An SJTAG Interface Perspective

BTW 2006

2006 IEEE 5th International Workshop on Board Test

Bradford G. Van Treuren

SJTAG

Lucent Technologies
Bell Labs Innovations

Outline

- Purpose of Presentation
- Uses of JTAG Interface
- Insights to Modularity
- Leaf Functions for JTAG
- Layers of Software for Traditional EBST
- Interface Boundaries
- Standardization of Interfaces
- SJTAG Data Perspective
- Conclusions

Purpose of Presentation

Inspire people to begin to discuss further how JTAG could be leveraged by their own current designs by providing what the SJTAG goals state – including data contents and formats communicated through all interface boundaries – as open, standard, vendor-independent, non-proprietary ways that are repeatable and predictable.

A Look at Software Interface Boundaries

IS IT POSSIBLE TO DEFINE INTERFACE BOUNDARIES BETWEEN FUNCTIONAL ELEMENTS IN OUR EMBEDDED BOUNDARY SCAN SYSTEMS?

Uses for JTAG Interface

- Testing
- Monitor/Sample
- Chain Selection/Management
  - Addressable Shadow Protocol (ASP)
  - Scan Path Link
  - Scan Bridge
- FPGA/CPLD Programming
- FLASH Programming
- IJTAG/Instrumentation
- Emulation
  - Board Debug Mode (BDM)
  - Nexus (Based on 1149.1)
  - Extended JTAG for MIPS (EJTAG)
  - Compact JTAG for Mobil Devices (cJTAG)

Fixed Set of Vectors

Fixed Set of Vectors with Optional GPIO

Fixed and Dynamic Sets of Vectors

Dynamic Set of Vectors with Optional GPIO

Insights of Modularity

*"**SJTAG** seems to be to **JTAG** what **iSCSI** is for **SCSI**."* Jeffrey Moore, EMC[2]

> *"An implementation of an SJTAG architecture in a system allows **Test Programs**, running on **Test Controller**(s), to operate on **Bscan controlled Functions** in **Bscan enabled components**."* Gunnar Carlsson, Ericsson (SJTAG Vice Chairman)

Leaf Functions for JTAG

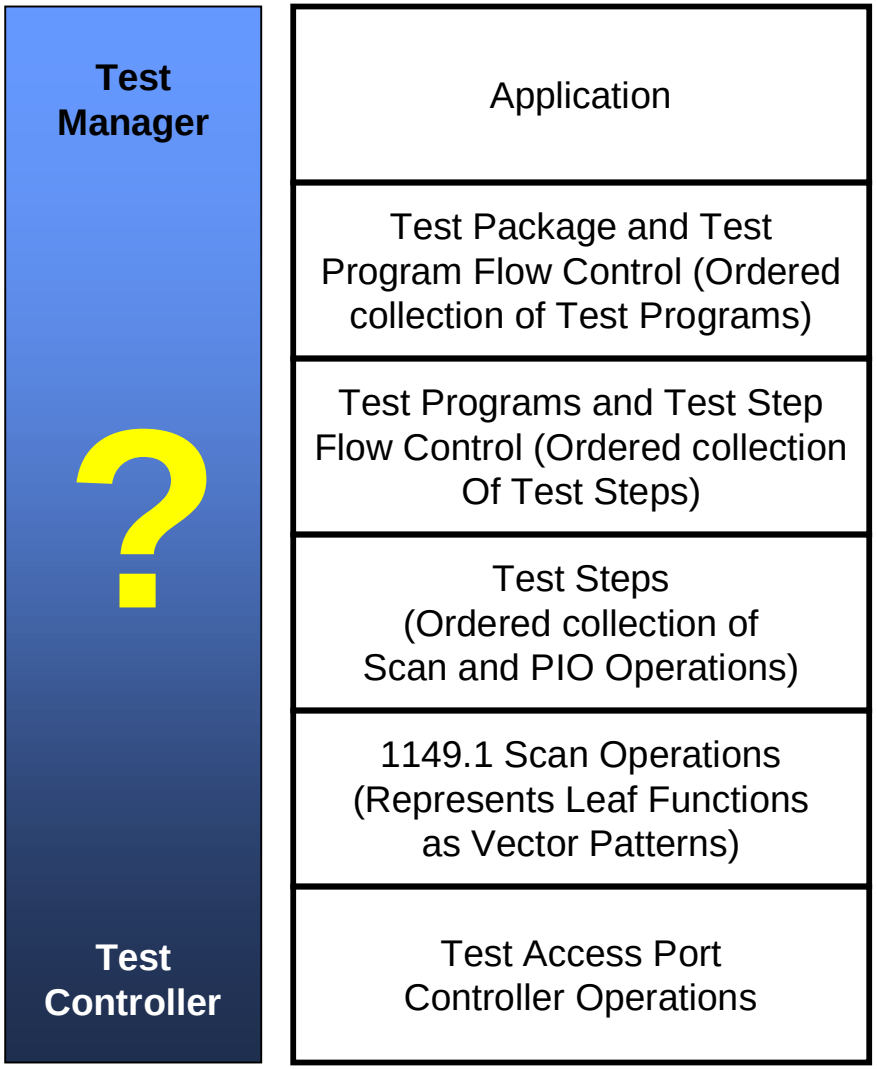
- Functions for configuration of scan chain segments
- Functions for accessing scan registers
  - E.g., PRELOAD BSR, LOAD BSR, SELECT BYPASS
- Functions for accessing built-in features
  - Execute the actual test (e.g., BIST operations)
  - Samples and Measurements (e.g., SAMPLE, 1149.4)
  - Etc.

➢ BSDL defines 1149.1 vector patterns for functions

➢ Languages, such as STAPL and SVF, define vector patterns and not functional behavior so intent is lost

➢ We need to manage the intent of the patterns in embedded environment

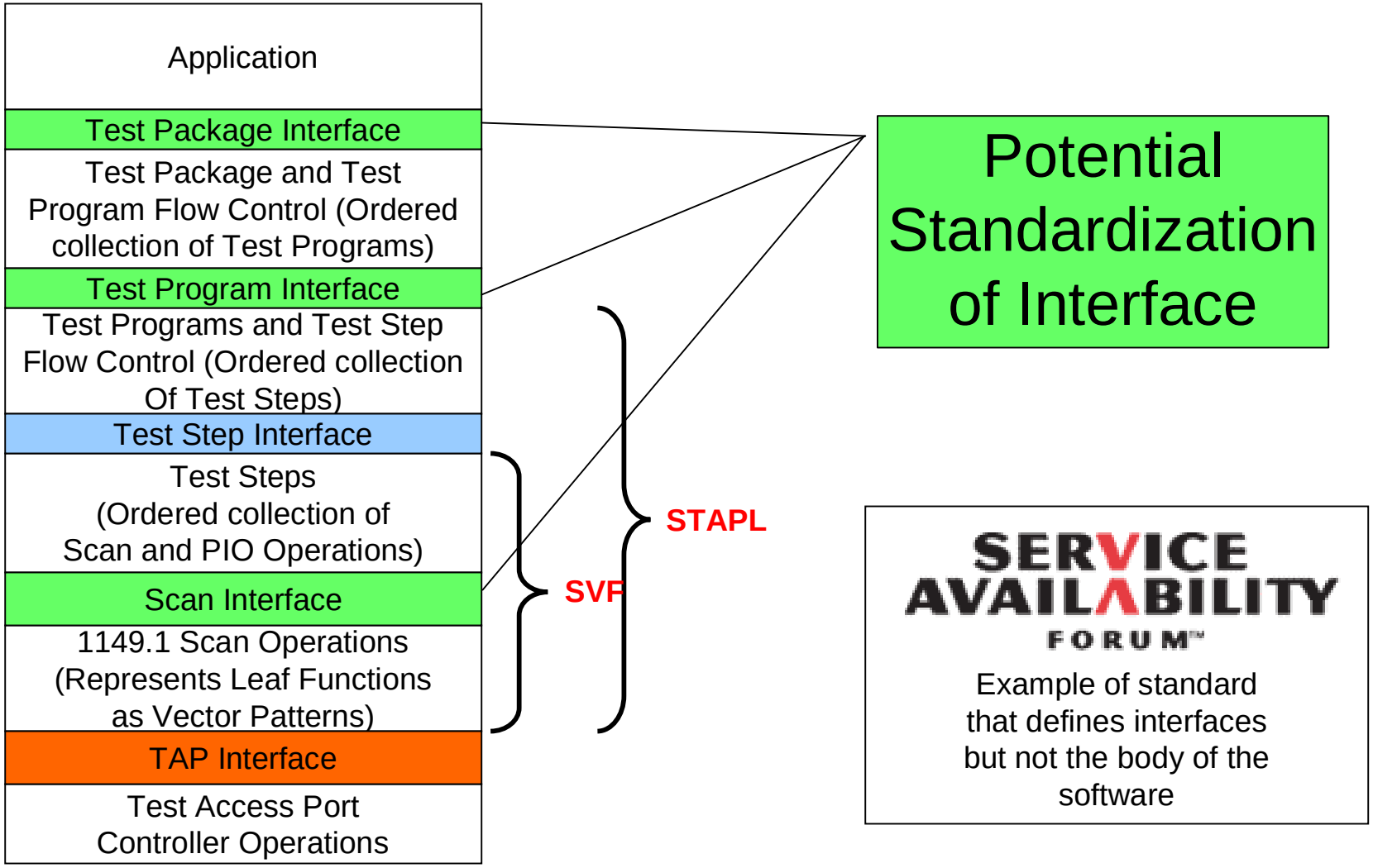
Layers of the Software for Traditional EBST



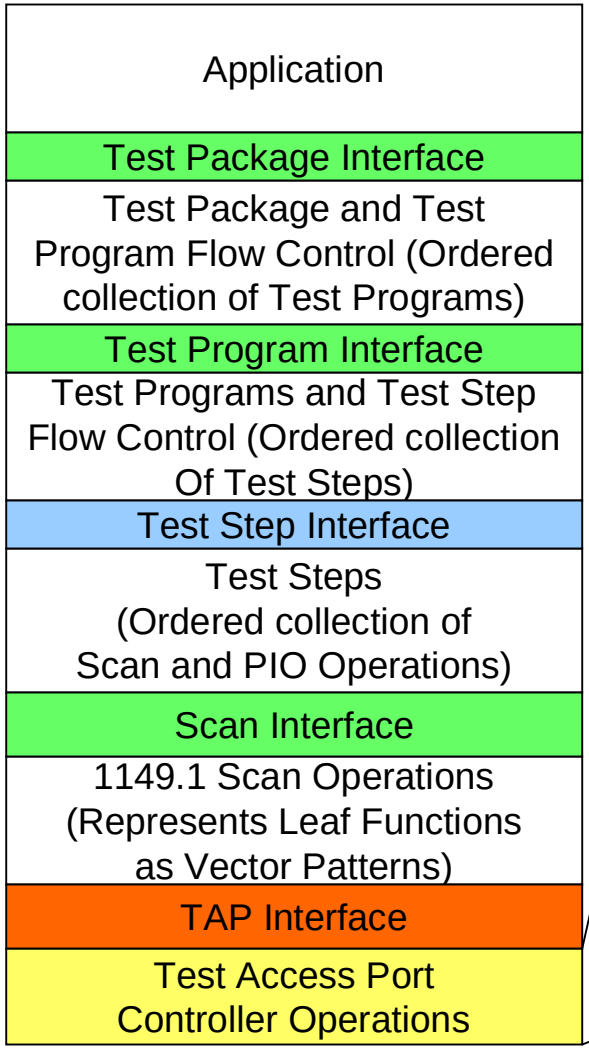
- Apply a set of vectors
- Capture a set of responses
- Compare responses to known expected values
- Conditionally apply next set of vectors based on response result of PASS or FAIL

➢ Fixed set of tests that are applied over and over again

Interface Boundaries of the Software



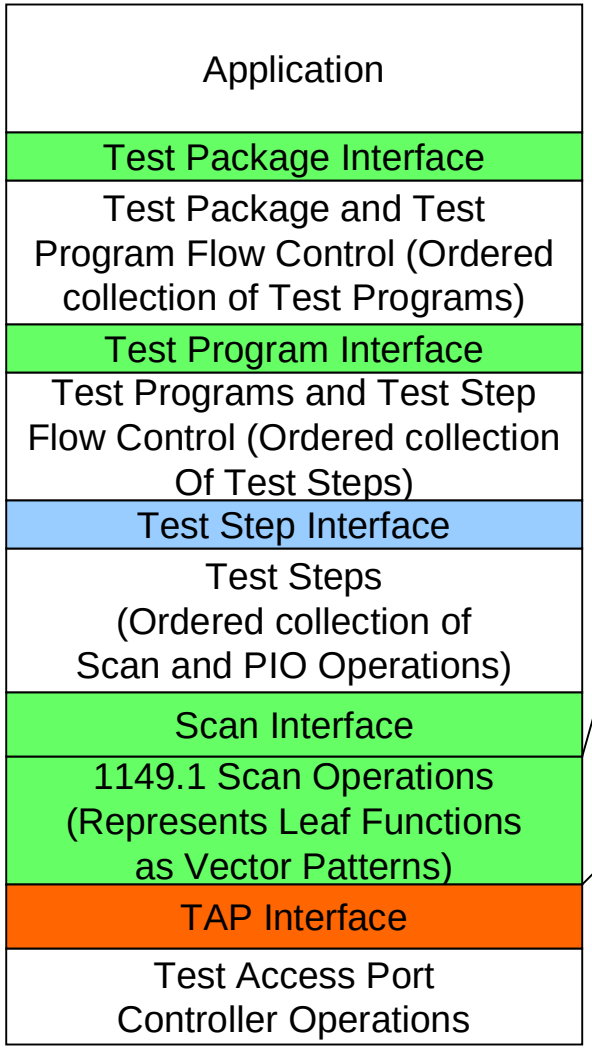
TAP Implementation Variability



- GPIO with Software base providing Parallel to Serial Conversion for IEEE Std. 1149.1 testing
- Specialized TAP Interface Device
- Specialized TAP Interface Intellectual Property (IP)
- Hybrid GPIO with Automated Data Shifters
- Uplink/Downlink TAP
- Scan Sequencers
- Scan Coprocessors

➢Not a candidate for standardization

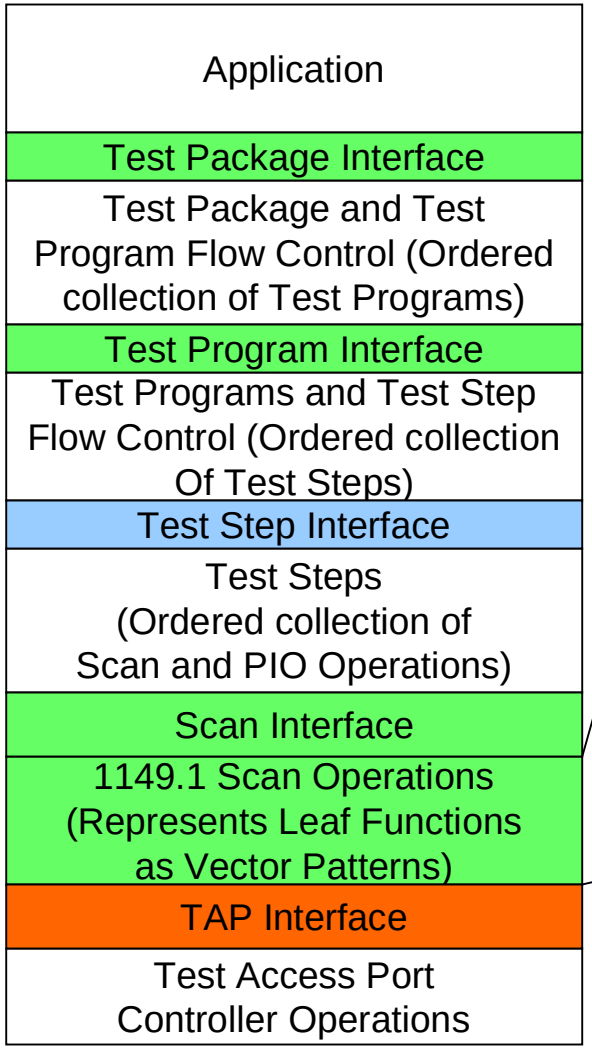
Traditional JTAG Functions



- IR Scan
- DR Scan
- GOTO State
- RunTest
- Frequency
- Async TRST
- Raw Bit Bang TAP

➢Good candidate for standardization of interface

Program/Emulator Specialization



- PIO Support
  - Functional
    - HRESET
    - SRESET
    - Write Pulse (WP)

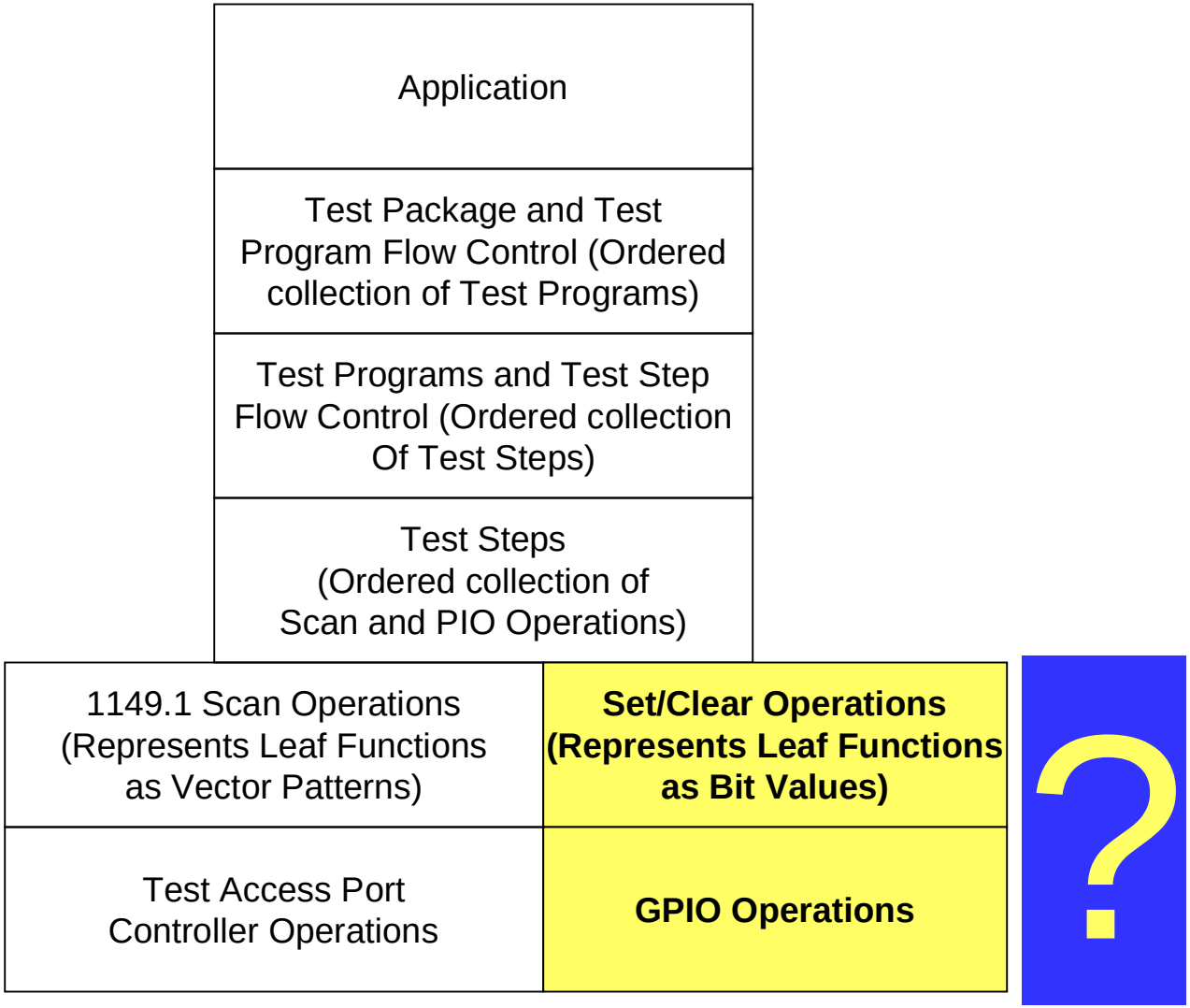
AND/OR

  - General Purpose
    - SetBIT
    - GetBIT
    - SetBITSET
    - GetBITSET

➢Could these be supported through extension functions?

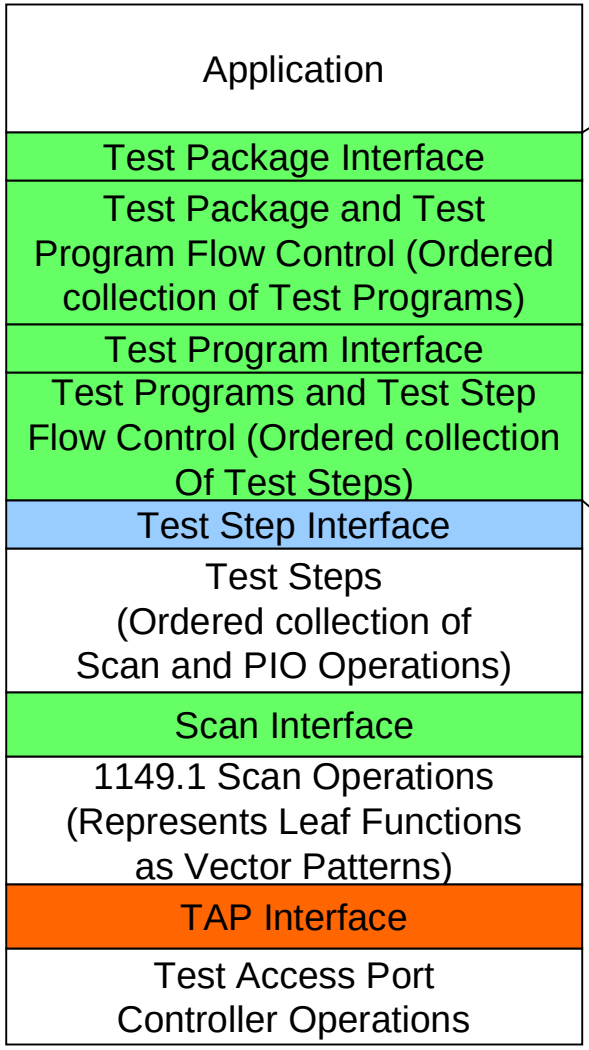
➢How would a language access these functions?

Non-Scan Extensions?



SJTAG Test Programs and Procedures

The intent behind the test program



TFCL™

- Each Test Step represents **smallest diagnosable Function/Action/Activity** for a UUT (e.g., Interconnect Test, ERASE, PROGRAM, VERIFY)
- Test Programs **manage the execution order of** and **results from** Test Steps

➢Management interface to Test Programs and results could be standardized

➢Management interface to Test Packages could be standardized

What about the data we use?

IS IT POSSIBLE TO MIGRATE TEST DATA INTO THE EMBEDDED ENVIRONMENT IN A USABLE FORM TO SUPPORT DIAGNOSTICS?

Analysis of Interconnect Test for EBST Diagnostics

Thinking outside the box!

- What can we mine/extract from this analysis?
  - Constant chain topology for entire test
  - Test vectors do not preserve test intent
  - Drivers cannot be deduced from test vectors
  - Observers can be deduced from MASK values
  - Failures detected by observer miscompares
- Is there a way to use this information to simplify diagnostics reporting in the EBST environment?

SJTAG Data Perspective

Interconnect ATPG Example

- What data do we need for test generation?
  - Netlist
    - Set of signals
    - Set of devices
    - Mapping of signals to device pins
    - Mapping of devices to characteristics
  - BSDL
    - Set of ports
    - Set of pins
    - Set of cells
    - Mapping of cells to cell types
    - Mapping of control cells to driver cells
    - Mapping of cells to ports
    - Mapping of ports to pins

➢ Net – **Device Pin** – Device Cell – Chain Cell

SJTAG Data Perspective

Interconnect ATPG Example

- What data do we really need from ATPG process for Failure diagnostics?
  - BSDL
    - Position of cells in device
    - Mapping of cells to device pins
    - Number of cells in the device's configuration
  - Netlist
    - Mapping of device pins to signals
    - Position of device in the scan chain (Yields position of cell in chain)

SJTAG Data Perspective

Interconnect ATPG Example – Data Representation

➢ Net – **Device Pin** – Device Cell – Chain Cell

- Databases store data in tables
- Related information contained on same row
- Can we use a table for diagnostic data storage in EBST?

Table: DIAGDATA			
Chain Cell	Device Cell	Device Pin	Nets
5	IC3.5	IC3.A5	SIG1
137	IC26.7	IC26.10	WRITE
138	IC26.8	IC26.12	CE
…	…	…	…

Conclusions

- We can learn from other industries how to better analyze SJTAG roles.
- We need to be thinking outside the manufacturing test mentality for EBST diagnostics
- We can find places for standardization in the software – SW Interface definitions
- Non-Test applications may reuse/leverage lower level standardized interfaces of the SW stack without a common implementation
- It is possible to provide real-time failure diagnostic feedback from an EBST environment