations
- Operating multiple instruments concurrently, reducing test execution times or instrument operation times for multiple instruments
- Coordinating instrument operations
- Other types of instrument activities yet to be defined

One of the key elements defined in the IEEE 1687 IJTAG standard is the SIB (Segment Insertion Bit). The composition of a SIB is shown in Figure 6 below. A SIB is similar to an IEEE 1149.1 boundary-scan shift/update cell, but the SIB dynamically configures an on-chip 1687 IJTAG scan path to meet the requirements of a particular set of test vectors. 'Selecting' a certain SIB can activate a portion of the chip's IJTAG scan path and consequently activate the instrument(s) on that segment of the scan path. Conversely, 'de-selecting' a SIB will deactivate a portion of the chip's overall scan path and render the instruments on that segment inaccessible. Instruments on a deactivated segment of the scan path cannot be accessed as long as the scan path segment is deactivated, but they can still hold a state that operates an embedded instrument while their network segment is deselected (the rule is that a deselected scan segment must hold state for all operations except reset). This feature allows an instrument to be started, then sequestered away from other instruments while other operations are being conducted on the active scan path (as opposed to parking in the Run-Test-Idle state until the instrument has completed its function). The overall effect is that the length and composition of the scan path is dynamic. It becomes longer when SIBs open or activate segments of the path and shorter when SIBs close off or deactivate one or more segments of the scan path. As a consequence of this feature, access times for operating instruments can be adjusted by activating (opening) or deactivating (closing) segments of the network; and multiple instruments can be dealt with concurrently since deactivating a segment allows continued scan access of other segments while the deactivated segment's instrument(s) continues running. Effectively, this provides engineers operational tradeoffs for the implementation of a network of embedded instruments.

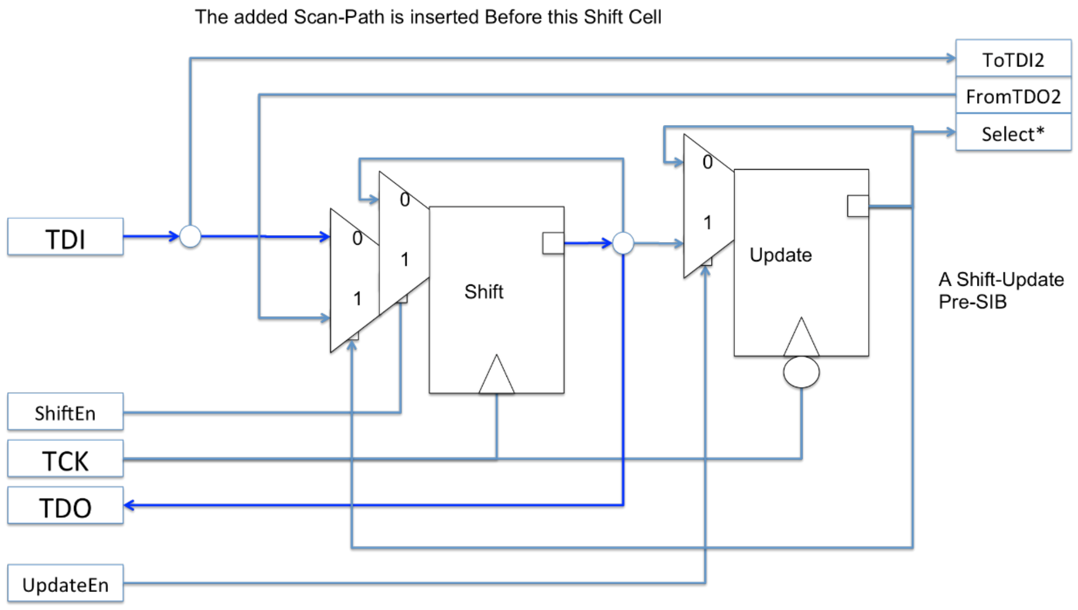


Figure 6: A scan path can be added before (pre-) or after (post-) the Shift Bit.

Note that a pre-SIB (an inserted scan path before the shift-cell) such as the one shown in Figure 6 or a post-SIB (an inserted scan path after the shift-cell) are not the only IJTAG scan path reconfiguration mechanism. In the 1687 IJTAG standard, control points are separate entities from insertion points. Therefore, an IJTAG serial scan architecture can be crafted with one or more control bits in one place or distributed throughout the instrument network while the actual multiplexor that performs the insertion is actually located somewhere else on the network. To facilitate this, 1687 IJTAG describes separately the scan registers that are the source of the multiplexor enable, the multiplexor and any logic function needed to create the actual multiplexor select signal.

Since IEEE 1687 IJTAG does not require a centralized instruction register, as does IEEE 1149.1 JTAG and IEEE 1500 ECT, IJTAG must have a mechanism to change the behavior of the serial access network without a JTAG IR or 1500 WIR. The 1687 IJTAG solution is to enable bits in the active scan path to configure and modify the behaviors of other bits in the active scan path. For example, an update bit associated with a scan shift cell could generate a signal to block the reset signal from operating on an identified group of bits in the active scan path.

Tradeoffs

One of the fundamental drivers of the IEEE 1687 IJTAG standard was to enable more engineering and operational tradeoffs with regards to embedded instruments. The IEEE 1149.1 boundary-scan (JTAG) and IEEE 1500 ECT standards are based on a fairly rigid architecture that includes a centralized instruction register. IJTAG SIBs enable two kinds of tradeoffs involving an IJTAG on-chip network and the instruments that make it up. First, SIBs provide operational tradeoffs by enabling the addition or subtraction of scan path segments to the active scan path. Second, SIBs also provide engineering tradeoffs by generating the Select from a location that is physically close to the instrument. In contrast, centralized instruction architectures like those based on the 1149.1 JTAG or 1500 ECT standards, will require that the instruction decode be physically close to the instruction register, which is usually within the centralized TAP Controller module. In a distributed 1687 IJTAG architecture, distributed SIBs generate the Select signal, which in turn controls the operational status (active or inactive) of the targeted TDR that interfaces to an instrument. Good design practices call for a SIB to be physically located close to its associated embedded instrument's TDR. (See Figure 7 below.) As a result, the decode process in an IJTAG network will be distributed across much of the chip. Specifically, a SIB will provide access to an embedded instrument wherever on the chip an instrument is located. Distributed decode allows a chip design to be optimized according to critical engineering considerations such as silicon area, timing, routing, power consumption and even thermal impact.

Another tradeoff that the IEEE 1687 IJTAG standard enables is the distribution of the ability to modify the scan network, as opposed to generating instrument interface modification instructions in the central instruction register before distributing these instructions throughout the network.

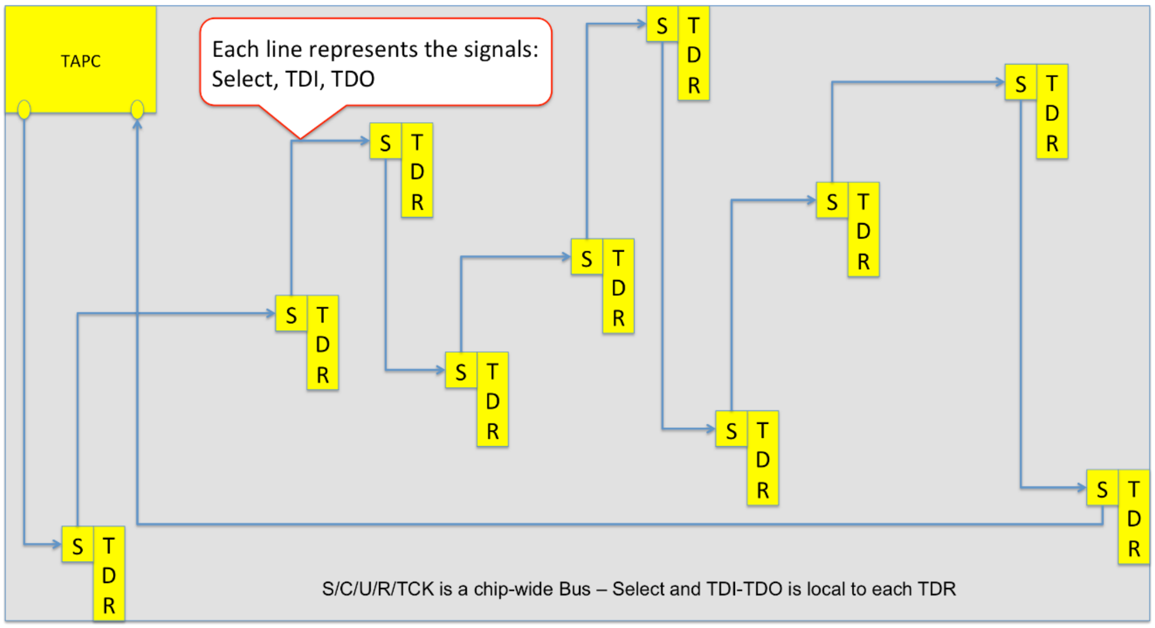

TAPC
Each line represents the signals: Select, TDI, TDO
S
TDR
S/C/U/R/TCK is a chip-wide Bus – Select and TDI-TDO is local to each TDR

Figure 7: Distributing TDR control to SIBs

Instrument Portability and Re-Use

Figure 7 above not only shows physical tradeoffs but illustrates several of the basic concepts of an IEEE 1687 IJTAG on-chip architecture. In the upper left corner is the TAP Controller (TAPC) as defined by the IEEE 1149.1 boundary-scan (JTAG) standard. As far as the 1687 IJTAG network is concerned, the TAP provides two basic functions: 1) the TAP's finite state machine (FSM) produces the control signals that operate the network; and 2) the TAP provides access from the outside world to the IJTAG architecture. The TAP's Test Data In (TDI) and Test Data Out (TDO) pins are connected in a scan path to a set of Test Data Registers (TDRs) which acts as a read/write interface to an IEEE 1687 IJTAG embedded instrument located anywhere within the chip.

Within the context of the 1687 IJTAG standard, there are several degrees of instrument portability. Wrapped instruments are more portable than unwrapped instruments. The term unwrapped or 'naked' instruments means that the instrument is provided only with its signal interface and nothing else. In contrast, a wrapped instrument already supports an IJTAG-compliant TDR by including the TDR's 1687 IJTAG separable interface signals with the instrument's module. A further degree of portability is added when an embedded instrument is delivered with its TDR, its SIB and possibly a Local-Reset or other advanced functions. This can be seen as a best practice. So, as shown in Figure 8, an instrument/TDR/SIB combination as a selectable wrapped instrument can easily connect to an IJTAG network since only the control signals need to be connected and the Select is simply handled by the SIB. For portability, IJTAG's ICL language defines the TDR and the read/write signals that interface the TDR to the embedded instrument. And IJTAG's PDL defines the actions, test vectors and other operations that the instrument executes. All of these are included with a wrapped instrument. This simplifies the integration of the instrument into the overall IEEE 1687 IJTAG network and lets the creator of an embedded instrument write its PDL with extended functions such as instrument selection and TDR reset, further simplifying the integration of a portable embedded instrument.

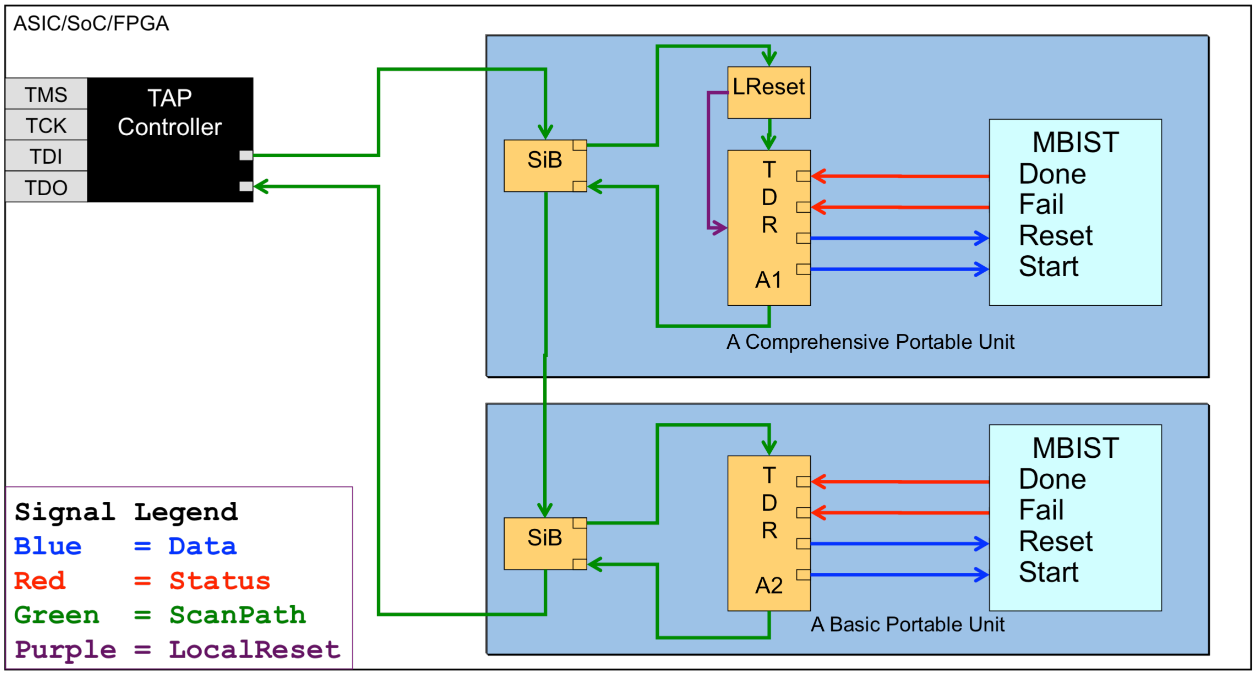


Figure 8: A portable IJTAG instrument with TDR and SIB and Local Reset

An engineering budget involves items such as the silicon area, the amount of routing and routing congestion, and the impact on timing associated with IEEE 1687 IJTAG network components. The IEEE 1149.1 JTAG method of design required very strict adherence to cell types (such as the BC_2 cell type) and required that all instruction decode be associated with the JTAG IR, which is located within the TAP controller. As the number of embedded instruments grows, the number of instructions generally grows linearly, eventually resulting in a routing congestion problem if all instructions for all of the chips embedded instruments originate within the TAP controller. IEEE 1687 IJTAG's SIB-based design allows the decode process to select an embedded instrument located close to the SIB, which should be physically close to the embedded instrument. The IEEE 1687 IJTAG Standard also does not stipulate which types of cells must comprise the network. The standard only defines the required behaviors, such as mandatory shift; optional capture and update; and mandatory reset on update types of cells.

These requirements mean that the IJTAG architecture and behaviors can be driven by various needs that can be quantified as metrics, such as the following:

1. Access time; scan path length; active scan path segment.
2. Number of independent IJTAG AccessLinks; instruction routing congestion; instruction decode logic area/gate count.
3. Scan path control signal routing and routing congestion, scan path timing, scan path data signal routing length

These metrics can drive the types of cells in an IJTAG architecture, the configuration of cells, the locations of SIBs and how much hierarchy is involved with SIB access.

Advanced 1687 On-Chip Architectures

On their surface, simple IEEE 1687 IJTAG networks may appear similar to IEEE 1149.1 boundary-scan (JTAG) or IEEE 1500 ECT architectures that have had a few SIB-based structures added. However, the IJTAG standard was created to solve some of the more prevalent and common architectural problems that arise with core-based design and logic re-use. For example, IEEE 1500 ECT architectures are often delivered with an 1149.1 JTAG FSM, which makes the core an embedded TAP that must be integrated into the same chip design along with the chip's primary boundary-scan JTAG TAP. Unfortunately, the IEEE 1149.1 boundary-scan JTAG standard does not formally support multiple-TAP environments, even though it does describe the selection of different TAPs using added compliance enable chip pins. The IEEE 1687 IJTAG standard does specify how multiple TAPs can be integrated into the same environment and how to document this architecture so that each TAP can be accessed and operated.

In addition, at times an IEEE 1149.1 JTAG scan chain, IEEE 1500 ECT wrapper or an IEEE 1687 IJTAG instrument's TDR must be reset without resetting JTAG for the entire chip or board. For this reason, IEEE 1687 IJTAG has defined a Local Reset that can reset a portion of the IJTAG network, a specific part of the IEEE 1149.1 JTAG scan chain or a certain IEEE 1500 wrapper. In addition, the JTAG serial access scan path may have problems that require debug operations. However, the JTAG operation sequence of Capture, then Shift and then Update is adverse to debug operation. In most cases, diagnostic sequences may be required without conducting the typical JTAG Reset, Capture, or Update operation.

Local Reset

As mentioned, engineers frequently need a local reset to manage board- and chip-level reset operations, but the IEEE 1149.1 JTAG architecture does not support local reset. As a result, resetting one instrument in a JTAG architecture requires resetting all of the chips on the board that are daisy-chained together in the active JTAG architecture. That is, all chips on the active JTAG scan chain must enter JTAG's Test-Logic-Reset (TLR) state, which will reset the entire board. With IEEE 1687 IJTAG's local reset capability, a particular instrument may be reset through one or more bits in its TDR (LR-Bits), which are local to the instrument on the scan path. To avoid breaking the scan path by disabling its ability to shift the scan path, the IJTAG standard requires that the LR-Bit that asserts at UpdateDR must self-clear before reaching the CaptureDR state of the operation sequence (2.5 TCK clock cycles on the 1149.1 JTAG FSM).

Multiple TAPs

One of the realities of modern core-based design is that sometimes an IP core will be delivered with an entire IEEE 1149.1 JTAG architecture. As mentioned previously, the 1149.1 JTAG standard does not formally allow two active TAPs in the same chip. So, when a core with a TAP is integrated into a chip design that already has a primary TAP for the entire chip, the integrator has two choices: modify the core by removing its TAP and TAP controller, or leave the core as is and integrate it into the chip design along with its TAP and TAP controller. If the core is a hard core (layout macro), the engineer has no choice. The TAP and TAP controller must remain in the hard core. If the core is a soft core (HDL or RTL code), the core can be modified, but this may render the vectors delivered with the core inoperable. Many integrators decide to leave the core intact. This raises the issue of how the chip design will support and operate multiple TAPs and TAP controllers within a single chip. The main concerns are how to maintain 1149.1 JTAG compliance at the chip-level while supporting a multiple TAP environment.

A common example of this situation is an IEEE 1500 wrapped core complete with an IEEE 1149.1 boundary-scan (JTAG) TAP and TAP Controller, which includes JTAG's FSM. Generally, this is done for self-contained timing purposes. This makes the TAP in the core an embedded TAP when it is integrated into a chip design with its own primary TAP. If several of these cores each with its own TAP are integrated into the same chip design, then a multiple TAP environment results. For example, four cores each with a TAP could result in five TAPs on one chip when the cores are integrated into a design that already has a primary TAP. Fortunately, the IEEE 1687 IJTAG standard specifies how multiple TAPs can be integrated into the same environment and how to document this architecture so that each TAP can be accessed and operated.

It must be noted, however, that IEEE 1687 IJTAG supports the basic scan path rule that when a scan path or scan path segment is not selected, the operations of the TDRs on the non-active or deselected scan path segments are frozen. That is, the TDRs do not shift, capture, or update; they simply hold their states. This rule also applies to IJTAG network architectures, including TAPs that may be selected and deselected. A deselected TAP must remain in Run-Test-Idle [RTI], whereas a reset TAP must remain in Test-Logic-Reset (TLR).

IJTAG can enable a multiple TAP environment through a top-level direct selection by an AccessLink instruction. Multiple TAPS are included in an active scan path through several possible types of architectural constructs, including the following:

1. A SIB construct (also known as a ScanMux);
2. More than one AccessLink instruction.
3. A configuration TDR.

Figure 9 shows a traditional IEEE 1149.1 JTAG connection from the chip design's primary TAP to an embedded TAP (eTAP). Such connections are enabled by the IJTAG standard. The primary TAP includes AccessLink instructions to select other eTAPs. In this type of configuration an

IEEE 1687 IJTAG AccessLink instruction can select one or more eTAPs. This configuration allows the primary TAP (the TAP that includes the chip-level BYPASS, IDCODE, EXTEST, and other chip-based instructions) to select an AccessLink instruction from among several possible AccessLink instructions. This would add one or more of the eTAPs into the active TDI-TDO scan path. When the primary TAP selects BYPASS, the architecture is compliant with the 1149.1 JTAG standard because one bit represents the bypass register in the chip's TDI/TDO path and the eTAP is deselected, causing it to remain inactive in the Run-Test-Idle (RTI) state. When the primary TAP selects an AccessLink, it may connect directly to an eTAP or to a secondary selection mechanism such as a configuration register or an IJTAG Segment Insertion Bit (SIB). The primary TAP IR can select an eTAP through an AccessLink or an eTAP SIB. When the AccessLink or SIB are asserted with an UpdateIR from the primary TAP, then the eTAP will wake up from either being in a TLR or in RTI state. The eTAP's parking state depends on the primary TAP controller; turning off TMS parks the eTAP in RTI; putting the primary TAP in TLR forces the eTAP into TLR within three TCK cycles (immediately if eTRST* is supported).

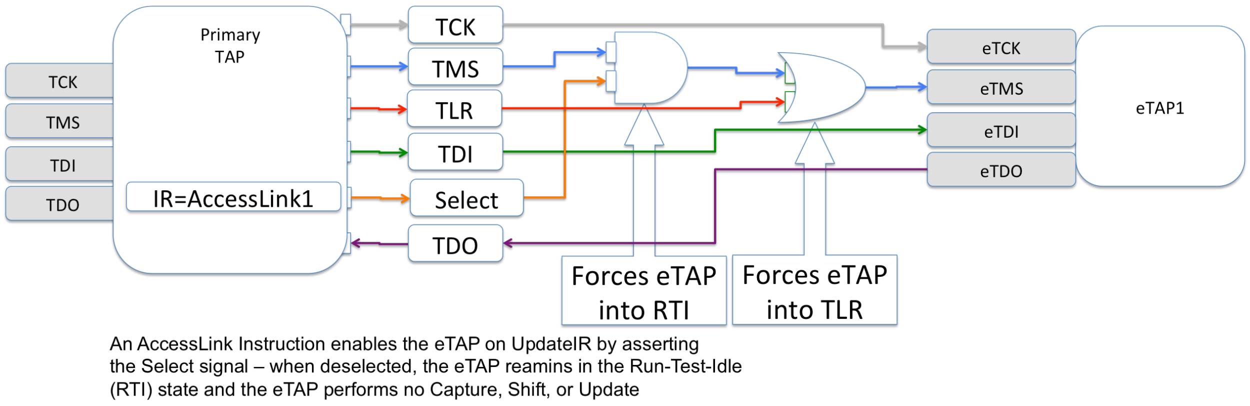

Primary TAP
TCK
TMS
TDI
TDO
IR=AccessLink1
TCK
TMS
TLR
TDI
Select
TDO
Forces eTAP into RTI
Forces eTAP into TLR
eTCK
eTMS
eTDI
eTDO
eTAP1
An AccessLink Instruction enables the eTAP on UpdateIR by asserting the Select signal – when deselected, the eTAP reamins in the Run-Test-Idle (RTI) state and the eTAP performs no Capture, Shift, or Update

Figure 9: A single eTAP enabled by an IJTAG AccessLink instruction

Figure 10 shows an extension of the eTAP selection concept. Here, a ScanMux or SIB is added to the architecture shown in Figure 9 (without the reset gate to minimize clutter in the drawing). In Figure 10, the default selection of AccessLink selects eTAP2. When the SIB is asserted (a logic 1 placed in the Update cell during a DR-Scan), eTAP1 is activated and is included into the overall scan path by the SIB generating the Select signal. If the SIB is closed (de-asserted with a logic 0), eTAP1 is removed from the scan path on the falling-edge of TCK in the UpdateDR state and eTAP1 is placed in RTI.

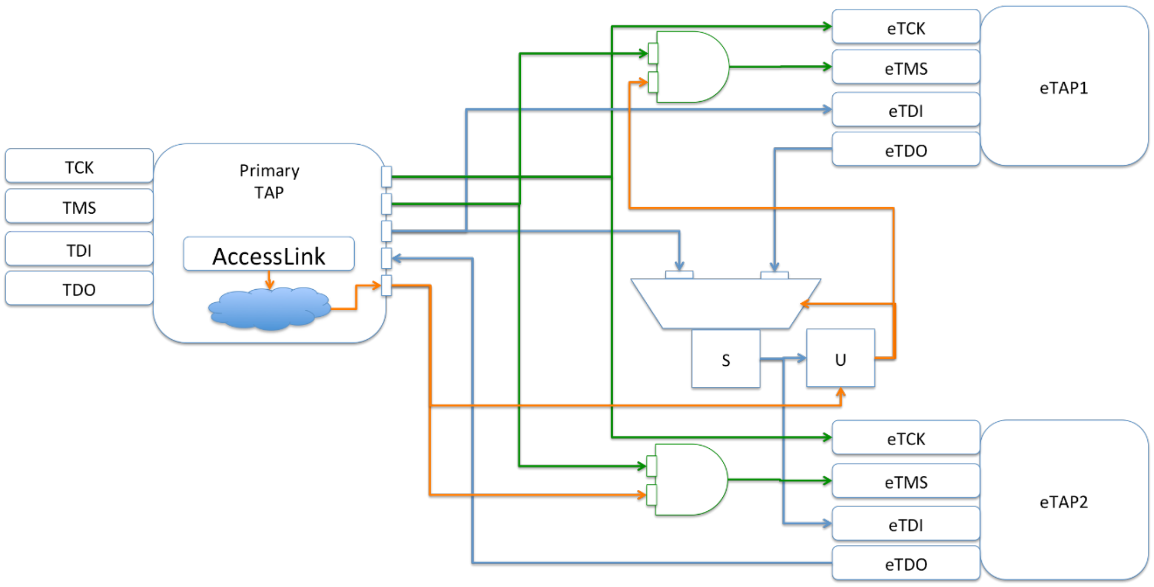


Figure 10: A multi-TAP architecture showing how a ScanMux (SIB) can include eTAPs into the scan path.

Other selection mechanisms are also possible. For example, different AccessLink instructions can select individual or groups of eTAPs. Or a configuration register can select eTAPs for inclusion in the active scan path. In the case of a configuration register, any one of several or many eTAPs can be associated with a bit in a configuration register that is either in the active scan path or accessible from a different AccessLink instruction.

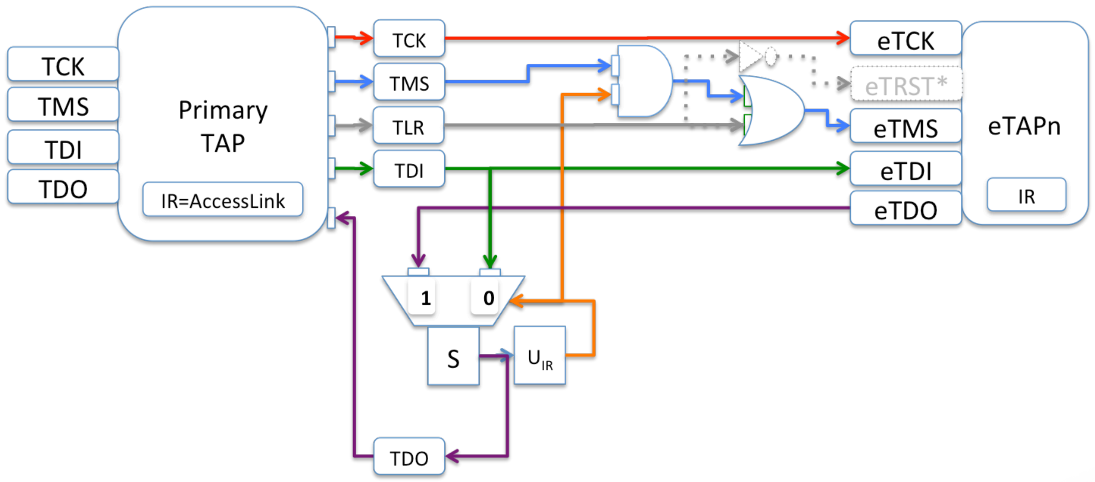


Figure 11: An eTAP can be parked in either RTI or TLR

Note that eTAPs, to be compliant with IEEE 1687 IJTAG (Figure 11), must be parked in Run-Test-Idle (RTI) when they are deselected. However, the eTAP must also be placed in a reset state, Test-Logic-Reset (TLR), when directed to do so by the primary TAP, whether selected or not. Parking in TLR is supported by applying the Global Reset signal to the 1687 network. Global Reset is generally produced when the IEEE 1149.1 JTAG FSM is in the TLR state but may also include the TRST* if the primary TAP supports the asynchronous reset signal.

Alternative Controllers

Although not included in the 2014 version of the IEEE 1687 IJTAG standard, the standard was developed so that alternative controllers besides the 1149.1 JTAG TAP could be included in the standard in future versions. At such time, the standard could be expanded to allow for even greater flexibility and scalability. IJTAG could evolve in a similar fashion to IEEE 1500 ECT, which allows for its architecture to be driven from an IEEE 1149.1 boundary-scan/JTAG TAP Controller FSM, directly from an automatic test equipment (ATE) system or other type of test equipment. To this end, the IJTAG AccessLink instruction was developed as a way to identify the specific controller that might be currently providing access to a 1687 IJTAG network interface and to set up that controller so that it would provide operations and access to the IJTAG network. For example, a very popular embedded instrument controller at the board level is the Serial Peripheral Interface (SPI). Engineers may want to embed cores that include an IJTAG network and instruments into chips that have only SPI interfaces and no IEEE 1149.1 JTAG TAP port.

IJTAG Description Languages

The IEEE 1687 IJTAG standard defines two languages:

1. Instrument Connectivity Language (ICL): for describing the access mechanism and network of embedded instruments
2. Procedure Description Language (PDL): for describing instrument operations or test vectors.

These IJTAG languages were created to be distinct from the IEEE 1149.1 JTAG Boundary-Scan Description Language (BSDL) and Serial Vector Format (SVF) languages for several reasons. The initial thought process of the IEEE 1687 IJTAG working group was that the IJTAG standard should not overload the IEEE 1149.1 JTAG languages with information that end-users (board test personnel) may not need, since many embedded instruments will only be implemented for IC test and not board test. IJTAG's ICL and PDL were created as separate languages and files so they could be delivered if requested by the board test or validation engineer. Another reason involved the nature of 1149.1 JTAG. JTAG does not document or describe instruments and their operations as IJTAG does; instead, JTAG is more concerned with creating chip-centric instructions, test modes, TDRs and ensuring that all of these are compliant with the IEEE 1149.1 standard. The IEEE 1687 IJTAG standard is more focused on describing the instrument interface and its related vectors and operations, and then documenting the pathway to the instrument interface. The main purpose of 1687 IJTAG is to enable the retargeting of instrument vectors to the pins of the chip. (Actually, 1687 IJTAG vector or operation retargeting may be applied from any module boundary to any other higher-level module boundary.) To facilitate this retargeting, a simple IJTAG language to represent the vectors or operations was created (PDL) and it was linked to the description of the instrument interface and the instrument access network (ICL).

Instrument Connectivity Language (ICL)

The basic goal of automated, software tool-based retargeting is that vectors or operations can be described at some signal interface or register. Furthermore, these vectors can be resolved through an access mechanism to a physically accessible interface such as the pins on an IC package or an edge connector on a board. To do this, the signal interface and access mechanism must be described so that a software tool can automate the process. However, some in the industry are concerned about divulging too much information about the specifics of a chip design or the instrument. For this reason, ICL was designed to describe the behavior of the network, not the physical representation of the network, the instrument itself or the instrument's target such as the memory targeted for testing by a memory BIST instrument.

The basic description element of ICL is the module, which is a software structure. Any IJTAG network description must contain at least one ICL module. Each different module should be given a unique name. A physical network may be made up of embedded TAPs, black-box modules, TDRs, SIBs and shift-path multiplexors. Instruments may include a signal interface, interface registers, interface logic, data multiplexors and clock definitions. So, an ICL module can contain a small set of building blocks that describe the network and the instrument interface. The overall ICL for a chip is organized into groupings of modules and looks like a hierarchical netlist description language. Each ICL module contains two basic types of information: 1) hierarchical network structure and instrument interface building blocks, which are required; and 2) parameters, aliases, enumerations, and attributes, which are optional. The optional elements can improve readability and reusability. For example, alias and enumerations are included in ICL modules to simplify the writing of PDL.

The language components and ICL optimizations are shown in Table 2 below:

Table 2: ICL language elements with proper syntax.

Index	ICL Description Elements
1	Module <ModuleName> {…}
2	<Func>Port <PortName> {…}[2]
3	ScanInterface <ScanInterfaceName>{…}
4	Instance <InstanceName> of <ModuleName> {…}
5	ScanRegisterName <ScanRegName> {…}
6	ScanMuxName <MuxName> SelectedBy <Selector> {…}
7	OneHotScanGroup <SignalName> {…}
8	AccessLink <InstanceName> of <LinkType> {…}
9	LogicSignal <SignalName> {…}
10	DataRegister <DataRegName> {…}
11	DataMux <MuxName> SelectedBy <Selector> {…}
12	ClockMux <MuxName> SelectedBy <Selector> {…}
13	OneHotDataGroup <SignalName> {…}

	ICL Optimization and Organization Elements
14	Alias <AliasName> = <Element> {…}
15	Enum <EnumName> {…}
16	Parameter <parName> = <parValue>;
17	Attribute <attName> = <attValue>;
	ICL Comments
18	// single line comment uses double slash
19	/* multi-line block comments */

NOTE: Two items in Table 2 are not fully expanded: 1) <func> Port can actually expand into a number of different port functions; and 2) the curly braces with ellipses, {…}, means that more information can be represented within the curly braces. If no extra information is available, the statement can be terminated with a semicolon.

A number of description elements are associated with the serial access network portion of the 1687 IJTAG standard. An IJTAG network may be delivered in several forms: as a whole network in a fabricated chip; as a portion of a network in an IP core, for example, or as just a TDR. As a result of this, most IJTAG network elements can be considered building block primitive elements that represent ports and items on the scan path. The physical elements that can make up a scan network are scan-path registers (TDRs), scan-path multiplexors, a scan interface or scan signal ports, and an access-link instruction. These items are described in Table 2 as items two through eight. Each is fairly self-explanatory.

Several of the other elements listed in items 2 through 8, such as Instance and OneHotScanGroup, are not self-explanatory. Instance will create the netlist or schematic view of the architecture. For example, a generic module named 32BitTDR may be used four times in an architecture. Each of these four modules are identical, but the four different applications of the module must be unique even though the same module is instantiated four times. The example ICL code below shows how two instances of the module 32BitTDR is made unique by naming one instance 'red' and other 'blue'.

```
Instance RedTDR32 of 32BitTDR;
Instance BlueTDR32 of 32BitTDR;
```

A OneHotScanGroup is just another version of a ScanMux, but the enable signals may source from multiple places in the architecture. So, these signals are listed within the curly braces in the OneHotScanGroup statement. An example of three signals from three different modules that select Mux1 is shown below:

```
     OneHotScanGroup Mux1 {
          Port RedTDR32.SO;
          Port BlueTDR32.SO;
          Port GreenTDR32.SO;
          }
```

Several ICL items describe the non-scan portions of the architecture. These commands are mostly associated with the embedded instrument, but some can also refer to the scan architecture. Each embedded instrument must have its signal interface described, in addition to any parallel registers or data multiplexors. Logic decode must also be described as well as the clock source or selection multiplexor. These items are listed as 9 through13 in Table 2 and are self-explanatory.

The ICL item that requires explanation is the LogicSignal statement. LogicSignal may actually refer to either the scan-path network or the instrument interface. LogicSignal represents a signal described as a Boolean expression and resolved to a single signal to enable a scan-multiplexor or data-multiplexor, for example. For instance, a LogicSignal would describe a real signal that sources from four different scan-path update-bits and these four bits resolve to a single scan or data multiplexor selection signal based on some AND, OR, Exclusive-OR, Not-AND, Not-OR, or Not combination. An example of this type of statement that reads "Bit1 AND NOT Bit2 OR Bit3 AND Bit4" would be:

```
LogicSignal MuxSel1 {
     Bit1 & ~Bit2 | Bit3 & Bit4;
}
```

Procedure Description Language (PDL)

IJTAG's Procedure Description Language (PDL) documents the operations of embedded instruments. PDL can be associated with different levels of an IJTAG network, but its primary task is to describe the operations or procedures for each instrument on the network (Table 3). In conjunction with ICL's network and instrument interface descriptions, PDL allows automated tools to retarget an instrument's procedures or test vectors to the chip's physical pin interface through the device's test controller. This allows IJTAG instruments to be portable, drop-in objects because no matter how many times an instrument is deployed or where it is placed within an IJTAG network, the device's ICL will map a pathway for the PDL procedures to the chip pins. From the chip pins, the PDL procedures can be extended further to the edge of the board where they can be controlled and managed by an IJTAG tool. PDL commands can be divided into two levels. Level-0 is the basic language and Level-1 is considered an advanced set of features that are described according to the rules of the Tool Command Language (Tcl) language.

The Level-0 PDL (PDL-0) is a small group of commands that provides the basic structure of the language and the basic vector actions needed by instruments. PDL-0 commands are evaluated by either a compiler or a Tcl interpreter. These basic Level-0 commands can be organized into four groups:

1. software structure commands that define the organization and actions of the language;
2. action commands that describe or define the operations of an instrument;
3. instrument configuration commands that define the environment associated with instruments;
4. resource commands that involve the sharing, isolating or describing of resources than can be associated with multiple instruments or the network.

Table 3: PDL Level-0 commands by group

Index	PDL Software Structure Commands	PDL Group Description
1	iPDLLevel	PDL commands associated with the software structure or actions of written PDL or PDL files
2	iPrefix	
3	iProc	
4	iProcsForModule	
5	iUseProcNameSpace	
6	iNote	
7	iCall	
PDL Action Commands		
8	iRead	PDL commands associated with the actions or operations involving instruments or the network
9	iWrite	
10	iScan	
11	iApply	
12	iReset	

13	iRunLoop	
PDL Definition Commands		
14	iOverrideScanInterface	PDL commands associated with defining the conditions under which PDL actions operate
15	iClock	
16	iClockOverride	
PDL Advanced Resource Commands		
17	iMerge	PDL commands associated with resource management of instruments or the network
18	iTake	
19	iRelease	
20	iState	

On the other hand, PDL Level-1 commands (Table 4) are a more sophisticated set of commands than PDL-0. PDL-0 commands can be thought of as 'flat' or 'static statements that map simple linear actions associated with an instrument. However, real-world operations require more complex actions, such as flow-control operations like 'if-then', 'do-while' and 'for-next'. Since PDL-1 is based on Tcl, which is a widely adopted and supported language, some of the PDL-1 operations actually come from the Tcl language. Ultimately though, a wide range of operations must be supported for the test and debug process and Tcl allows users to define their own commands. The PDL-1 commands defined in the 1687 standard are described in Table 4 below.

Table 4: PDL Level-1 commands defined in 1687.

CMD #	PDL Level-1 Command	Command Definition
1	iGetReadData	Return (as a string in the specified unsized number format) the value from the most recently applied iRead operation on register or output port (or an alias consisting of either or both). May contain x-values.
2	iGetMiscompares	Return (as a string in the specified unsized number format) the XOR of the value from the most recently applied iRead operation on a register or output port (or an alias consisting of either or both) and the value expected for that iRead operation. May contain x-values.
3	iGetStatus	Return the decimal number of iApply miscompares that have occurred since the last time that iGetStatus –clear was issued. Clear the count afterwards if directed.
4	iSetFail	Return the message string to the controlling program to indicate an unexpected condition, with the optional directive to abort execution.

It must be noted that the PDL can be written so that an embedded instrument may be used in several different ways. PDL may be written at the level of a complete test. For example, the PDL code may 'run_instrument_to_end' or 'run_instrument_to_fail'. Alternatively, PDL may be written at an atomic level. That is, the PDL code may instruct the instrument to perform certain functions, such as ’start’, ’pause’, ’stop’, ’algorithm_1’, ’change_test_data’ and so on. When PDL is delivered as a complete test, the end user doesn’t have any options other than to run the test. This is especially true if the ICL has been obfuscated to limit the user’s understanding of the instrument. When PDL is delivered as atomic functions, end users must develop complete tests on their own. Of course, this will require that users understand the operations of the embedded instrument. The last option would be for the PDL provider to deliver both a complete test and the atomic functions that allow end users to create their own complex functions.

Coordinating IEEE 1687 IJTAG at the Board Level

As the IJTAG ecosystem continues to develop, many of the initial IJTAG embedded instruments will likely be chip-centric instruments intended for ATE and system test, chip diagnostics, and possibly chip characterization and tuning. Of course, each instrument must have a viable return-on-investment (ROI), or it will not be utilized. If an embedded instrument saves time, effort, cost, or it can reduce the cost and complexity of the hardware associated with test and characterization, then the adoption of the embedded instrument makes sense. This adoption process has been going on since the 1990s and many types of embedded instrumentation such as Scan, Scan Compression, Memory Built-in Self-Test (MBIST), Memory Built-in Self-Repair (MBISR), PLL-BIST, and SerDes-BIST have become commonplace (Figure 12).

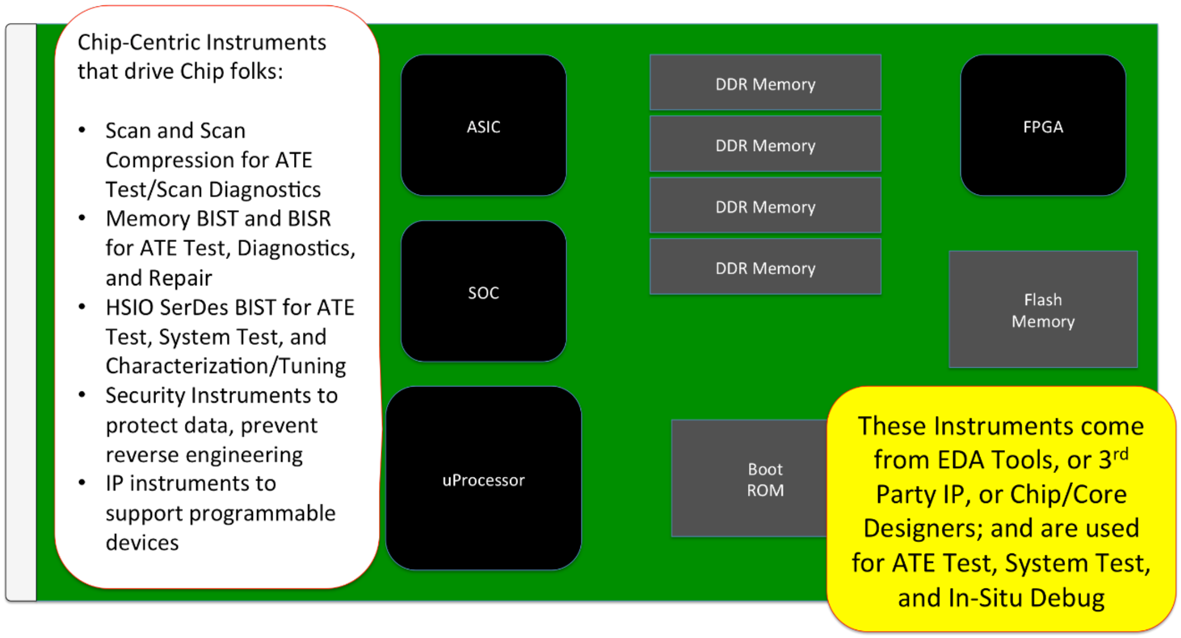

Chip-Centric Instruments that drive Chip folks:
• Scan and Scan Compression for ATE Test/Scan Diagnostics
• Memory BIST and BISR for ATE Test, Diagnostics, and Repair
• HSIO SerDes BIST for ATE Test, System Test, and Characterization/Tuning
• Security Instruments to protect data, prevent reverse engineering
• IP instruments to support programmable devices
ASIC
SOC
uProcessor
DDR Memory
DDR Memory
DDR Memory
DDR Memory
FPGA
Flash Memory
Boot ROM
These Instruments come from EDA Tools, or 3rd Party IP, or Chip/Core Designers; and are used for ATE Test, System Test, and In-Situ Debug

Figure 12: Chip-centric embedded instruments for chip provider purposes

Eventually, embedded instruments will also migrate to support other applications in addition to chip-centric verification and characterization. Inevitably, IJTAG embedded instruments will service the board and system environments. Chip designers would include these instruments within chips, but for the purposes of supporting board and system level applications such as validating chip-to-chip communication or providing on-board diagnostics. These types of instruments are becoming necessary at the board level because of the engineering challenges associated with modern board design. For example, shrinking board sizes, increases in chip complexity, increases in signal speeds and the loss of physical test points are limiting the ability of external equipment to test and characterize circuit boards. Most of the features requested by the board test community today are all centered around high-speed signals or reducing test time.

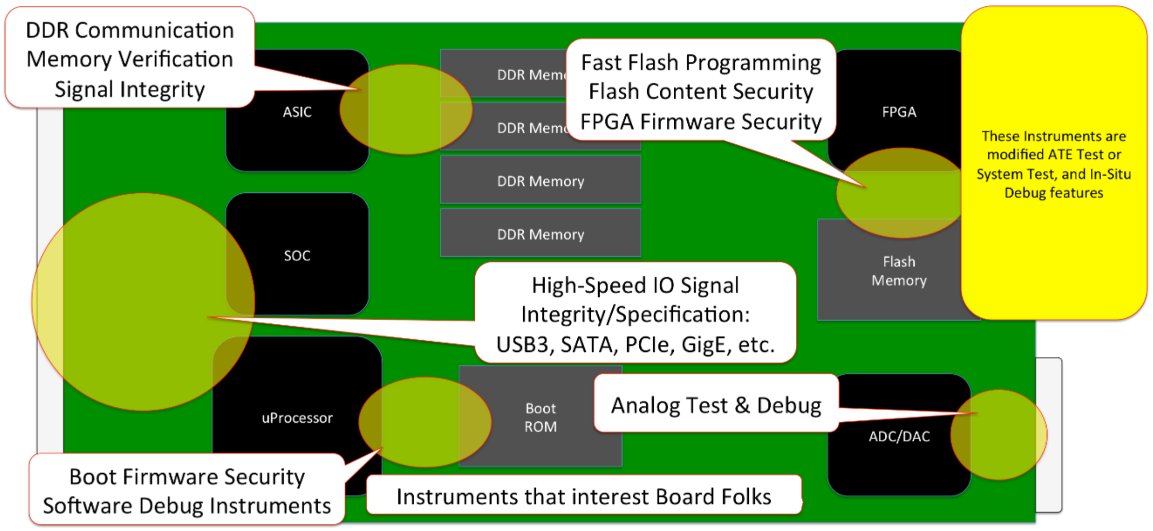


Figure 13: Embedded instruments requested by board test organizations

Even though both are included within chips, the main difference between chip-centric IEEE 1687 IJTAG instruments and board/system-centric 1687 IJTAG instruments (Figure 13) is the coordination at the board level that board-centric 1687 IJTAG instruments will require. For example, a SerDes BIST applied within a chip may include an in-chip loopback, but a SerDes BIST that is applied at the board level to validate board-level resources will require one chip to operate its embedded SerDes BIST instrument and another chip on the board to either be placed in a loopback mode or to receive and respond to the SerDes BIST traffic from the source chip.

Summary and Conclusions

Although the IEEE 1687 IJTAG standard was ratified in 2014, the concepts embodied in the specification are not new. In fact, interim versions of the standard have been applied by many early adopters years before the standard was ratified.

Widespread adoption of the IEEE 1687 IJTAG standard in the industry is anticipated, as evidenced by the support that technology providers have already given to the IJTAG ecosystem.

Any standard's adoption is always accelerated by the presence of a tools ecosystem, which supports the standard and simplifies its adoption for working engineers. Collaborative efforts among several major tools providers, including Synopsys, Siemens EDA, ASSET InterTech and others is proof that an ecosystem for the IEEE 1687 IJTAG standard has already emerged. At the chip level, EDA tools are already accommodating the simulation and verification of IJTAG resources and architectures, in addition to generating PDL for deployment on ATE testers. However, unleashing the vast potential of IJTAG-networked embedded instruments for testing systems-on-a-chip (SoC) and circuit boards, or even for field service applications, will require a complete bundle of resources, including Boundary-Scan Description Language (BSDL) files, ICL and PDL. With this sort of a package, the end user engineer will be able to employ IJTAG tools running on JTAG tester hardware. This pathway has already been proven by collaborative interoperability efforts involving EDA providers, IC suppliers and IJTAG tools companies.