

Alfred L. Crouch
Chief Technologist & Director of IJTAG R&D
Vice-Chair IEEE P1687 "IJTAG" Working Group



P1687 Hardware Update

The "Latest & Greatest" IJTAG Work:
Gateways, Instruments, Connections,
Tradeoffs, and Bandwidth

ASSET

Presentation Plan

- 1. Problem solved by P1687 – embedded content not made for board test clogs up the true 1149.1 standard implementation; BISTs and Scans and Vector Compression and Debug Logic require more description than available with BSDL
- 2. 1149.1 and 1687 have hardware separation by using a Gateway – the left side of the Gateway has the TAP, TAP Controller, BSR, and Instruction Register and is described in BSDL; the right side of the Gateway has BIST, Scan, Wrappers, Vector Compression, Debug Logic, and Environmental Monitors and is described in something other than BSDL (such as CTL – 1450.6)
- 3. Each 1687 Instrument has a defined Normative interface; and one of the defined Normative connection schemes to the Gateway – this allows architectures to be made that can meet engineering tradeoffs (area, logic, concurrency, power, etc.)
- 4. The stuff still in work are the parallel connections which allow instrument-to-instrument communication and asynchronous events such as triggers, breakpoints, assertions, and global communication – these are defined as data and control transfers called READs and WRITEs

The Committee

Active P1687 (IJTAG) Working Group

- Chair: Ken Posse (Consultant)
- Vice Chair: Al Crouch (Asset-InterTech)
- Editor: Jeff Rearick (AMD)
- Std. Liaison: Ben Bennetts (Bennetts Assoc. – Ret)
- Web Master: Michael Laisne' (Qualcomm)

- Current Working Group Members:
- Jason Doege (AMD); Mike Ricchetti (ATI); Srinivas Patel (Intel);
- Bill Eklow, Hongshin Jun, Ted Eaton (Cisco);
- Songlin Zhuo (Qualcomm); Pradipta Ghosh (Broadcom);
- Hugh Wallace, Rick Nygard (Agilent); Scott Hartranft (Tektronix)
- Thomas Rinderknecht, Paul Reuter (Mentor); J.F. Cote (LogicVision);
- Rohit Kapur (Synopsys) – 1450.6 CTL Liason;
- Bill Tuthill (Intellitech); Stylianos Diamantidis (GlobeTech);
- Brad Van Treuren, Michele Portolan, Suresh Goyal (Alcatel-Lucent);

What is IJTAG?

- IJTAG is the P1687 Proposed Standard which will be designed to enable more efficient access and optimized interconnect to embedded logic and electrical/environmental monitors inside the chip:
 - for all purposes: functional configuration, test, debug-diagnosis, yield
 - for all environments: wafer probe, package test, die-stack, board test, in-system
- P1687.0 will stipulate the 1149.1 JTAG port as the primary access mechanism
- Embedded content is defined as instruments and includes items such as:
 - FCN: Bus Configuration, Pin Configuration, Power Modes, Clock Modes
 - DFT: MBIST, LBIST, Scan-Compression, Test-Wrappers, Clock-Controllers
 - DFD: Logic Analyzers, Bus Monitors, Traffic Monitors, Trace-Buffers, Hardware Assertions, Embedded O-Scopes
 - DFY: Voltage Sensors, Temperature Sensors
- The goal is to enable standard control and configuration (architectures, connections, and interfaces) and to also:
 - allow higher bandwidth data delivery to instruments that need it
 - allow instrument-to-instrument communication
 - allow TAP-asynchronous instrument operations (e.g. a fail flag)

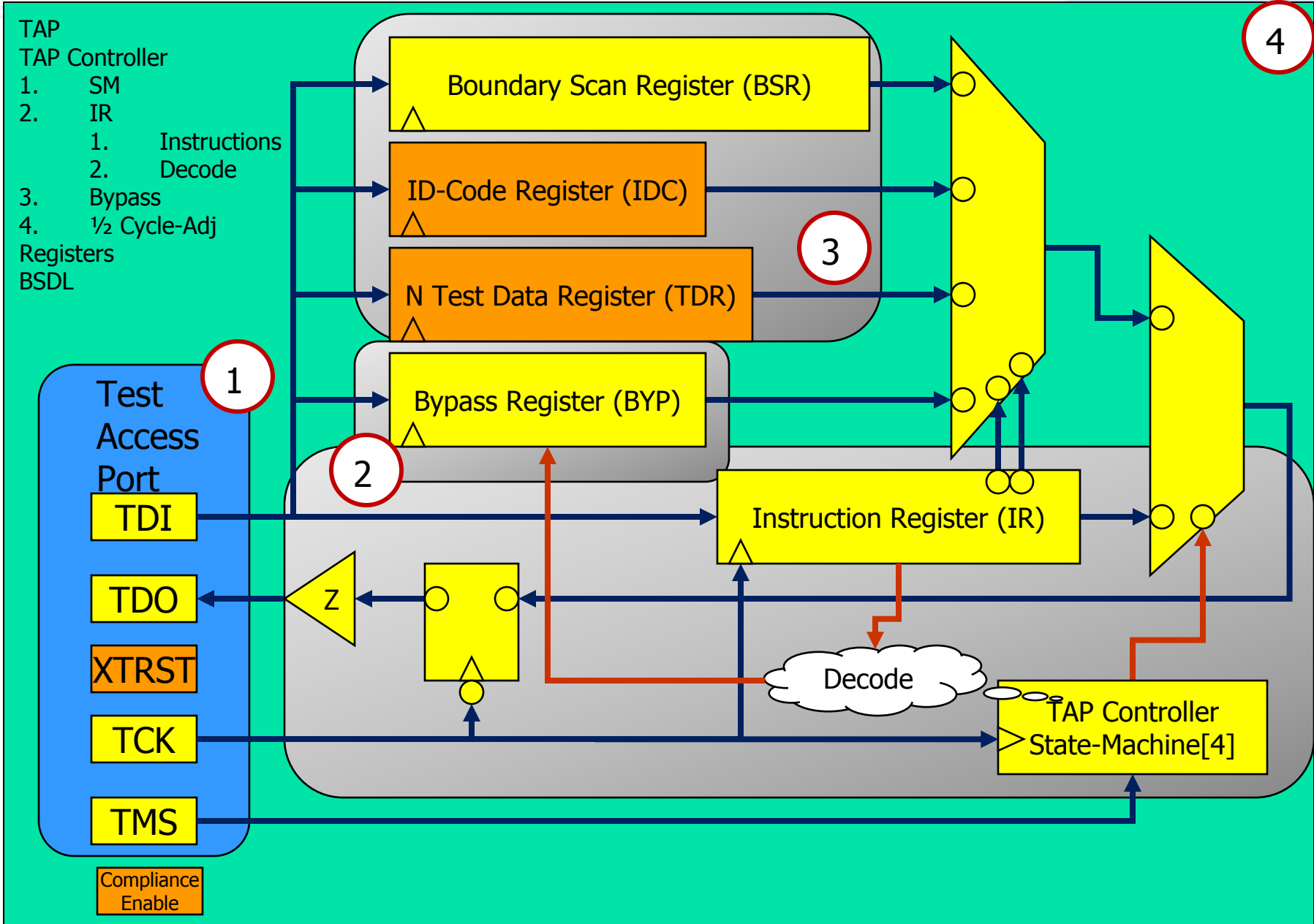
What Problem Does JTAG Solve?

➤ JTAG solves two main problems:

1. Specifying how to Include embedded logic not meant for board test under the 1149.1 TAP and TAP Controller without causing problems with 1149.1 Compliance
 - Clogging up the 1149.1 BSDL with non-board test content – higher probability of bad BSDL
 - Badly or inadequately describing complex instruments
 - Growing the Instruction Register with tens, hundreds, or even thousands of instructions
 - Dealing with one-hot instructions (instead of IR encoding)
 - Dealing with hierarchical nesting and changing scan-path lengths
2. Enabling optimization, efficiency, and tradeoffs to be applied to the growing volume of embedded logic and electrical monitor content (instruments) in modern chips
 - Allows a variety of connections instead of just JTAG daisy chain (all-at-once serial) or star (one-at-a-time)
 - Allows organization of embedded content into groups and hierarchies...
 - ...enables reuse of IP that may have entire instrument architectures (as opposed to TAPs)
 - Offloads the non-board-test content out of the 1149.1 Instruction Register

Standard JTAG Flow-Through Model

- 1. TAP
- 2. TAP Controller
 - 1. SM
 - 2. IR
- 3. Registers
 - 1. Instructions
 - 2. Decode
- 4. BSDL



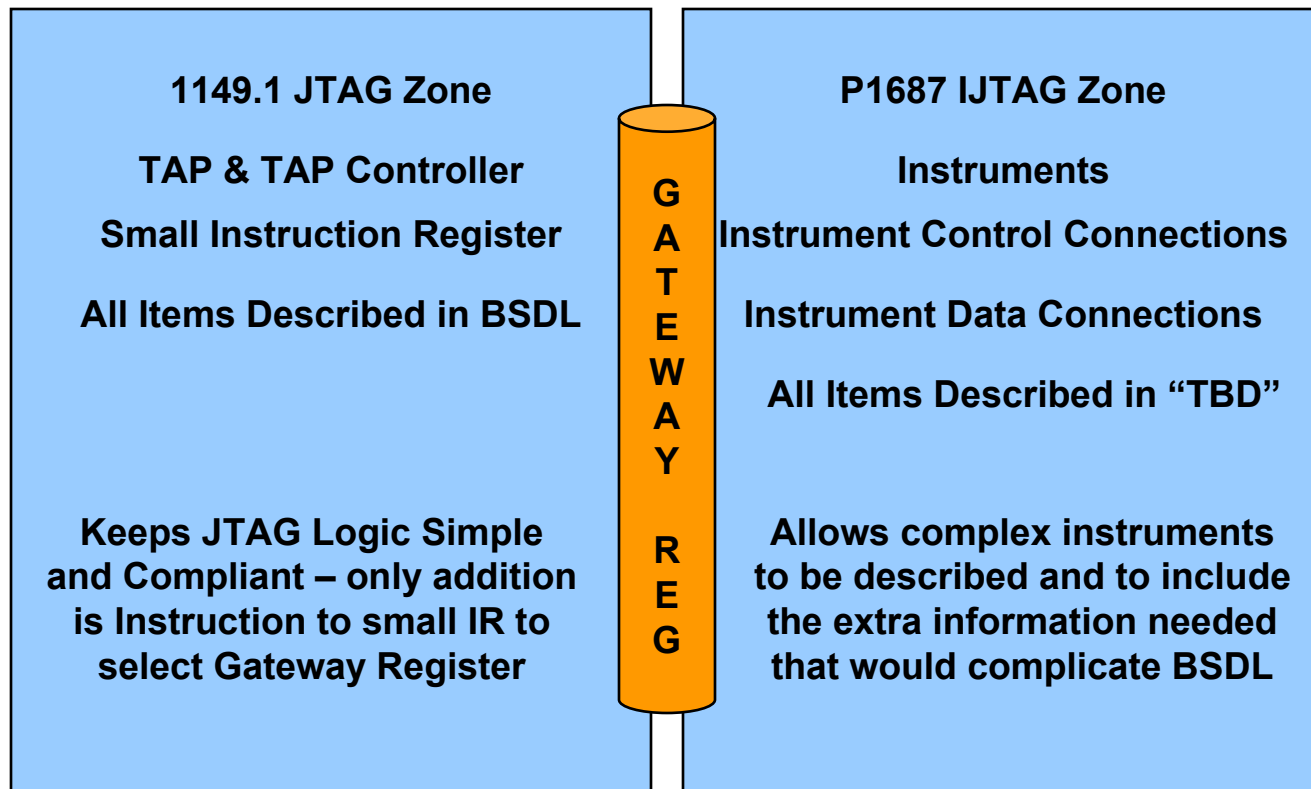
4

1

2

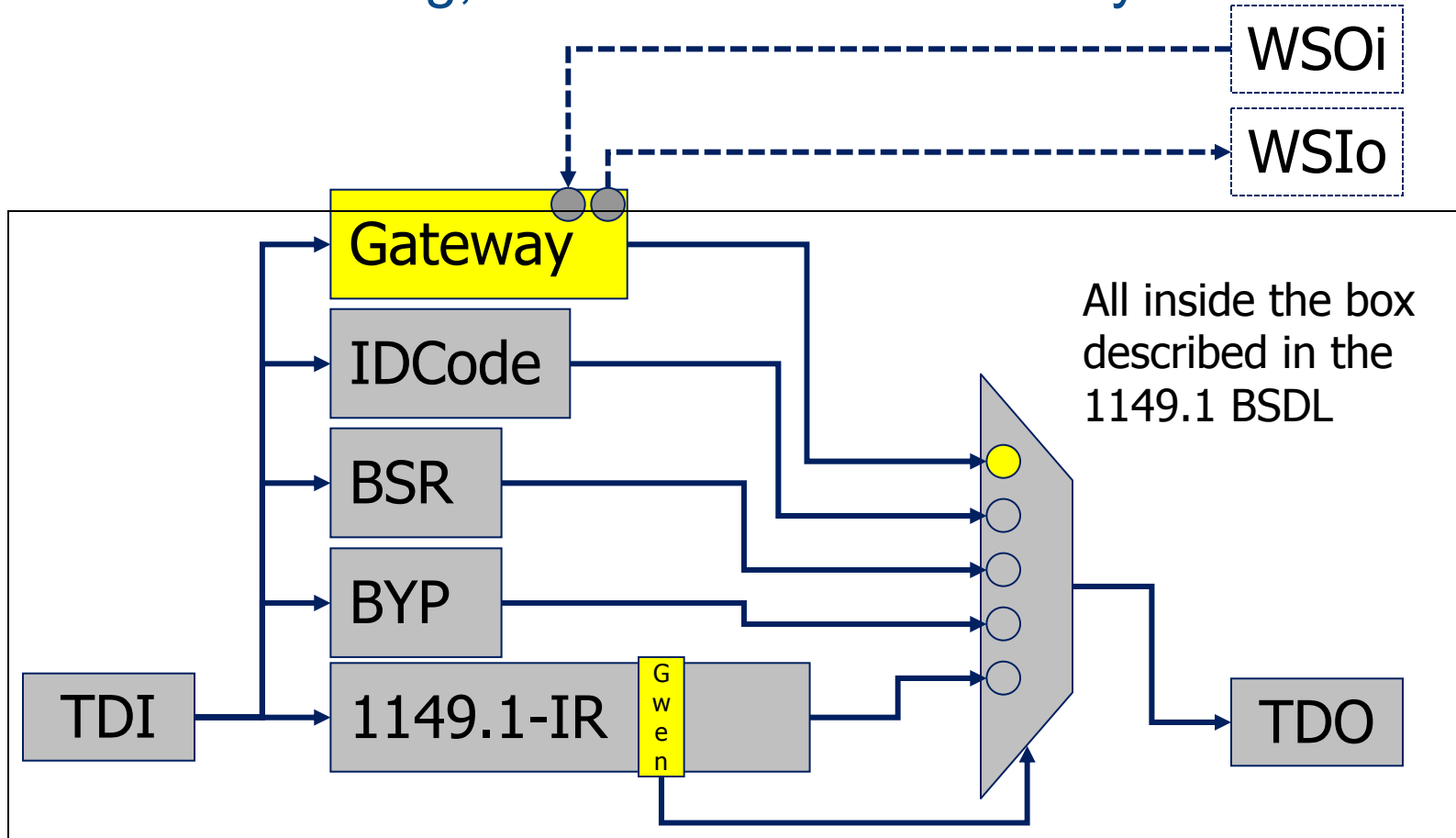
3

The IJTAG Differences



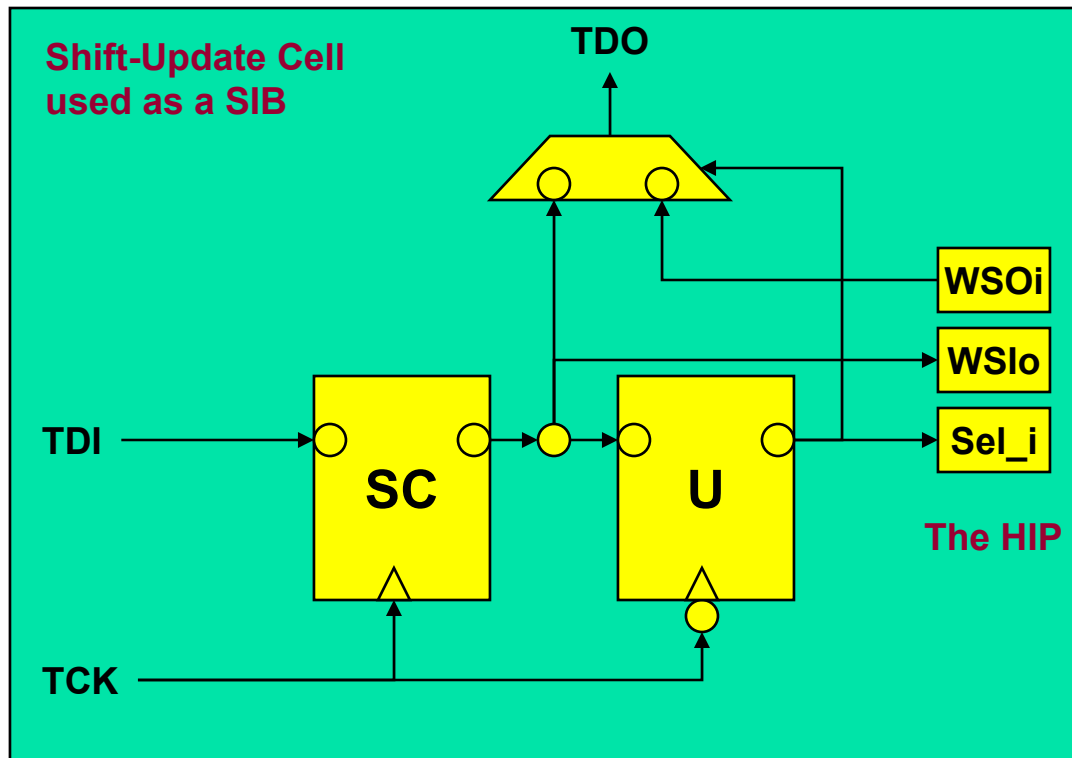
1149.1 Supports P1687 by...

- By including a TDR called a Gateway and an instruction (GWEN: GateWay ENable) to select it in the 1149.1 Instruction Reg; to describe the Gateway in the BSDL



The Gateway Select-Instrument-Bit (SIB)

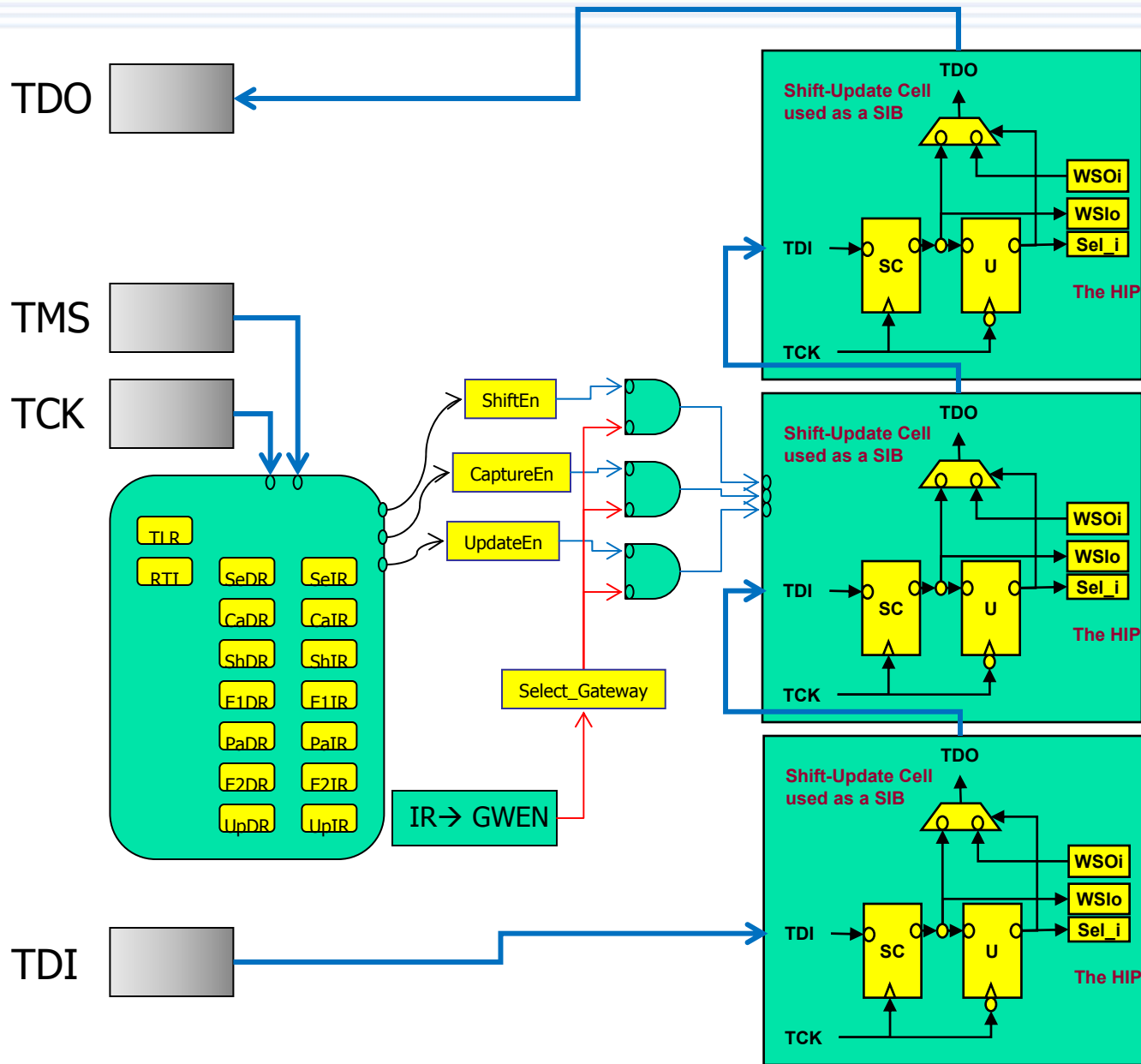
The Key Element for Adding, Organizing, Managing Embedded Content



The Hierarchical Interface Port (HIP) opens up a connection to embedded instruments when the an Assert value is placed in the U-Cell after a TAP State- Machine Update-DR

Scan Path Management Bit

A Closer Look at a Gateway



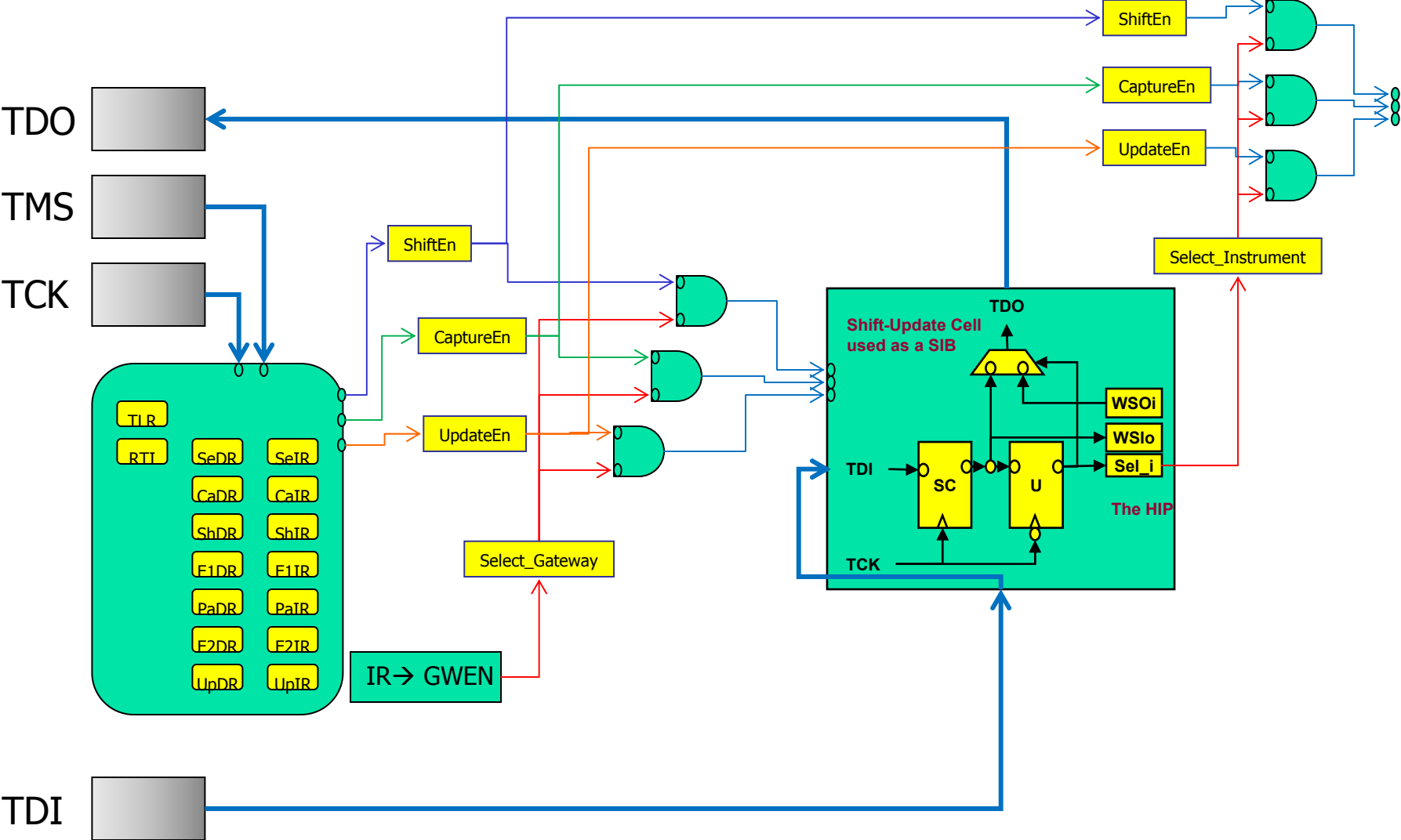
The BSDL describes the Gateway and the GWEN Instruction – the Gateway is represented as its collapsed length

All Instruments are accessed the same way the Gateway is: Shift, Capture, Update is gated by a Select_i signal local to the instrument

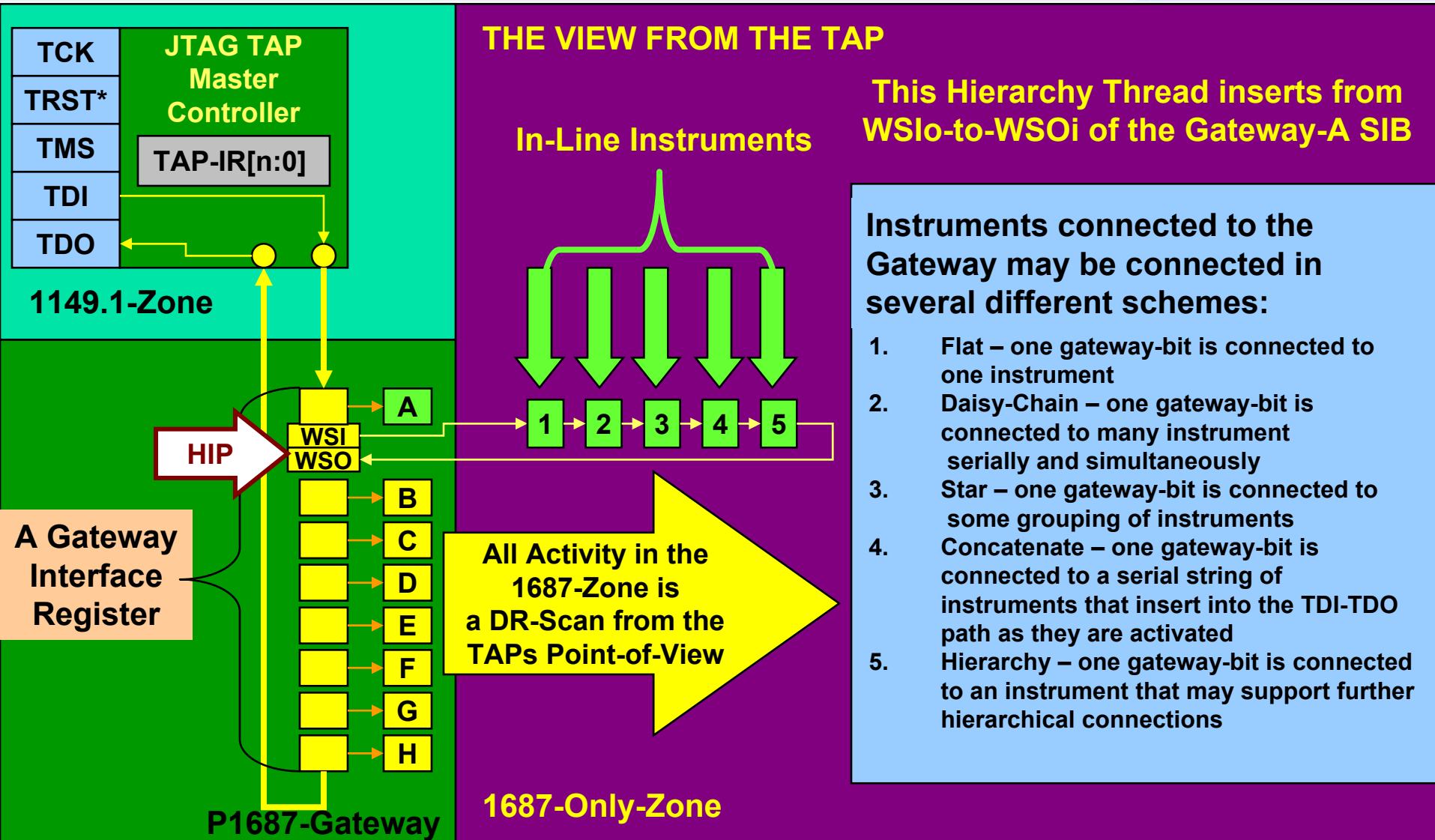
A Gateway can select an instrument or another Gateway

Instruments are required to have interfaces that are similar to 1500 WIRs using: Select_Instrument, Reset, ShiftEn, CaptureEn, UpdateEn, TCK, WSI, WSO,

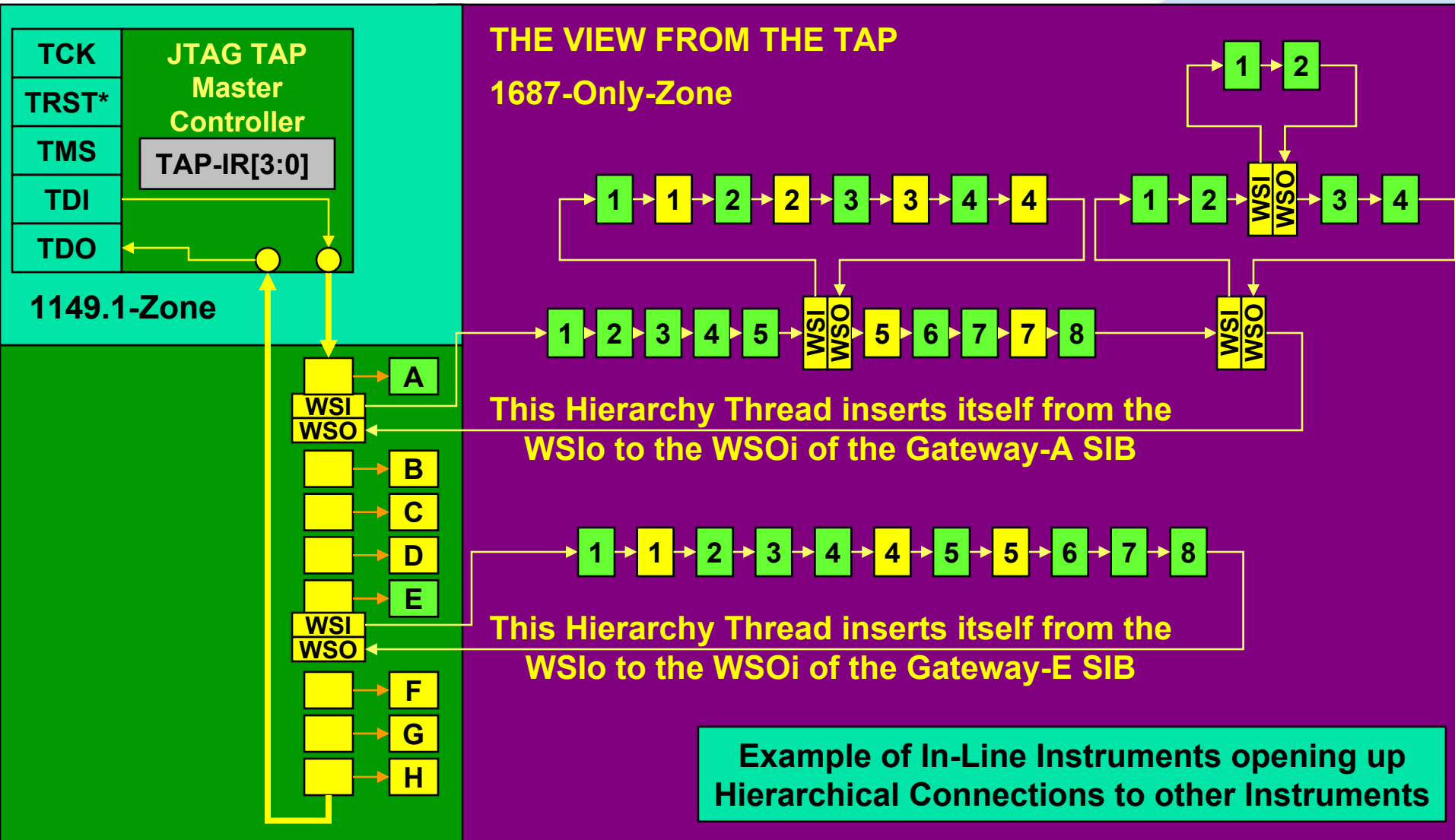
Passing Generic Shift-Capture-Update



1687 Hardware Architecture



1687 Hardware Architecture



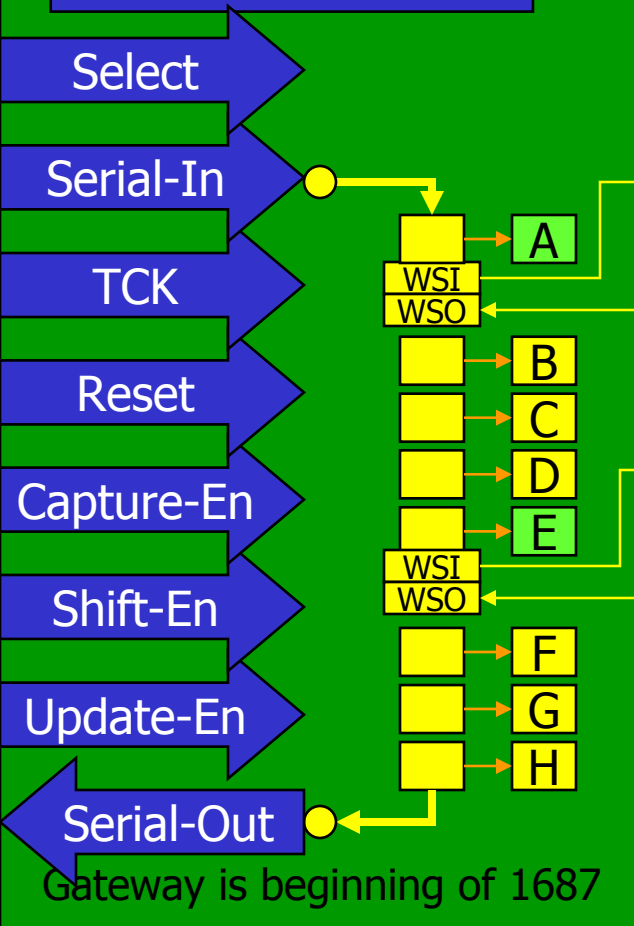
The Connections and Tradeoffs

- 4 Non-Hierarchical: Flat, Daisy-Chain, Star, Concatenate
- Hierarchical: Use of the SIB to open nested Gateways

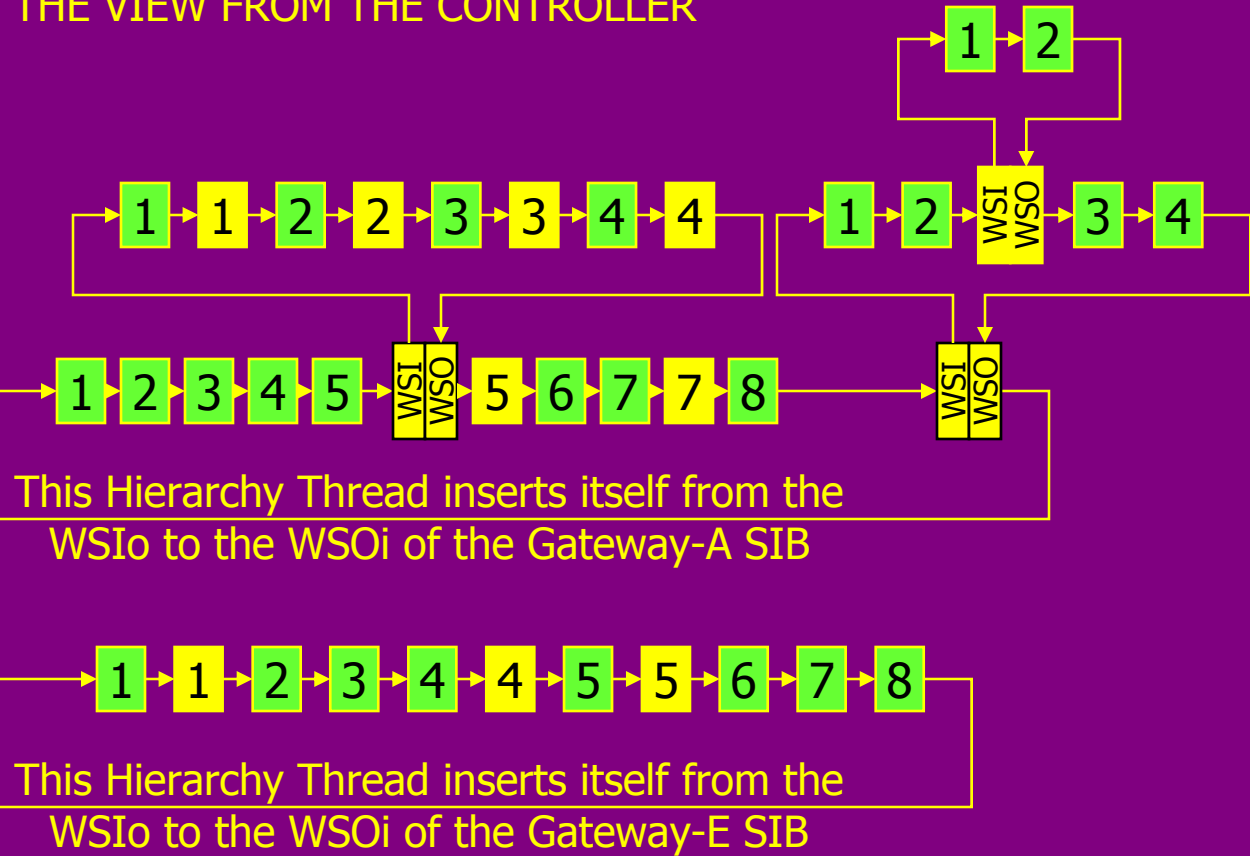
- Tradeoffs:
 - Engineering:
 - Area, Timing, Routing
 - Power-Thermal
 - Risk
 - Compliance/Efficiency:
 - IR-Depth, Scan-Path-Depth, Scan-Path-Depth-Stability
 - Utility/Automation:
 - Concurrency
 - Post-Silicon Flexibility
 - Protocol-Complexity
 - Language-Complexity

1687 Hardware Architecture

Example of Signals Required to operate a 1687 Gateway and Connected Instruments



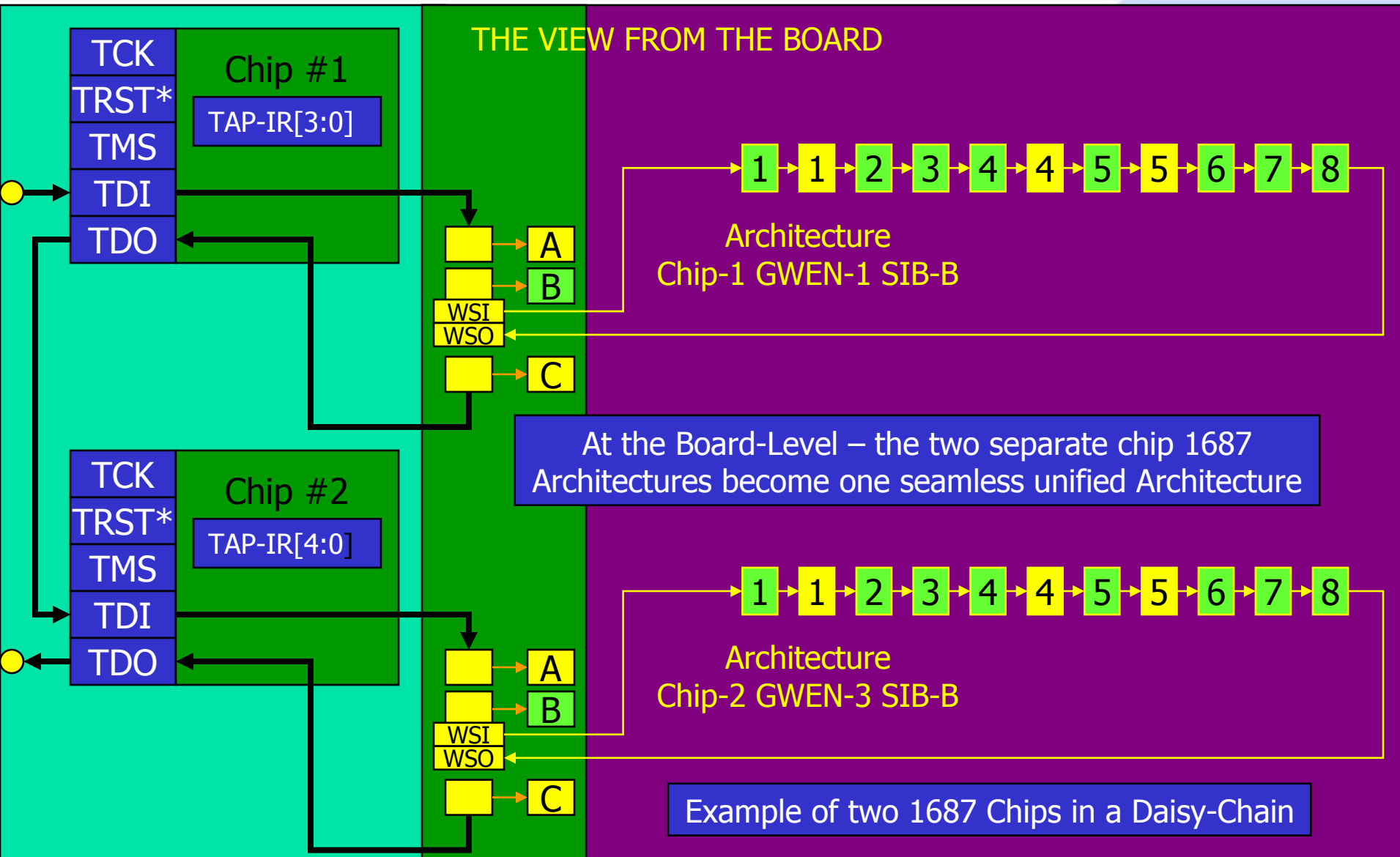
THE VIEW FROM THE CONTROLLER



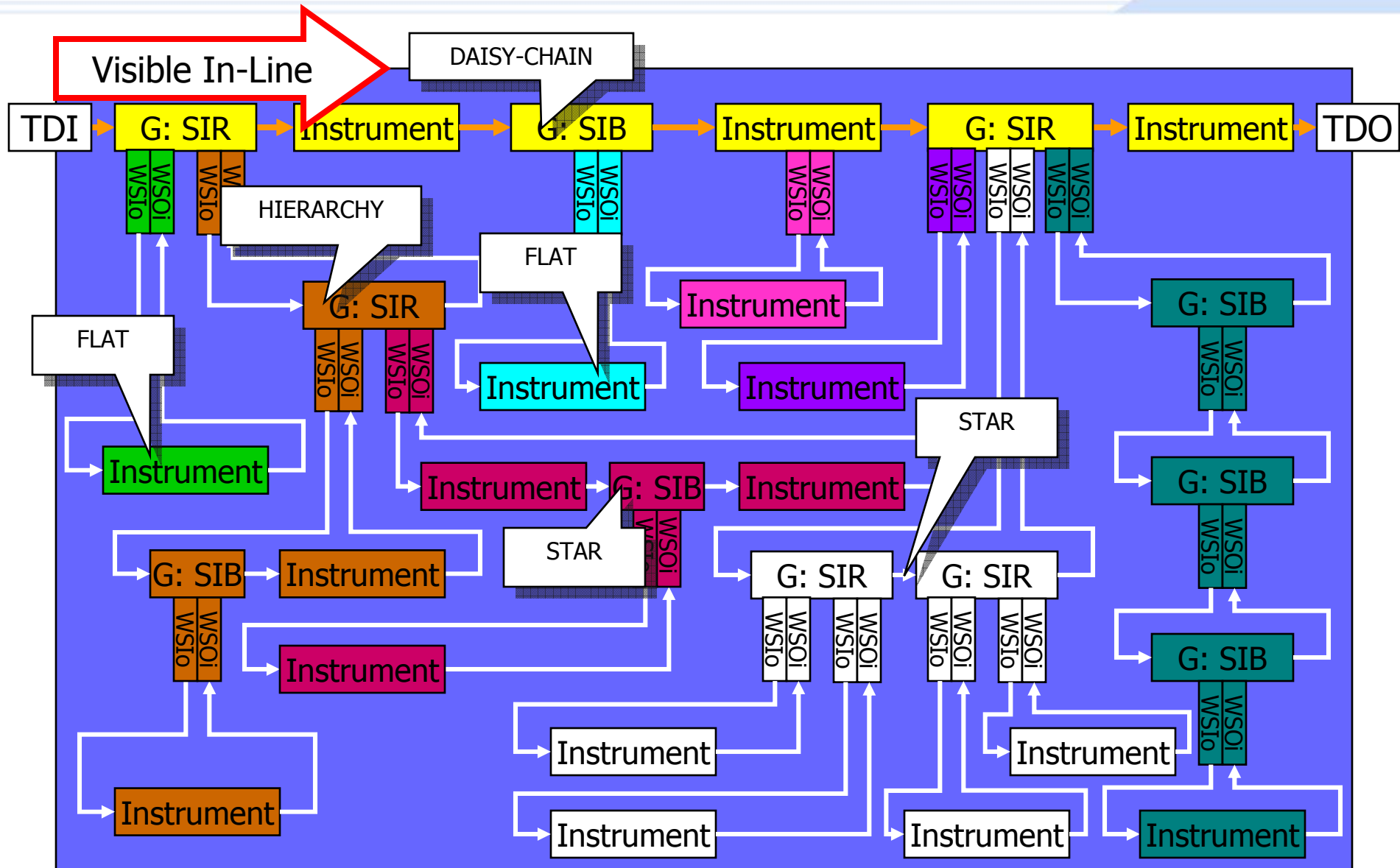
Example of use of a non-compliant TAP and TAP-Controller or other State-Machine Gateway may be TDR-like or 1500-TAM-like

1687-Only-Zone

1687 Hardware Architecture

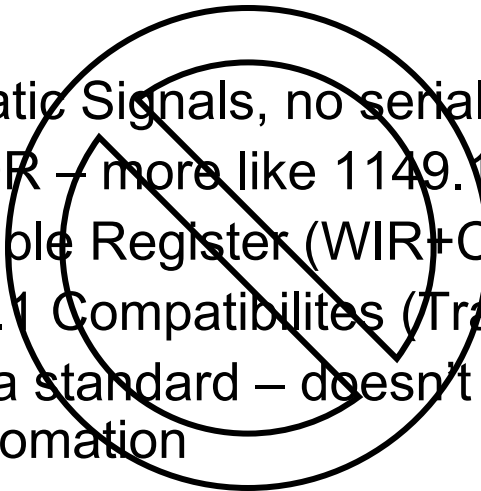


1687 Hierarchy Connections Example



Normative Instrument Interfaces

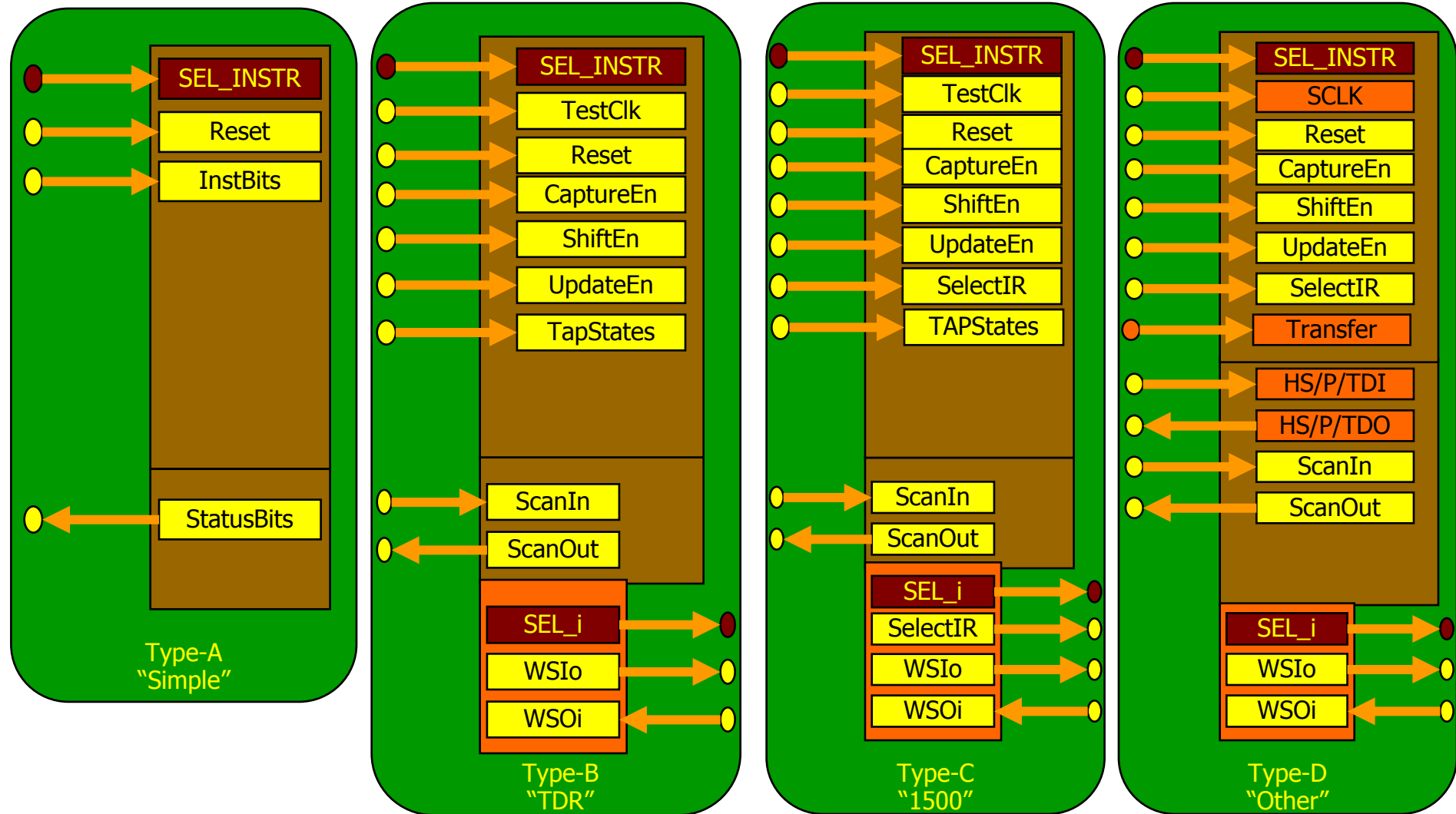
➤ Originally supported A, B, C, D for all existing legacy interfaces

- A=Simple Static Signals, no serial access
 - B=Simple TDR – more like 1149.1 BSR
 - C=1500 Multiple Register (WIR+QDR+WDR...)
 - D=Non-1149.1 Compatibilites (Transfer, SysClk, PDI-PDO)
 - Not much of a standard – doesn't lend itself to efficiency/automation
- 

➤ Can get what we need with a B-Plus

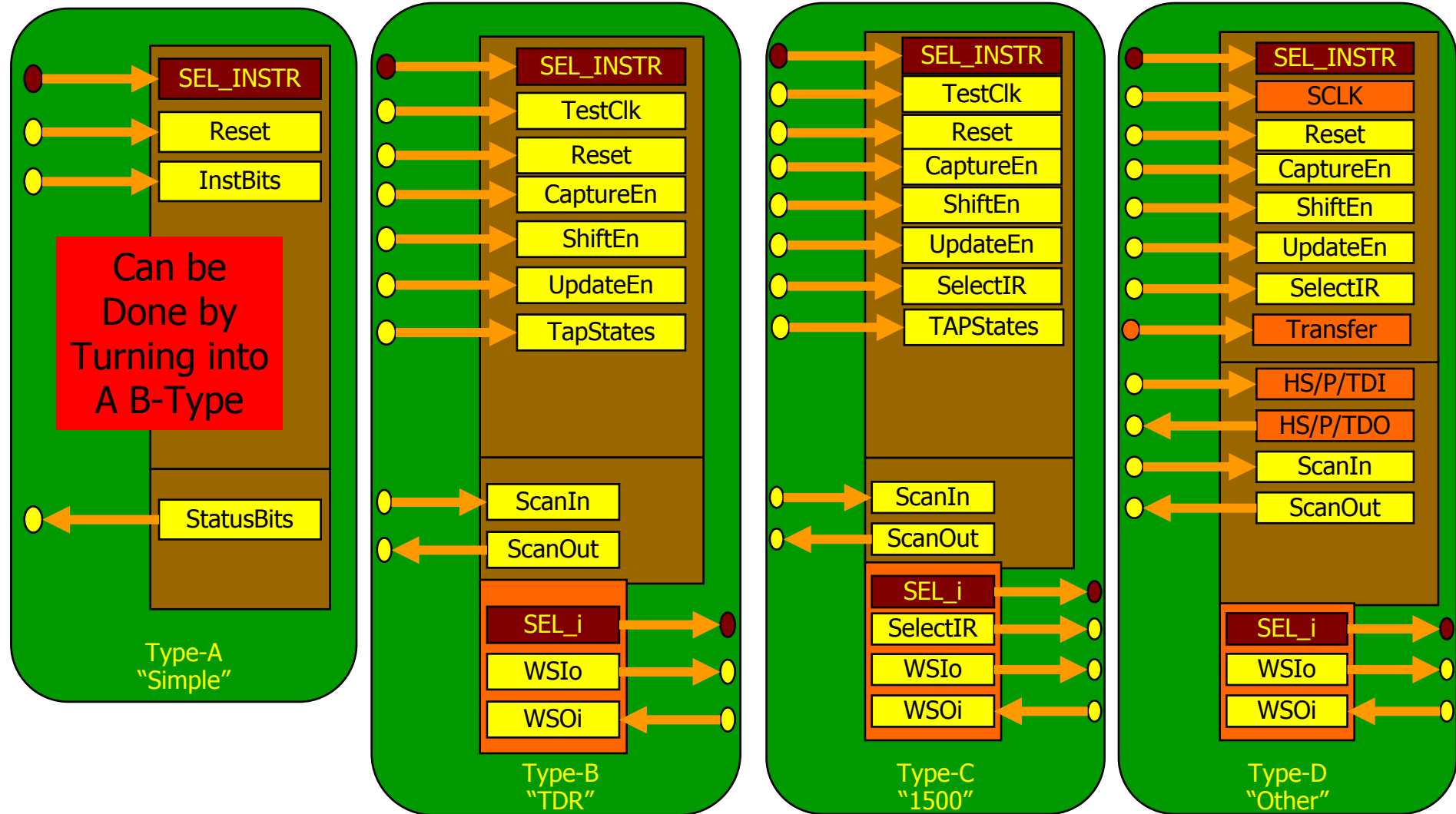
- A-Type can be supplied by requiring a B-Type
- The difference between a B and a C is the Select_WIR signal which can be locally generated with a SIB as the 1st scan bit
- D-Type Non-1149.1 actions are defined with a parallel access

Original Allowed 1687 Instrument Interfaces



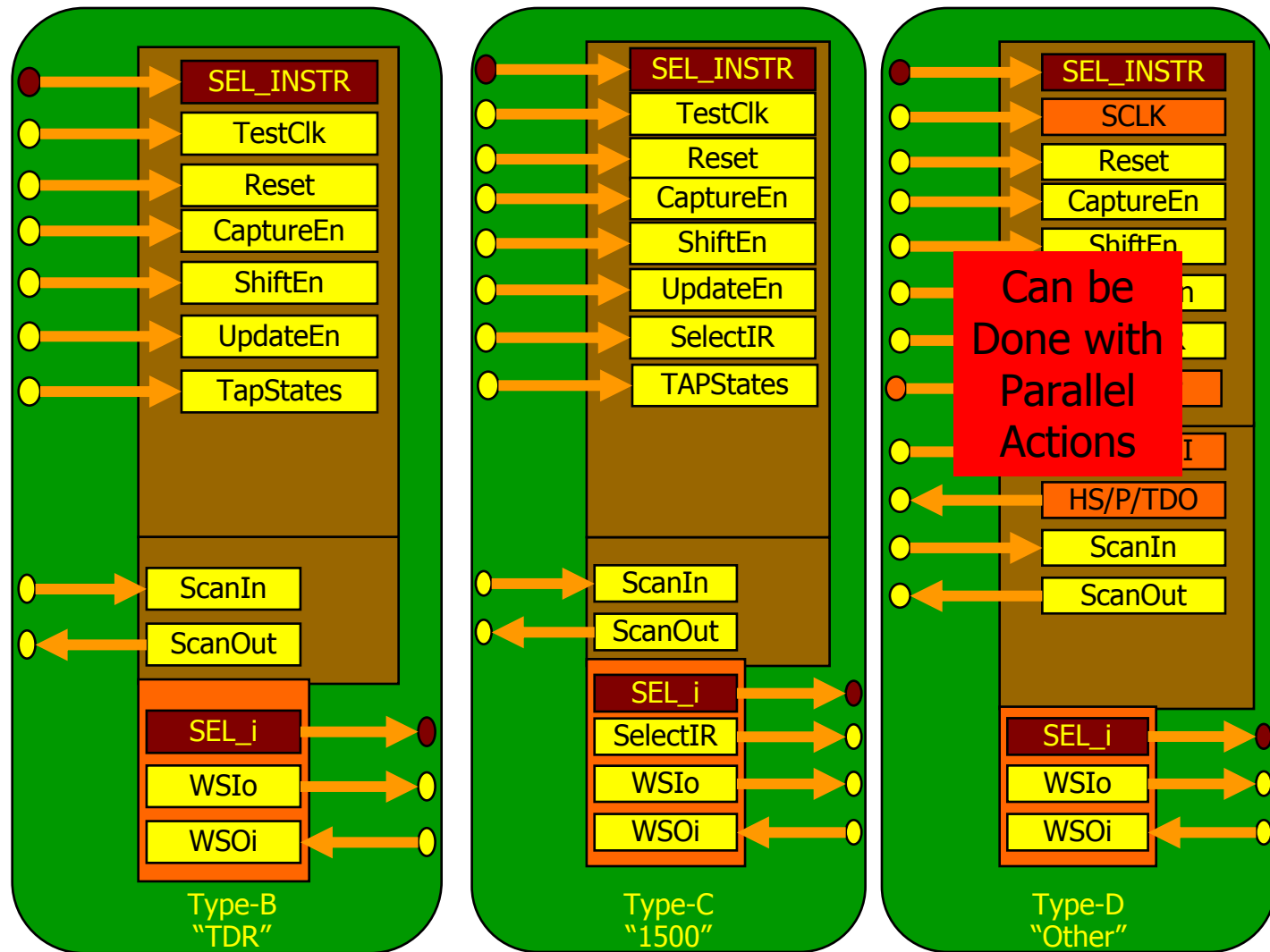
B, C, and D may also support Hierarchical Connections through HIPs

Original Allowed 1687 Instrument Interfaces



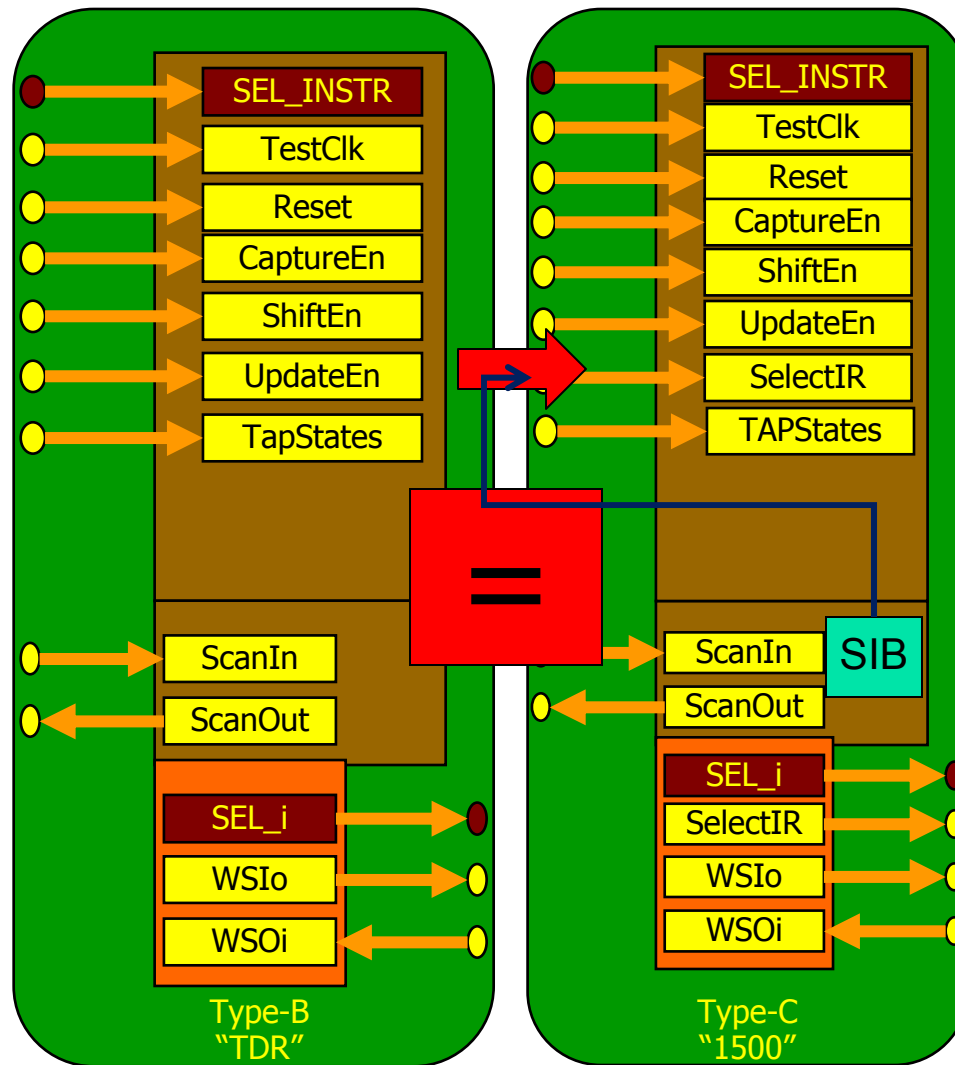
B, C, and D may also support Hierarchical Connections through HIPs

Original Allowed 1687 Instrument Interfaces



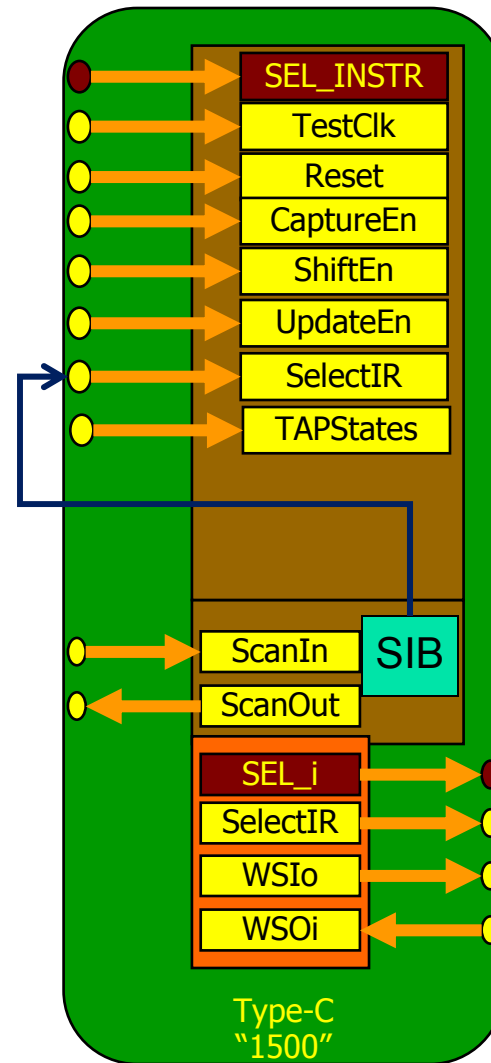
B, C, and D may also support Hierarchical Connections through HIPs

Original Allowed 1687 Instrument Interfaces



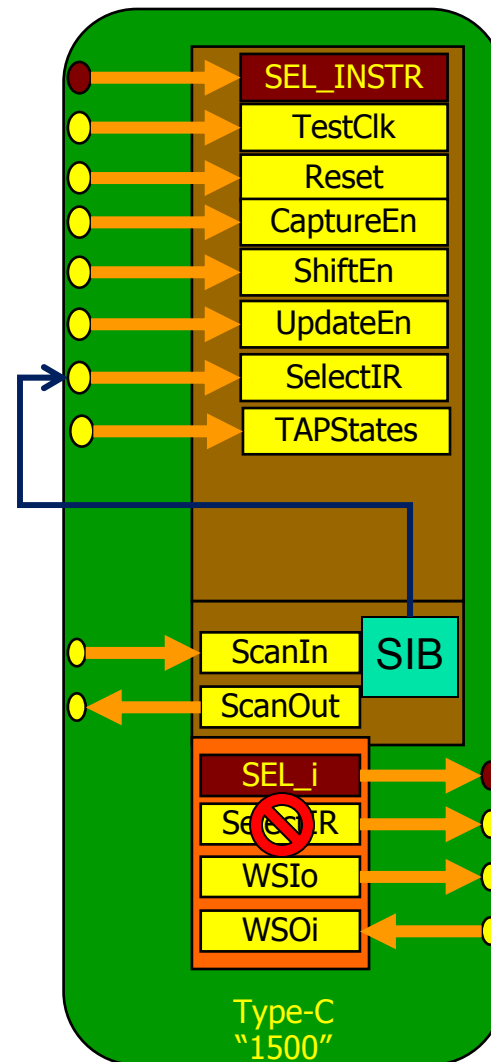
B, C, and D may also support Hierarchical Connections through HIPs

Original Allowed 1687 Instrument Interfaces



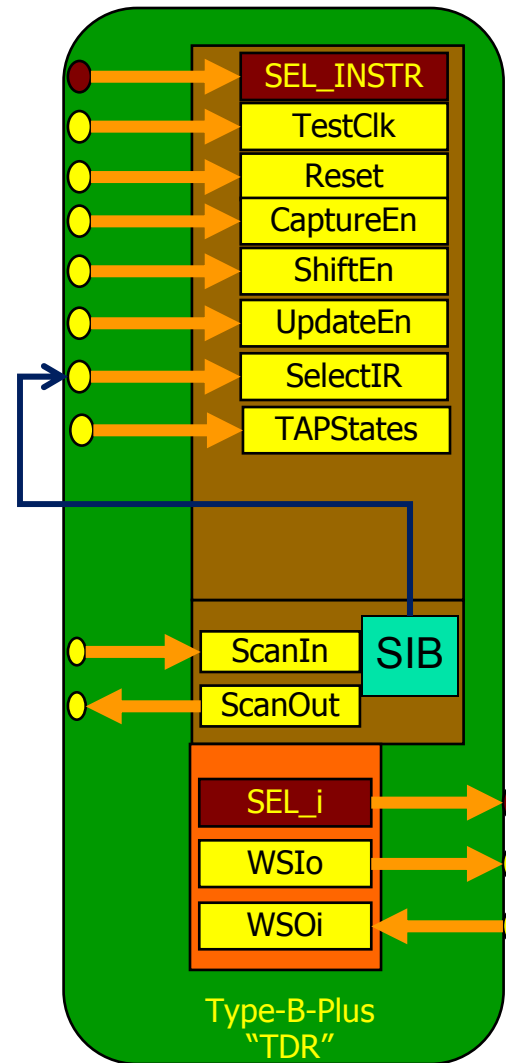
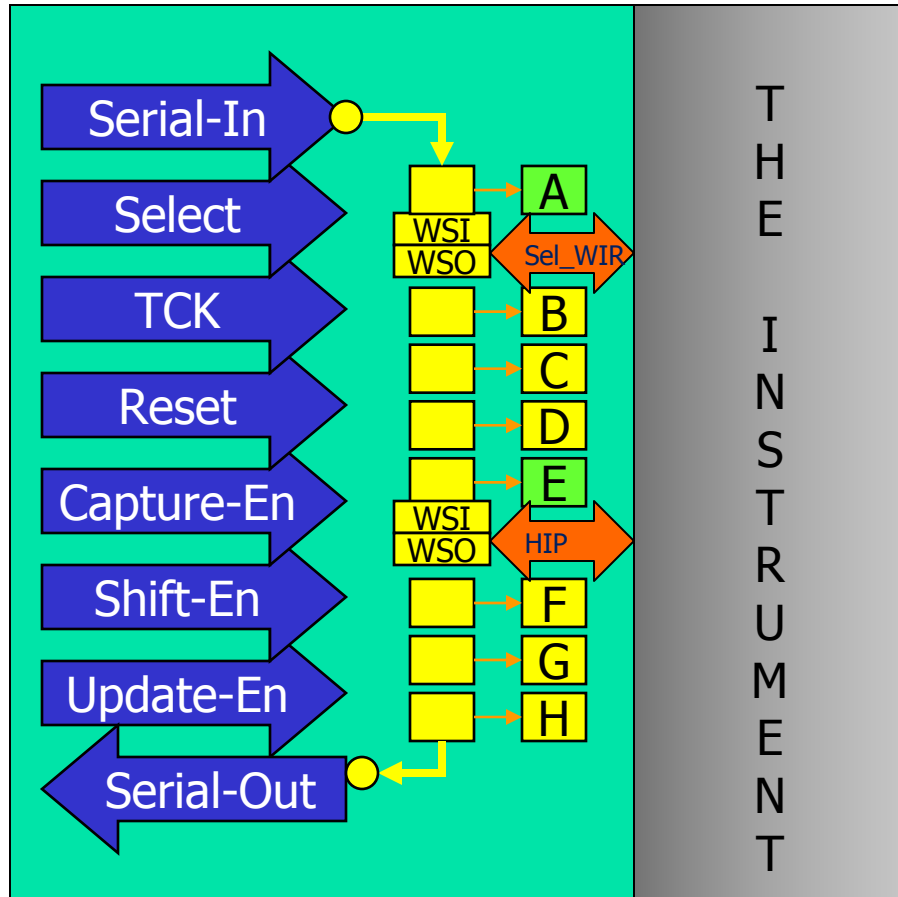
B, C, and D may also support Hierarchical Connections through HIPs

Original Allowed 1687 Instrument Interfaces

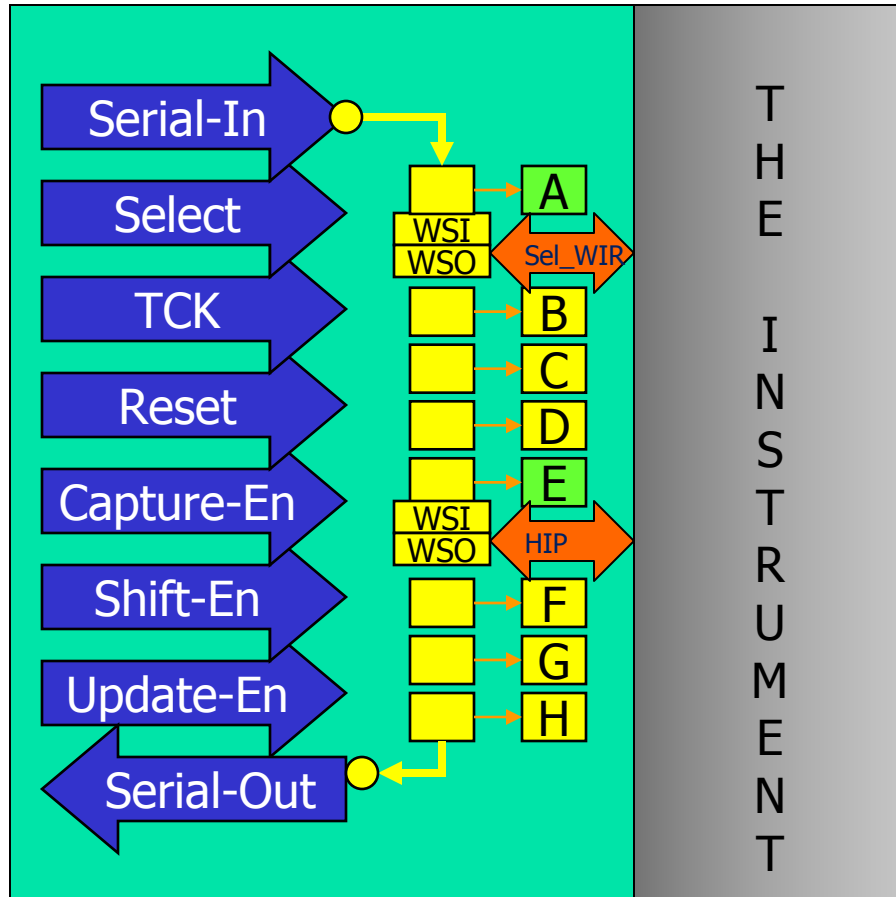


B, C, and D may also support Hierarchical Connections through HIPs

New Allowed 1687 Instrument Interface



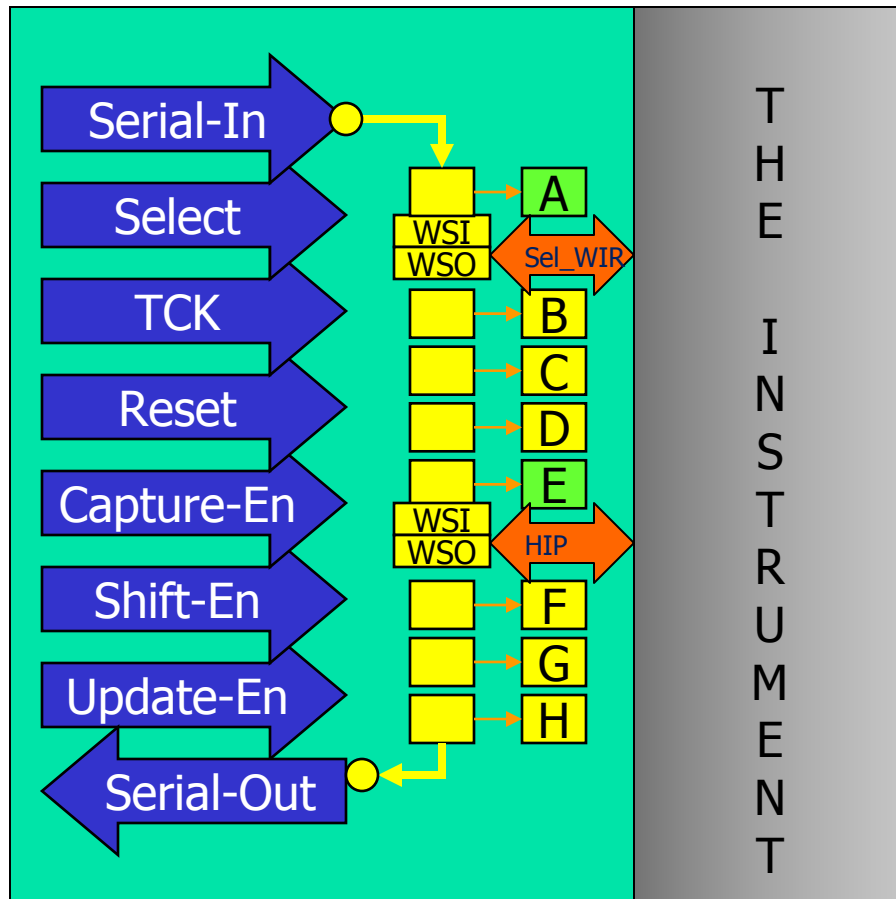
New Allowed 1687 Instrument Interface



Now the Instrument Interface matches the Gateway Interface

For Documentation Purposes this makes the development of the Hardware Description much easier – which Gateway SIB is the source – how many SIB bits is the Instrument Interface

New Allowed 1687 Instrument Interface

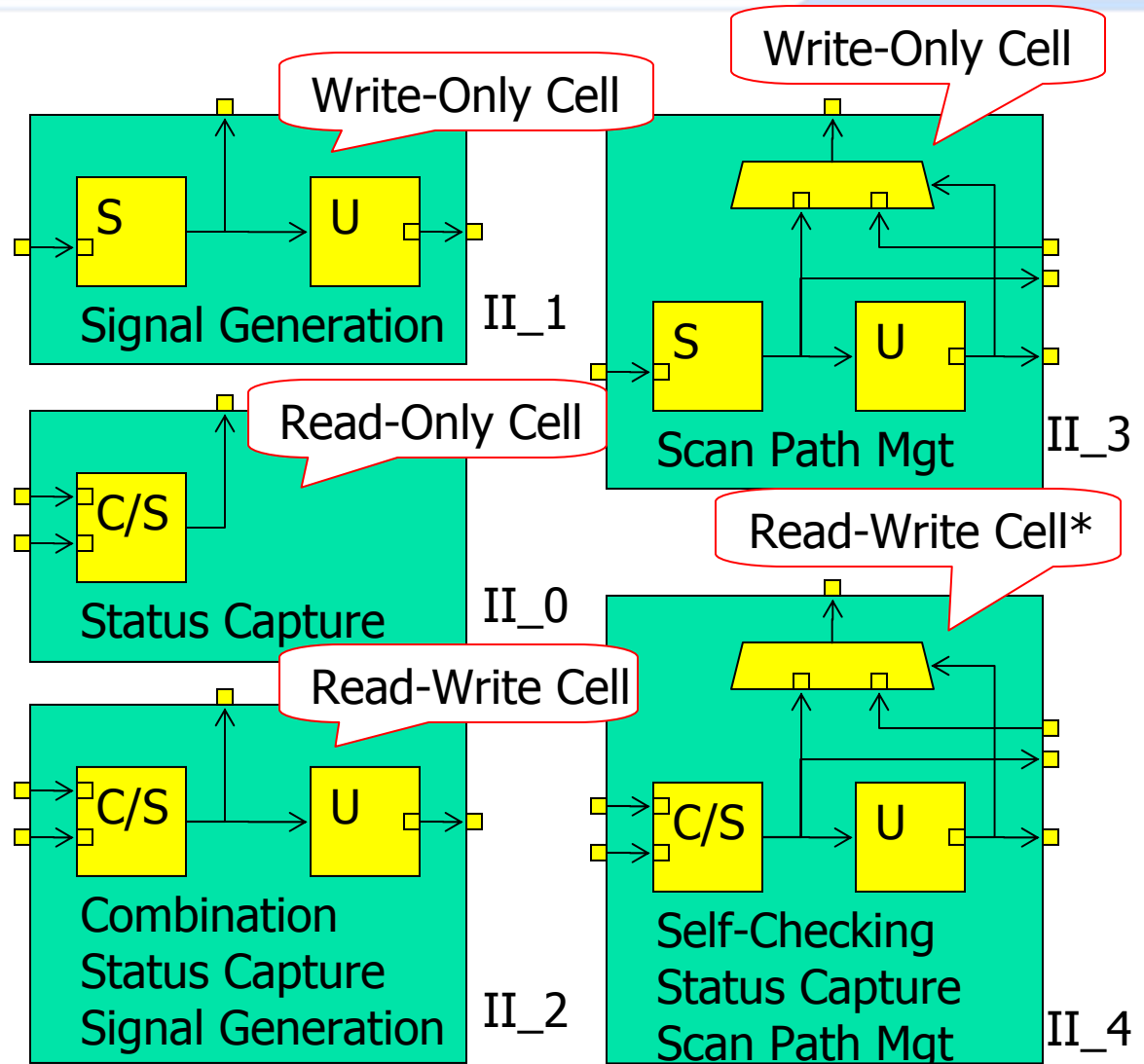
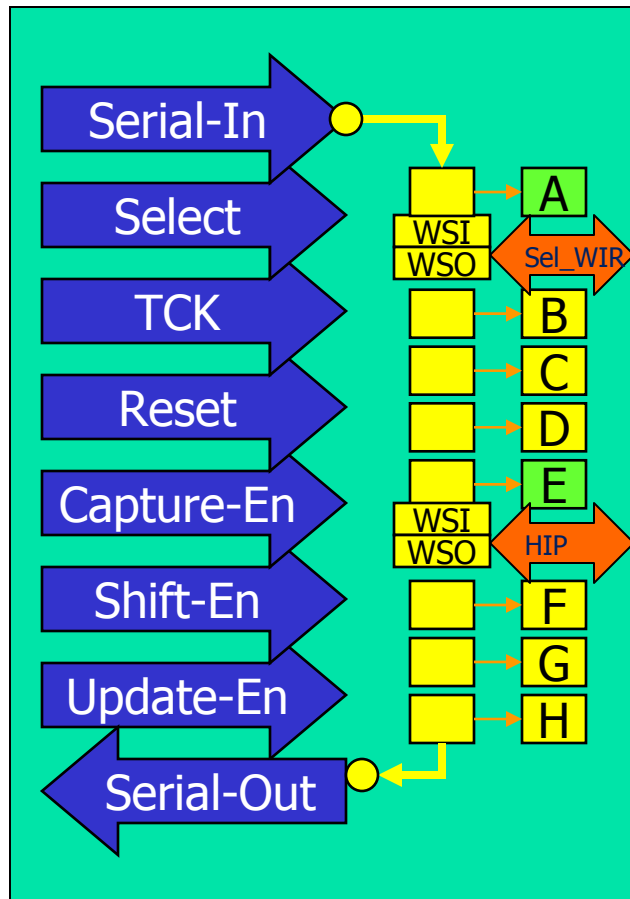


Note that the Instrument Interface may be comprised of just a few kinds of "bits":

- 1.A Signal Generation Bit
- 2.A Capture Status Bit
- 3.An "Add a Scan Path" Bit
- 4.A Combination Bit

The basic documentation should identify the location of the interface in the architecture, which bits are associated with which instrument, and the type of bit at each bit location in the interface

1687 Instrument Interface Bit Definitions



* May Require Capture to be only from U-cell output as Diagnostic

Bit Signal Categories

- The Bits in the Instrument Interface can have different purposes, but should fall into obvious classes
 - **Instrument Control Signals:** Inputs – signals such as Invoke, Reset, Pause, etc. that commit actions when asserted and require a “safe” or “off” state be defined
 - **Instrument Configuration Signals:** Inputs – signals such as Mode1, Retention_On, Enable_Diagnostic, Algorithm_Select, etc. that place the instrument into an operating mode or configuration (which implies different input signals or registers and/or output signals/values or registers)
 - **Operation Status Signals:** Outputs – signals such as Done, Fail, Error_Out, Paused, etc. that report on the activity, action, or result of operating an instrument
 - **Configuration Status Signals:** Outputs – signals such as March_Mode, Ten_Chain_Mode, Diagnostic_Out_Enabled, etc. that report on the configuration of the Instrument
 - **Data Values:** Inputs or Outputs – signals such as Data_In, Data_Out, Memory-Background, Signature_Out, Mem_Diag_Data, LBIST_Seed, etc. that represents data required by, or produced by, the instrument
 - **Access Architecture Configuration:** Inputs – signals that may change the flow or length of active scan registers that are included in the access architecture

Bit Signal Standard Labels

- No matter what the instrument signals are actually named – some of them may fall into standard categories, and therefore, may have standard defining labels
- **Reset:** Input – control signal that will put the instrument into a default state or mode (from operation, paused, or stopped – may corrupt the last state or any data used or produced within the instrument)
 - **Load:** Input – control signal that enables a data or configuration register to pass data to the instrument
 - **Pause:** Input – control signal that gracefully suspends operation (does not destroy the state or corrupt the data) in such a way that operation may be resumed later
 - **Resume:** Input – control signal that gracefully resumes suspended operation from a paused state
 - **Stop:** Input – control signal that will suspend operation, but with no requirement to resume or to retain operating state; there may be a requirement to retain “reporting data” at the point of stopping
 - **Run:** Input – control signal that invokes the operation or function of the instrument
 - **Private[n]:** Input – control or configuration signals that are not to be documented
 - **Enable_Output:** Input – configuration signal that enables an output signal or port
 - **Enable_Input:** Input – configuration signal that enables an input signal or port
 - **Open:** Input – a scan path management bit that opens a scan path
 - **Close:** Input – a scan path management bit that closes a scan path
 - **Bypass:** Input – a scan path management bit that enables a bypass path

Example Instrument Interface Definition

Bit #	Bit Type	Dir	Signal Name	Signal Type	Safe	Dir	Signal Name	Signal Type	OK	Link
5**	II_1	In	Reset	InstrControl	0					
4**	II_1	In	Reset	InstrControl	0					
3	II_3	In	SelectAddr	Access Config	0					AReg#
2	II_3	In	SelectData	Access Config	0					DReg#
1	II_2	In	RunBIST	InstrControl	0	Out	Fail	OpStatus	0	
0*	II_4	In	SelectAlgo	InstrConfig	1	Out	Done	OpStatus	1	

II_0 = Static Status Capture Bit

II_1 = Static Signal Drive Bit

II_2 = Combination Drive/Capture Bit

II_3 = Scan Path Add-In Bit

II_4 = Combination Scan Path Add-In/Capture Bit

* Bit 0 is the bit closest to WSO

**Bits 4-5 represent an encoded signal – 2 bits to create reset

#AReg and DReg are separate registers with tables like this one

A-Reg is a 32-Bit Register that is added in between bit-3 and bit-2 when bit-3 is asserted with a logic 1;

similarly D-Reg is also a 32-Bit Register added between bit-2 and bit-1 when bit-2 is asserted with a logic 1

But “Serial and JTAG-Like” Isn’t Enough!!!

- Many have expressed concerned with Bandwidth and non-1149.1 Sequences
 - More instruments means more data – need higher bandwidth for some instruments
 - Coordination between instruments – need instrument-to-instrument communication
 - Non-1149.1 Operations – need to conduct and describe TAP-asynchronous instrument operations such as a “fail flag” or “action” that occurs when a fail happens, not when the State-Machine happens to be in Capture-DR

Parallel Operations

➤ They already occur

- Only the serial shift-in and serial shift-out are not parallel operations
- Capture is a parallel load into the Shift/Capture Cell
- Update is a parallel load into the Update Cell
- JTAG Operations such as Extest, Intest, Clamp, HighZ are parallel operations

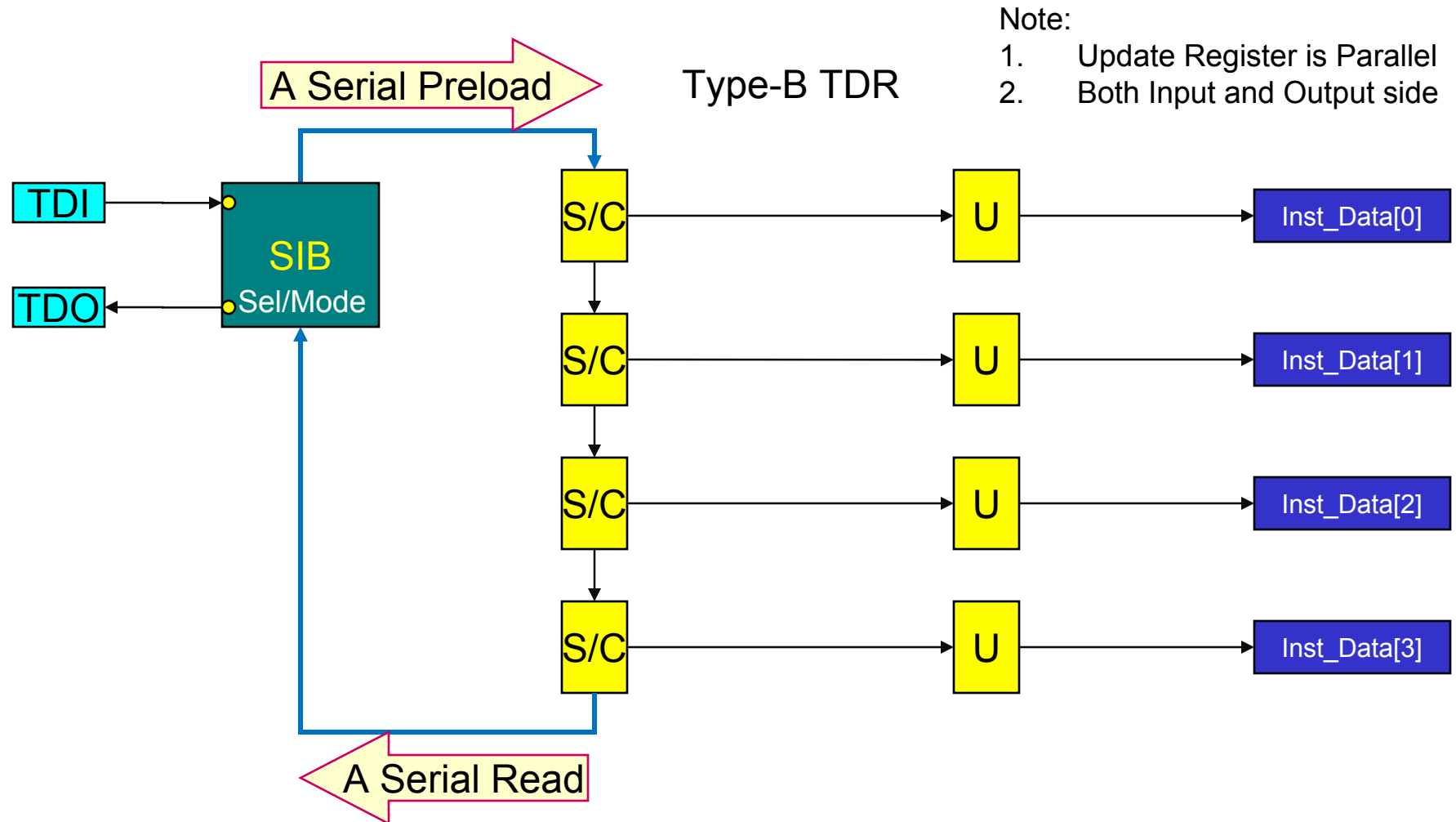
➤ To make use of these operations, terminology needs to be defined

- **Read:** currently capture+scan-out of the Shift/Capture Cell
- **Write:** currently scan-in+update of the Update Cell
- **TAP Synchronous:** Read or Write aligns with State-Machine
- **TAP Asynchronous:** Read or Write not aligned with State-Machine
- **Data-Operation:** Read or Write involving Data
- **Control-Operation:** Read or Write involving WIR

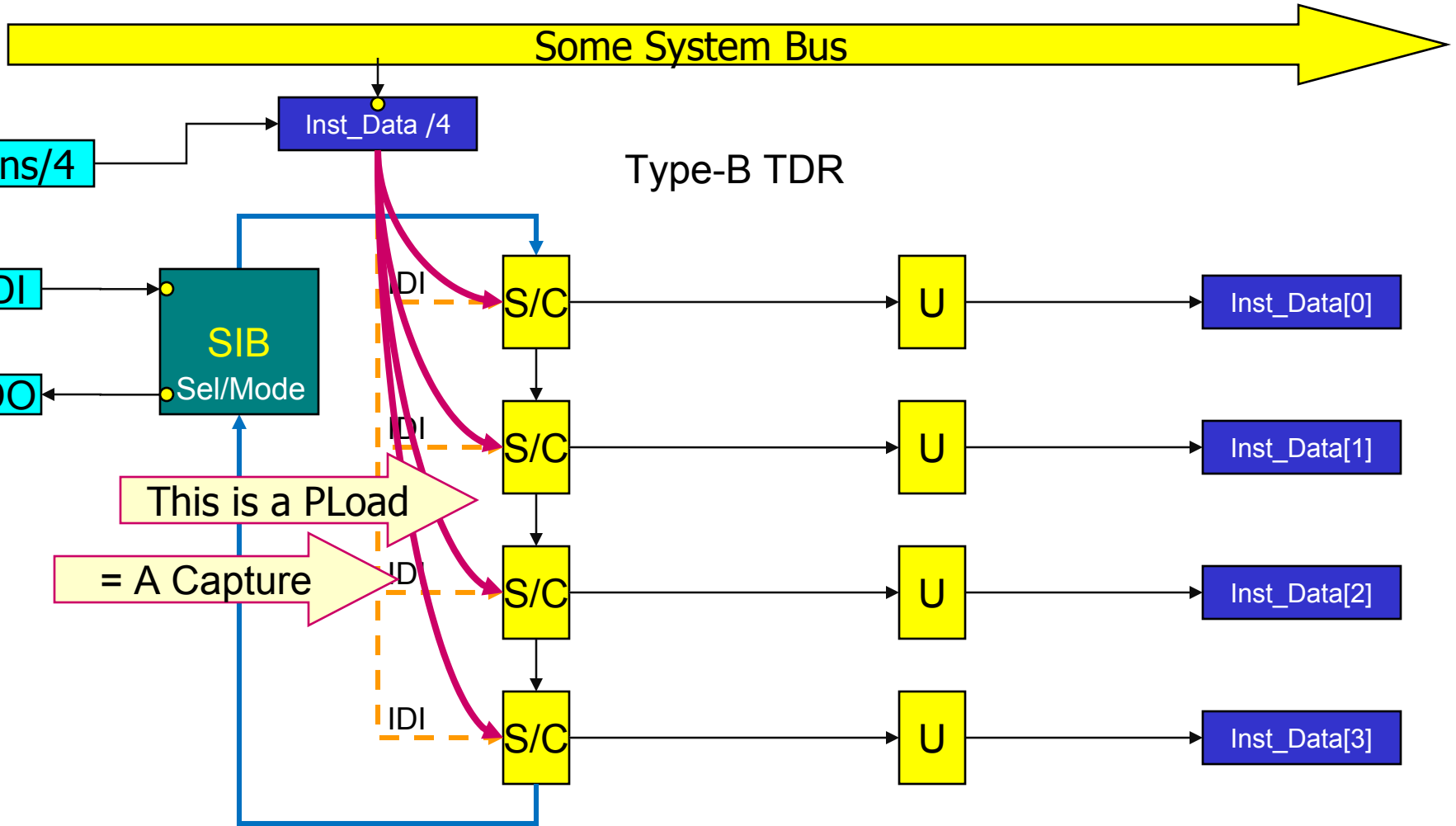
Parallel Notes

- How does this replace the Type-D Instrument Interface?
 - Loading/Reading/Writing the parallel registers directly turns a multi-clock-cycle serial operation into a single-clock-cycle operation – Bandwidth
 - Parallel Reads and Writes that are TAP-Synchronous use TCK to synchronize the data transfers
 - Parallel Reads and Writes that are not TAP-Synchronous can use any clock or trigger to synchronize the data transfers
 - Instruments can create the triggers that other instruments would use to conduct data/control transfers – facilitates Instrument-to-Instrument communication

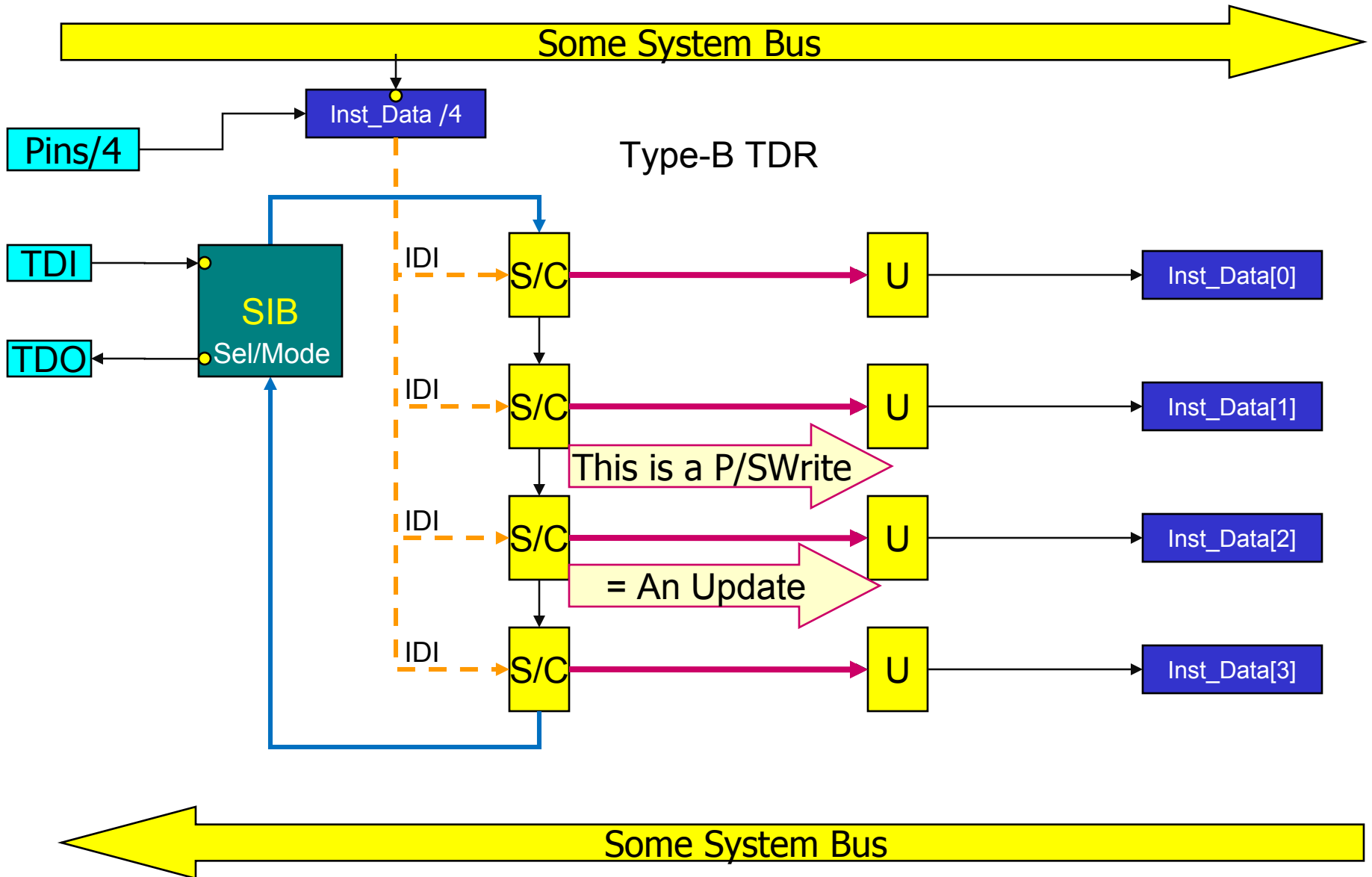
The Example TDR: The Serial Operations



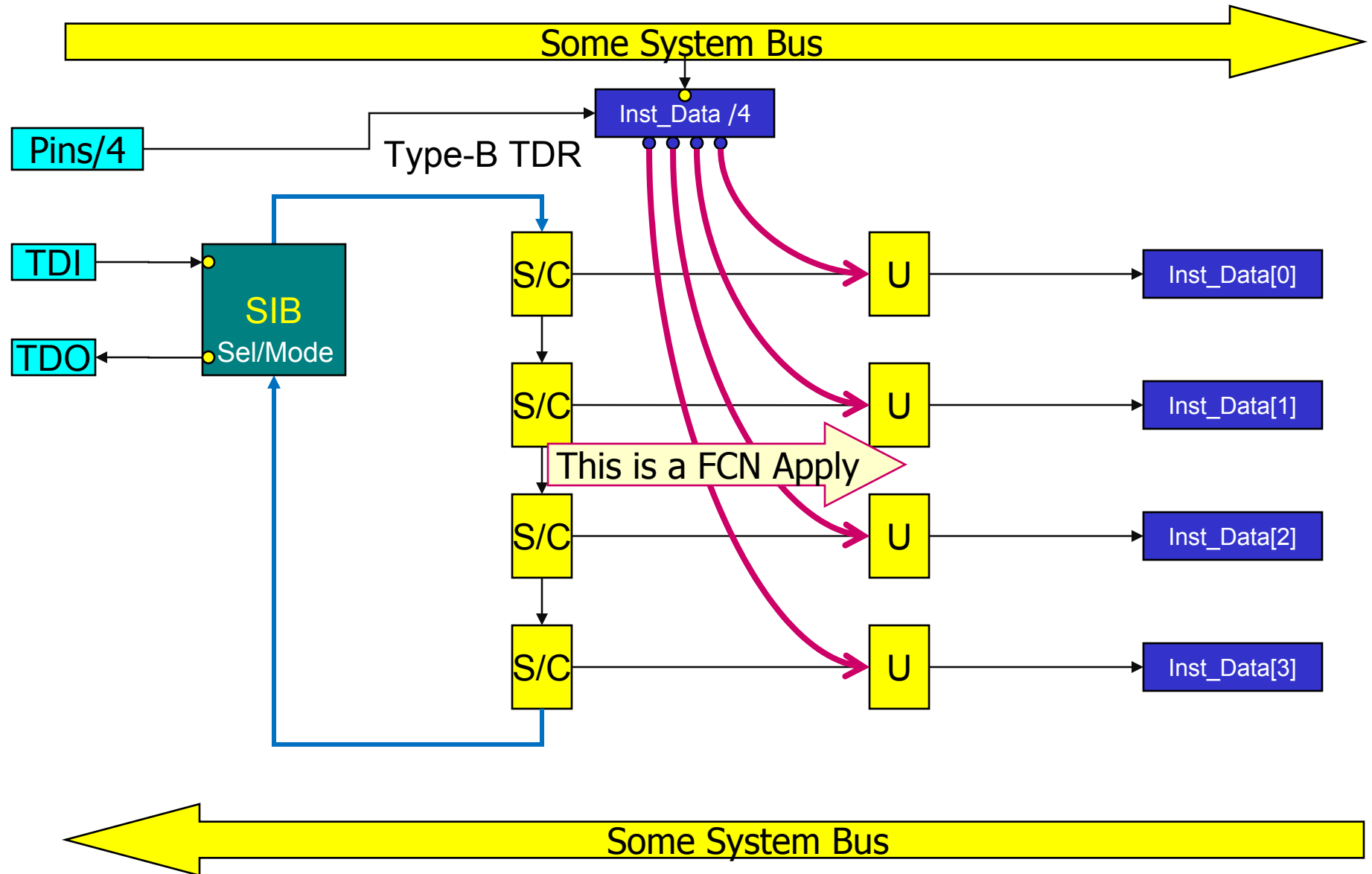
The Example TDR: A Parallel Load



The Example TDR: A Write



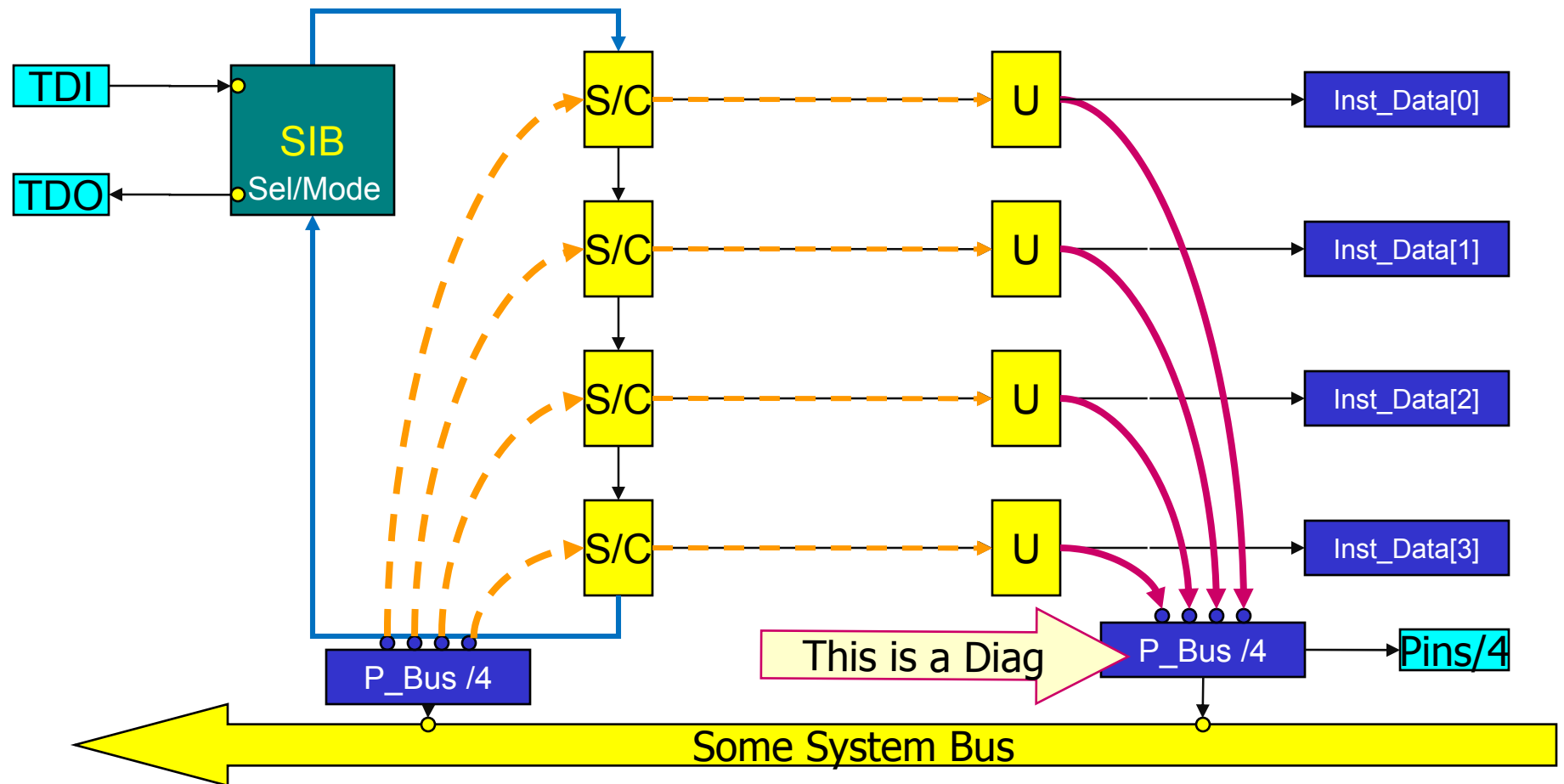
The Example TDR: A Functional Apply



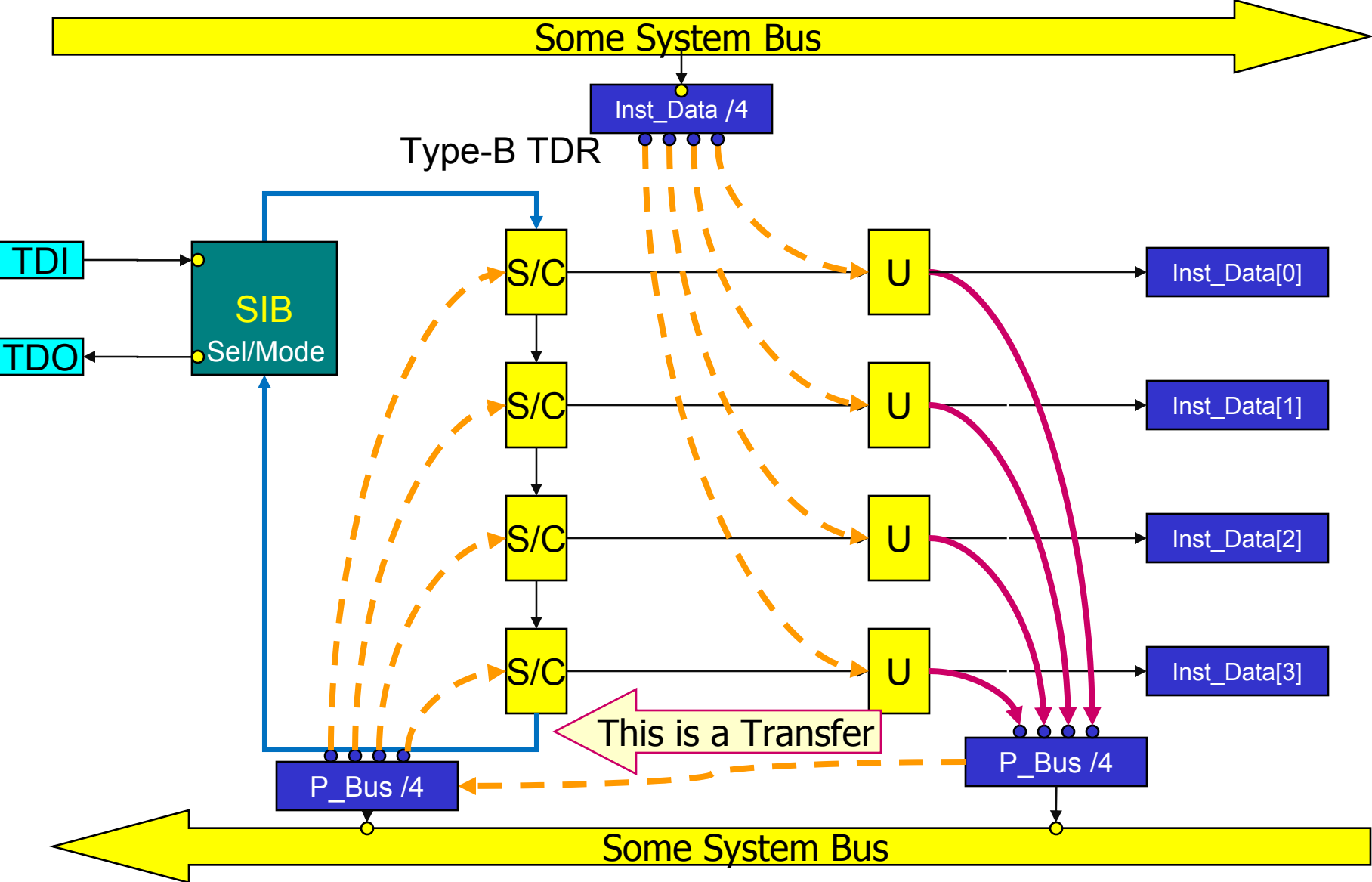
The Example TDR: A Parallel Diagnostic

Some System Bus

Type-B TDR



The Example TDR: A Parallel Transfer



Idea: Standard Instrument Commands

➤ Given the Instrument Bit and Signal Definitions, there are a limited set of standard operations that can be applied to define the use and sequencing:

- **Reset:** place the instrument into a default state or mode
- **Read OpStatus:** read the instrument operation status register
- **Read IData:** read an instrument data register
- **Read IConfig:** read an instrument configuration register
- **Read SPConfig:** read a scan-path configuration register
- **Read All:** read all status bits associated with an instrument

- **Write IControl:** write an instrument control register
- **Write IData:** write an instrument data register
- **Write IConfig:** write an instrument configuration register
- **Write SPConfig:** write a scan-path configuration register
- **Write SafeState:** write the defined safe-state to all instrument control registers
- **Write DefaultState:** write the defined default-state to all instrument registers

- **While:** looping construct designed to watch for the end of an activity/event/assertion
- **Wait-Until:** looping construct designed to watch for an event or assertion signal
- **For-Next:** looping/counting construct designed to count clocks or events
- **For-Each:** looping/counting construct designed to count instruments, bits, other objects
- **If-Then-Else:** decision making construct to allow flow control

Summary-Conclusions

- **Several Companies are already implementing JTAG Concepts**
 - They've already run into the “volume of instruments” problem
 - They are beginning to merge DFT, DFD, DFY into Design-for-Access
 - They are already having “efficiency” and “scheduling” problems
- **The concepts presented have been filtered through real designs and tradeoff criteria**
 - Separation of 1149.1 and P1687
 - The Gateway; various budget-based connectivity schemes
 - Instrument Interfaces; parallel data transfers, instrument-coordination
 - Architecture implementation is still very simple
- **The committee work now is focused on Language:**
 - Describe the architecture (instrument interfaces, Gateways, TAP)
 - Describe instrument modes or features
 - Generation and Retargeting of vectors