


SJTAG Meeting at EBTW

SJTAG Meeting at EBTW, Tallinn, Estonia

25 May 2005

Ben Bennetts
ben@dft.co.uk

EBTW 2005, Tallinn, Estonia
Slide 1



Attendees

Name	Affiliation	E-mail	Name	Affiliation	E-mail
Jukka ANTILA	Nokia	Jukka.anyila@nokia.com	Ville HASSINEN	Ericsson	Ville.hassinen@ericsson.com
Ben BENNETTS	Bennetts Associates	ben@dft.co.uk	Peter HORWOOD	Firecron	Peter.horwood@firecron.com
Gunnar CARLSSON	Ericsson	Gunnar.Carlsson@ericsson.com	Eugen MULLEN	Firecron	Eugene.mullen@firecron.com
Pete COLLINS	JTAG Technologies	petec@jtag.co.uk	Benard SUTTON	B Sutton Consulting	B-sutton@btconnect.com
Bill EKLOW	Cisco	beklow@cisco.com	Anders UGGLA	SAAB Test Systems	Anders.uggla@testsystems.se
Billy FENTON	ITT	Billy_fenton@intertech.com	Reg WALLER	ASSET InterTech	rwaller@asset-intertech.com
Ken FILLITER	National Semiconductor	Ken.filliter@nsc.com	Thomas WENZEL	Goepel	wenzel@goepel.com

EBTW 2005, Tallinn, Estonia
Slide 2



Purpose of the Meeting (1)

Several systems companies, including Ericsson, are proposing or are already building quite sophisticated approaches to system level test based on the use of a suitable backplane test bus, such as 1149.1 or an alternative e.g. I2C. The assumption is made that the boards already contain boundary-scan chains and that some form of multi-drop architecture allows access to individual boards and even individual devices on a board for test application (re-use of single-board tests or board-to-board tests) or in-system configuration/reconfiguration of on-board PLDs. Such techniques support various field service requirements and can be managed remotely (over a wireless network) or locally (over a wired connection) or be fully embedded.



Purpose of the Meeting (2)

The field-service Development and Diagnostic Test Manager can be a free-standing system based on an extension to existing PC-based single-board boundary-scan testers and the question has arisen - is there a suitable JTAG Test Command and Data Language (TCDL) capable of covering the following basic requirements:

- Represent** embedded test data (e.g. vectors) efficiently
- Write** to, **Read** from and **Manage** test data stored on the board
- Run** an embedded test
- Configure** and **Validate** an on-board PLD
- Capture** the result of the test and **Compare** with the expected result
- Log** test execution details (time, date, result, etc)
- Send** specific test reports and service logs to the Test Manager.



Purpose of the Meeting (3)

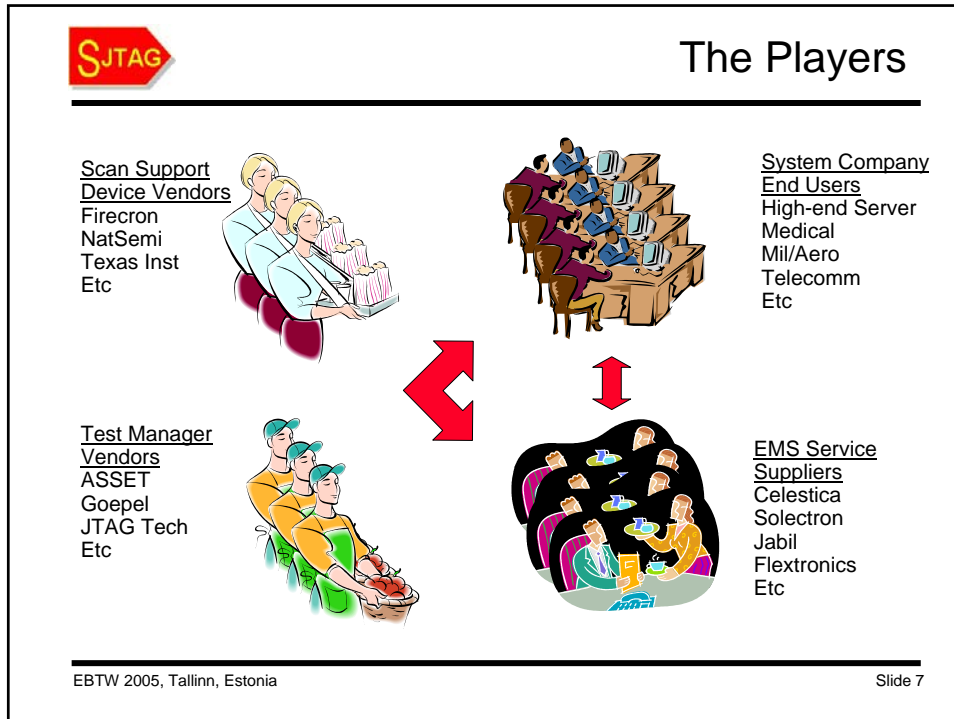
- An initial survey has suggested that the answer is “no” but that there is sufficient interest amongst several companies to join together to intensify the search and, if necessary, start a process of creating a specification for the language.



Initial Presentations, From ...

- Gunnar Carlsson, Ericsson
- Brad van Treuren, Lucent Technologies
- Jim Webster, BAE Systems
- Mike Westermeier, ASSET InterTech
- Steve Harrison, Motorola Networks

- See separate presentations



-
- SJTAG**
- ### Notes From The Meeting
- "We do not need new data formats. Use existing formats." Gunnar Carlsson
 - "The biggest need is the ability to return data in a common format supported by the Test Manager Support Vendors." Peter Horwood.
 - "EDIF + CDF is probably powerful enough for future diagnostics." Reg Waller, but see additional note supplied after the meeting.
 - "The language must be driven by maintenance requirements as well as test requirements." Ken Filliter
 - "Lucent's Test Flow Control Language (TFCL) is a proprietary solution which is not available for donation. But can be seen as a working model."
- EBTW 2005, Tallinn, Estonia Slide 8



JESD32 Standard for Chain Definition


- More formally known as **JESD32 Standard for Chain Description File**. This file format is meant describe the connection of arbitrary programmable devices in a serial chain. It is somewhat confused in execution, precariously trying to balance a description of both non-1149.1 and 1149.1 types of serial chains in the same language. It also notably is unable to describe complex boundary scan chain configurations like hierarchical or multi-drop types of architectures

- Taken from the Xilinx web site



Quick Overview of SJTAG (Bill Eklow)

- Software based initiative
- Standardized test vector format and protocol for remote communication
- Optimize reuse of vectors between Test/ATPG platforms
- Optimize transfer to and execution of test vectors on remote systems
- Assumes remote, on-board boundary scan controller




Actions

- Create a white paper describing the exact problem and possible solutions. Authors:
 - Scan Support Device Vendors: Peter Horwood, Ken Filliter
 - Test Manager Vendors: Pete Collins, Mike Westermeier (mwestermeier@asset-intertech.com), Thomas Wenzel
 - System Users: Gunnar Carlsson, Jim Webster (jim.webster@baesystems.com), Steve Harrison (Steve.Harrison@motorola.com)
 - Lead author: Gunnar Carlsson
 - First draft available by 29 July 2005

- Ben: Set up SJTAG meeting at ITC 2005 in Austin, November

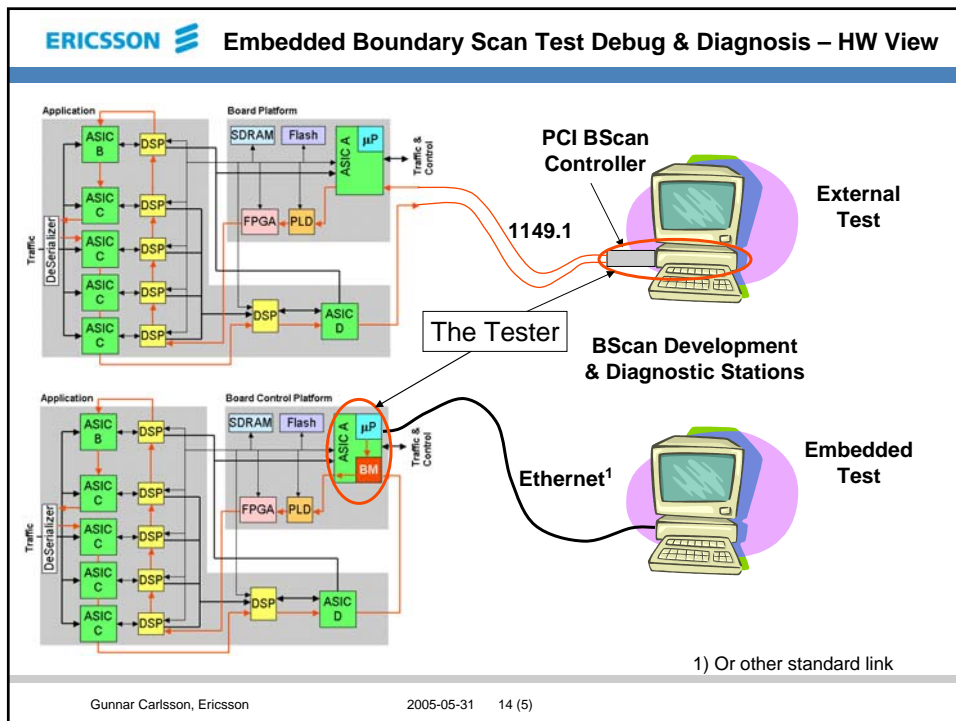
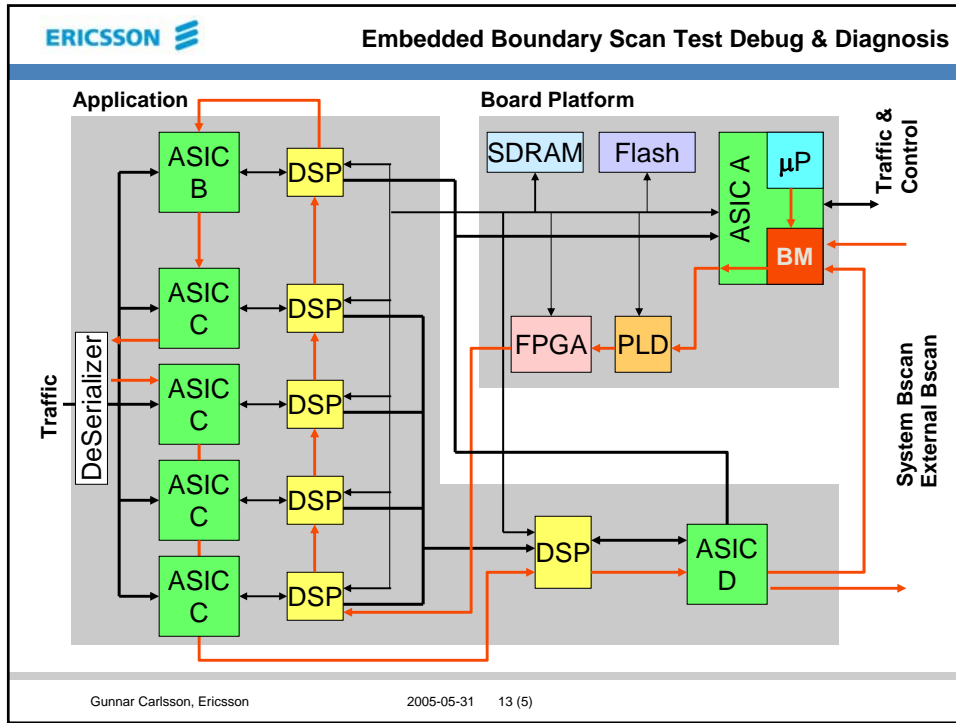
EBTW 2005, Tallinn, EstoniaSlide 11

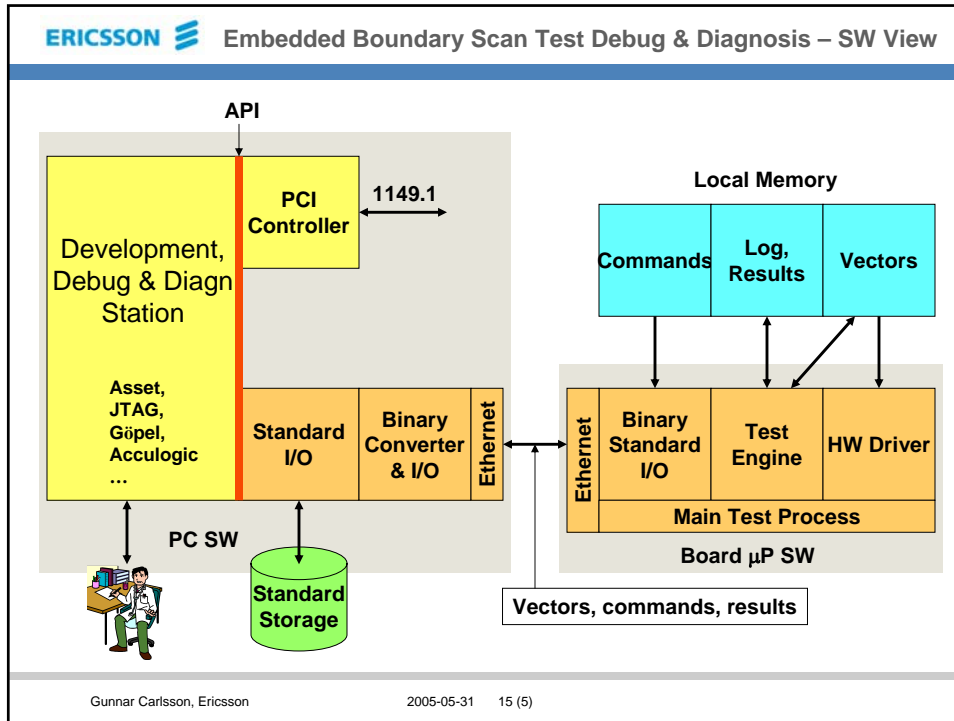


SJTAG Meeting at EBTW

Additional Presentations

EBTW 2005, Tallinn, EstoniaSlide 12





SJTAG – Lucent's Test Flow Control Language

Tallinn, May 2005

Brad Van Treuren – Lucent Technologies

05/23/05 V2.1

Goals of Test Flow Control Language

- Provide test engineers with the control flexibility they have in the manufacturing environment while supporting test in all embedded system test environments where tooling and resources are limited.
- The infrastructure for this language must support various forms/formats of tests as test steps and hardware interfaces as well as be reusable across multiple instances of a unit under test (UUT)
 - Primary focus for test step types (but not limited to) are: Boundary-Scan based tests, including SVF, STAPL, and the TI Addressable Scan Port (ASP) to support multi-drop testing
 - Primary focus for hardware types (but not limited to): any Boundary-Scan TAP controller including the use of General Purpose I/O to drive the TAP as a low cost interface.

Test Flow Control Language, TFCL

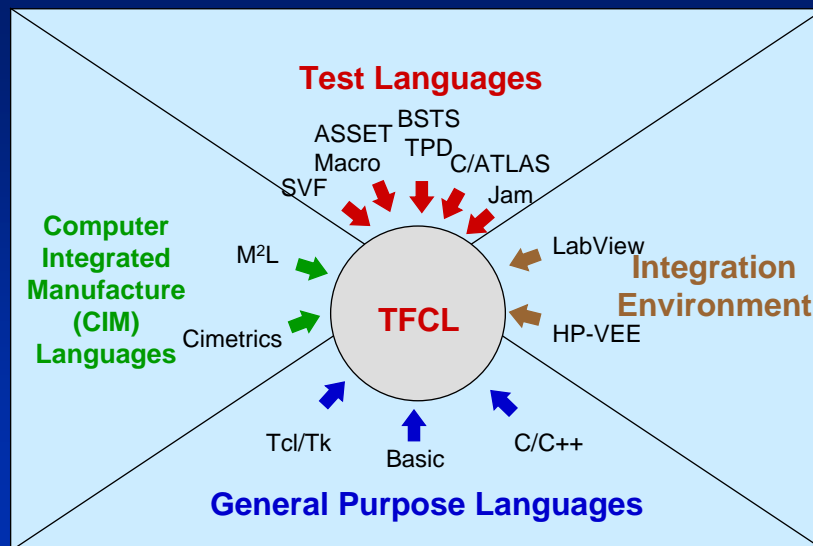
- **The first TCFL applications were developed to support the application of boundary-scan tests across a Lucent multi-drop system test architecture based on the TI Addressable Scan Port gateway device and Lucent's own Boundary Scan Master (BSM) test bus controller. The language is not in the public domain.**
- **TFCL is capable of supporting the following system-level test activities.**
 - Test call up and application: SVF or STAPL formats, structural or functional styles
 - Conditional branching and looping
 - ASP addressing
 - Message display
 - Diagnostic data down to SVF line number and, where available, device pin and board net
 - Synchronization calls
 - In-system configuration of CPLDs and FPGAs with SVF or STAPL configuration data
 - BSM, and BSM2, assembler programs
- **TCFL is written in C++**

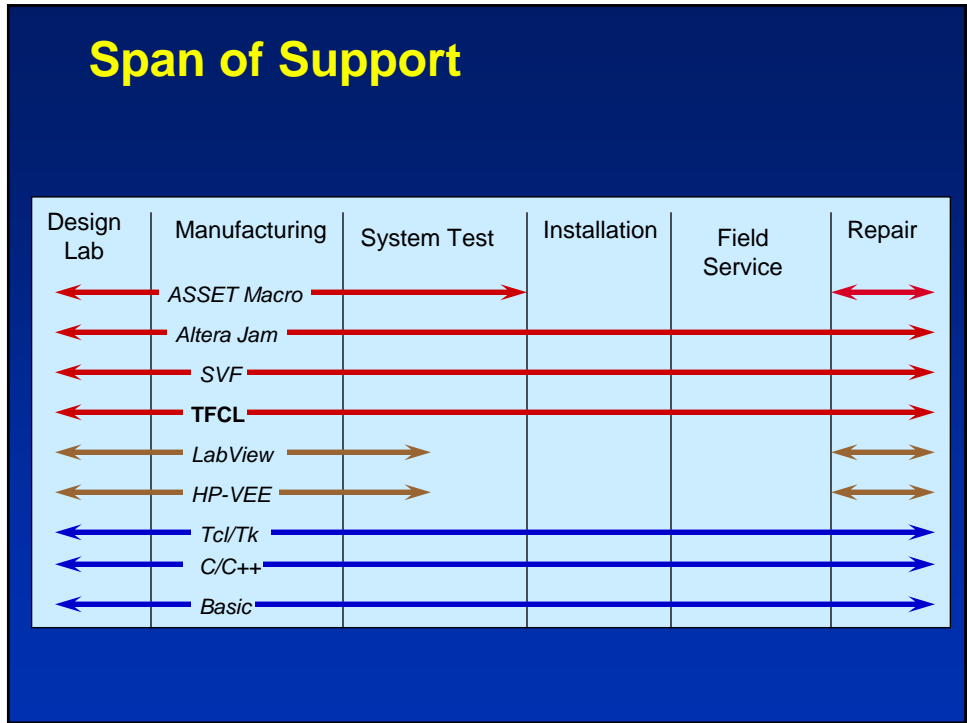
B Van Treuren and J Miranda,
 "Embedded Boundary Scan", Design
 & Test of Computers, March-April
 2003, pp. 20 -24.

Some Additional Comments (1)

- Early investigation for TFCL is summarized by looking at the languages and environments shown in the two upcoming diagrams.
- The BSTS TPD is the AT&T Boundary-Scan Test System: Test Package Description Language we used in our internal tool
- M²L is an automation controller language – AT&T Modular Manufacturing Language.
 - I was one of the developers for M²L. It was used quite extensively in AT&T, a few external companies and several universities for controlling robotic and vision systems in a coordinated manner. The use of abstraction of controller types was quite heavily used in this language as well. However, the method for decoupling was not as elegant as it is in TFCL. M²L was written in C using object oriented techniques prior to the release of C++.
- Computer Integrated Manufacturing (CIM) represents the environment and infrastructure required to perform Just-In-Time manufacturing/assembly.

Test Flow Control Language Background





Simplified Test Language Assessment (1)

- SVF – Excellent language for representing ordered vector data and easily portable to embedded environment, but lacks ability for flow control
- ASSET Macros – Excellent language for representing dynamically ordered vector data, but difficult to port to embedded environment
- Boundary Scan Test System (BSTS) Test Program Description – Provides excellent sequence/flow control capability as well as separation of control from tests, but has too much dependence on the use of a file system availability

Simplified Test Language Assessment (2)

- Common/Abbreviated Test Language for All Systems (C/ATLAS) - Provides excellent sequence/flow control capability as well as separation of control from tests, but requires different commands for each test type and is far too large to support an embedded environment efficiently
- Jam/System Test and Programming Language (STAPL) – Excellent language for representing dynamically ordered vector data that is easily ported to embedded environments, but does not separate out flow of control from test data and only supports Boundary-Scan operations

Some Additional Comments (2)

The key concepts to keep in mind are:

- Separation of control from the tests themselves
- Language independent of test types (might be difficult with the patent we have)
- Simple language that a programmer is not required to use
- More than just GO/NO-GO diagnostics (need device pin and net) is required for full product life cycle beyond the use of smallest field replaceable unit (FRU)
 - Prototype testing (Models), Integration testing, Functional Testing, Environmental Testing, and Repair Depots all require better diagnostics than Field Service that traditionally just needs to know what part to replace.

TFCL Example

```
ENTITY sys
FLOW sys IS
APPLY ASP FROM 0x1 TO 0x1 DIAGNOSE;
IF NOT FAIL THEN
  APPLY SPATH DIAGNOSE VECTORS;
  IF FAIL THEN
    PRINT("Scan path integrity test failed!");
  ELSE
    PRINT("Scan path integrity test passed!");
  APPLY inter1 DIAGNOSE LINE PIN NET;
  IF FAIL THEN
    PRINT("Interconnect test failed!");
  ELSE
    PRINT("All tests passed!");
  END IF;
END IF;
ELSE
  PRINT("Unable to connect to board!");
END IF;
END FLOW;
END ENTITY;
```

Some Additional Comments (3)

- There is more in the D&T article than in the BTW paper on the language
- There are other aspects to the language like exception handling and test step adornments that are not really shown in the article to keep from clouding the issues.
- In reality, our embedded tests are not much different or complex than what you find in the article and the example in this presentation.

Some Additional Comments (4)

- The problem I have with SJTAG, as described in the flyer, is that the language is only a piece of the puzzle.
- The interface into how to invoke the language and manage the tests is just as important if the language will truly be used.
- Take Java for example. Without the specification for the Java Virtual Machine, the language would not have been used by anyone other than the Sun inventors. With a specification about how to run it and build it, the language took off.
 - NOTE: We also looked at Sun's joint proposal with Xilinx for JTAG via Java (Java Bits – Jbits) when it came out after we had TFCL
 - Jbits suffers from the same problems as STAPL – No separation of control from the test. However, separation could come with the support of dynamic linking, but expensive to support in an embedded environment – especially a stand-alone test environment with no operating system present.

Main Reference

- Brad Van Treuren and Jose Miranda, “Embedded Boundary Scan”, Design & Test of Computers, March-April 2003, pp. 20 -24. (This paper is based on paper 2.3 presented at the 2002 Board test Workshop and available at www.dft.co.uk/BTW2002).



SJTAG – Yet Another Standard?

Tallinn, May 2005

Jim Webster - BAE SYSTEMS

04/28/03 V6.1

Lets look at what JTAG has to offer

- **Scan Path Integrity**
- **Interconnect testing**
- **Device programming**
 - **Flash, FPGA, CPLD**
- **Memory device testing (e.g. SDRAM)**
- **Only 5 dedicated signals**
- **Test vector application**
- **IEEE Standard compliance**

What about System Test Requirements?

- **Performance/parametric testing**
- **Functional testing**
- **Real time operation**
- **User interaction/interfaces**
- **Meeting the requirements specification**
- **Software/firmware execution**
- **Built In Test (BIT) requirements**

What can **SJTAG add to System Test?**

- **FPGA / CPLD reconfiguration**
- **Flash Reprogramming**
- **Indirect (board-to-board) backplane testing**
- **In-situ board test (covers on)**
 - **Test vector re-use from board test**
- **but..... this is only in the digital domain**

Bringing JTAG & System test together

- **System Testing offers**
 - Test Management
 - Usually operator interactive
 - Diagnostics to “sub-system” level
 - Test data logging
 - Results processing
- **JTAG offers**
 - Serial test vectors at relatively low speed
 - Diagnostics to net/pin level

What does **SJTAG** need

- **So**
 - A new test language
 - or “hooks” into an existing one?

Test Command and Data Language (TCDL)

- **Features – 2 kinds**
- **Necessary**
 - Run/execute an embedded test
 - Including device re-configuration, flash programming
 - Write/Read test data and results
- **“Nice to have”**
 - Log test data (date/time/results etc.)
 - Send/manage data to Test Manager

Macros or scripts? (1)

- **Macros need to be extensions of chosen language (C++, LabWindows/CVi, Borland Builder, Pascal etc.)**
 - How do existing languages cope with large serial vector manipulation not only for generating them, but more importantly translating the returning vectors into useful diagnostic information?
 - Is the overhead in processing acceptable?
 - Does it slow down the real-time operation?
 - How do we handle parameter passing?

Macros or scripts? (2)

- **Scripts can be language independent**
 - No syntax checking in target application
 - A lot more data handling will be necessary
 - User can use own syntax, so long as the data protocols/information transfers are handled correctly between script and target application language

What are the pitfalls? (1)

- **JTAG is a manufacturing test tool**
 - It tests for “process defects”, not functionality
- **SJTAG may be yet another extension to an already growing standard i.e.**
 - 1149.1, 1149.4, 1149.6, 1532
- **Backplane testing is all very well, but what about mixed vendor systems using different “gateways” on the boards (Firecron, TI, Nat Semi, etc.)?**

What are the pitfalls? (2)

- **Device programming can also be “non-standard” due to customised vendor solutions to get extras such as flash programming speed increases.**
- **JTAG diagnostics usually depend on Netlists/Schematic captures for creating fault lists**

What are the benefits?

- **A standard interface to many differing programming languages**
- **Common data transfers methods to Test Managers**
- **Needs to be independent of the JTAG tool vendors environments**



SJTAG – What’s wrong with STAPL?

Tallinn, May 2005

Mike Westermeier- ASSET InterTech
(ex- Tellabs, STK)

04/28/03 V6.f

Lets look at what STAPL has to offer

To follow-up, I would say the language already exists and it is STAPL *

- If you embed a STAPL player and then use STAPL as your test language than you can:
 - Write, Read and Manage test data
 - Run embedded actions
 - Configure/Validate CPLDs
 - Capture the result of the test and compare
 - Log test execution details
 - Send specific reports

* EIA/JEDEC Standard Test & Programming Language, JESD71, August 1999, available from www.jedec.org

STAPL Deficiencies (1)

- About the only deficiency in STAPL is that the EXPORT command is limited which is why we enhanced it at Tellabs.
- That was the brain child of Don Lillienkrantz and was needed to support a Tellabs unique system implementation.
- Don also leveraged the VECTOR command to support the TI ASP protocol for one of the planned systems.

STAPL Deficiencies (2)

- We also wanted to enhance STAPL to have the ability to import and export files but instead just ran a script to do this as we couldn't get a sponsor for that feature on the JEDEC standard for STAPL.
 - An example of why we wanted this was reading an Intel HEX file for programming FLASH. The script instead grabbed the data and then parsed it into a STAPL array so again we found a work-around.
- We later decided that the work around was sufficient as it kept the embedded player small

STAPL versus SVF

- **STAPL was a huge improvement over SVF as it provides:**
 - Decision Making
 - Looping
 - Array Compression
- **It's a JTAG language not a vector format**

STAPL: Go For It !!

- **So I think if you look at STAPL, you have your SJTAG language.**
- **I know between Don and myself we felt that it was the best choice and at most would need some minor tweaking.**
- **Other good news:**
 - Source code availability for embedded use
 - Most tools support STAPL
 - Documented and released as a JEDEC standard.

Mike



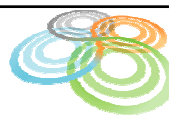
MOTOROLA NETWORKS

System JTAG

25th May Tallinn

Stephen Harrison
steve.harrison@motorola.com

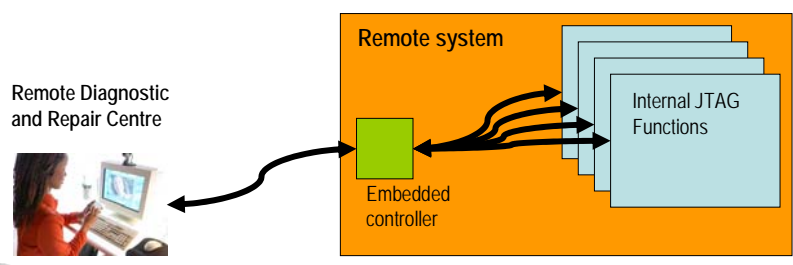
System JTAG



What is System JTAG?

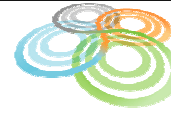
Motorola's perspective of SJTAG is the ability for a remote diagnostic and repair station to access and control a JTAG-enabled system anywhere on a system network.

The accessed system would have an embedded SJTAG controller to access internal JTAG sub-architectures and features.



The diagram illustrates the System JTAG architecture. On the left, a 'Remote Diagnostic and Repair Centre' is shown with a person at a computer workstation. An arrow points from this center to a 'Remote system' box. Inside the 'Remote system' box, there is a green 'Embedded controller' box. From the 'Embedded controller', multiple arrows point to a stack of light blue boxes labeled 'Internal JTAG Functions'. A Motorola logo is visible in the bottom left corner of the slide.

System JTAG



System JTAG solutions are available now.

What is preventing a wider use of this test methodology?

The aim of this presentation is to highlight what Motorola sees as the limiting factors and roadblocks to SJTAG's wider adoption.

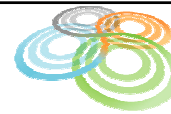
Have we been here before ? Why did IEEE1149.5 fail ?

I suggest the following reasons:

- **An immature JTAG market place**
- **Limited availability of commercial SJTAG support devices**
- **High SJTAG embedded controller cost**
- **Limited commercial SJTAG software support tools**
- **Lack of an adopted and common standard board description and test vector and sequence format.**



System JTAG



An immature JTAG market place

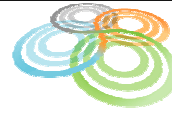
IEEE1149.1 devices and tools and implementation are now main stream and still gaining in momentum.

User are more aware of the possibilities and benefits of JTAG from initial product prototypes through to products and field repair and diagnostics.

After all, that's why were having this discussion !



System JTAG



Limited availability of commercial SJTAG support devices

SJTAG devices and implementations are becoming main stream with new devices and IP being added continually providing cost effective and practical solutions.

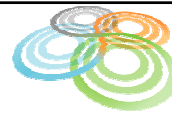
Motorola rejected an in-house bespoke SJTAG solution as it required increased levels of support and maintenance throughout the product life cycle.

A commercially-available SJTAG solution was selected which provided a hardware and software framework to work within with external software and hardware support with wider user base.

This previous limitation is now being removed.



System JTAG



High SJTAG embedded controller cost

FACT : In today's cost sensitive markets, devices that don't contribute to core product functionality will be targeted for removal

- Any SJTAG solutions must be low cost.
- Increased device/solution complexity will increase SJTAG implementation cost.
- Cost reduction comes through simplicity and volume.

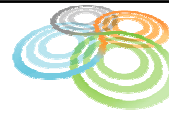
It's critical that vendors have freedom to chose cost-reduced controllers, varying memory size, external interfaces etc., to meet market place requirements and cost expectations.

However a basic core set of functionality should be defined with a superset of instructions for more complex support devices.

Start small to build acceptance.



System JTAG



Limited commercial SJTAG software support tools

JTAG tool vendors historically did not embrace IEEE1149.5 Why?

Will this change with a new SJTAG specification?

Without commercially available tools to manage and support SJTAG it will eventually fail again.

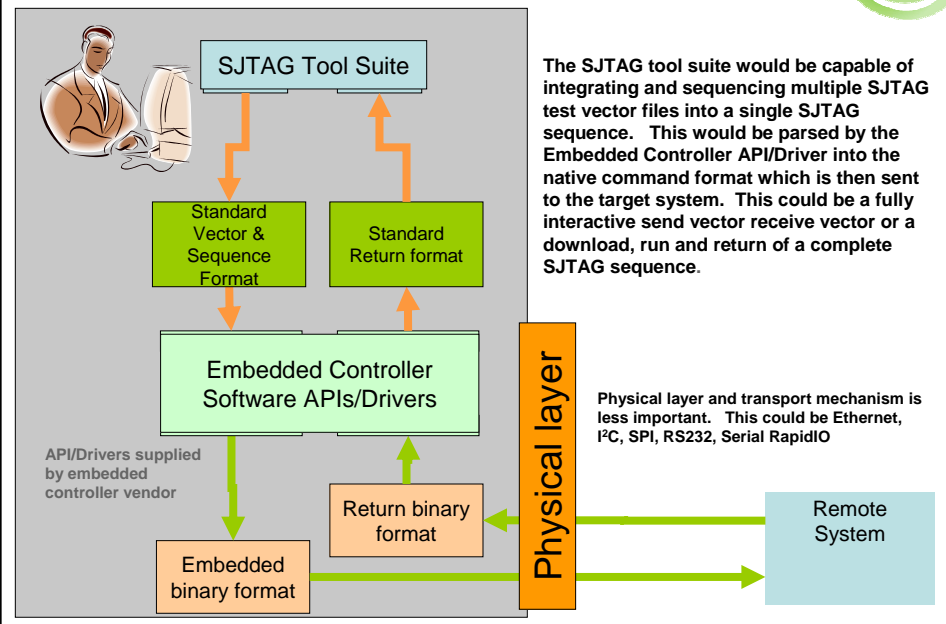
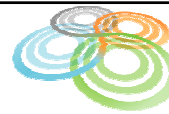
Motorola prefers not to design a bespoke SJTAG software solution with multiple conversion steps and processes. It would rather purchase a commercial off the shelf solution. It sees an opportunity for an enterprising tool vendor to extend its existing product portfolio to fill this requirement.

Motorola sees a value proposition in supplying its customers and support centres with added services after standard hardware delivery, allowing upgraded firmware and software to be enabled, providing increased functionality and remote diagnostics reducing the need for actual site visit.

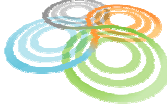


System JTAG

Concept Overview



System JTAG




Standard Requirements

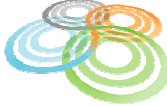
Any SJTAG tools should be built where possible on existing standards.

- Test vector and sequence format - JESD71 (STAPL) ?
- A board description format - EDIF netlist ?
- An agreed scan chain description format - JESD32 (CDF)?
- An agreed test vector return format ?

Can this be achieved ?
Do APTG vendors want to support it ?
End users want and need it !



System JTAG




Standard Requirements

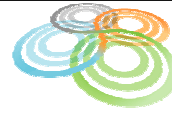
STAPL Standard Test & Programming Language

STAPL

- Is targeted at IEEE1149.1 test & programming
- Has an ASCII & Binary format
- Has sufficient instructions for test sequencing and control
- APTG vendors are already familiar with the format
- Has an already established set of tools available from multiple sources, not just APTG vendors
- Supports JESD32 Scan Chain Description Format
- Has support for embedded integration



System JTAG



Standard Requirements

EDIF net List

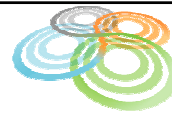
EDIF

- Has all the information required to describe a board's connectivity and parts list
- Common industry format
- APTG vendors are already familiar with the format
- Has an already established set of tools available from multiple sources

Some additional definitions may have to be added to ensure a clear link between IEEE1149.1 devices and their BSDL files e.g. an include file BSDL file statement.



System JTAG



The end user community needs an agreed SJTAG specification/roadmap to move this test methodology forward.

We can't wait years for a new specification.

We have all the main building blocks already.

Test generation is not the issue. Test description, execution and control is the issue.

In essence users need the tool vendors to agree and move forward to establish a common portable test vector description and sequence format. Designing in-house systems/software is not the answer.

**We know what we want, can you supply it ?
It's in our own best interest to grow IEEE1149.x**

