IEEE P1687 (IJTAG) Draft Standard for Access and Control of Instrumentation Embedded Within a Semiconductor Device

A Presentation Based on Slides Created by Members of the IEEE P1687 (IJTAG) Working Group and presented by

Ben Bennetts, DFT Consultant

Tel: +44 1489 581276  E-mail: ben@dft.co.uk
http://www.dft.co.uk/
Outline

- Background: use of 1149.1 Test Access Port to access embedded instruments
- IEEE P1687 (IJTAG) evolution, scope, motivation, problem statement
- Evolution of embedded instrumentation – revisited
- Describing embedded instrument features: documentation, BSDL, access protocols
- P1687 Application Program Interface
- Secondary interfaces: the HUB concept
- Conclusions, status, issues, getting involved
Disclaimer

- This presentation is a personal view of what is happening within the IJTAG initiative. It is not necessarily an IJTAG-endorsed presentation. Other members of IJTAG may have slightly different views of the objectives and status of IJTAG. This is the nature of formative activities.
P1687 Evolution

- Began as separate and independent discussions between ASSET and Inovys, and Agilent and Cisco.
- First meeting at ITC04 BTTAC (BTAG) meeting.
- Working group formed shortly after VTS05.
- IEEE Project Authorization Request approved on 16 March 2006. Allocated P1687 project number *
- Now regular face-to-face meetings at conferences and workshops and weekly telephone conference calls.
  - 8 core members.
  - Over 80 “extended” WG members.

* Isaac Newton, *Principia Mathematica* published
June 2006 – Who’s Involved?

- Agilent Technologies
- ATI Research
- Cadence
- CISCO
- DAFCA
- IBM
- Inovys
- Lucent Technologies
- Novas
- SiliconAid
- ST Micro
- Synopsys
- Plus a number of independent consultants

Core Group:

Ben Bennetts, Bennetts Associates
Al Crouch, Inovys
Jason Doege, DA-Test
Bill Eklow, Cisco Systems
Mike Laisné, Qualcomm
Mike Ricchetti, ATI
Ken Posse, Consultant, Chairman
Jeff Rearick, Agilent Technologies
Draft Standard for Access and Control of Instrumentation Embedded Within a Semiconductor Device:

Scope: “This Standard will develop a methodology for access to embedded test and debug features, (but not the design of the features themselves) via the IEEE 1149.1 Test Access Port (TAP) and additional signals that may be required. The elements of the methodology include a description language for the characteristics of the features and for communication with the features, and requirements for interfacing to the features”
Defining Test, Debug and Instruments

- **Test**, usually taken to be structural (manufacturing) test
- **Debug**, usually taken to mean functional (validation) checkout.
- **Instrument**, very broadly:
  - Any on-chip circuit for test, debug, diagnosis, monitoring, characterization, configuration, or functional use that can be **accessed by**, **configured from**, or **communicated with** an IEEE 1149.1 TAP and TAP controller.
  - Note: an instrument is often called an **Intellectual Property, IP**.

Examples of instruments: next slide …
TAP Access to Embedded Instruments

- Internal scan chains, LBIST and MBIST
- I/O BIST (PRBS, Jitter Test, crosstalk test)
- Process Monitors (used to identify systemic problems)
- Voltage Monitors (used to identify IR-drop problems)
- State Dump
- Built in Logic Analyzer or O-scope embedded instrumentation
- IEEE 1500 Wrapped Cores
- BUT – no standard directions on how to access these features

Systemic: affecting the body generally
Motivation for P1687

- **Board/System:**
  - Facilitate test and debug development
  - Interoperability of tests across multiple test-prep-and-apply vendors and across multiple test processes (prototype board validation, volume manufacturing)

- **Chip instrument designers/providers:**
  - Require a standard way of defining the use of an embedded instrument

- **Chip ATE:**
  - Facilitate test and debug development
  - Promote low-cost chip ATE
Example: Creating a Board/System Test

- **Complex components**
  - Multiple SOC and High-End ASICs
  - Microprocessors
  - Embedded and standalone memories
  - Programmable logic devices

- **Complex design features**
  - High-speed I/O
  - Backplane connections
  - Multiple configurations
Goal: Re-use Component Instruments …

… at board and system level:

- Re-run device embedded memory tests
- Re-run device logic BIST
- Run ASIC-based external memory tests
- Run chip-to-chip High-Speed IO Pseudo-Random Binary Sequences tests
- Monitor internal signal waveforms
- Capture internal chip states (scan dump)
- Use chip test features to assist board test

… and even at chip level
Daunting Task: Assembling IC Info

- BSDL (Boundary Scan Description Language) files
- Initialization sequence(s) and clock control
- Logic BIST and MBIST recipes
  - Setup, launch, checking procedures
  - Diagnostic routines
- List of other test and debug instruments and access methods.
For each high-speed link
  - Method to setup, launch, and check Bit-Error Rates
  - Ability to apply different crosstalk, jitter, noise, data content conditions

For each parallel bus
  - Method to setup, launch, and check signal integrity properties.
  - Patterns for crosstalk, glitches, etc.

For each backplane configuration
  - ...
Accessing Instruments: Cisco’s Experience

Accessing Test Features is Painful
- Multiple ASIC vendors
- Multiple memory vendors
- Multiple test methodologies: structural and functional.
- Multiple ATE platforms
- Multiple test languages

- Bottom line: chip test re-use at board/system?
  - It’s tough!
There is currently no standard method to describe or interact with chip design, test and debug instrumentation.

It is currently very difficult to re-use chip-level DFTest and DFDebug instruments at the single-board and multi-board (system) levels.

There is a growing supply of test and debug instruments and a growing need to re-use it at higher levels.

So, what is a design, test and debug instrument?
The Evolution of Embedded Instruments

HSIO BIST Engines
- PLL and Clock Configuration
- Enabling/Disabling Memory Lock
- Configuration of Polynomials
- Configure Reduced Pin Count Modes

Internal Scan Chains
- Configure Input Isolators
- Configure Input Pin Isolators
- Configure Access to Test Bus
- Configure Clock Domain

Boundary-Scan Logic (IR, DRs, Controller, etc.)
- Selection of Output Compactor
- Configure Scan Dump Modes
- Configure Clock Chop Ratio
- Configure Functional Units
- Crosstalk Generation

1149.1 Test Access Port
- Configure Input Isolators
- Configure Clock Chop Ratio
- Configure Access to Test Bus
- Configure Reduced Pin Count Modes

Memory BIST Engines
- Core Instrument Interface
- Current Control
- Current Measurement
- Configure External Memory BISTs
- Configure Internal Memory BISTs
- Configuration of Memory BISR

Configuration of Functional Units
- Configurable Control LFSR
- Re-Seeding
- Configuration of MISRs
- Configuration of Functional Units
- Selection of Access to Test Bus
- Asserting Test Bus

IO_WRAP
- Configure Tristate Bus Controllers
- Configure Scan-In/Scan-Out Ports
- Alternate IDCode Register
- Configure 1500 Wrappers

Assertion Checking
- Configure Scan Wrappers
- Configuration of Decomp Units

Signal Tapping
- Configure Tristate Bus Controllers
- Configure Scan-In/Scan-Out Ports
- Alternate IDCode Register
- Configure 1500 Wrappers

Current Meter
- Configure Tristate Bus Controllers
- Configure Scan-In/Scan-Out Ports
- Alternate IDCode Register
- Configure 1500 Wrappers

Logic Analyzer
- Configure Tristate Bus Controllers
- Configure Scan-In/Scan-Out Ports
- Alternate IDCode Register
- Configure 1500 Wrappers

O-Scope
- Configure Tristate Bus Controllers
- Configure Scan-In/Scan-Out Ports
- Alternate IDCode Register
- Configure 1500 Wrappers

Configuration of Incremental and Final Signatures
- Configure Tristate Bus Controllers
- Configure Scan-In/Scan-Out Ports
- Alternate IDCode Register
- Configure 1500 Wrappers

The Evolution of Embedded Instruments

© 2006, IEEE P1687 Working Group
Last revised: 14 June 2006
Types of Embedded Instruments

- 1500 wrapper structures
- Clock control
- Power control/Measurement
- Voltage Measurement
- Temperature Measurement
- I/O configuration
- In-Circuit Emulation
- Functional Configuration
- Internal Memory BIST
- External Memory BIST
- Logic BIST
- IO - PRBS
- IO - BERT
- IO - Jitter
- IO – SSO/Crosstalk
- IO – Parametric Adjustment
- Logic Analyzer
- O-Scope
- External Trigger selection
- Performance Monitors
- PLL and Clock configuration
- Power Control
- Scan Dump Control
- X-Mask control, for TDC
- IBIST
- Inter-domain Synchronizers
- Internal Test Management
- Waveform Generation/Analysis
- Packet Generation
- Internal Counters/Status Registers
- Chip/Die ID
- Analog Muxing

More Details ..
TAP-based Access to Instruments

- uP/ASIC/ASSP/FPGA
- Internal test instruments (BIST, LAs, O-scope, clock controllers, etc)
- Register access
- Internal interface
- IEEE 1149.1 TAP
- High band width
- Standard Protocol
- hand shake
- Latest Protocol
- Test Data

© 2006, IEEE P1687 Working Group
Last revised: 14 June 2006
Fully-Embedded Instruments

- Fully embedded instruments simply require a mechanism for initialising the instrument, running the instrument, and collecting the final result e.g. memory BIST in Pass/Fail mode.
Partially-Embedded Instruments

- Partially embedded instruments require a real-time connection to monitor the progress of the instrument and, possibly, cause a mode change e.g. analysis and display of Logic Analyser signals.
Key Focuses of P1687

- **Documentation: architectural descriptions**
  - Identify accessible embedded instruments
  - Specify characteristics of the instrument

- **Access protocols: procedure descriptions**
  - Describe how to communicate with an instrument
  - Facilitate re-use through portability

- **“Enhanced”, secondary access/interface:**
  - Service instruments not easily handled solely by the TAP (i.e. use high bandwidth I/O)
  - Simplify hierarchical test architectures
Communication between provider and consumer is ad-hoc

There is no mechanism to specify which instruments are included in the chip
  - In many cases available test instruments are not used due to lack of knowledge

Details often included in long specs (no common format)
  - Many times specs are communicated “word of mouth”
Survey of Cisco BIST Documentation

Cisco study found ....

- **Supplier MBIST and IOBIST**
  - Brief description of logic
  - Boundary scan tester macros

- **Cisco Internal Logic BIST**
  - No HW documentation
  - Verilog test bench

- **Supplier High-Speed IO Pseudo-Random Binary Sequences**
  - No documentation (application engineer)

- **Cisco Internal Scan Dump**
  - Macros provided by ASIC DFT group

- **Cisco External Memory BIST**
  - 50+ page specification
  - Full HW and access protocol definition
End user can easily identify and program embedded instruments
Facilitate automated tests based on machine readable descriptions
Minimize “Time to Understanding” and “Time to Bring Up”
Describe the architecture of internal instruments i.e. how to use, but not what they do.
- Instrument name, type and instance.
- Register definition: location, length, serial or parallel access, clocks, type.
- Control action details: initialize, execute, wait time, results collection.
- Data and instruction formats.
- Internal and external dependencies.

Provide an inventory of all instrument content on a chip: basically, to identify the number and location of each instrument.

Provide enough information for a programmer to determine how to perform low level instrumentation functions.
Today’s “common” interfaces for embedded instruments:

- TAP-based
- I²C, or some other bus.
- Custom CPU interface e.g. ARM CoreSight
- Internal core-based e.g. 1500 wrapper
- Custom protocol

Each with their own language!
Facilitate re-usable code across boundary-scan platforms and different test processes.

Simplify program development by building on several lower level procedures e.g. IR-Scan, DR-Scan, etc.

Application Programming Interface versus a Language such as CTL (IEEE 1450.6), STAPL, SVF, …

- Instruction and data protocols.
- Action sequencing.
- Pass/Fail and error reporting.
- Be EDA/ATE vendor independent.
IEEE P1687 API Proposal

- P1687 procedures can be thought of as an API:
  - Can be called from a variety of higher-level environments.
  - Delivered as a package by the IP provider (instrument designer): initiate actions, set up parameters, collect responses.
  - Expose only those features that IP provider chooses.
  - Hide low-level details from user.
  - Instrument actions embedded in a set of standard P1687 function calls based on BSDL/HSDL analysis.

- Layers:
  1. test / measurement process
  2. exported instrument procedures
  3. register writes and reads
  4. TAP commands

End user

IP provider

Compiler
P1687 Static Test Assembly Flow

Test program calling IP procedures (your favorite language)

IP procedures (reg read/writes)

1149.1 TAP-based test assembler

Scanpath configuration

Stream of TAP instructions/data

Assumes Static Test
IP procedures (register read/writes)

TAP interface DLL / API

Interactive program calling IP procedures in terms of 1149.1 TAP interface box API

Chip/Board/System JTAG connection

Chip/Board/Sys scanpath configuration
Secondary Interfaces/HUB* - Intent

- Allow hand-off to an interface with higher bandwidth than the 1149.1 TAP for data-intensive operations
- Allow interoperation between individual instruments that may require asynchronous signaling
- Allow an avenue for support of legacy (non-TAP) instrument interfaces

* Still deciding on a name: router, gateway, nub, kernel, cynosure (guide), nukulus, …
Phase 1: simple instrument-to-TAP interface
- Need: spec for standard interface to instruments
- Solution: scan-based: TAP Test Data Registers (TDRs)

Phase 2: higher bandwidth I/O
- Need: hand off data transfer to another interface with higher bandwidth than TAP
- Solution: mux controls to configure I/Os

Phase 3: instrument intercommunication
- Need: instrument-to-{instrument/ATE} communication
- Solution: hierarchical “HUB” with asynchronous signals
Phase 1: Test DR-Based Access

Phase 1: simple instrument-to-TAP interface
Phase 2: TDR + I/O + Polling signals

Phase 2: higher bandwidth I/O

Registers for A

Registers for B

Note:
REQ = start
ACK = done

Instrument A

Instrument B

some interface
Phase 3: HUB Approach

Phase 3: instrument intercommunication
HUB-less P1687 Interface (Phase 1)

Slide 36

© 2006, IEEE P1687 Working Group
Last revised: 14 June 2006

Simplest 1687 Interface
HUB: One Instrument, One Phy

Note: the HUB can re-use the registers in the instrument.
HUB with Two Instruments (Phase 3)

Could be synchronous

Could be asynchronous
P1687 Interface Hierarchy
Hub Summary & Issues

- Goal: Make the HUB just simple enough
  - Enable TAP-based interaction with instruments
  - Enable high-bandwidth data transfer (optional)
  - Enable instrument interaction (optional)
  - Enable cascading hierarchically (optional)

- Status: Closing in on an architecture but …

Oh dear!!

Diagram:
- Main Core “JTAG” Port
- Sub Core 1500 WSP
- LogicVision’s Wrapper TAP
- Custom Port

© 2006, IEEE P1687 Working Group
Last revised: 14 June 2006
Summary: The Three Pillars of P1687

Architectural Descriptions
[BSDL/HSDL]

Interface Handoff
[HUB]

Procedure Descriptions [API or
CTL/SVF/STAPL/…]
Open Issues

- Exact architecture of the HUB
  - 1149.1 TAP to lower level ports (1500, WTAP, JTAG, custom)
  - Do we need to handle hierarchy?

- BSDL/HSDL for architectural description: syntax, specs

- Patterns and protocol procedural language choice:
  - Application Program Interface, or …
  - Full-featured programming language?

- Compliance checking

- Looking for help
  - Staff subcommittees;
  - Provide real world examples
To get involved/learn more/register your interest, contact Ken Posse (IJTAG Chairman) at kepos@comcast.net
To Probe Further …

- Jeff Rearick et al., “IJTAG: a Step on the Evolutionary Path” ITC 2005, P32.4
Any Questions?
## Appendix 1. In More Detail ….

<table>
<thead>
<tr>
<th>Power management</th>
<th>Turn on/off input isolators (pass gates)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Turn on/off pullup/pulldown resistors</td>
</tr>
<tr>
<td></td>
<td>Turn on/off entire clock domains</td>
</tr>
<tr>
<td>Clock control</td>
<td>Select chop-clock ratios for transition and path delay testing</td>
</tr>
<tr>
<td></td>
<td>Select chop-clock ratios and dividers for functional operation</td>
</tr>
<tr>
<td>Chip configuration</td>
<td>Capture of alternate chip IDCode register (chip or mask version)</td>
</tr>
<tr>
<td></td>
<td>Selecting Functional Units on the chip to be disabled/enabled</td>
</tr>
</tbody>
</table>
### Memory test

- Selection of Memory BISTs to be run in parallel (1-hot bit per BIST)
- Selection of memory BIST algorithm
- Selection of memory BIST background (e.g., 3-C, 5-A, 0-F, 9-6)
- Selection of Memory BISTs to be run in diagnostic mode
- Data collection from a memory BIST operating in diagnostic mode
- DMA (connecting BIST muxes to chip-level busses with access to pins)
- Enabling-Disabling memory lock for test and debug (OE and R/IW)
## … Continued

<table>
<thead>
<tr>
<th>Scan test</th>
<th>Selection of chip-level scan mode or individual partition scan mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Enabling-disabling-configuring internal tristate busses</td>
</tr>
<tr>
<td></td>
<td>Reconfiguration of scan chains</td>
</tr>
<tr>
<td></td>
<td>Reconfiguration of scan wrappers around cores</td>
</tr>
<tr>
<td></td>
<td>Selection of pins to be used as scan-ins and scan-outs</td>
</tr>
<tr>
<td></td>
<td>Configuration of pins used as scan-ins and scan-outs</td>
</tr>
<tr>
<td></td>
<td>Subsuming AC scan operation completely within JTAG</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Logic BIST</th>
<th>Selection of STUMPs Logic BIST units</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Selection of polynomial of PRPG LFSRs</td>
</tr>
<tr>
<td></td>
<td>Scanning in of seeds for re-seeding or debug</td>
</tr>
<tr>
<td></td>
<td>Selection of 'final signature compare' vs. 'incremental signature extraction'</td>
</tr>
<tr>
<td></td>
<td>Scanning in of signature to be compared</td>
</tr>
</tbody>
</table>
### Debug & Diagnosis
- Loading an internal counter used as a breakpoint
- Shadow capturing key registers (with a SAMPLE-like function)
- Masking or overwriting key registers (with an EXTEST-like function)
- Replacing data in key registers (with and UPDATE-like function)
- Selection of scan dump mode (lock memory, enables scan-out)

### PLL control
- Access to control registers for gating Scan-Enable and Scan-Clock PLL outputs

### Reduced pin count test
- Access to low-frequency IOs
- Control PLL bypass for clock control
- Control of internal scan chain configurations (detect versus locate)
### Access for embedded instrument outputs

<table>
<thead>
<tr>
<th>Fault insertion</th>
<th>Drive IO faults</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Drive internal faults</td>
</tr>
<tr>
<td>Control of embedded instrumentation</td>
<td>Control of use of embedded instruments</td>
</tr>
<tr>
<td></td>
<td>Access for embedded instrument outputs</td>
</tr>
</tbody>
</table>
Appendix 2: Memory BIST Procedure

- Function declaration
  - e.g.: int run_mbist_ram256x32sp(int repair_enable, int background);

- Function body
  int run_mbist_ram256x32sp(int repair_enable, int background);
  {
      start_clock(MCK);
      done_reg = 0;
      pass_reg = 0;
      repair_enable_reg = repair_enable;
      background_reg = background;
      start_BIST = 1;
  }
Appendix 3: Four Technical Challenges

- **Bandwidth**
  - Bus sizing and data rates for instruments

- **Sequencing**
  - Temporal staging of instrument actions

- **Synchronization**
  - Coordination of chip resources and instruments

- **Interoperation**
  - Connectivity with external resources (e.g. ATE)
Bandwidth

- Communication bottlenecks:
  - Inside chip to outside world: state dump
  - Outside world to inside chip: memory preload
  - Inside chip to inside chip: BIST

- Control vs. Data bandwidth
  - Control precedence? The ability to interrupt data?

- Scalability across instruments
  - Go/NoGo vs. massive data dump
Sequencing

- Simplistic approach to instrument staging:
  - Initialize, launch, check

- Complications
  - Multiple launches
  - Interruptions
  - Destructive checking
  - Diagnostics
  - Power limitations

- Language requirements
Synchronization

- Coordination of chip activity with instruments
- Coordination of board/sys activity with instruments
- Coordination across multiple instruments
- Possible need for real-time interaction
- Time stamping with IEEE 1588
- Cross-clock domain data transfers
- Synchronization to TAP clock domain
Interoperation

- Connection to external resources (ATE, controllers, measurement devices, etc.)
- Control and data exchange protocol and language
- Access to instruments during mission mode
- Master/Slave relationships with multiple instruments
- Security
Instrument definition

attribute INSTRUMENT_DEF of <device name>:
  entity is
  "Core1 IP1," &
  "Core2 IP1";

Chain definition

attribute CHAIN_DEF of <device name> entity is:
// first register in chain is R1 of Core 1 and it is 4 bits
  "Chain1 (Core1.R1,4)," &
  "Chain1 (Core1.R2,4)," &
// first two elements of Chain2 are R1 and R2 of Core2
  "Chain2 (Core2.R1,4 Core2.R2,4);"

Register access

attribute REGISTER_ACCESS of <device name> entity is:
  "Chain1 (MEMTST1)," &
  "Chain2 (MEMTST2);"