

IJTAG Scope & Purpose, Challenges & Deliverables

Ben Bennetts, Al Crouch, Jason Doege, Bill Eklow,
Michael Laisne', Ken Posse, Jeff Rearick

IJTAG Proposed Working Group

Presentation Outline

- Scope and Purpose
- Basic Problem Concept
- Three Major Foci
- Four Major Technical Challenges
- Two 900 lb Gorillas
- Deliverables
- Questions

The Current Purpose¹

■ Purpose

Currently, the IEEE 1149.1 standard specifies circuits to be embedded within an IC to support board test, namely the Test Access Port (TAP) and boundary scan registers. In practice, the TAP and TAP controller are being used for other functions well beyond boundary scan in an *ad hoc* manner across the industry to access a wide variety of internal chip test and debug features. The purpose of the IJTAG initiative is to provide an extension to the IEEE 1149 standard specifically aimed at using the TAP to control the configuration and communication to and from on-chip instrumentation.

1. Still subject to modification

The Current Scope¹

■ **Scope**

This standardization effort is intended to address the **access** to on-chip instrumentation, not the instruments themselves. The elements of standardized access include a description language for the characteristics of the instruments, a protocol language for communication with the instruments, and interface methods to the instruments.

1. Still subject to modification

The Definition of Instrument¹

■ **Instrument**

Any on-chip logic for test, debug, diagnosis, characterization or functional use that can be accessed, configured, or communicated with by a TAP Controller.

1. Still subject to modification

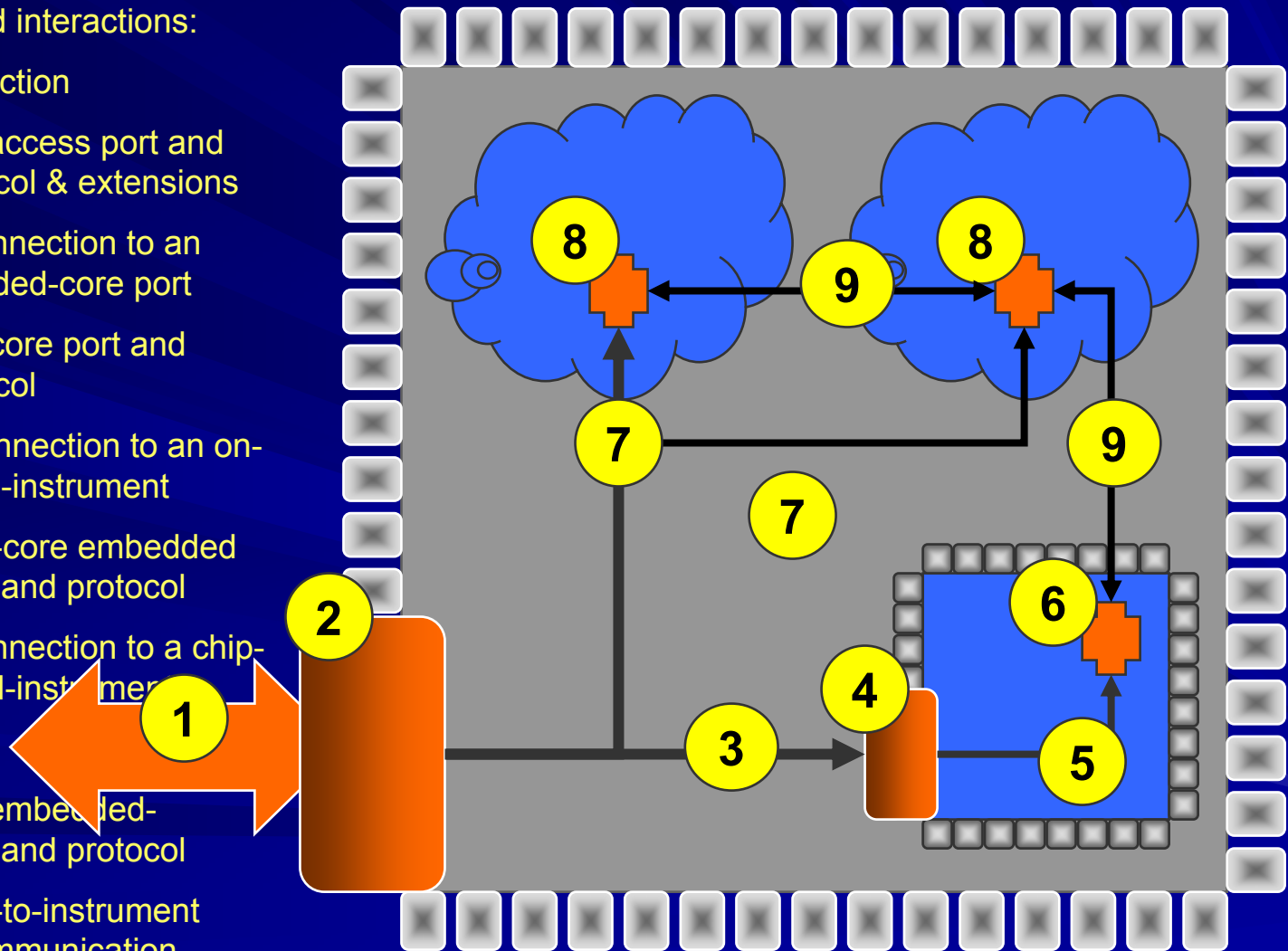
Current Status

- *Face-to-Face at ITC (this meeting)*
- *PAR form submitted and past first hurdle (Rohit Kapur and copyright forms) – passed on to IEEE NESCOM*
- *Core Team almost complete:*
 - Ken Posse (Chair) – Al Crouch (Vice-Chair)*
 - Jeff Rearick (Editor)*
 - Bill Eklow – Mike Laisne' (Contributors)*
 - Ben Bennets (Contributor & English Language Consultant)*
- *Still looking for IC Mfg Core Contributor(s)*
- *Approx. 30 in Extended Reviewer Category*

50K-Foot Problem View

Define ports, protocols, descriptions, interactions for internal chip instruments and interactions:

1. The chip-connection
2. The chip-level access port and operation protocol & extensions
3. The on-chip connection to an Internal embedded-core port
4. An embedded-core port and operation protocol
5. The on-core connection to an on-core embedded-instrument
6. Any internal on-core embedded instrument port and protocol
7. The on-chip connection to a chip-level embedded-instrument
8. Any chip-level embedded-instrument port and protocol
9. Any instrument-to-instrument direct cross-communication



The 3 Major Foci

- **Can all known and proposed DFT/Test Instruments & Architectures be described:**
 - To the extent needed for documentation and automation?
 - With a language and format at the appropriate level?
- **Can the Chip-Level Protocol and connections be defined and described:**
 - To the extent needed for documentation and automation?
 - With a language and format at the appropriate level?
- **Can the mismatches between the operating needs of DFT and Test instruments/architectures, the needs of the ATE, and the limitations of the chip-level interface/protocol and the be resolved?**
 - DFT/Test: Bandwidth, Sequencing, and Synchronization?
 - ATE/Test Platform: Data Rate, Vector Volume, Sequencing, Clock Synchronization, Data Synchronization

The 4 Major Technical Challenges

- **Bandwidth** — *bandwidth includes both chip-level I/O and internal distribution and includes frequency/data-rate, bus-sizing, storage*
- **Sequencing** — *Sequencing includes the “protocol” solutions of enabling, configuring, accessing, controlling, and adjusting, “multiple” sequences, instruments, activities and architectures*
- **Synchronization** — *Synchronization includes the “coordination” and “interoperability” of chip resources and instruments – involves clocks in one sense, instrument communication in another*
- **ATE Interaction** — *includes the “coordination” of chip resources and instruments with ATE/test platform resources and instruments or coordination of JTAG control with operation control*

The 4 Major Technical Challenges

- **Bandwidth:** *includes chip-level I/O, internal storage and distribution – involves frequency/data-rate, bus-size, priority, scaling*
 - *means that there is a wealth of data that must pass between the outside of the chip and the inside of the chip; and*
 - *there is a wealth of data that must be moved around or stored inside of the chip;*
 - *inside the chip, what are the priorities and precedences of delivering the control and data to the various instruments? Can data/control be delivered simultaneously? And is the bandwidth scalable for large vs small vs data hungry instruments?*

The 4 Major Technical Challenges

- **Sequencing:** *includes the “protocol” solutions of enabling, configuring, accessing, controlling, and adjusting, “multiple” sequences, instruments, activities and architectures*
 - *means that the board-level compliant JTAG TAP Controller stipulates or restricts some operations*
 - *are there operations that instruments need that cannot be accomplished with a compliant TAP Controller?*
 - *are there compliant work-around solutions, that are not engineering gymnastics, that can be standardized?*

The 4 Major Technical Challenges

- **Synchronization:** *includes the “coordination” and “interoperability” of chip resources and instruments*
 - *is different than Sequencing and has several meanings*
 - *one meaning is getting instruments and protocols that source from one clock domain to work with the controller or other instruments and protocols from different or other clock domains [Clock Synchronization]*
 - *another meaning is that one instrument, process or architecture needs to do something in coordination with another instrument, process, or architecture – what is it that synchronizes the start, stop, coordination and data transfer involved between any two instruments, processes, or architectures?*

The 4 Major Technical Challenges

- **ATE/Test Platform Synchronization:** *includes the “coordination” of chip resources and instruments with ATE or test platform resources and instruments*
 - *how does the ATE know when it must deliver data or control in conjunction with an embedded function?*
 - *how does the ATE know when to expect response or fail data in conjunction with an embedded function?*
 - *how are multiple clock domains or multiple data rate requirements handled?*
 - *if the ATE drives the parallel pins and a JTAG TAP driver controls the TAP, how do the two separate test systems coordinate?*

The 2 Unavoidable Gorillas

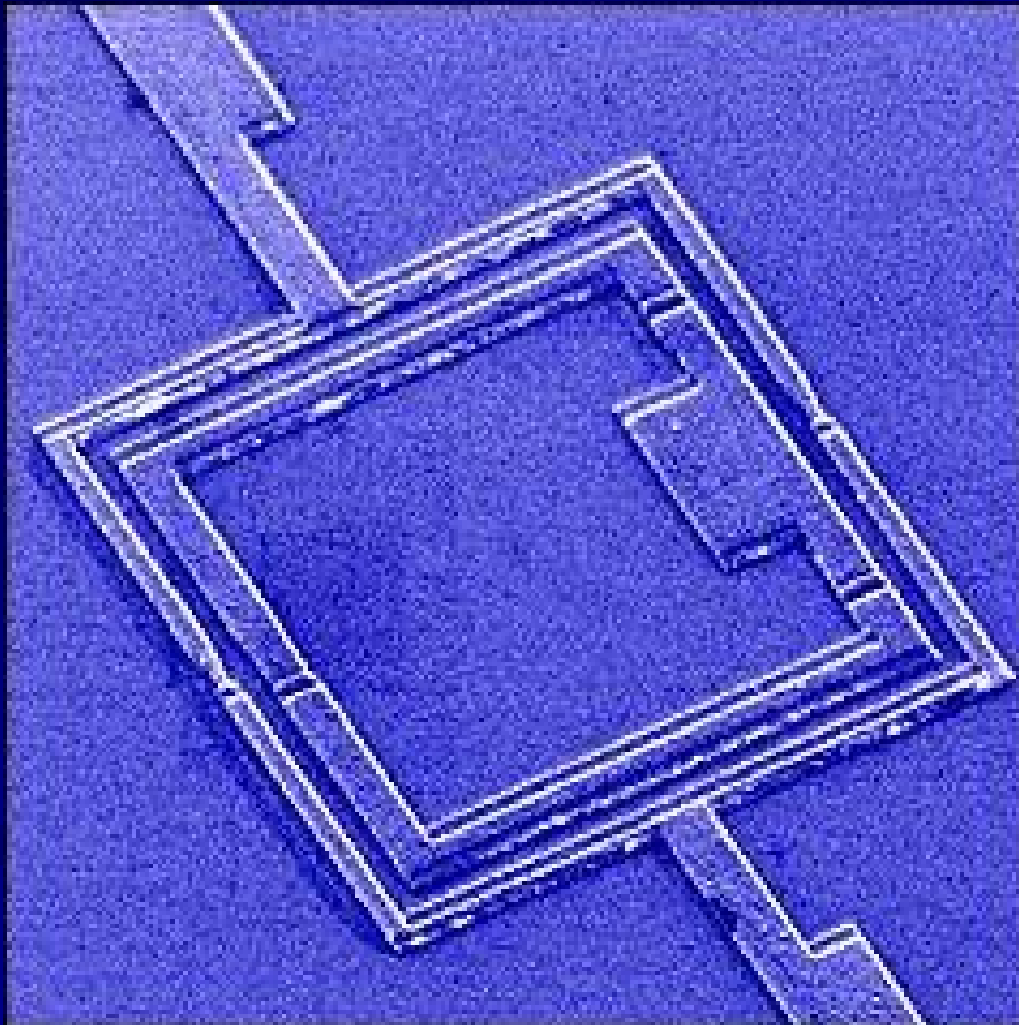
■ Instructions:

- Mandatory? Optional?
- Organization of the Instruction Register

■ Instruments:

- Required hardware?
- Optional hardware?
- Does everything look like Boundary Scan Cells?
- What about complex things like Embedded Logic Analyzers or things with their own State Machines?

The Showing of the Instruments



Bandwidth

- **Bandwidth**: *main possible considerations:*
 - *More than TDI-TDO pair for digital data;*
 - *Different data rates for different functions*
 - *High-speed I/O (e.g. SerDes);*
 - *On-chip data storage within memory elements*

Bandwidth

■ Comments:

“Using JTAG for Test Data is like sucking the ocean dry with a straw!” – Tom Williams at a P1500 meeting

JTAG for Boards may get away with a serial streams, but IJTAG has to deal with MBIST fail data, Scan input-output, Vector Compression & Logic BIST signatures, and Debug Trace data

Sequencing

■ **Sequences:** *main complaints:*

- *“AC Scan” requires multiple shifts and multiple captures, but shiftDR not directly connected to captureDR and captureDR does not loop;*
- *“Parking memory BIST in RTI” when the reality is complex test scheduling requires JTAG TAP activity while MBISTs, LBISTs, etc. are running;*

TAP State Machine

Park in RTI
but need to
install and
coordinate
other actions

2.5 clocks
to get back
to capture and
no loop in
capture

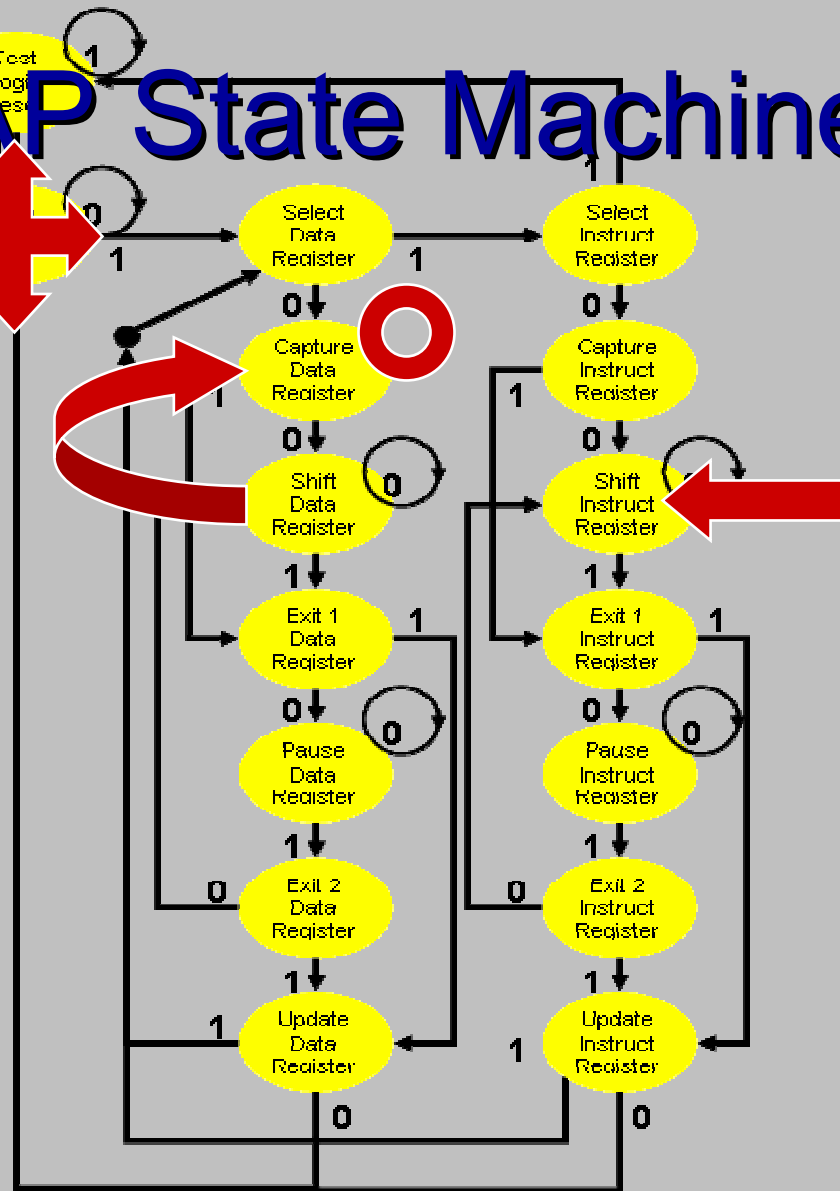


Figure [1]: TAP State Machine

Cross-Communication

- **Cross-Communication:** *also known sometimes as Instrument-Synchronization, is the act of using signals from one instrument, feature, process, or architecture to start, stop, configure, pause, or in some way interact with another instrument, feature, process, or architecture.*
- *An extended case of this is if one instrument cross-communicates with the outside of the chip: a tester, test platform, debug board, emulation board, test instrument, O-scope, or system development setup.*

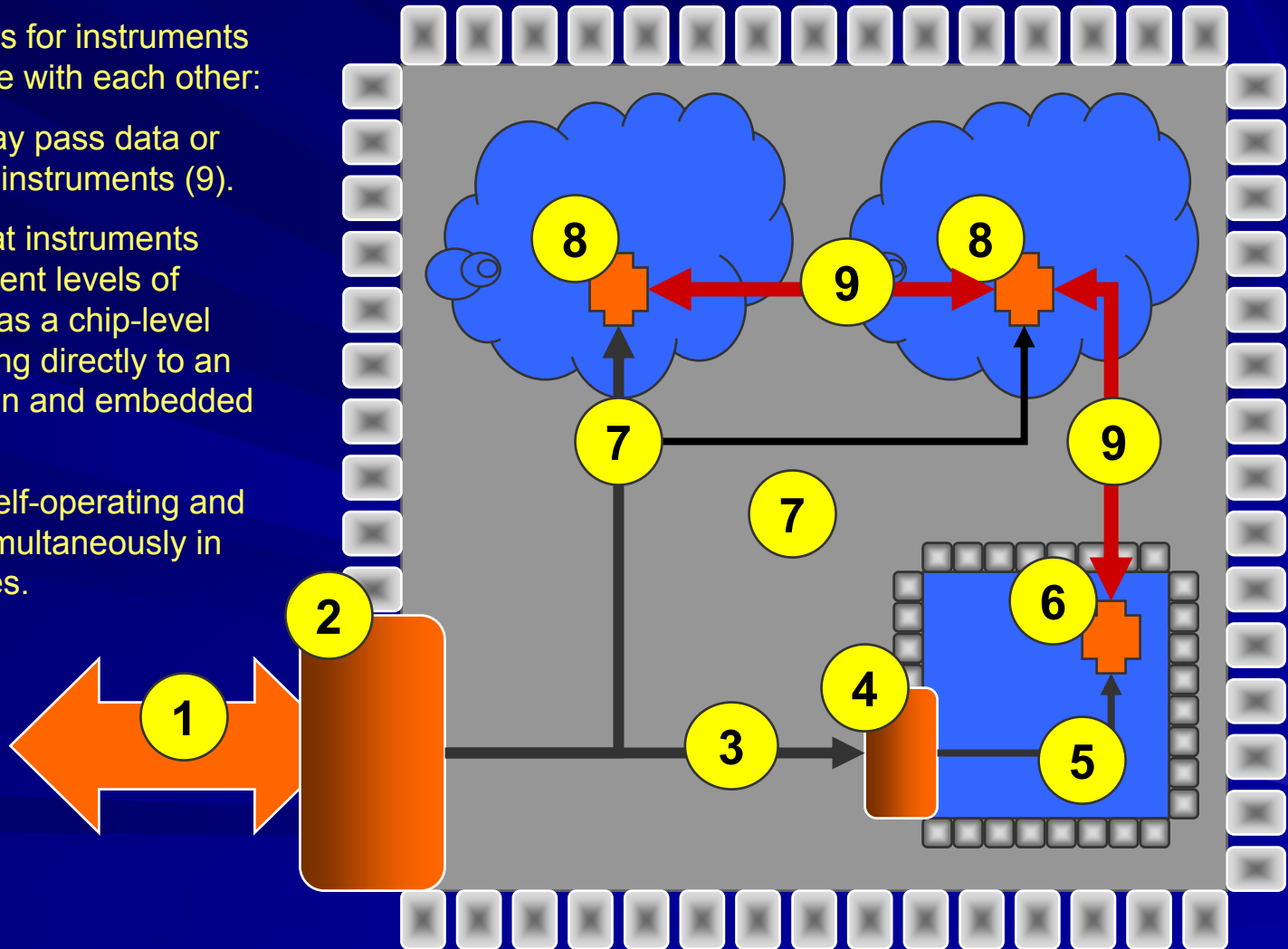
Hierarchy Problem: Cross-Communication

There may be reasons for instruments to communicate with each other:

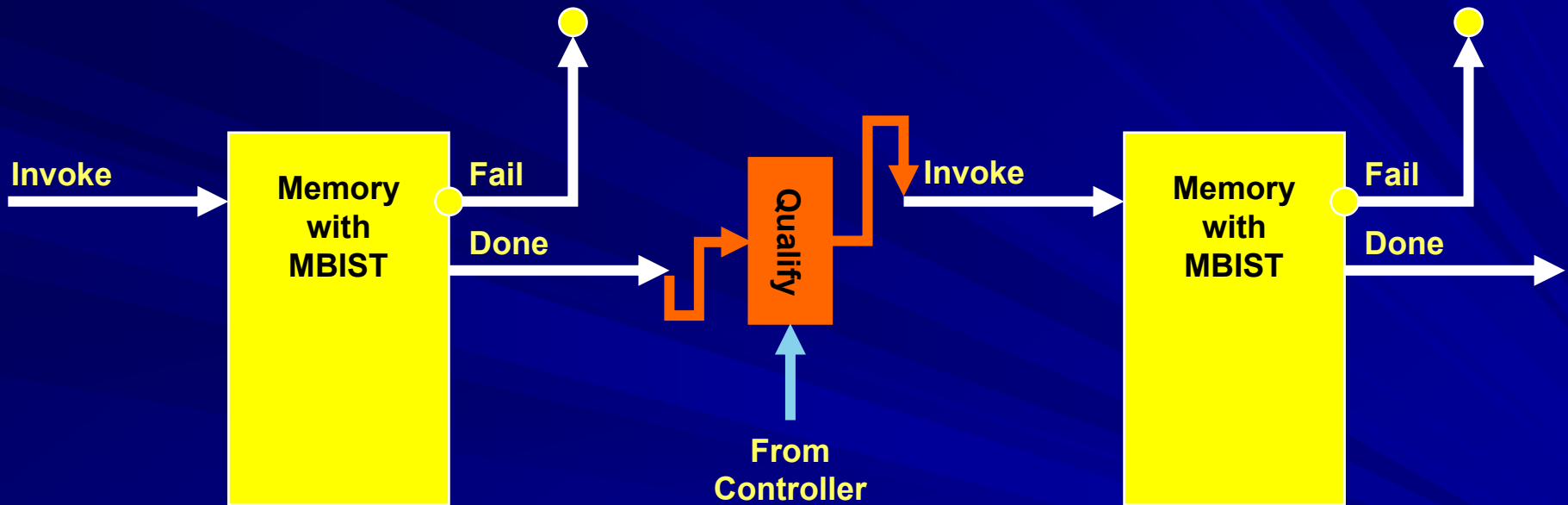
Some instruments may pass data or control to other instruments (9).

Complications are that instruments may be at different levels of hierarchy such as a chip-level instrument talking directly to an instrument within and embedded core.

Instruments can be self-operating and may operate simultaneously in certain instances.



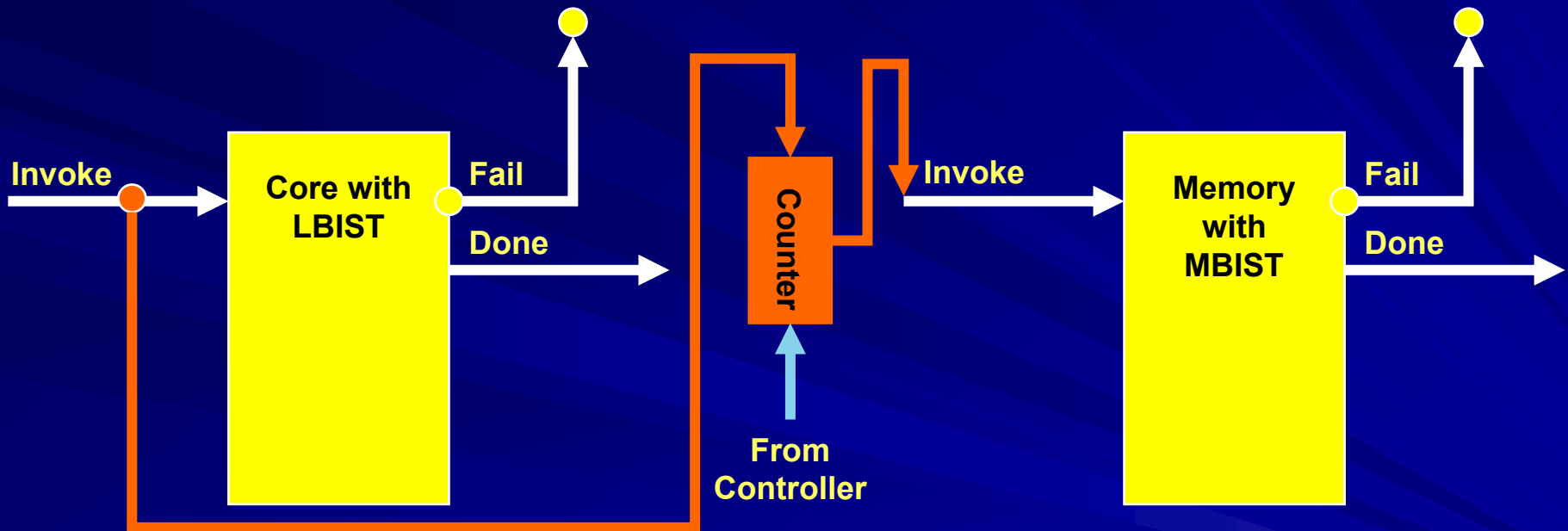
Example Instrument Communication Start on Finish/Fail



- When an MBIST or Bank of MBISTs complete, then this will automatically trigger the next MBIST or Bank of MBISTs...
- ...when enabled by the controller (may be gated by selection of “engineering vs production” or Fail assertion from previous Bank)
- The “Done” signal is the “Trigger”

Example Instrument Communication

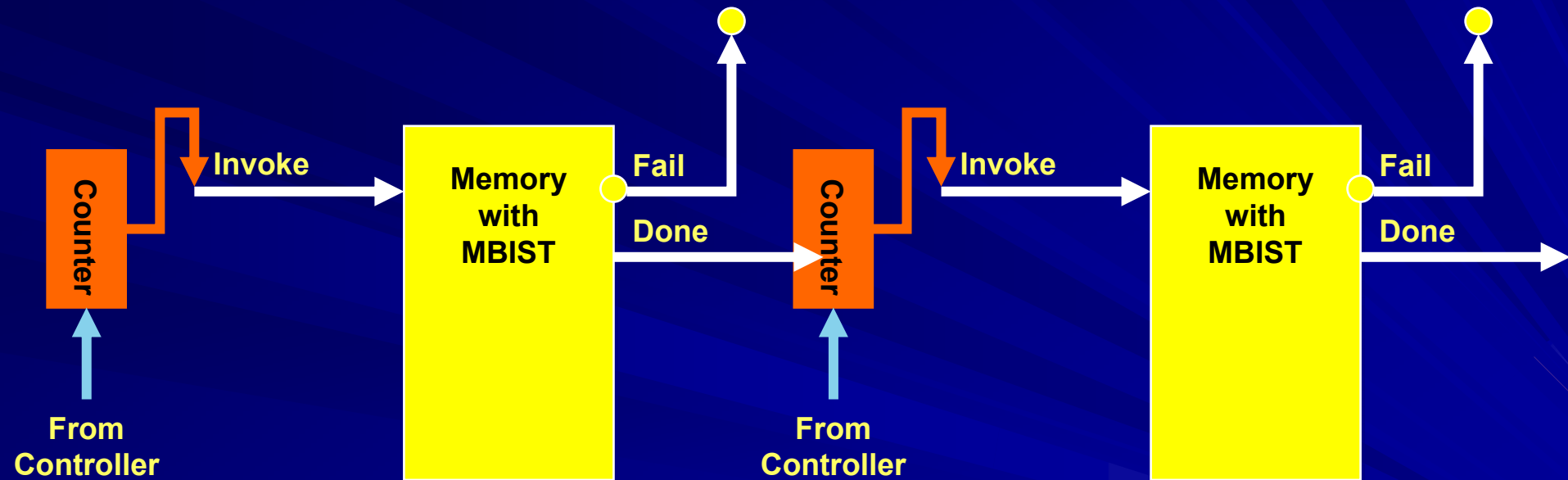
Start on Count



- When a test instrument starts, another instrument can be started after a number of cycles when enabled by the controller...
- ...for example, don't start MBIST during the Scan Shift Test portion of the LBIST; wait until after n-cycles (count installed by controller)
- The "Invoke" signal is the "Trigger"

Example Instrument Communication

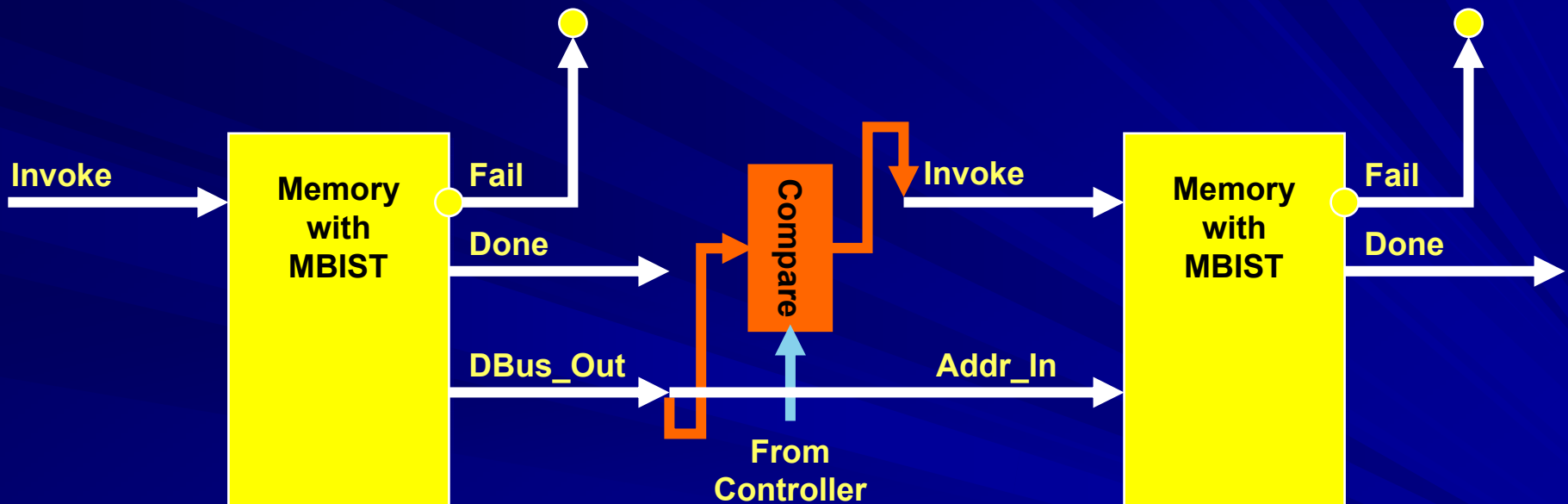
Start on Cycle



- Test Instruments can all be started at a certain cycle or within certain cycles if the controller places absolute cycle values...
- ...or instruments can be scheduled with relative cycle counts if the Invoke, Done, Fail, and similar signals are used as Triggers

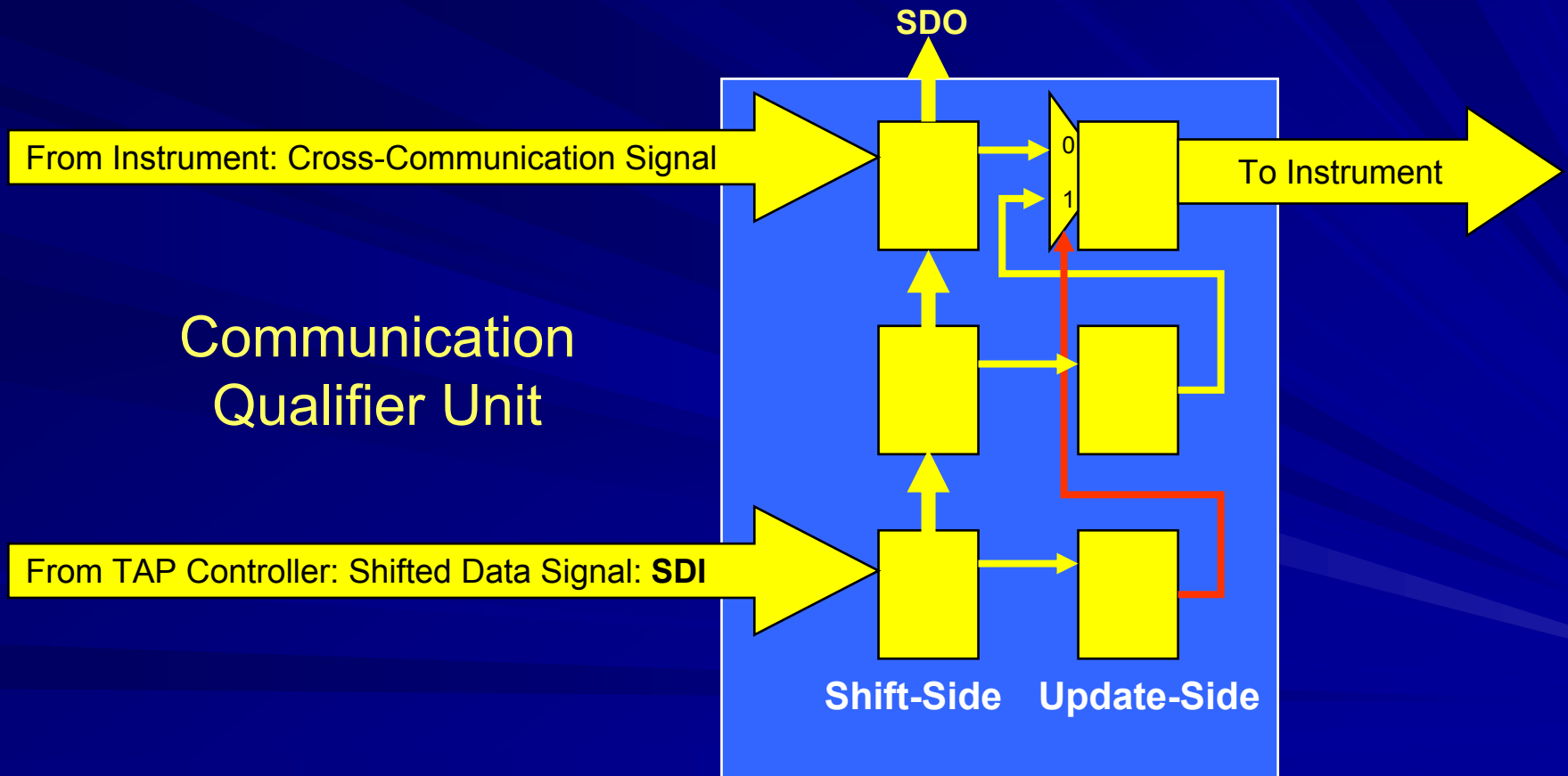
Example Instrument Communication

Start on Event



- When a value on one instrument (for example the data in a Tag Memory) achieves a predicted or known state...
- ...then this may enable the test application of another instrument
- The “Data Bus” or similar “event” signal is the “Trigger”

Example Communication Enabler



IJTAG Cross-Communication Classes

- **Communication actions:** may be organized as follows:
 - Start Actions or Triggers
 - Stop Actions or Triggers
 - Pause Actions or Triggers
 - Continue Actions or Triggers
 - Modify Actions or Triggers
 - Flow Change Control or Triggers
 - Clock Change Control or Triggers
 - Result or Status Enable Control or Reporting

IJTAG External Feedback Signals

- **Communication actions with ATE:** External Communication can only be “internal-instrument-to-external-pin” – all incoming control must be through the TAP (and external pins can be borrowed)
 - Start Actions or Triggers (tells ATE to start an operation)
 - Stop Actions or Triggers (tells ATE to stop an operation)
 - Pause Actions or Triggers (tells ATE to suspend an operation)
 - Continue Actions or Triggers (tells ATE to resume operation)
 - Modify Actions or Triggers (tells ATE to modify an operation)
 - Flow Change Control or Triggers (tells ATE to modify test flow)
 - Clock Change Control (tells ATE to apply specified clock)
 - Result or Status Enable Control (tells ATE to prepare for data)
 - Result or Status Reporting (provides ATE with process status)

Instructions

■ **Instructions**: *main confusion:*

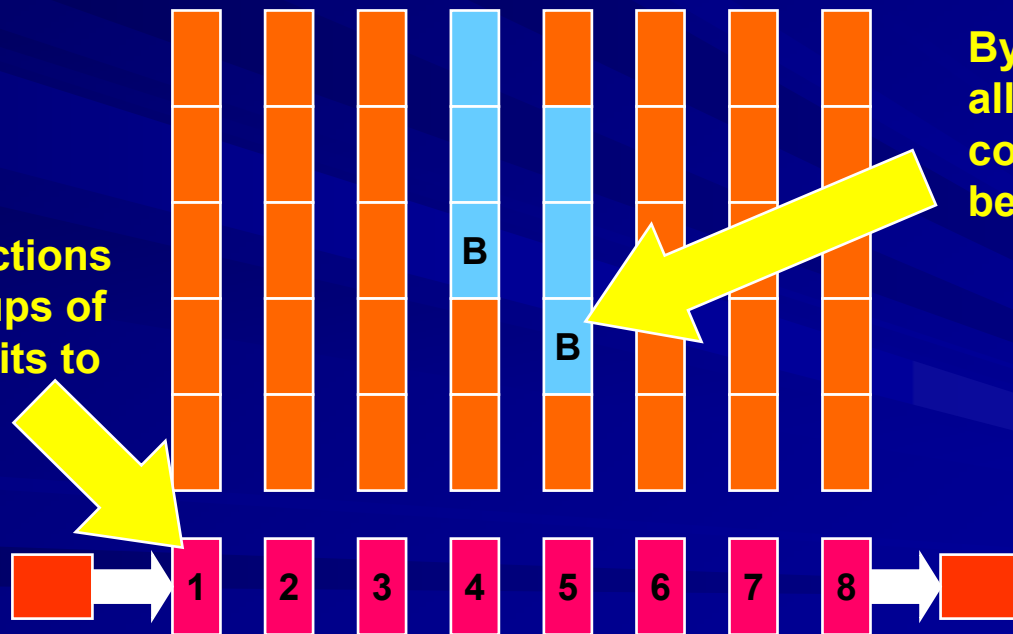
- *300 memories with Production BIST and with Engineering BIST requires 600 Instructions??? So, doing anything with the IR takes hundreds of clocks!*
- *Instructions are encoded? Instructions are one-hot?*
- *BSDL has to have pages of instruction encodings listed with each instruction???*

Instructions



Long and Ugly: Board Test Folks will Kill You

One-Hot Instructions (1-8) allow groups of configuration bits to be selected



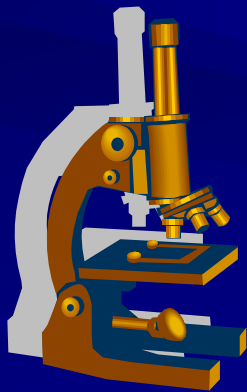
Bypass Bits in-line allow groups of configuration bits to be bypassed

Short & Sweet: Harder to Map

Instruments

■ **Instruments**: *main confusion:*

- *What the heck is an instrument???*



The 4 Major Instrument Classes

- **I-1: Simple Instrument under TAP-SM Control**
 - Selected by Instruction
 - Reacts to Capture-DR, Shift-DR, Pause-DR, Update-DR

- **I-2: Simple Self-Contained Process**
 - Selected by Instruction
 - Operates when TAP-SM is in RTI

- **I-3: Complex Instrument or Process**
 - Selected and Configured by Instruction
 - Operated, Configured, Changed, Adjusted, using Instruction changes (e.g., one hot encodings) and TAP-SM states

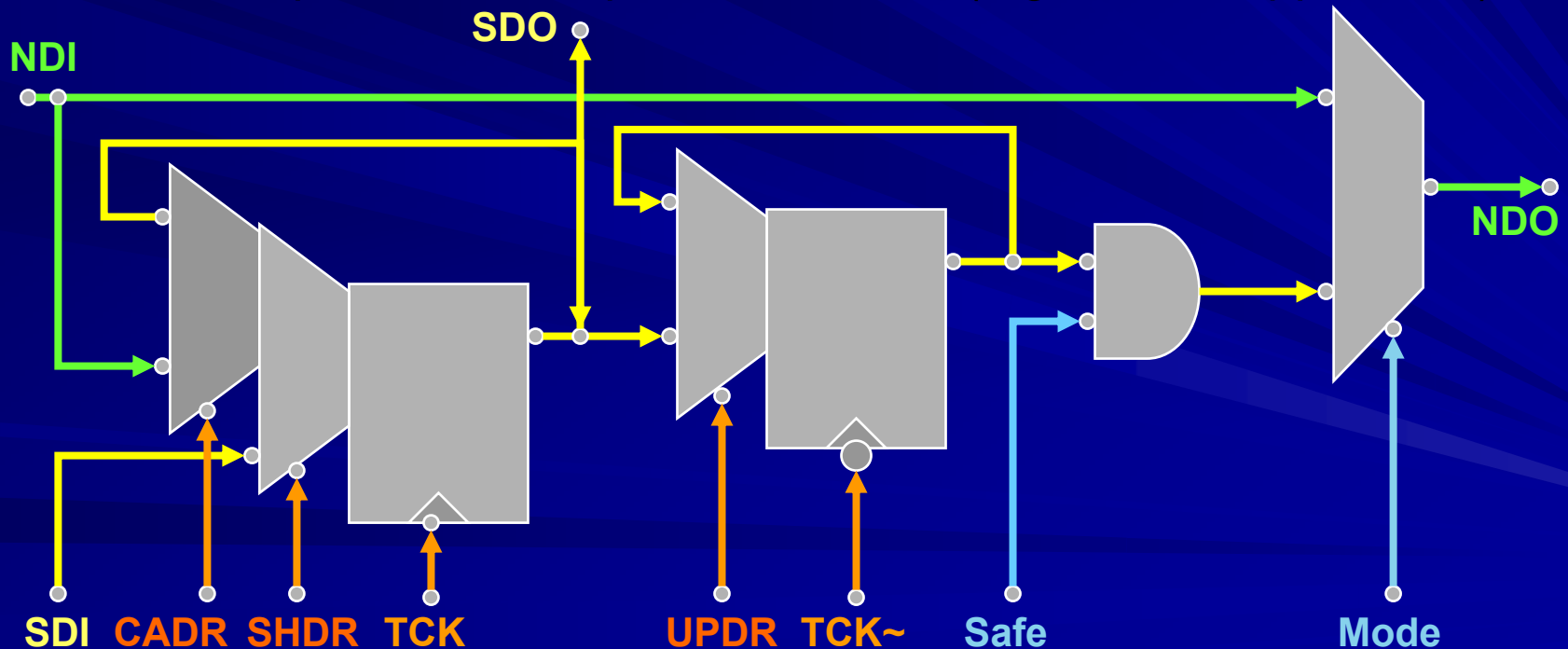
- **I-4: Complex Self-Contained Process or Processor**
 - Selected and/or Configured by Instruction
 - Operated, Configured, by Config-Reg or Instrument-Instructions

The 4 Major Instrument Classes

- Color Coding Legend for example instruments
- Green: Functional Connection
- Yellow: Test Connection
- Orange: TAP State Machine Control or Clock
- Blue: Instruction Control

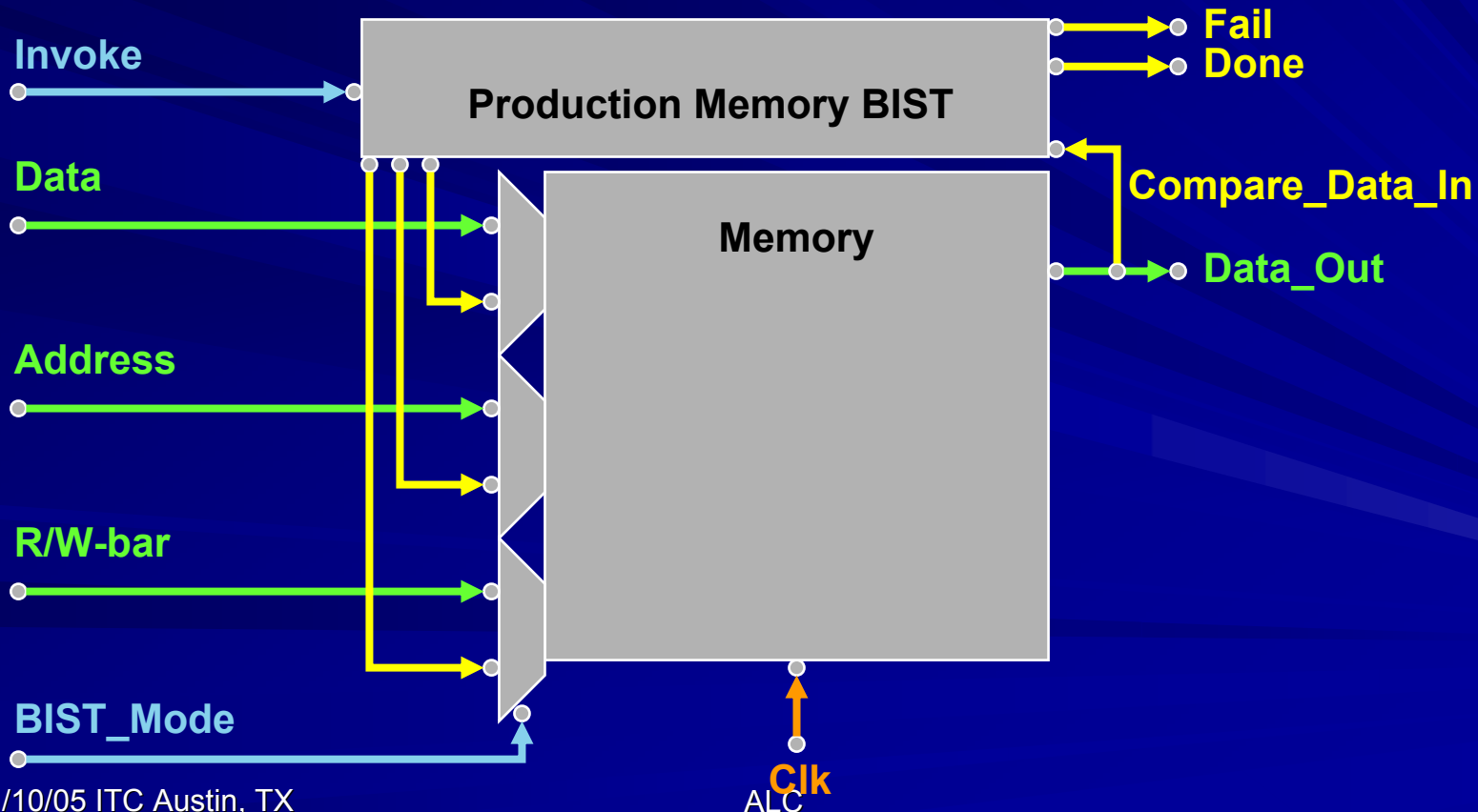
Instrument Class I-1: Simple Instrument

- I-1: Simple Instrument under TAP-SM Control
 - Selected by Instruction or Instructions
 - May react to Capture-DR, Shift-DR, Pause-DR, Update-DR
 - Other operations independent of TAP (e.g., vector application)



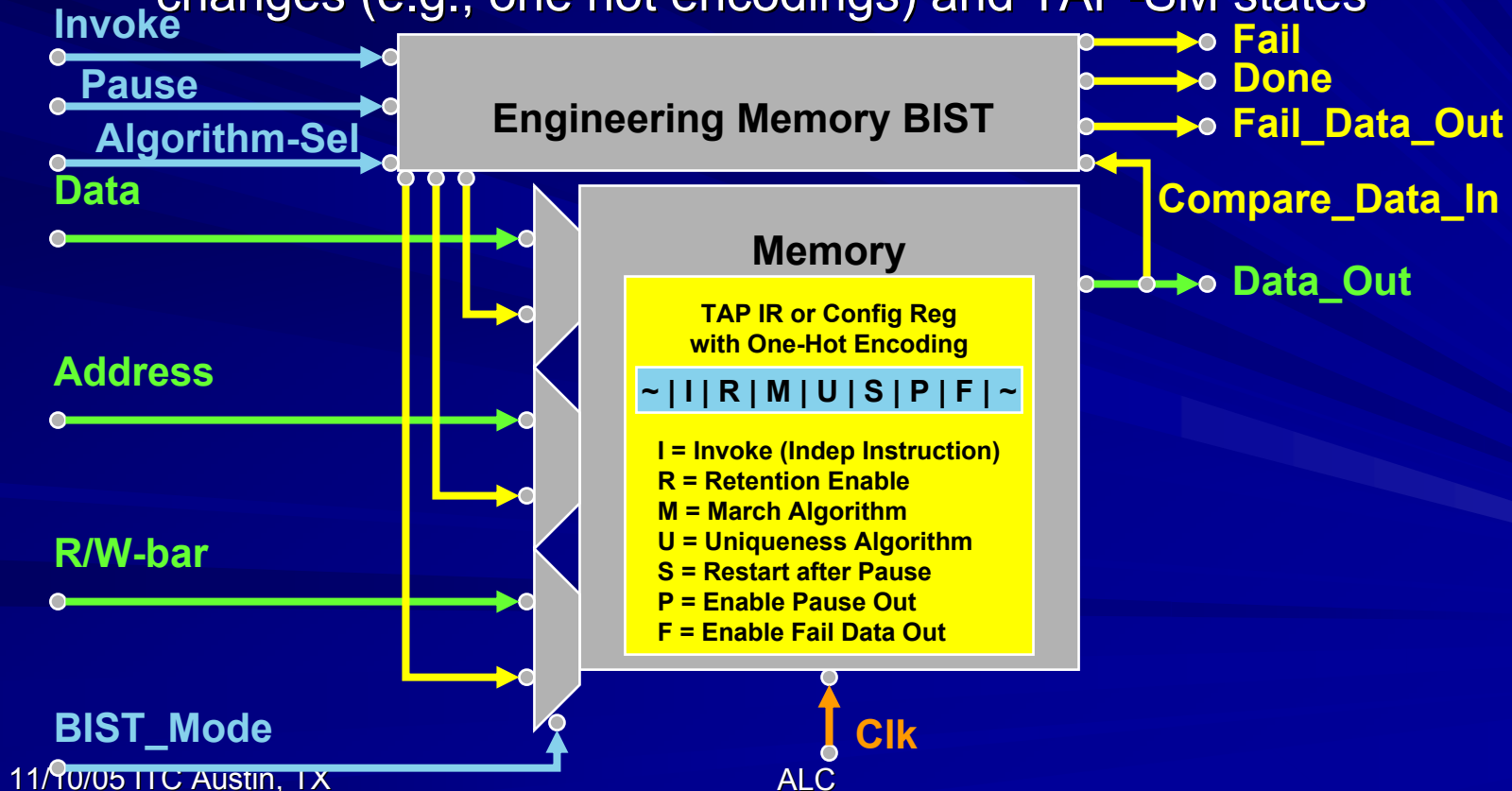
Instrument Class I-2: Simple Process

- I-2: Simple Process under TAP-SM Control
 - Selected by Instruction or Instructions
 - Operates independent of SM control – while SM is in RTI



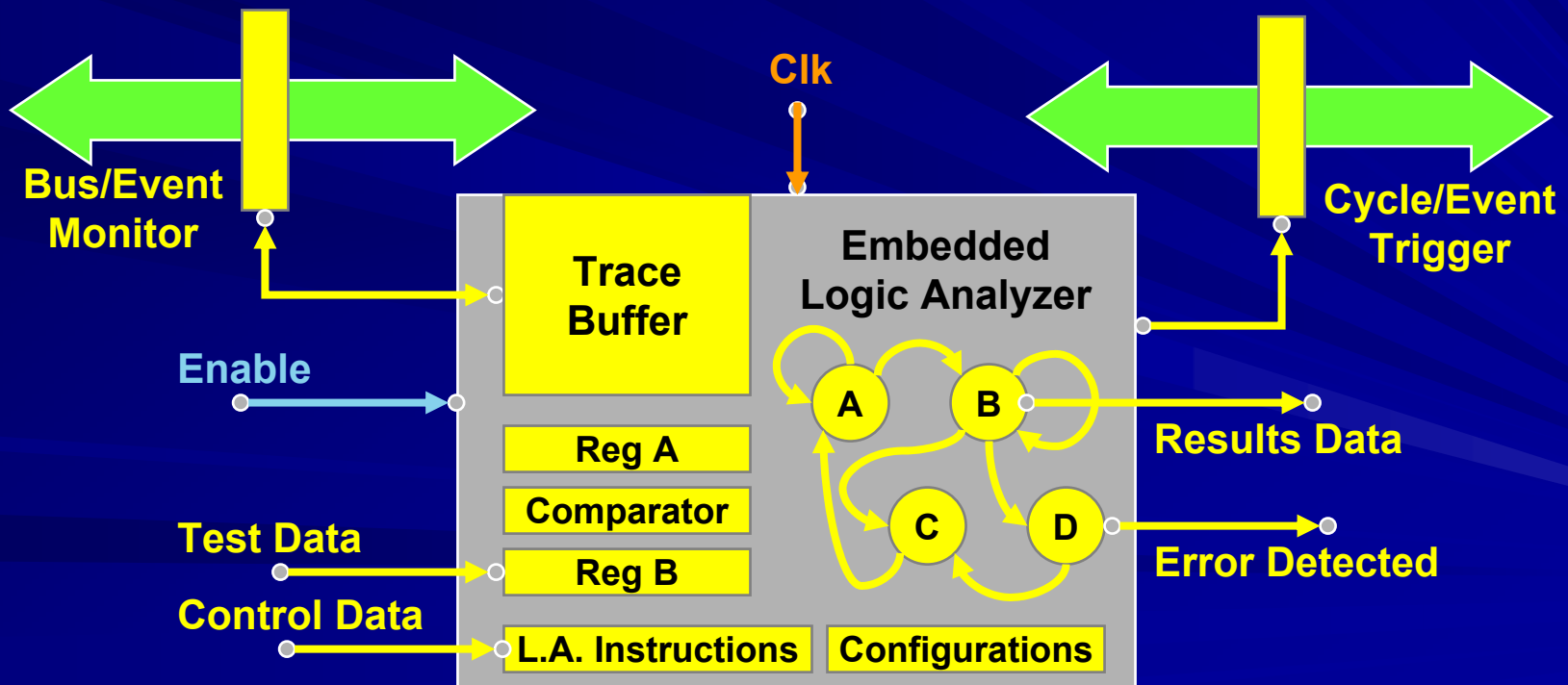
Instrument Class I-3: Complex Instrument

- I-3: Complex Instrument or Process
 - Selected and configured by Instruction
 - Invoked, Configured, Modified, Adjusted, Operated using IR changes (e.g., one hot encodings) and TAP-SM states



Instrument Class I-4: Complex Processing Element

- I-4: Complex Self-Contained Process or Processor
 - Selected and/or Configured by Instruction or Instructions
 - Operated, Configured, by Config-Reg or Instrument-Instructions



Common Instruments Classified

- DC Scan Architecture with borrowed I/O pins, Clk, SE
 - Viewed as an I-1 – mostly Configuration
- DC Scan Architecture bundled under the TAP
 - Viewed as an I-1 – mostly Config & Reactions to TAP
- AC Scan Architecture with borrowed I/O pins, Clk, SE
 - Viewed as an I-1 – mostly Configuration
- AC Scan Architecture bundled under the TAP
 - Viewed as an I-3,4 – mostly Configuration, Reactions to TAP-SM States, and extra self-contained functions or modified TAP-SM functions to handle @spd sample

Common Instruments Classified

- **1500 Compliant Wrapper**
 - Considered an I-1 with exception of mismatched capability
- **Embedded Vector Compression or Logic BIST**
 - Considered an I-2 if self-contained, I-3 if configurable
- **Memory BIST for Production**
 - Considered an I-2 if MBIST runs while TAP-SM is in RTI state; an I-3 if Repair is required; an I-4 if a standalone MBIST processor is used
- **Memory BIST for Engineering**
 - Considered an I-3 or I-4 depending on complexity and type of configuration and where and how the functions are implemented

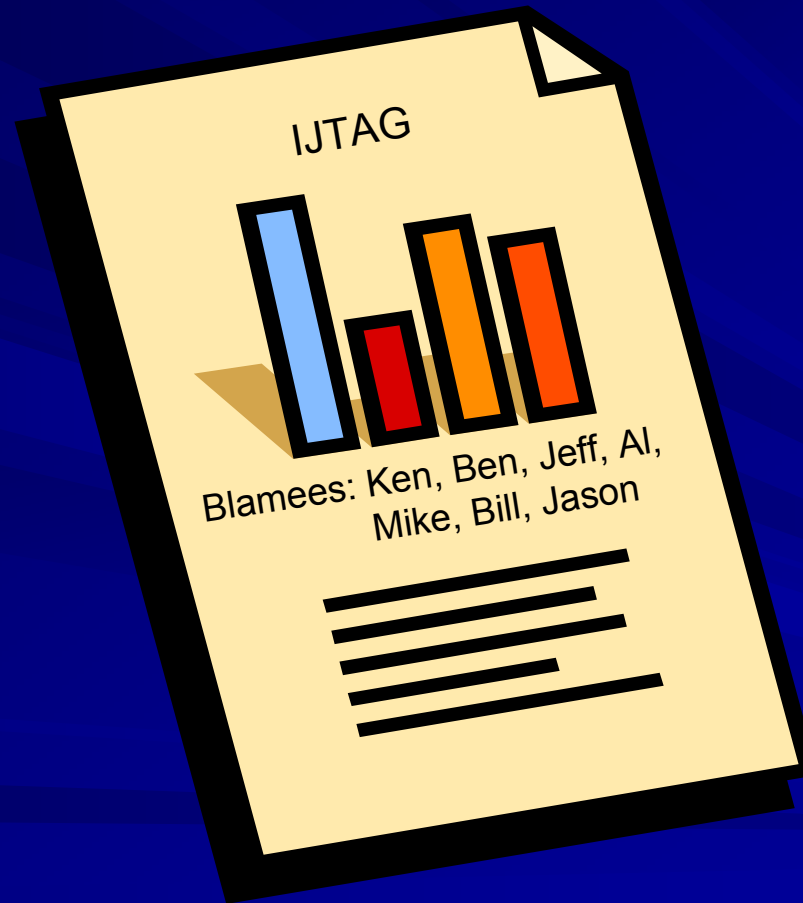
The Job's not done until the Paperwork is Over



You are here

And this may not
be a good thing

The Deliverables



The Deliverables: Languages

- **Architecture, Process, and Instrument Language:**
 - a specification for the language used to describe the instruments (format; level of description)
 - or a pointer if the language already exists; and any changes or additions to needed
- **Chip-Level and Internal Protocol Language:**
 - a specification for the language used to describe the protocol (format; operations.)
 - or a pointer if the language already exists; and any changes or additions to needed

The Deliverables: Specifications

- **The Architecture, Methods, Techniques Specification:**
 - an inclusion of the 1149.1 TAP and compliance as the chip-level protocol
 - from the outside of the chip it looks just like a compliant 1149.1 TAP
 - Includes the defined board-test instructions/features
 - a description of the preferred ways to address internal instruments
 - connections and protocols
 - new defined Standard Instructions and hardware structures
 - data specification for separate configuration registers

The Deliverables: Specifications

- **The Architecture, Methods, Techniques Specification:**
 - A description of the preferred ways to solve the technical problems of bandwidth, sequencing, synchronization, ATE or test platform synchronization
 - Preferred methods to turn on more bandwidth (enabling other pins, selecting a data clock, using internal memories);
 - Preferred methods to alleviate the limitations of using the compliant TAP (TAP extensions, extended processes)
 - Preferred methods to get two different items to talk to each other (an embedded communication module?);
 - Identified other/new signals that can communicate directly with the tester or test platform to indicate that processes are done, paused, waiting, etc...

Questions?



al.crouch@inovys.com