



System JTAG

Supporting eXternal and Embedded Boundary Scan Test (XBST, EBST)

A white paper from the SJTAG Core Group

Version 0.4

1 November 2005

Foreword

At the IEEE European Board Test Workshop held in Tallinn, Estonia, May 2005, a group of 14 board test professionals met to discuss a common set of problems associated with the extended test and configuration use of 1149.1 boundary-scan infrastructure within complex multi-board systems. As a result of this discussion, available at www.dft.co.uk/SJTAG/, it was decided to create a system-level JTAG initiative, called System JTAG (SJTAG). It also was decided to create a white paper to more clearly define the nature of these problems and hence their possible solutions. This is that white paper.

To become involved in SJTAG, send an e-mail to Ben Bennetts, SJTAG Chair, at ben@dft.co.uk.

1. Who's who?

SJTAG Core Group

Ben Bennetts, Bennetts Associates - Chair
Gunnar Carlsson, Ericsson
Pete Collins, JTAG Technologies
Bill Eklow, Cisco Systems
Ken Filliter, National Semiconductor

Steve Harrison, Motorola Networks
Peter Horwood, Firecron
Brad van Treuren, Lucent Technologies
Jim Webster, BAE Systems
Mike Westermeier, ASSET InterTech

Members of the core group are responsible for determining the scope and objectives of the SJTAG initiative and for defining future actions and deliverables. They also are responsible for amendment and general maintenance of the white paper.

SJTAG Extended Group

Adam Ley, ASSET InterTech
Steve Lakin, Motorola Networks

Zefei Yan, Huawei Technologies
... more to come

Members of the extended group will receive notification of any SJTAG activity (updates to the paper, upcoming meetings, etc.) and may, if they want, make comments on the content of the white paper and the general direction of the SJTAG initiative.

To become a member of the extended group, send e-mail to Ben Bennetts at ben@dft.co.uk.

Access to the SJTAG white paper

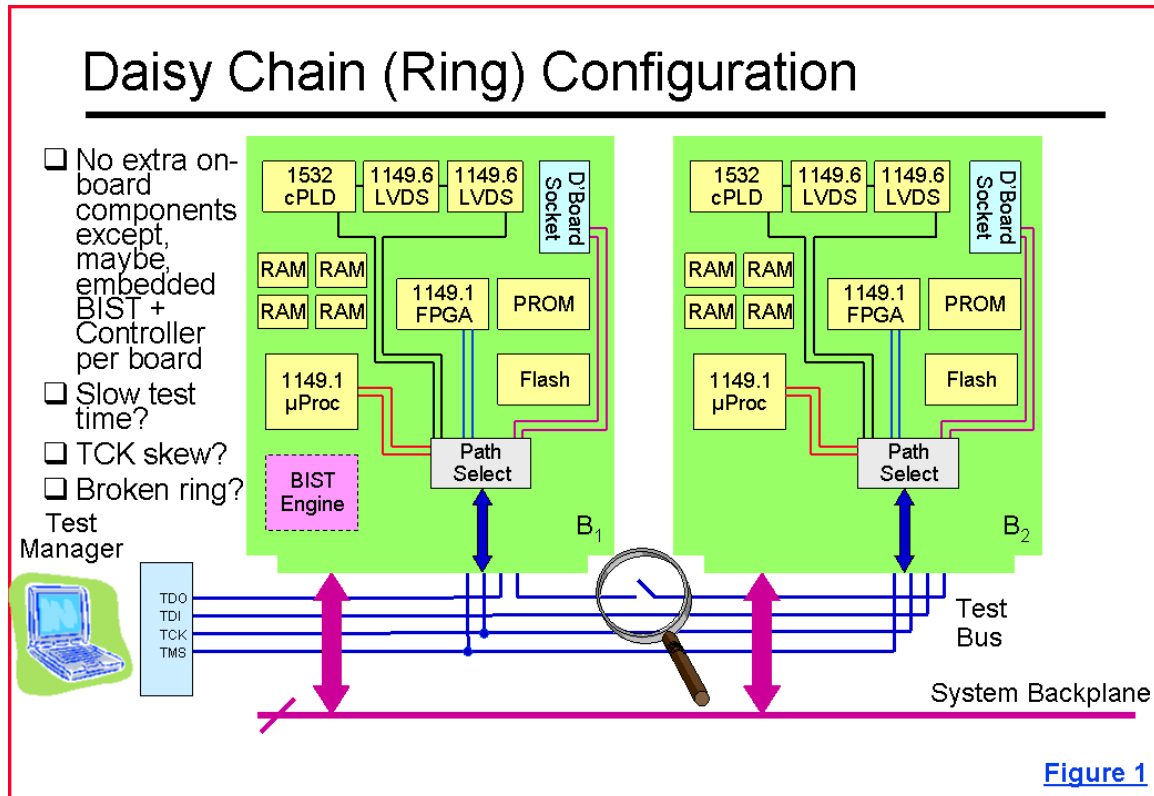
There are no restrictions on the distribution of the SJTAG white paper. To download the latest version, visit www.dft.co.uk/SJTAG/

Comments on this white paper

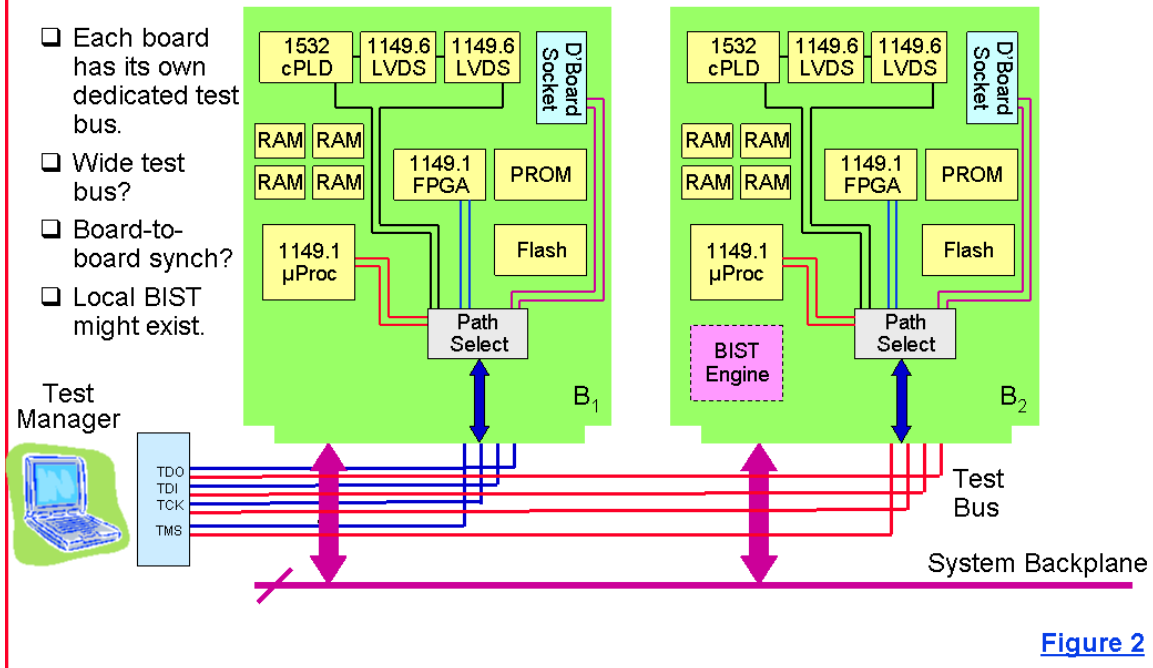
Any comments on this white paper should be sent to ben@dft.co.uk

2. Introduction: Definition of Basic Terms

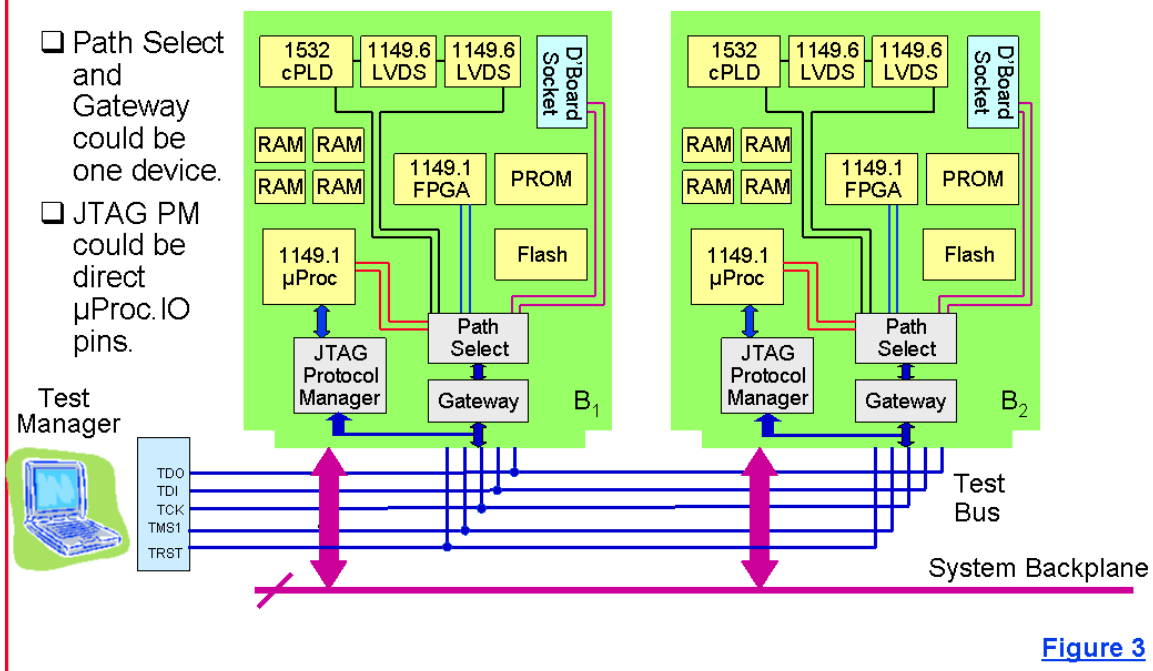
In this white paper, we address the issues associated with test and configuration operations of multi-board systems such as base stations, routers, mainframe computers, etc. using a boundary-scan infrastructure based on IEEE Std 1149.1 [1] (commonly known as "JTAG"). We will assume that the system-level approach to testability can be based on either a **daisy-chain (ring)**, **radial (star)**, or **multi-drop architecture**, as shown below. We also will assume that the backplane test bus, if it exists, can be based on a 4- or 5-wire 1149.1 protocol or on some other bus protocol (e.g., PCI, TCP/IP, etc). There currently is no restriction in this document on the overall style for system testability nor the associated system-level bus used for test and configuration purposes.



Full Radial (Full Star) Configuration



Multi-Drop Architecture (Full-Featured)



We will use the following definitions of commonly-used terms.

Unit-Under-Test (UUT) is used to mean anything that is the current focus of a test or configuration operation. In general, a UUT can be a device, a single board, an assembly of boards, a chassis (or shelf in the language of ATCA [2]), or a field-replaceable unit (FRU) such as an ATCA server blade or shelf management module. In this document we typically will use UUT to mean a single board but other interpretations may fit the context of the discussion.

The word **System** is used to mean the highest level of complexity of the manufactured product (i.e., a complete server or base station). Anything less than the complete system will generally be referred to by its component name (e.g., a server blade).

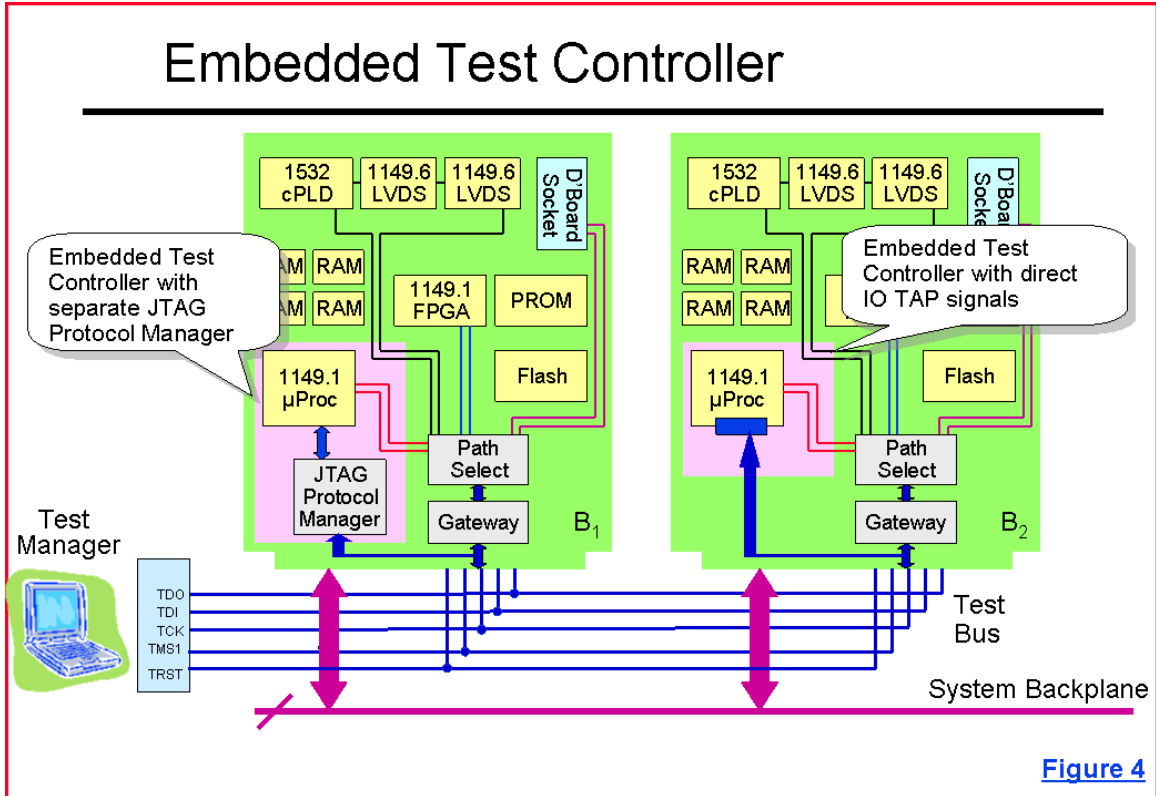
The words **Test** and **Configuration** are generally used in their wider senses. Unless otherwise qualified, “test” could mean functional test, structural test, on-line health test, any use of measurement instrumentation, etc. and “configuration” could mean the loading and re-loading of firmware into on-board RAM or flash or it could mean configuration or re-configuration of on-board CPLDs or the PROMs associated with FPGAs.

Test Manager is the term used to describe any combined hardware/software test control system that is used either as a free-standing off-line test program developer and/or as a free-standing or integral on-line runtime controller for UUT test or configuration operations. Free-standing Test Managers are typically PC-based boundary-scan program-development and test-application controller products that provide the ability to generate tests and apply them to the UUT using an external connection to the UUT.

A runtime-control Test Manager may be external to the system or integrated into the system. It can contain a number of application programs, for example:

- Learning the configuration of the system: what UUTs are present, in what slots, etc.
- Execution control of the application of a number of UUT and system-level test and configuration actions.
- A program for capturing and analysing responses.
- Providing overall executive control of all test and configuration operations.

Additionally a runtime-control Test Manager contains a suitable hardware interface to deliver the low level JTAG protocol signals and receive responses.

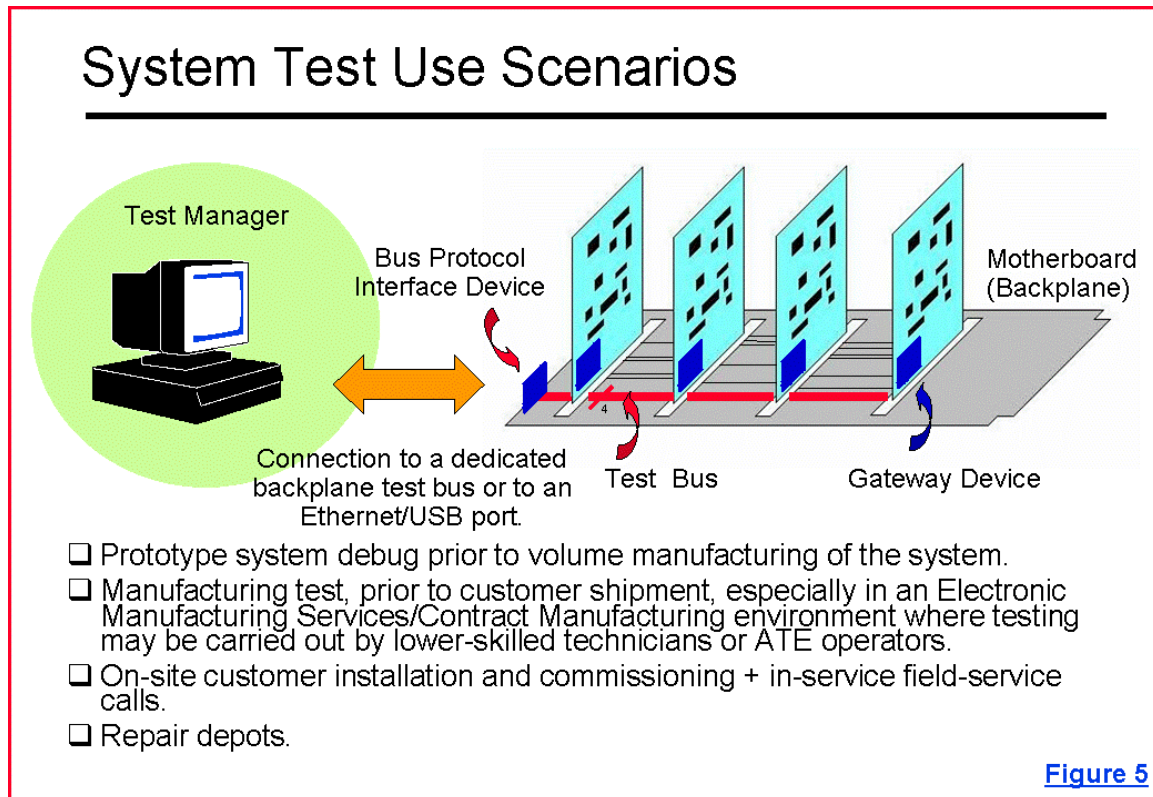


If some, or all of the functionality of a runtime-control Test Manager is built into the UUT, then this is referred to as an **Embedded Test Controller (ETC)**, and defined to be the combination of an on-board host microprocessor plus, if needed, a special device called a **JTAG Protocol Manager (JTAG-PM)**. A **JTAG Protocol Manager** is the hardware interface between the microprocessor and the boundary-scan infrastructure on the board delivering and receiving the low-level JTAG protocol signals and responses.

But note that the JTAG-PM function could be based on a software application running within the host microprocessor where its IOs are interfaced directly to the JTAG TAP signals, rather than a special hardware device.

Finally, individual boards within a multi-drop system will require a unique identifier addressing scheme in order to be properly selected by a Test Manager for the purpose of test and configuration operations. This unique address is often achieved using an on-board **Gateway** device. The purpose of a gateway device is to listen to the address that is broadcast on a backplane bus and to recognise its own address thereby connecting the board to and from the backplane bus. Commercial gateway devices are available from several suppliers.

3. Definition of eXternal Test and Embedded Test



In this section, we will introduce and define the terms **eXternal Test** and **Embedded Test**.

Conventional **eXternal Test** is carried out by applying test stimuli or configuration data to the Unit-Under-Test (UUT) from some external source under the control of an external Test Manager and evaluating the results by observing the response from the UUT and comparing with expected results back at the external Test Manager. The connection between the external runtime-control Test Manager and the UUT can be direct (wired) or remote (wireless) over a communication path such as the Internet.

Embedded Test means that an **Embedded Test Controller** based on an available processor located within the UUT becomes responsible for executing test and configuration operations. The controller uses a UUT-resident application and associated test data for stimulation or configuration of the devices within the UUT and for evaluation or verification of responses from the UUT. A basic device driver is all that is required to support the vector delivery. Test data may be stored remotely and the results may be examined remotely. With embedded test, the UUT is potentially autonomously capable of testing itself (maybe with limited capability of testing the logic involved in the test execution control, such as the processor). Traditionally, embedded test exercises vital functionality and interfaces but, as we will see, embedded test can also be used to apply, and re-apply, structural tests and to control configure and re-configure operations.

Embedded test is typically used in complex systems. The tests may be initiated from built-in UUT sources (BIST), by external signals coming from an external Test Manager or by UUT-external system software. The results may be reported through external indications (e.g., LEDs) or signals, or to UUT external system software.

Embedded test can be used in a variety of use scenarios:

- Prototype-system debug prior to volume manufacturing of the system and its components UUTs.
- Manufacturing test, prior to customer shipment, especially in an Electronic Manufacturing Services/Contract Manufacturing environment where testing may be carried out by lower-skilled technicians or ATE operators.
- On-site customer installation and commissioning.

- In-service field-service calls.
- Repair depots.

Embedded test has value, such as:

- A reduction in time to test – it can be faster than external test, particularly in the field.
- It can be a covers-on test (i.e., no need for system disassembly unless a fault is confirmed).
- There is less need to hook up external test equipment.
- It is possible to obtain a more-accurate diagnosis down to the relevant smallest-replaceable element.

The test invocation may be:

- Triggered by alarms from supervision and monitoring functions.
- Activated at power-on.
- Activated on a regular basis to detect latent faults (sometimes called a health check).
- On-demand testing by remote or local support staff.

Finally, note that care should be taken with Embedded Test:

- It should not be able to be triggered by faults within the system.
- It must not leave conditions that can adversely affect operation after the test is complete.
- It cannot test Flash memory (corruption of contents) except to perform basic CRC or checksum tests.
- CPLD testing may have to be limited to installed configuration only. Similarly FPGA testing.
- The re-use of single-board manufacturing tests in a systems-test environment will have to cater for the changes to BSDL files to reflect the post configuration state. It is possible that manufacturing tests will use the un-configured state of the programmable device for higher test coverage. Such tests will not carry through as re-usable tests later on once the programmable device has been configured.

4. eXternal Boundary Scan Test (XBST) compared to Embedded Boundary Scan Test (EBST)

The Question



Question: should I have the external Test Manager do all the work and just have “bare-bones” additions to the system, or should I have a “fully-loaded” Embedded Test Controller system and just use the external Test Manager as an overall test and configuration controller?

Figure 6

Earlier traditional system-level embedded test was largely software based and targeted at functional checkout. However, current boundary-scan techniques to support production test of ICs and boards offer opportunities to complement the traditional methods with hardware-implemented test support, making the test more efficient in terms of detection capability, test time and diagnostic support and expanding the capability into the structural test domain: see sidebar below.

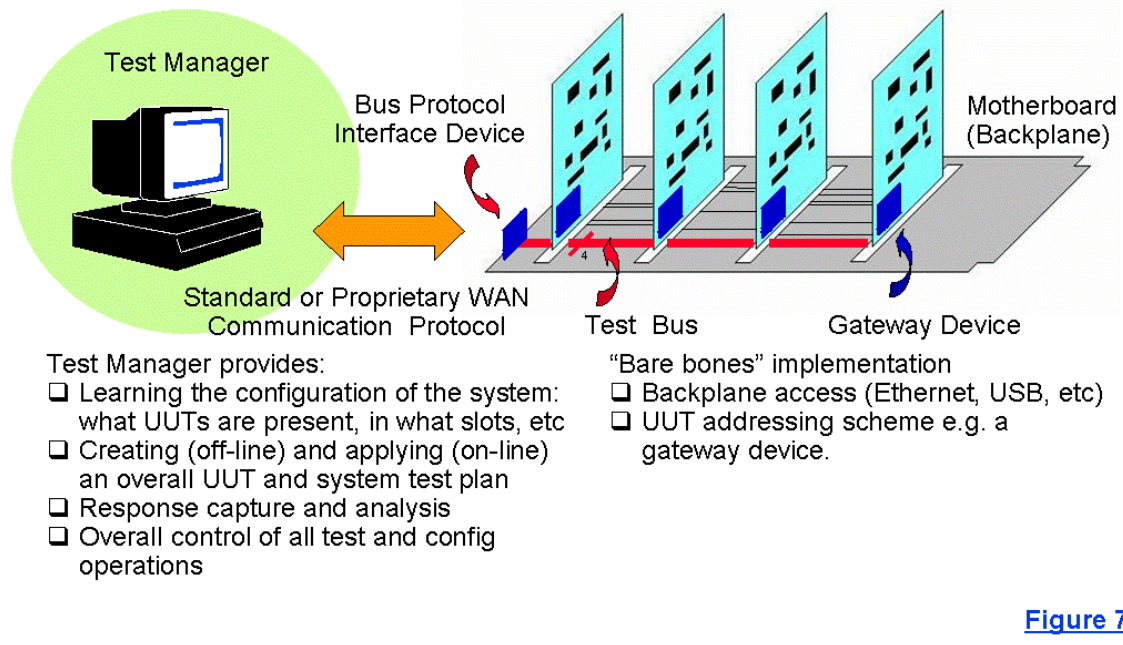
On the complimentary nature of board testing with on-board resources combined with the benefits of a test bus interface for system-level testing.

Ideally, both are needed. The first is for concurrency of testing the UUT (especially at power-up). The latter is for better test coverage (on-board test circuit may be included) and external/alternate to UUT diagnostic reporting mechanism (fail-safe backup for a dead/non-responsive board).

This extended application usually requires some form of a system backplane test bus, which can either be based on an existing system backplane bus such as the I²C bus at the physical layer of the ATCA Intelligent Platform Management Bus (IPMB) or an Ethernet or USB connection, or based on a separate backplane test bus such as a 4- or 5-wire bus based on the IEEE 1149.1 protocol. In addition, extra hardware and firmware is required on the boards, such as:

- Backplane-bus-to-board addressable Gateway devices for multi-drop systems.
- Single-primary-to-multiple-secondary Path Select devices for those UUTs with multiple scan chains.
- On-board JTAG Protocol Manager if low-level drive/sense capability is not provided by the host microprocessor.

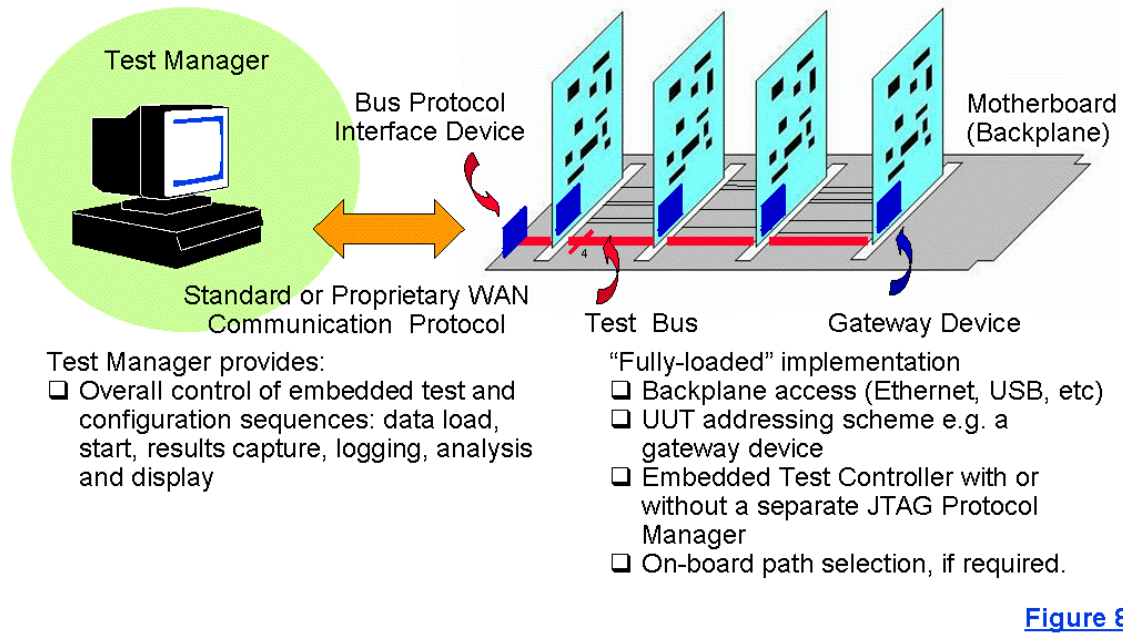
eXternal Boundary Scan Test (XBST)



A “bare-bones” implementation of system-level test using boundary-scan infrastructures would require just some form of backplane access, Ethernet/USB or similar, plus a UUT addressing scheme based on a gateway device per UUT (for multi-drop systems). Everything else would be set up and controlled by an external Test Manager. This scenario, called **eXternal Boundary Scan Test (XBST)** is shown above. The external Test Manager would be responsible for managing some or all of the following test and configuration actions, depending on the use requirements.

- Learning the configuration of the system: what UUTs are present, in what slots, etc.
- Applying the components of a test plan (e.g., for each UUT and, possibly, UUT-to-UUT) including:
 - UUT infrastructure Integrity + Device ID test;
 - UUT interconnect tests, including BS-to-non-BS device interfaces;
 - UUT virtual-access cluster tests: memory, random logic;
 - UUT device tests: RUNBIST, INTEST, INSCAN,
 - UUT In-System Configuration: CPLD, FPGA configuration data, Flash;
 - At-speed test of SerDes LVDS links if TX/RX pairs are equipped with TAP-controlled BIST engines: single UUT and, possibly, UUT-to-UUT;
 - Board-to-board interconnect tests.
- Overall management of all test and configuration operations: data load, start, results capture, logging and display.
- Result analysis support for debugging and diagnosis using system and UUT structural information.

Embedded Boundary Scan Test (EBST)



In the **Embedded Boundary Scan Test (EBST)** scenario, the operations on the board must be controlled from software running on an Embedded Test Controller responsible for executing the embedded tests. In many cases, the management of the boundary-scan protocol has hardware support through a JTAG Protocol Manager device.

In its most extreme form, EBST places the entire test environment into the UUT with only a console type interface to the outside world where an operator (the ultimate external Test Manager?) selects the tests and views the results. Note that this very often leads to sophisticated forms of BIST engines within the UUT to allow at-speed test of on-board components. The BIST engines are controlled and monitored by the UUT's Embedded Test Controller

A less extreme form of EBST is based on a server/client model where the hardware required to run the tests (the JTAG Protocol Manager) and supporting processor + software infrastructure are part of the UUT and a communications interface (TCP/IP or some other fast medium) is used to send packets of lower level commands to the embedded software to perform multiple operations that together constitute a test or configuration operation. The server software resides on an external Test Manager (PC-based or possibly a system controller in a shelf) to perform the higher level test management operations. There are some Test Manager vendors that already supply this type of interface and others may be contemplating it. This approach is sometimes referred to as the scan proxy interface and is often the approach used by companies at the beginning of embedded test facilitation.

Embedded Boundary Scan Test (EBST) is typically split into

- System test application and diagnostic software that is responsible for managing the validation of the board state for a test, the execution of the test, and reporting/logging the results of a test.
- Software for execution flow control, fault management and data logging.
- All the test vectors and configuration data.

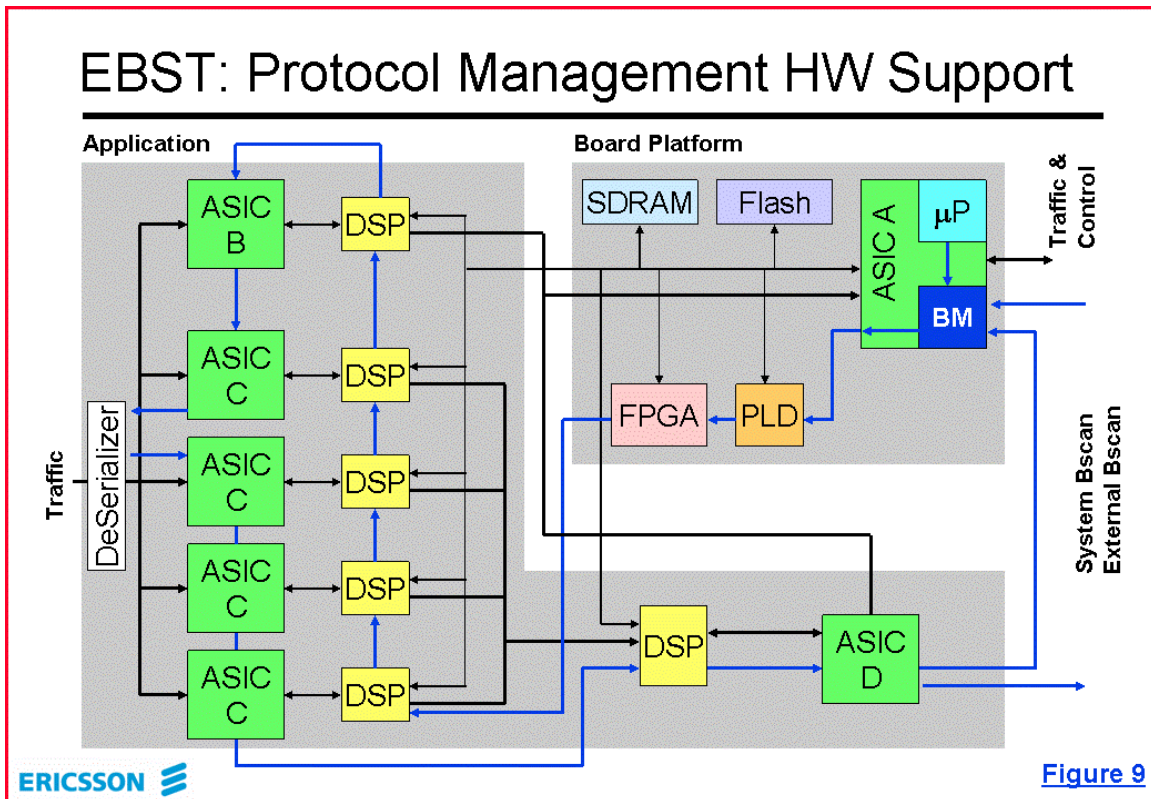
The software for a given processor (and hardware support) may be more or less generic, but the vectors are UUT specific.

In summary, the major difference between XBST and EBST is that in the EBST scenario, the Embedded Test Controller, protocol-management hardware and execution-control software have been moved from the external Test Manager to an internal hardware/software solution.

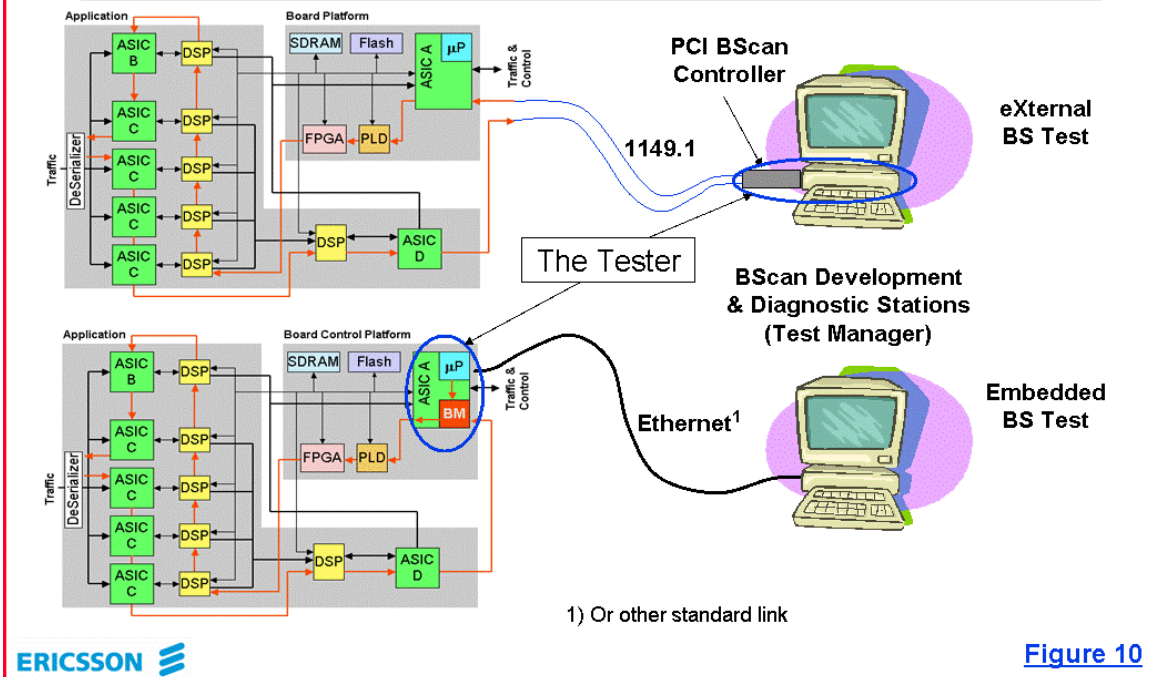
At the other end of the spectrum, the XBST approach has minimal impact on the design of the UUT and requires just a board addressing technique (for multi-drop systems), gateway device or otherwise, plus a communication interface. All test and configuration operations emanate from, are controlled by and terminate at the external Test Manager.

But, as noted, there are many variations between the two extremes of fully-XBST and fully-EBST architectures. **Appendix 1** shows the variants and comments on their pros and cons.

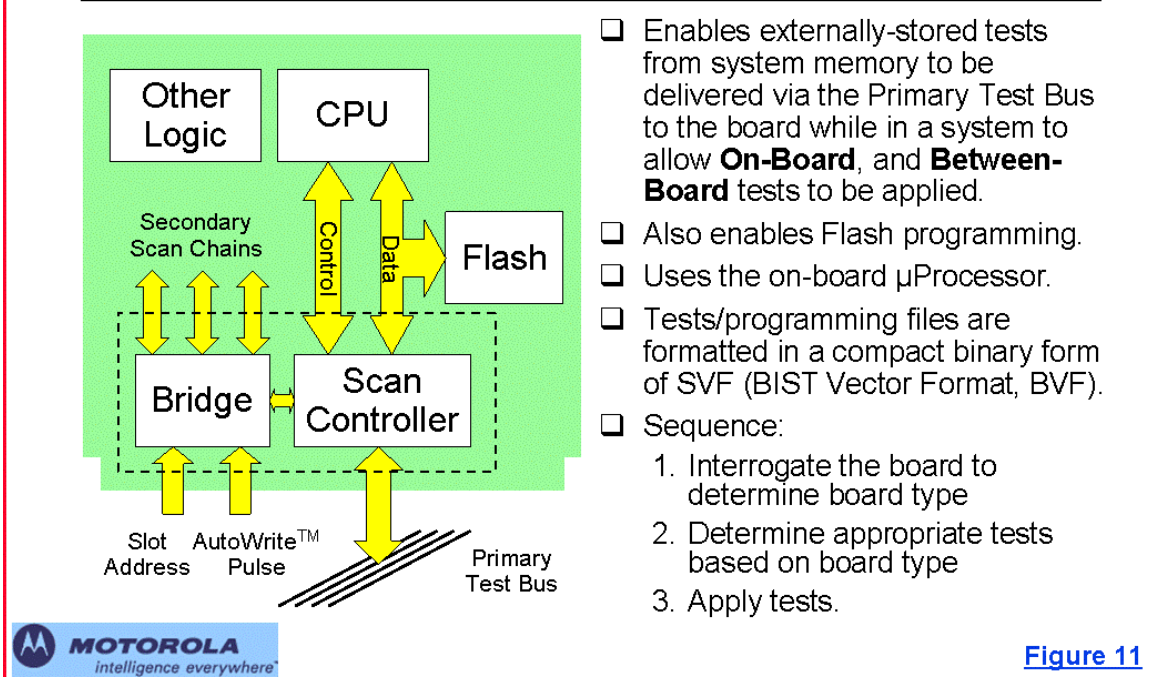
To illustrate the EBST scenarios, here are two examples taken from the EBTW Tallinn presentations and earlier ITC presentations from Ericsson and Motorola.



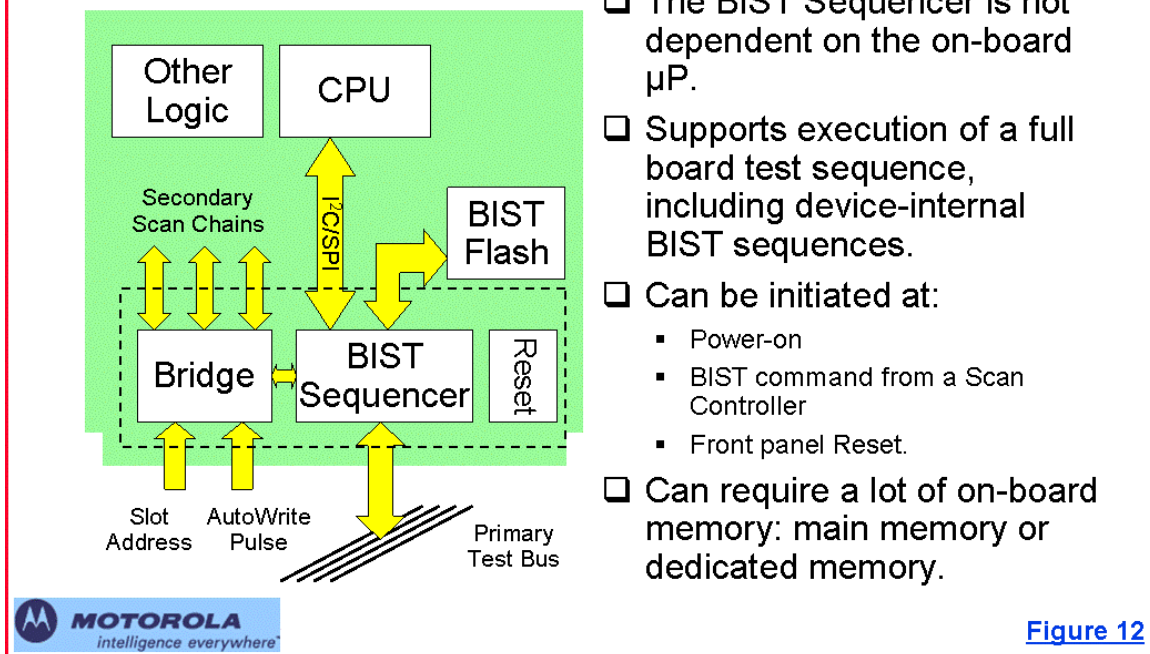
XBST HW Compared with EBST HW



EBST Scan Controller



EBST BIST Sequencer



- ❑ The BIST Sequencer is not dependent on the on-board μ P.
- ❑ Supports execution of a full board test sequence, including device-internal BIST sequences.
- ❑ Can be initiated at:
 - Power-on
 - BIST command from a Scan Controller
 - Front panel Reset.
- ❑ Can require a lot of on-board memory: main memory or dedicated memory.

Figure 12

5. With This As Background, What's the Problem?

XBST systems today have reasonable and comprehensive support in terms of:

- External hardware directly controlling the backplane test bus onto the UUT via a UUT connector.
- External execution control software (running on an external Test Manager).
- Automatic vector generation for most structural tests (e.g., integrity, interconnect, etc.) and for creating configuration data streams.
- Versatile graphical user interfaces for vector debug and fault trace at repair.

For EBST, it is possible to use existing Test Manager development platforms to create the vectors, but there are no standard means to support vector application, debug and diagnosis within the Embedded Test Controller.

The software for vector creation, analysis and graphical interfaces, used at debug and diagnosis, could more or less be common in both cases. But in the EBST case, a link between the Test Manager development platform and the Embedded Test Controller is required.

The following two figures illustrate Ericsson's view of the interfaces between an external Test Manager development platform and the associated EBST hardware, and a software view of the debug and diagnosis requirements.

EBST Development Environment

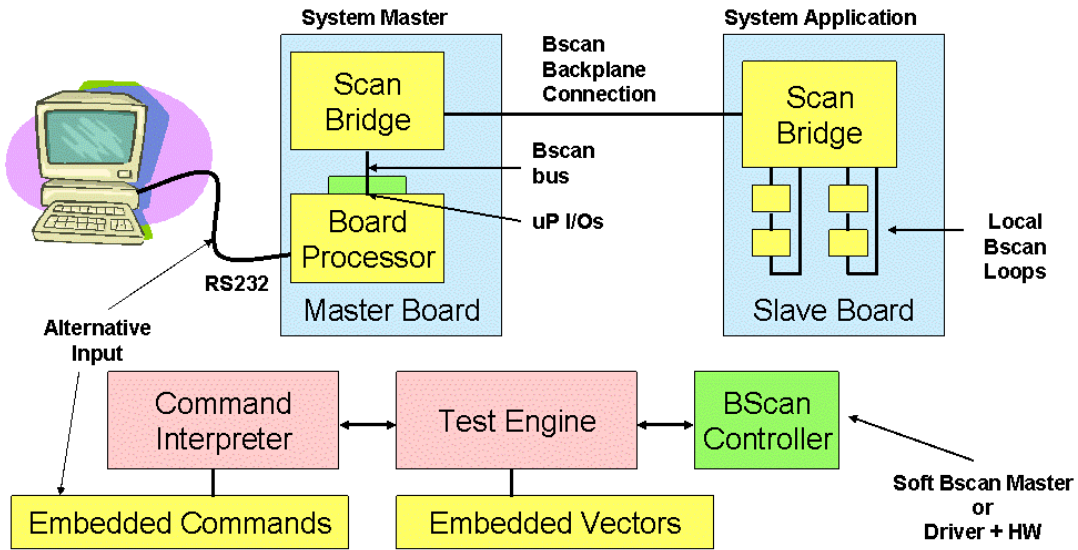


Figure 13

EBST Debug & Diagnosis – Software View

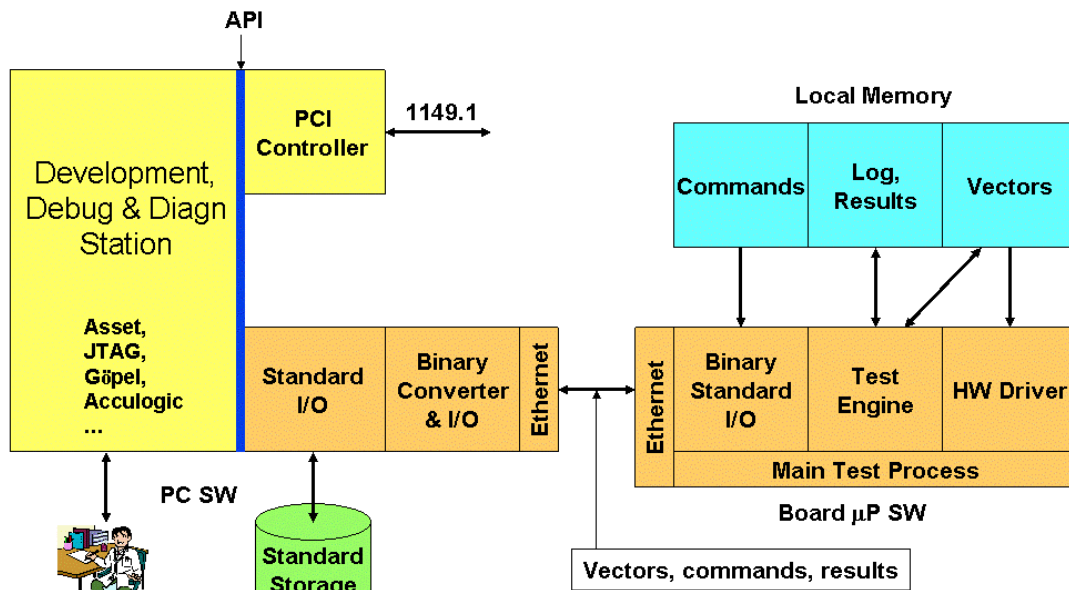


Figure 14

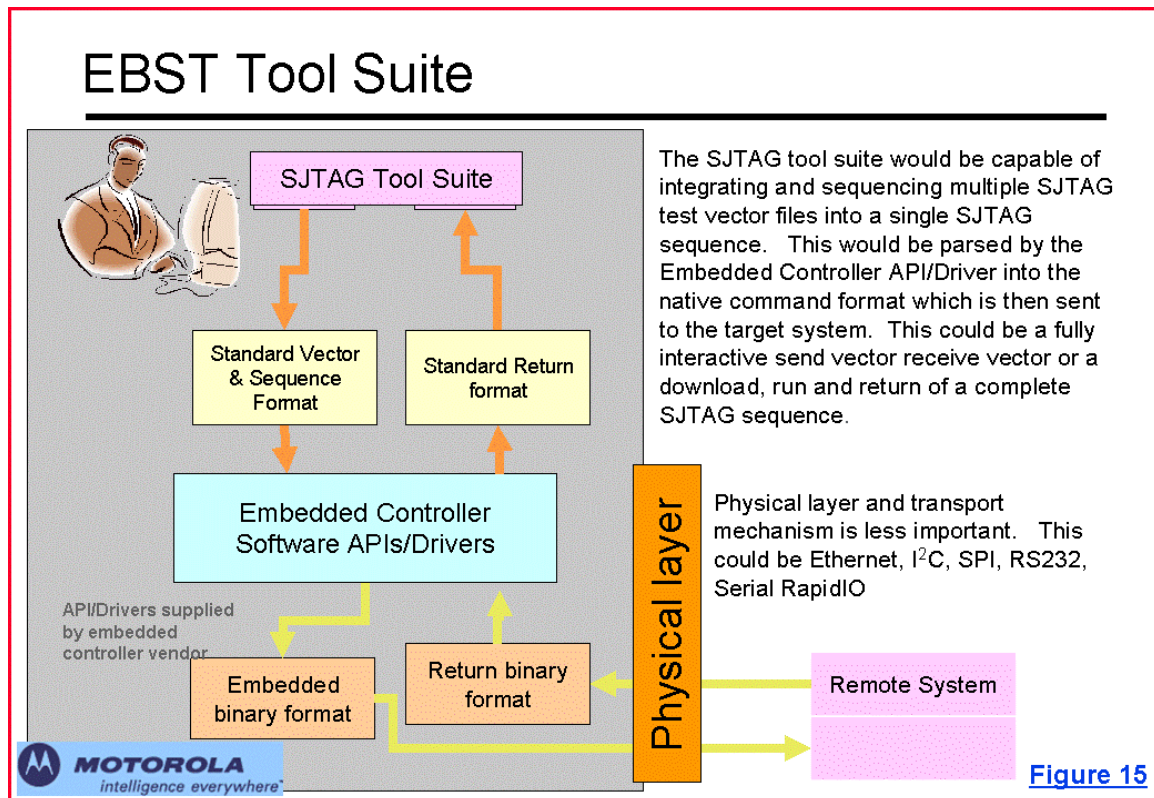
6. Communication between an EBST system and an External Test Manager used as a Development Platform

The EBST may work in two modes of operation

- An autonomous mode, which is the default mode, used in the field.
- An externally controlled mode, allowing external control over the execution flow, transfer of vectors in both directions and transfer of response from the UUT for analysis in the development platform.

Two comments about these modes

- The test execution time may need to be relatively fast if being carried out on a “live” system so as not to adversely affect the overall system performance (possibly an advantage of Embedded over eXternal).
- Care has to be taken here – any EBST test or configuration action needs to ensure that the system is returned either to the same state as before the test execution or to a safe reset state, particularly if being tested remotely and there is no power re-cycle after testing is complete.



The data communicated in the externally controlled mode could be grouped into categories

- Vector and configuration data – SVF or STAPL type (conceptually) of information to be executed in EBST: see sidebar below.
 - The vector data should be in a compact format for efficient storage on the UUT.
- Response data – raw TDO data (e.g., from a failing test)
 - The response data could perhaps be in the same format as the vector data.
- Log and status data – delivered from the EBST runtime software at request
 - Log and status data is simple text.
- Commands – controlling the EBST software operation from the external platform.
- The commands could be grouped into categories:
 - Vector management – writing, reading and deleting vectors, etc.
 - Execution conditions set-up – stop conditions, log conditions, etc.
 - Execution control – run a vector set, a subset, single step vectors, etc.
 - Response data retrieval – get the raw TDO data for external analysis.
 - Log and status data retrieval.

- UUT recovery from a test (hardware reset, power cycle, etc).

Sidebar on SVF and STAPL

Note: there are different levels of control for an application. There are also different requirements for vector storage – static vs. dynamic. There are very strong cases for testing with pure static vectors (Serial Vector Format [3] falls into this category). There are also some other cases for test where only a dynamically created vector is possible to use (what JEDEC's JES71 Standard Test And Programming Language [4], STAPL, may provide as well as many other vector languages from various vendors such as Xilinx JBits, IEEE 1532 ISC data file, etc). There are also combinations of static vectors with control flow decision points for deciding what vector needs to be applied next. Unfortunately, SVF only goes as far as supporting static test vectors. STAPL has some nice features but also falls short in some required cases (especially when sequencing with some external hardware conditions outside JTAG control needs to be done in combination with what needs to be done for a test such as gateway device selection or a test involving a functional code generated pattern with a boundary-scan sampled signature). It is also known that there are at least three different implementations of STAPL that are semantically different. Files that work for one vendor do not work for another. This is in spite of STAPL being a standard. The STAPL standard falls short of defining semantic details. In fact, the SVF specification also suffers in this area, but not as badly.

Structural Information

In conventional Boundary Scan test development platforms, structural information such as UUT and system netlists, BSDL files, and even schematics and layout information is used for vector generation, debug and diagnosis. The same information is needed in the embedded case, and the same interchange formats can be used (e.g., EDIF, BSDL, etc). However, since EBST will probably make extensive use of BIST and other, maybe *ad hoc* DFT functions, more information on such features may be required to better support vector creation and analysis. Such information is not in the scope of SJTAG, but may be handled by another group, the Internal JTAG (IJTAG) group [5]. We are currently exploring the potential relationship between IJTAG and SJTAG.

Note that there may well be an increased number of BSDL files for similar devices where they have different configuration files within a single board or where similar boards are configured differently for different operations within a system. For example processor boards can carry out different functions but may all be of the same design. They may even have different daughter boards fitted but then they would be treated as different board types anyway.

It is not envisioned that any run-time analysis that requires any structural information will be performed directly by the embedded software running on an Embedded Test Controller.

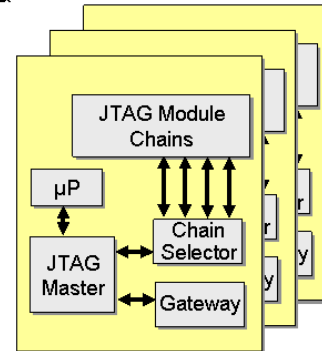
7. The Goal of SJTAG

The Goal of SJTAG



Test Manager

- Vector and configuration data
- Response data
- Log and status data
- Commands
 - Vector management
 - Execution conditions set-up
 - Execution control
 - Response data retrieval
 - Log and status data retrieval
 - UUT recovery from a test



The goal for SJTAG is: for all variants of XBST and EBST, to define the data contents and formats communicated between external Test Manager platforms and internal Embedded Test Controllers, and between ETCs and the UUTs they serve in an open-standard vendor-independent and non-proprietary way.

Figure 16

The goal for the System JTAG effort is: for all variants of XBST and EBST, to define the data contents and formats communicated between external Test Manager platforms and Embedded Test Controllers (ETCs) and between ETCs and the UUTs they serve in an open-standard vendor-independent and non-proprietary way.

The SJTAG Players

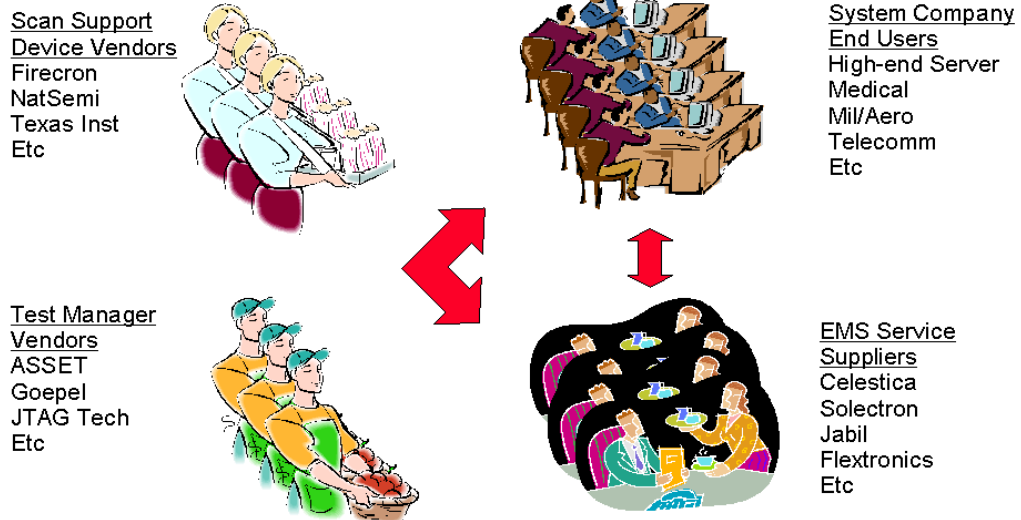


Figure 17

There are multiple players in the SJTAG initiative: scan-support device vendors (JTAG-PM, gateway, and path select devices), Test Manager vendors, system company end-users and the out-sourced board manufacturers. Each player has its own view of the problem and their part in the solution. We have a way to go before creating a solution that satisfies all players.

To Become Involved in SJTAG ...

Send e-mail to Ben Bennetts at ben@dft.co.uk



[Figure 18](#)

8. References

- [1] IEEE Std 1149.1-2001, *Standard Test Access Port and Boundary-Scan Architecture* (available from: standards.ieee.org, www.techstreet.com/info/ieee.html, or global.ihs.com)
- [2] Advanced Telecom Computing Architecture, Short Form Specification, *PICMG_3_0_Shortform.pdf* (available from www.picmg.org)
- [3] Serial Vector Format, supplied and maintained by ASSET InterTech (available from www.asset-intertech.com)
- [4] JEDEC Standard JES71 "Standard Test and Programming Language" (available from www.jedec.org)
- [5] Rearick et al., "IJTAG (Internal JTAG): a step towards a DFT Standard", ITC 2005, P. 32.4

9. Bibliography

Motorola Networks:

- [6] Steve Harrison et al., "Implementation of 1149.1 boundary scan test strategy within a cellular infrastructure production environment", ITC 2000, P2.3
- [7] Steve Harrison et al., "Hierarchical boundary scan: a scan chip-set solution", ITC 2001, P17.2
- [8] Steve Harrison et al., "Board level 1149.1 boundary scan: Built-In Self Test", BTW 2002, P2.1
- [9] Pete Collins, "Design considerations in using 1149.1 as a backplane test bus", BTW 2003, P6.1

BAE Systems

- [10] Clayton Gibbs, "Backplane test bus applications for 1149.1", ITC2003, P. 43.1
Ericsson & Linköpings Universitet
- [11] G. Carlsson, D. Bäckström, E. Larsson, "Remote Boundary-Scan System Test Control for the ATCA Standard", ITC 2005, P32.1

Lucent Technologies

- [12] Robert Barr et al, ITC 2000, P22.2
- [13] Brad Van Treuren et al., IEEE Design & Test of Computers, March 2003, pp. 20-25
- [14] Brad Van Treuren, B. Peterson, J. Miranda, "JTAG-Based Vector and Chain Management for System test", Proc. ITC 2005, P32.1
- [15] Ben Bennetts, "System JTAG: JTAG on the move", Electronic Component News, T&M Supplement, Oct. 2005
- [16] Brian Stearns (National Semiconductor), "Back to the basics: scaling JTAG to meet evolving embedded system needs", July 2005, available from <http://www.embedded.com/showArticle.jhtml?articleID=166401095>
- [17] Dave Bonnett (ASSET InterTech), "Boundary scan goes underground", Test & Measurement World, September 2005, available from <http://www.tmworld.com>
- [18] Reg Waller (ASSET InterTech), "Making moves from board to system test", Electronics Manufacture and Test, June 2005

Appendix 1. Variants of EBST

Appendix: XBST/EBST Variants

- There are at least four major system level options:
 - All Slaves with a Backplane: **eXternal Boundary Scan Test (XBST)**
 - Single Master with a Backplane
 - Multiple Masters with a Backplane
 - Multiple Masters without a Backplane
- } Variants of **Embedded Boundary Scan Test (EBST)**
- Reminder: potential system life-cycle use scenarios
 - Prototype system debug prior to volume manufacturing of the system.
 - Manufacturing test, prior to customer shipment, especially in an Electronic Manufacturing Services/Contract Manufacturing environment where testing is carried out by lower-skilled technicians.
 - On-site customer installation and commissioning.
 - In-service field-service calls.
 - Repair depots.

Figure 19

No Master – All Slaves (XBST)

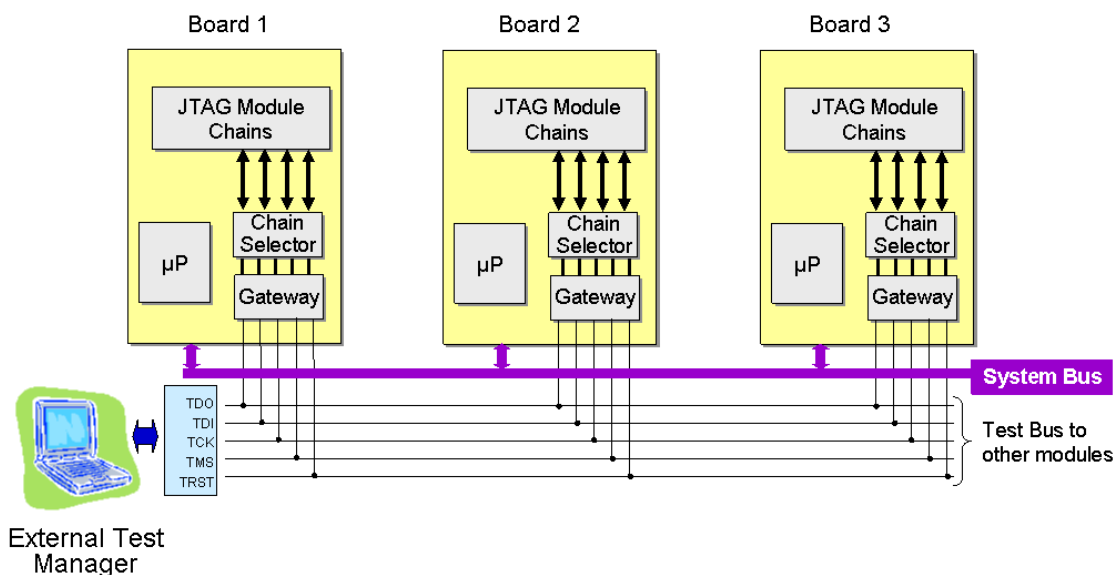


Figure 20

No Master – All Slaves

- External Test Manager provides all operation control, test execution and response analysis.
 - Learning the configuration of the system: what UUTs are present, in what slots, etc
 - Creating (off-line) and applying (on-line) an overall UUT and system test plan.
 - Response capture and data logging, analysis and report.
 - Overall control of all test operations.
- Advantages
 - No system processor resources or coordination with system SW development required.
 - No on-board RAM required for loading and applying tests and for recording responses.
 - Minimal chance of unintended impact on system design.
 - Minimal risk to system development schedule.
 - Supports system commissioning and field service tests..
- Disadvantages
 - “Expensive” external Test Manager required but this can be a shared resource.
 - Slow test data/config data upload/download times.

Figure 21

Single Master – Multiple Slaves (EBST)

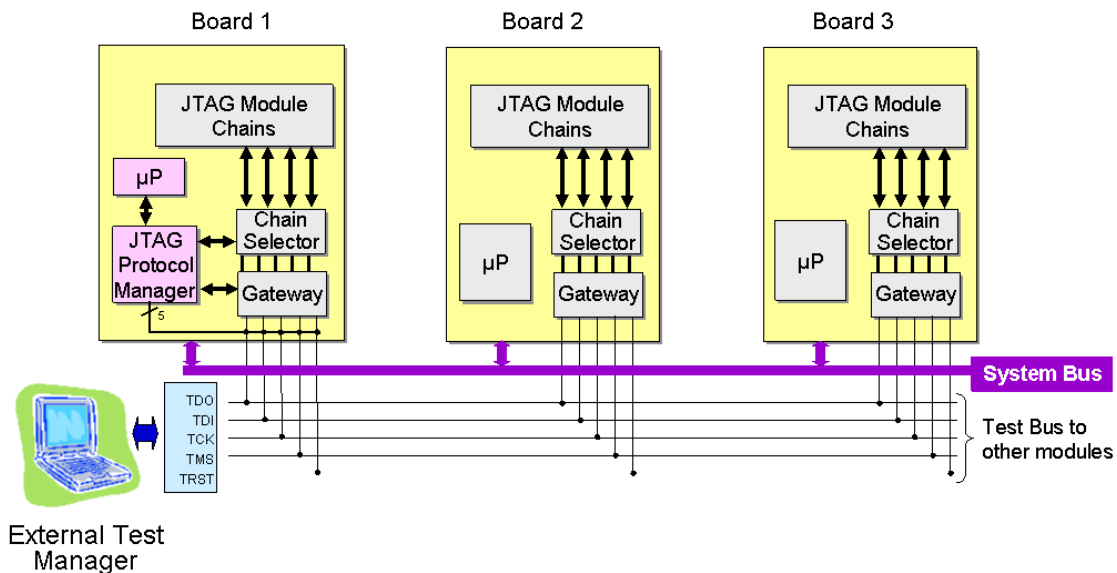


Figure 22

Single Master – Multiple Slaves

□ Single internal Embedded Test Controller, either placed on an operational board or on its own board: three variants

Next 3 slides

□ Advantages

- Simplifies the requirements of the remote Test Manager.
- Supports Customer with ability to initiate remote updates to PLDs, re-run embedded tests and perform real time monitoring.
- Supports system commissioning and field service tests.

□ Disadvantages

- Potential weakness of a single ETC: what happens if the ETC develops a defect?
- Requires coordination with the system software development team.
- Requires on-board storage for test: stimuli and responses.
- Running embedded tests under the control of an embedded test executive can disrupt the normal operating system and leave the system in a potentially dangerous state.

Figure 23

Single Master – Multiple Slave: Variant 1

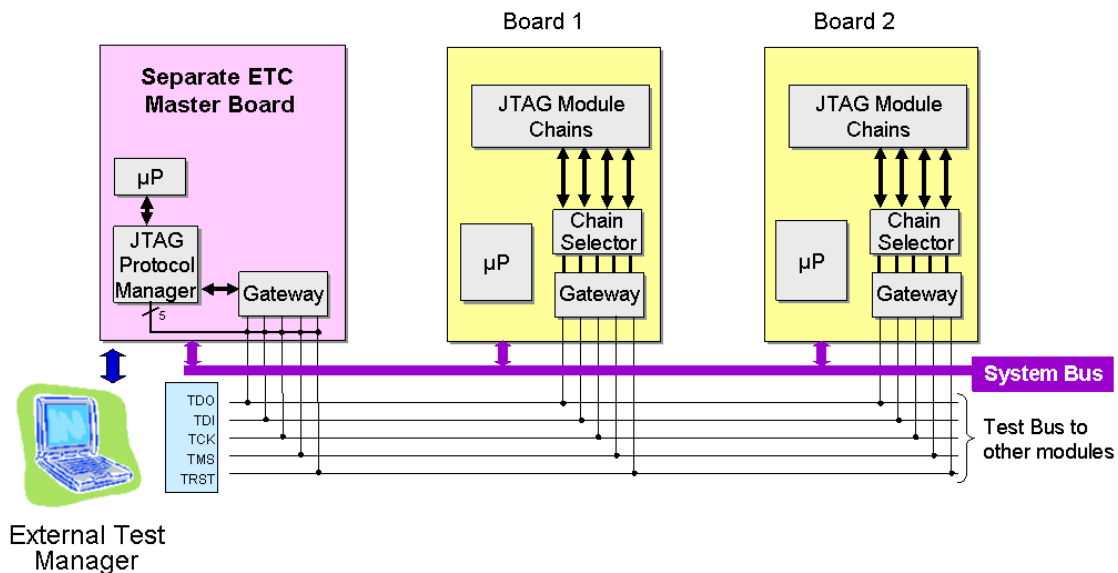


Figure 24

Single Master – Multiple Slave: Variant 2

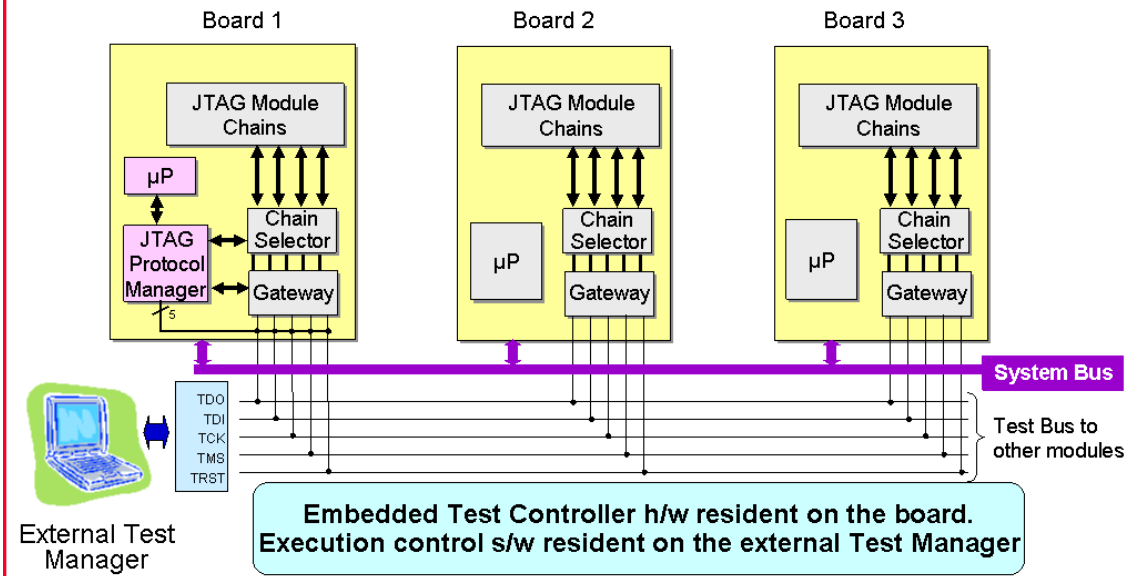


Figure 25

Single Master – Multiple Slave: Variant 3

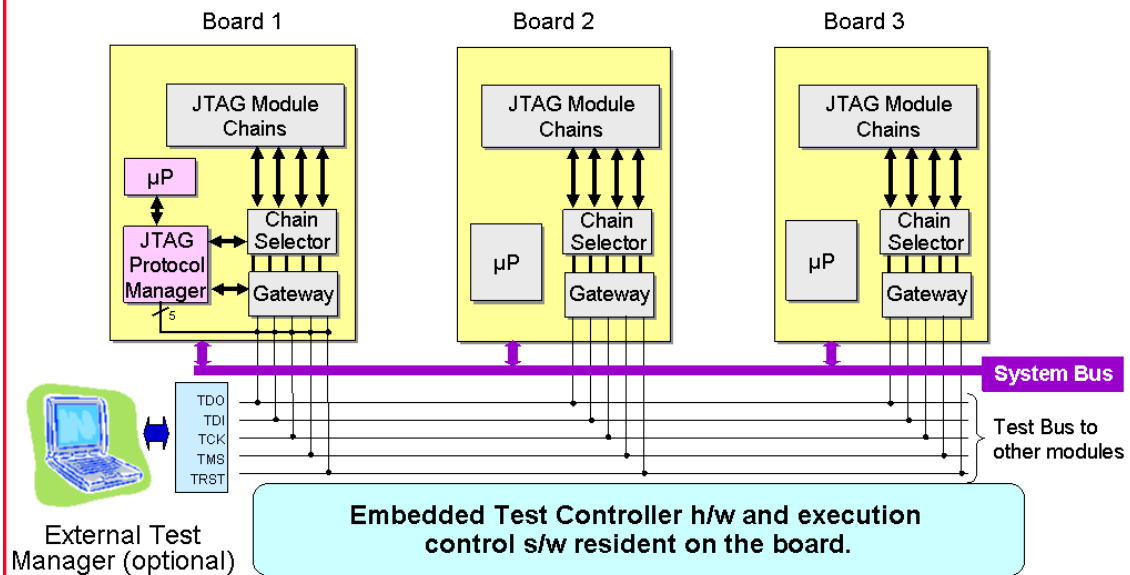


Figure 26

Multiple Masters with a Test Bus (EBST)

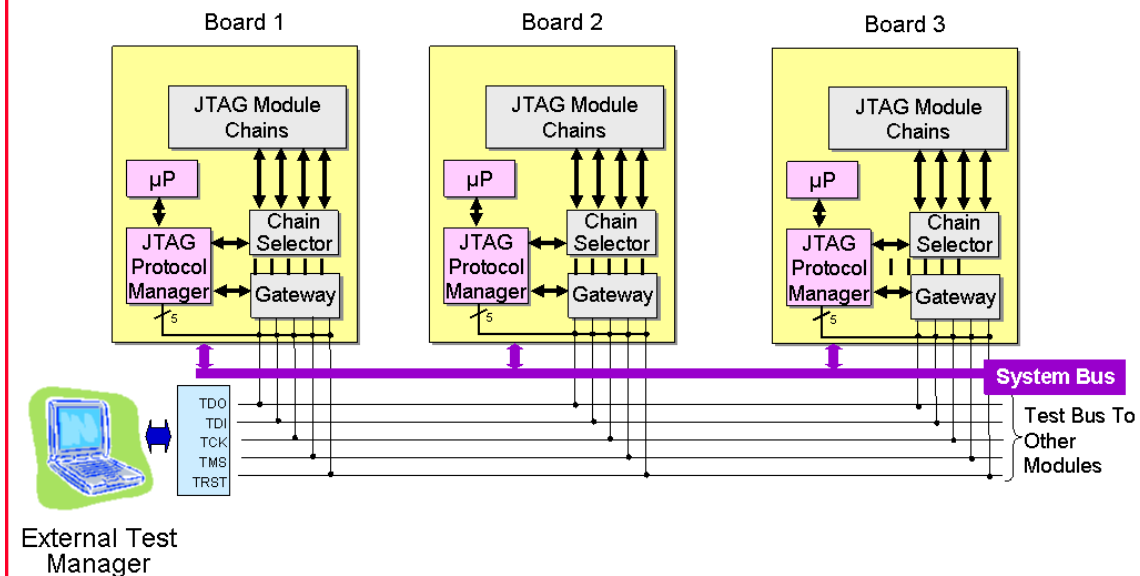


Figure 27

Multiple Masters with a Test Bus

- ❑ Similar to Single Master – Multiple Slave except that each board contains its own Embedded Test Controller for fail-safe redundancy in a mission-critical system.
- ❑ In multiple master cases there is a controller buffer to isolate the ETC from the backplane so that only one ETC is active at a time. This way an ETC can also test itself or its stand-by mate controller.
- ❑ Advantages
 - May permit concurrent testing of the UUT at the board level (useful for Power-On Self Test (POST)).
 - May permit independent testing of UUT from the external Test Manager using the test bus for non-responsive boards.
- ❑ Disadvantages
 - Brings all of the disadvantages of the Single Master – Multiple Slave system plus requires additional control software to handover from one master to another when necessary.

Figure 28

Single Master without a Test Bus (EBST)

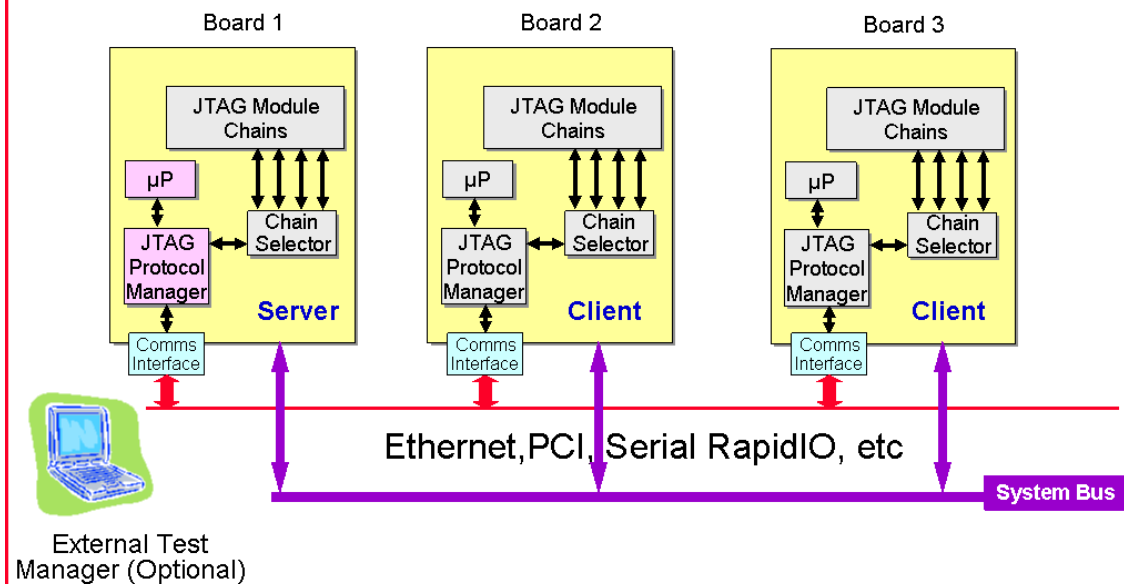


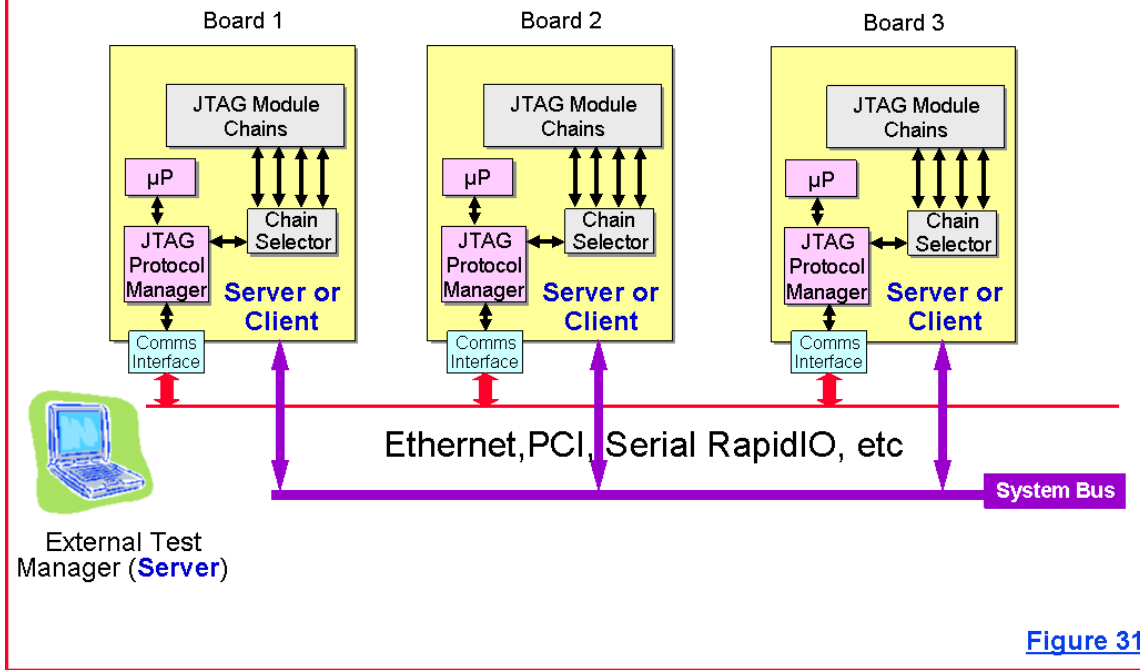
Figure 29

Single Master without a Test Bus

- ❑ The hardware required to run the tests and supporting processor and software are part of one of the UUTs.
 - ETC software server issues commands.
 - ETC software client interpretes the commands.
- ❑ The external Test Manager, if required, communicates across a standard high-speed communications interface e.g. Ethernet, PCIe, Serial RapidIO, etc.
- ❑ Advantages
 - Doesn't require a dedicated backplane test bus or gateway chip on each board (but assumes some UUT addressing facility).
 - Still supports remote monitoring, testing and re-configuration of PLDs.
 - Often a first point of entry into an EBST architecture.
- ❑ Disadvantages
 - Still requires some form of local or remote master controller but this can be very simple and connected into an existing LAN/WAN port.
 - Might be difficult to synchronize between two or more ETCs for, say, board-to-board interconnect tests.

Figure 30

Multiple Masters w/o a Test Bus (EBST)



Multiple Masters without a Test Bus

- ❑ The ultimate arrangement: each board contains its own Embedded Test Controller capable of working independently of any other ETC.
- ❑ External Test Manager may still required for top-level control and with communication access to each board.
- ❑ Advantages
 - Doesn't require a dedicated backplane test bus or gateway chip on each board
 - Still supports remote monitoring, testing and re-configuration of PLDs.
- ❑ Disadvantages.
 - Still requires some form of local or remote master controller but this can be very simple and connected into an existing LAN/WAN port.
 - Might be difficult to synchronize between two or more ETCs for, say, board-to-board interconnect tests.

Figure 32

Finally, a Radial Variant ...

