Differences between IEEE P1687 and 1149.1-2013

Al Crouch

Chief Technologist
ASSET InterTech, Inc.

Jennifer Dworak

Southern Methodist
Unversity

Is there a Fight Brewing?

- There is a perception that there are a number of IEEE Test Standards that overlap significantly
  - Is this true?
  - How did it happen?
  - Is this a problem?

So which standards are involved?

- 1149.1 was the original Test Standard
  - Started because of the surface mount packages (leads not accessible with probes)
  - Logic included in chips to assist with *board test*

- 1500 was created when *portable cores* became popular
  - How to include boundary scan and groups of test instructions per core
  - How to include several core within a single chip that is only supposed to have one 1149.1 TAP and TAP Controller

- P1687 was created when *embedded instruments* became popular
  - How to increase the number of embedded items accessed
  - How to create access networks that allow engineering tradeoffs

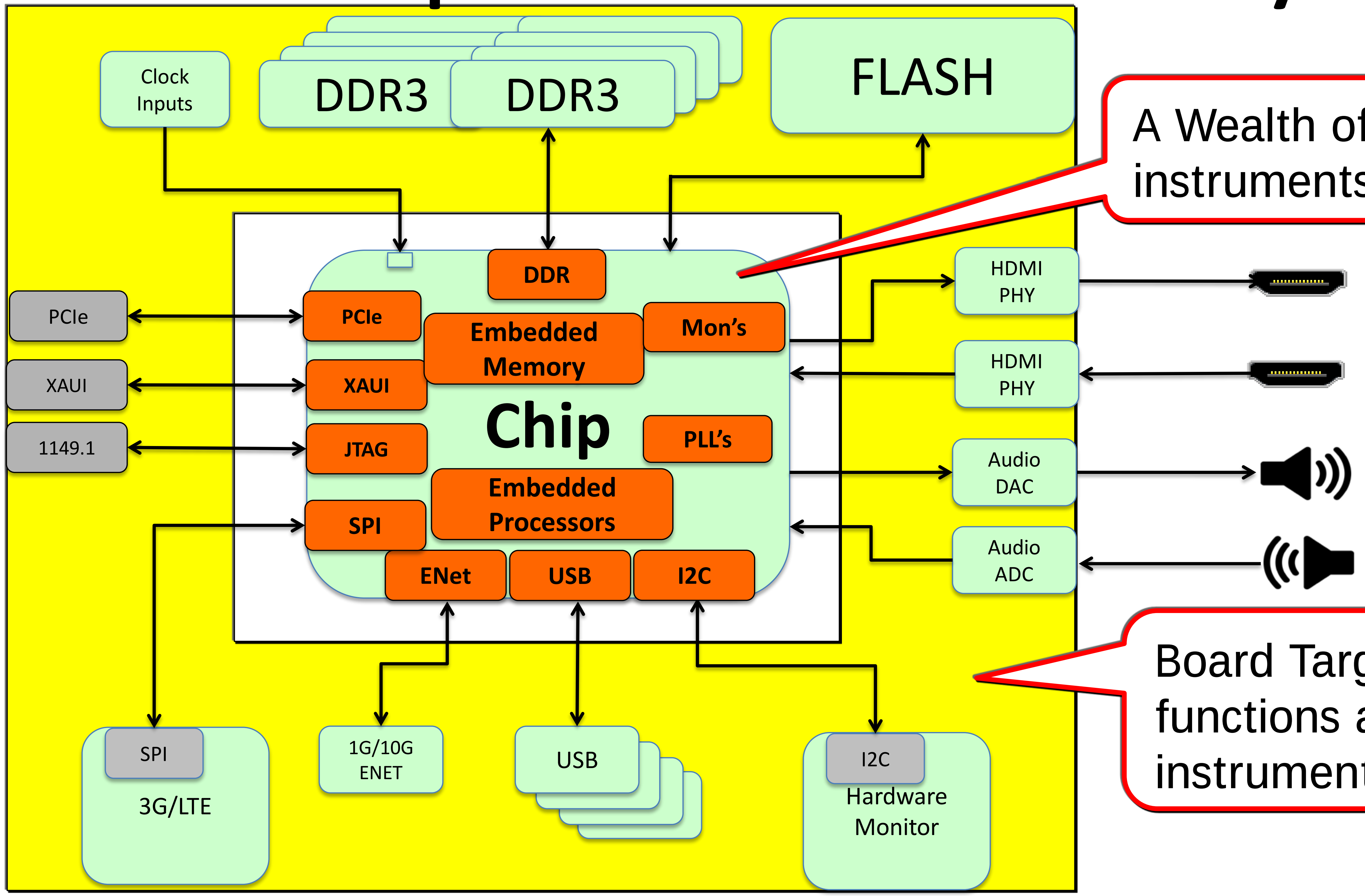
What do Chips look like Today?

Most have IP and embedded instruments

Most have high speed I/O with protocol and tuning

Chip may be an FPGA, ASIC, SOC, or Proc.

Common Practice: use of JTAG to access embedded instruments



A Wealth of embedded instruments in Chips

Board Targets of functions and instruments in Chips

What are Embedded Instruments?

- Test
  - Scan, Scan Compression, MBIST, LBIST, PLL-BIST, etc…
- Debug
  - Trace, Capture Buffers, Assertions, etc…
- Monitor
  - Voltage, Temperature, Frequency, etc…
- Functional Configuration
  - PLL settings, Power Modes, Bus Configurations, etc…

What is the Context?

- 1149.1 historically is perceived as being inefficient for scalability
  - Does not easily accommodate design tradeoffs
  - Number of instructions grows linearly with instruments for “one-at-a-time”; exponentially with “all possible configurations”
  - TAP Controller becomes the “routing clog” if all instruction decode is within the TAP Controller

- 1500 provides a method to place the TDR’s and Instructions locally with complex embedded cores
  - Created a new description language (CTL) & vector language (STIL)
  - Had a separable interface (replaced TMS with Shift/Capt/Upd/Rst)
  - Can place instruction decode local to Core – but instruction selection ambiguity
  - Observed all 1149.1 Rules (with exception of Transfer) – separation of Data/Instruction

What is the Context?

- 1149.1 historically is perceived as being inefficient for scalability
  - Does not easily accommodate design tradeoffs
  - Number of instructions grows linearly with instruments for “one-at-a-time”; exponentially with “all possible configurations”
  - TAP Controller becomes the “routing clog” if all instruction decode is within the TAP Controller

- P1687 provides a method to place the TDR’s and Instructions locally with embedded instruments
  - Created a new description language (ICL) & vector language (PDL)
  - Has a separable interface (replaces TMS with Shift/Capt/Upd/Rst)
  - Can place decode local to instrument – change TDR length with DR-scans
  - Violates 1149.1 Rules – separation of Data/Instruction, fixed length TDRs

What is the Controversy?

- During the development of P1687 – which has gone on longer than expected…
- 1149.1 opened their Standard and started updating their capability to include the ability to handle the boundary scan ring crossing low power domains…
- …and members of 1149.1 attended P1687 meetings…
- …and 1149.1 finished their update first - including many capabilities that P1687 was trying to standardize…
- …so 1149.1-2013 now seems to be more about IC Test…

Is this true?

- 1149.1-2013 Camp says:
  - We can now do everything that P1687 plans to do…
  - We had PDL first and P1687 does not align…

- P1687 Camp says:
  - This is not true – there are still advantages to P1687
  - And since 1149.1 is different from P1687, PDL is different

What is the history of P1687?

- Current Effort: Access to embedded instruments through the JTAG TAP – we call it 1687.0

- IEEE Proposed Standard development since 2005
  - To deal with Embedded Instruments
  - Make Access more efficient (scheduling, concurrence)
  - Unify multiple Access Mechanisms under a Standard
  - Incorporate some Legacy Mechanisms
  - Reduce IC Test impact on 1149.1 Board Test Architecture

What P1687 is NOT?

- P1687 is NOT a highly-restricted specified-method of access – identified cells, fixed-length TDRs, defined standardized instructions with fixed functions

- P1687 is not (just) JTAG on the inside of the chip
  - Designed to allow tradeoffs for designers
  - Minimize area (cells/gates), routing (data, control)
  - Adjust access time, scan length, operation frequency
  - Adjust power consumption, activity level
  - Return 1149.1 to its board test purpose

Comment #1 (from one who was there)

- Original idea in 2005 at start of P1687 was to combat the growing IC Test/Debug Embedded Instrument impact on Board Test Purpose of 1149.1:
  - Keep IR short (as opposed to hundreds to thousands of instructions – or long one-hot bit IRs) – goal was to add 1 P1687 Instruction to standard set
  - Keep BSDL to need only Board Test material (as opposed to including the IC Test/Debug TDRs, Instructions, Comments, etc. in BSDL – that is why a separate ICL/PDL)
  - Enable the IC Test/Debug portion to handle flexibility, tradeoffs, scheduling, and more "IC usage" efficiency – and to engender portable IP by having IP Vectors
  - At the time, it was felt that the embedded IC Test/Debug instruments were items that were "don't cares" to the Board Test community

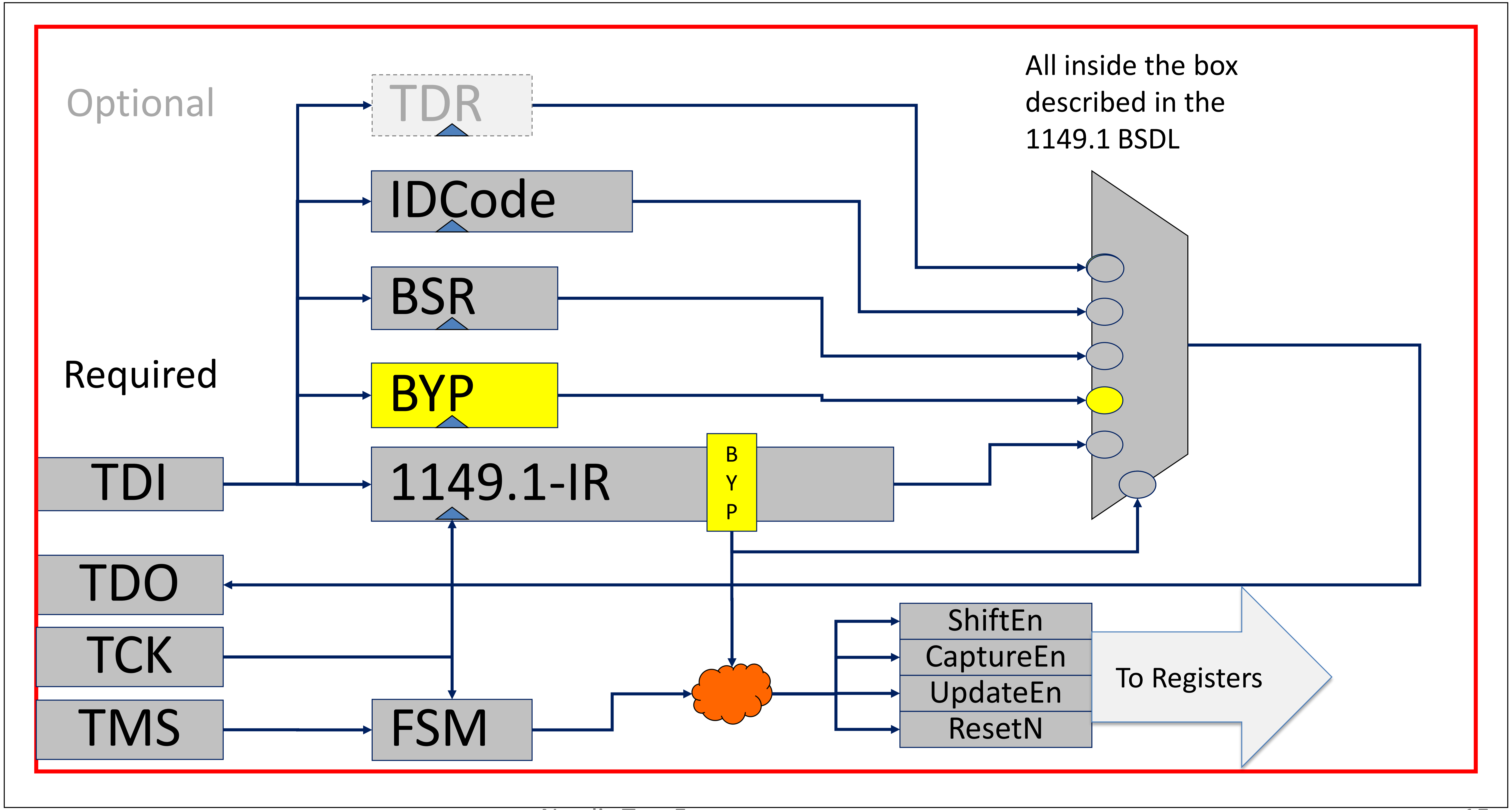
Comment #2

- The opening of **1149.1-2001** to add new capabilities undermined these P1687 goals:
  - The Perception by many was that 1149.1 was predominantly for Board Test
  - The Perception was that SVF was the defacto Vector language associated with 1149.1
  - The new Perception of 1149.1-2013 is that it is more for IC Test/Debug and less for Board Test – and Board Test simplicity has been sacrificed
  - The new Perception of 1149.1-2013 is that they only needed a small portion of PDL for Init-Sequence – but added a significant amount of PDL to migrate towards the IC Test/Debug goal

So, what do these different access choices look like?

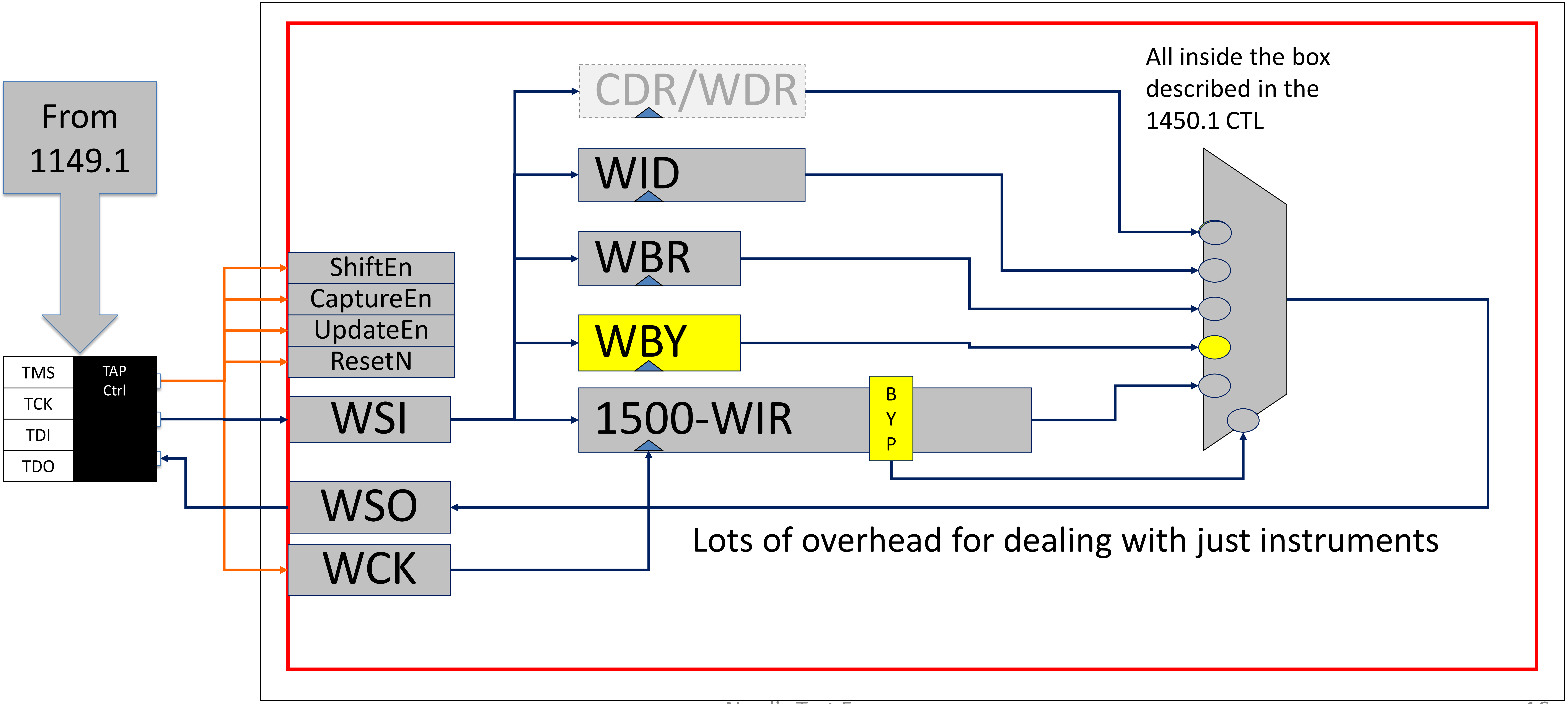
IEEE 1149.1 – Controller/Wrapper for Chips

- Instructions select registers and are used for scheduling



IEEE 1500 Wrapper for Cores

- Instructions select registers and are used for scheduling

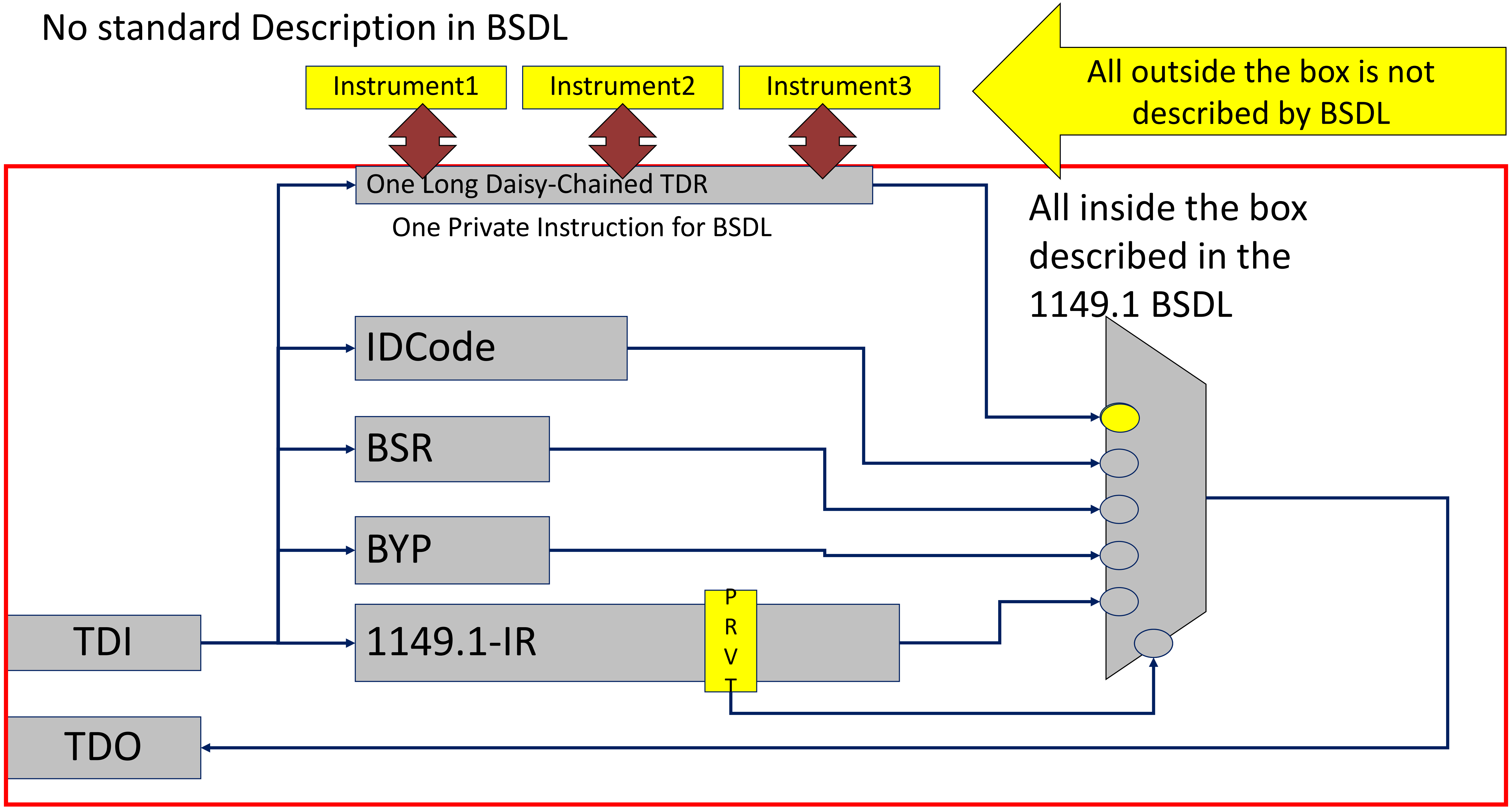


All inside the box described in the 1450.1 CTL

Lots of overhead for dealing with just instruments

1149.1/1500 has some Issues

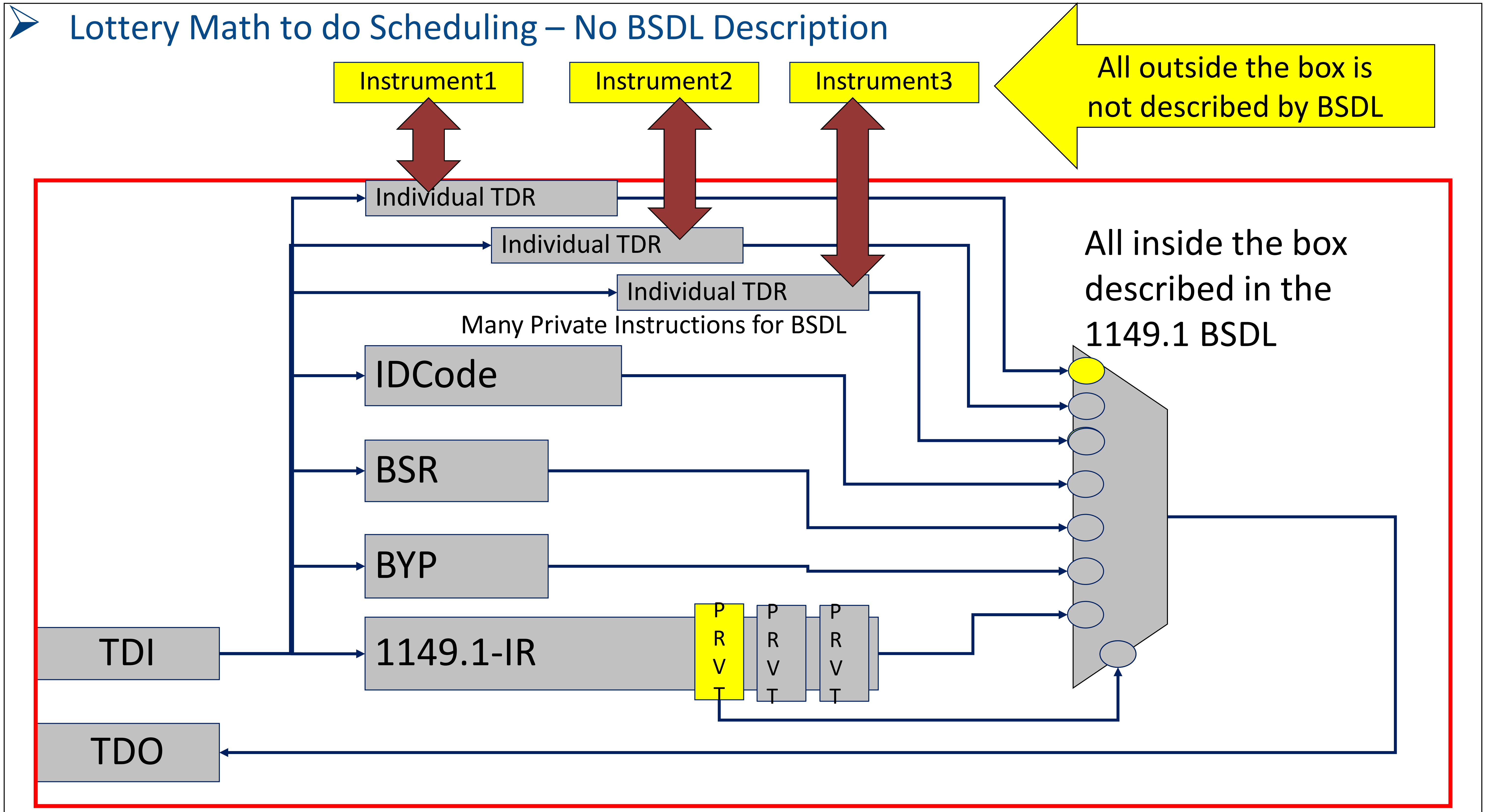
- High Risk if the Scan Path Breaks
- No standard Description in BSDL



Instrument1
Instrument2
Instrument3
All outside the box is not described by BSDL
One Long Daisy-Chained TDR
One Private Instruction for BSDL
All inside the box described in the 1149.1 BSDL
IDCode
BSR
BYP
TDI
1149.1-IR
PRVT
TDO

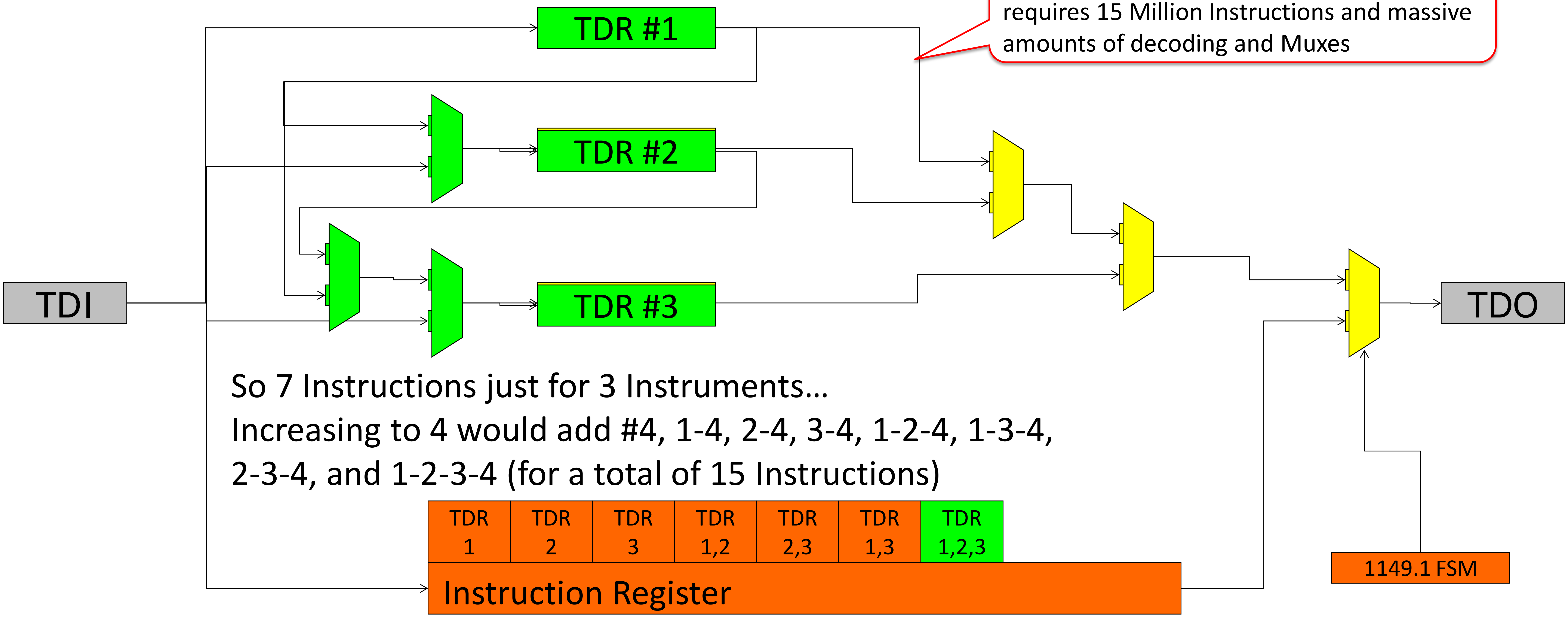
1149.1/1500 has Issues

- Lottery Math to do Scheduling – No BSDL Description


Instrument1
Instrument2
Instrument3
All outside the box is not described by BSDL
Individual TDR
Individual TDR
Individual TDR
Many Private Instructions for BSDL
All inside the box described in the 1149.1 BSDL
IDCode
BSR
BYP
TDI
1149.1-IR
PRVT
PRVT
PRVT
TDO

Example of Instruction Lottery Math

- Example of 3 Instruments

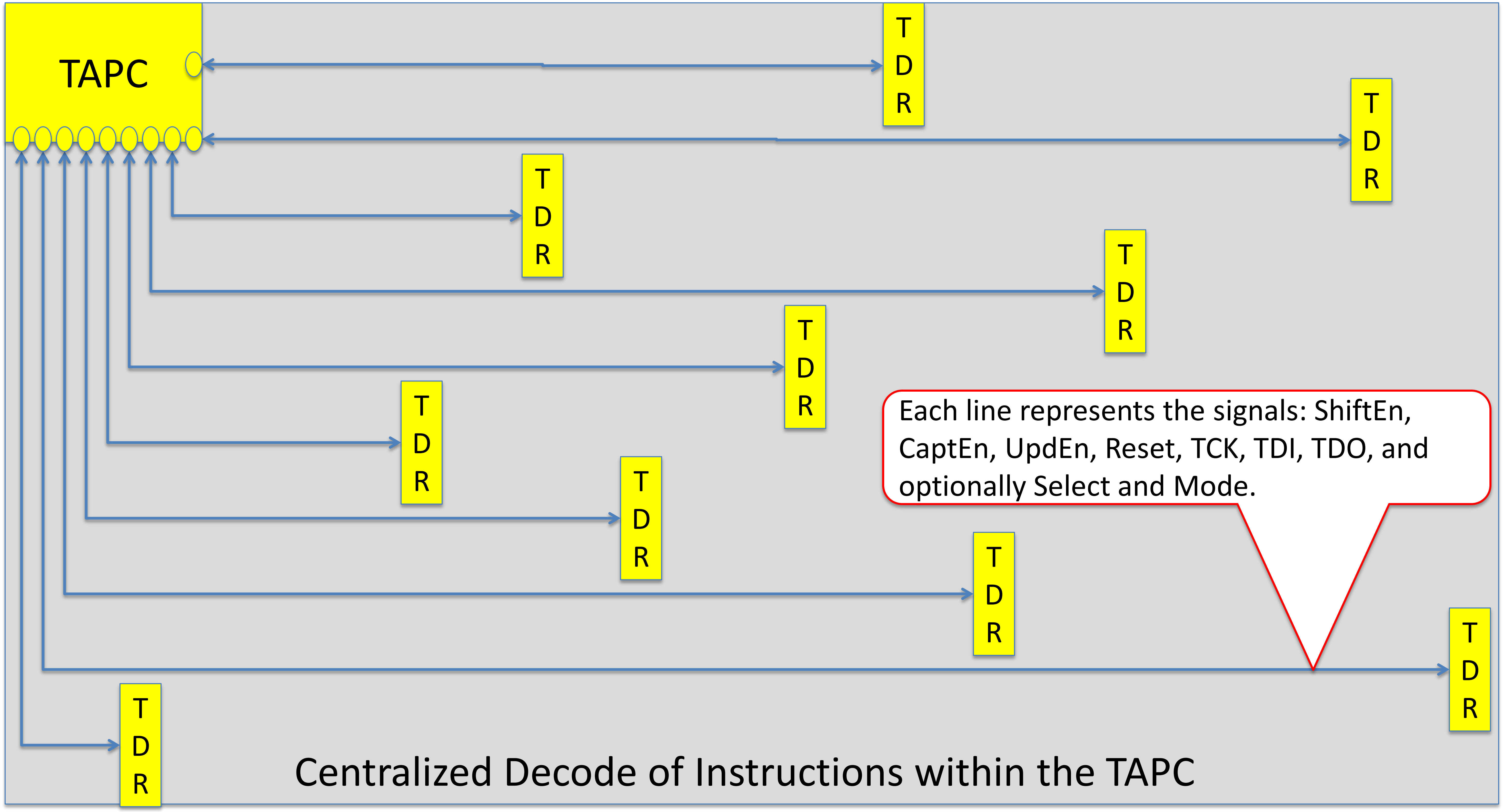


Scheduling just 6 instruments out of 50 requires 15 Million Instructions and massive amounts of decoding and Muxes

So 7 Instructions just for 3 Instruments…
Increasing to 4 would add #4, 1-4, 2-4, 3-4, 1-2-4, 1-3-4, 2-3-4, and 1-2-3-4 (for a total of 15 Instructions)

TDR 1	TDR 2	TDR 3	TDR 1,2	TDR 2,3	TDR 1,3	TDR 1,2,3

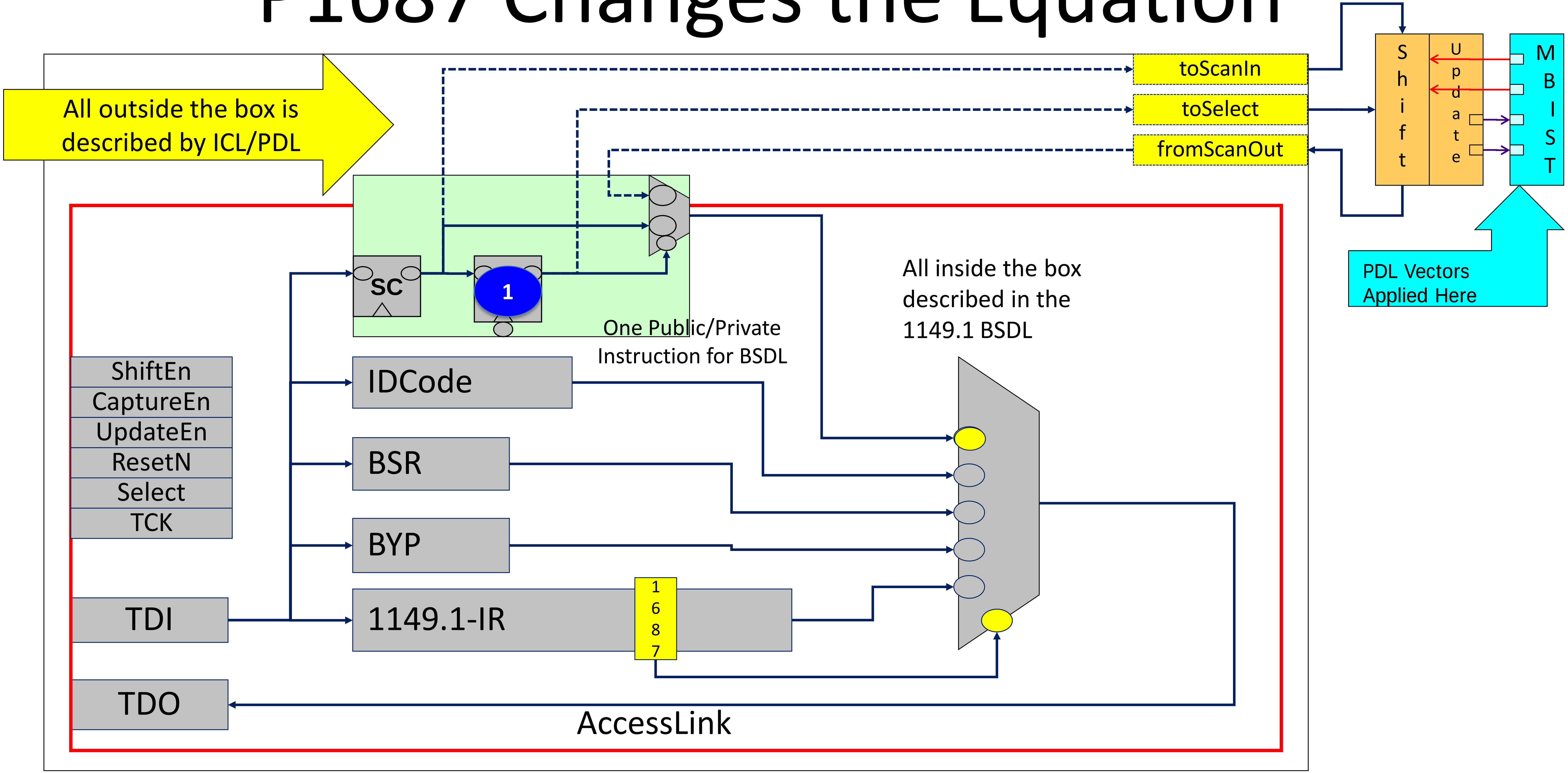
Physical Impact of Centralized Instructions



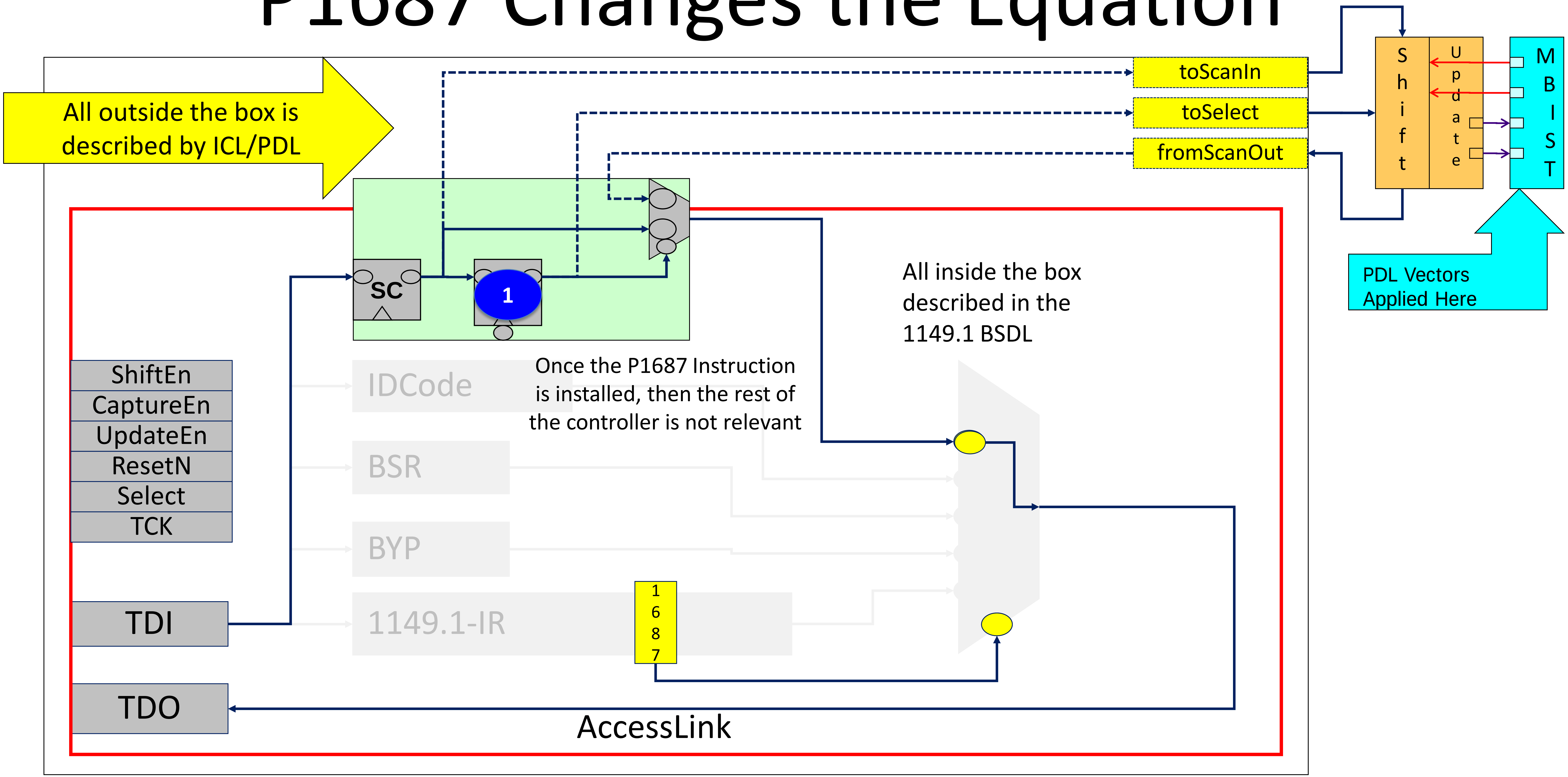
Each line represents the signals: ShiftEn, CaptEn, UpdEn, Reset, TCK, TDI, TDO, and optionally Select and Mode.

Centralized Decode of Instructions within the TAPC

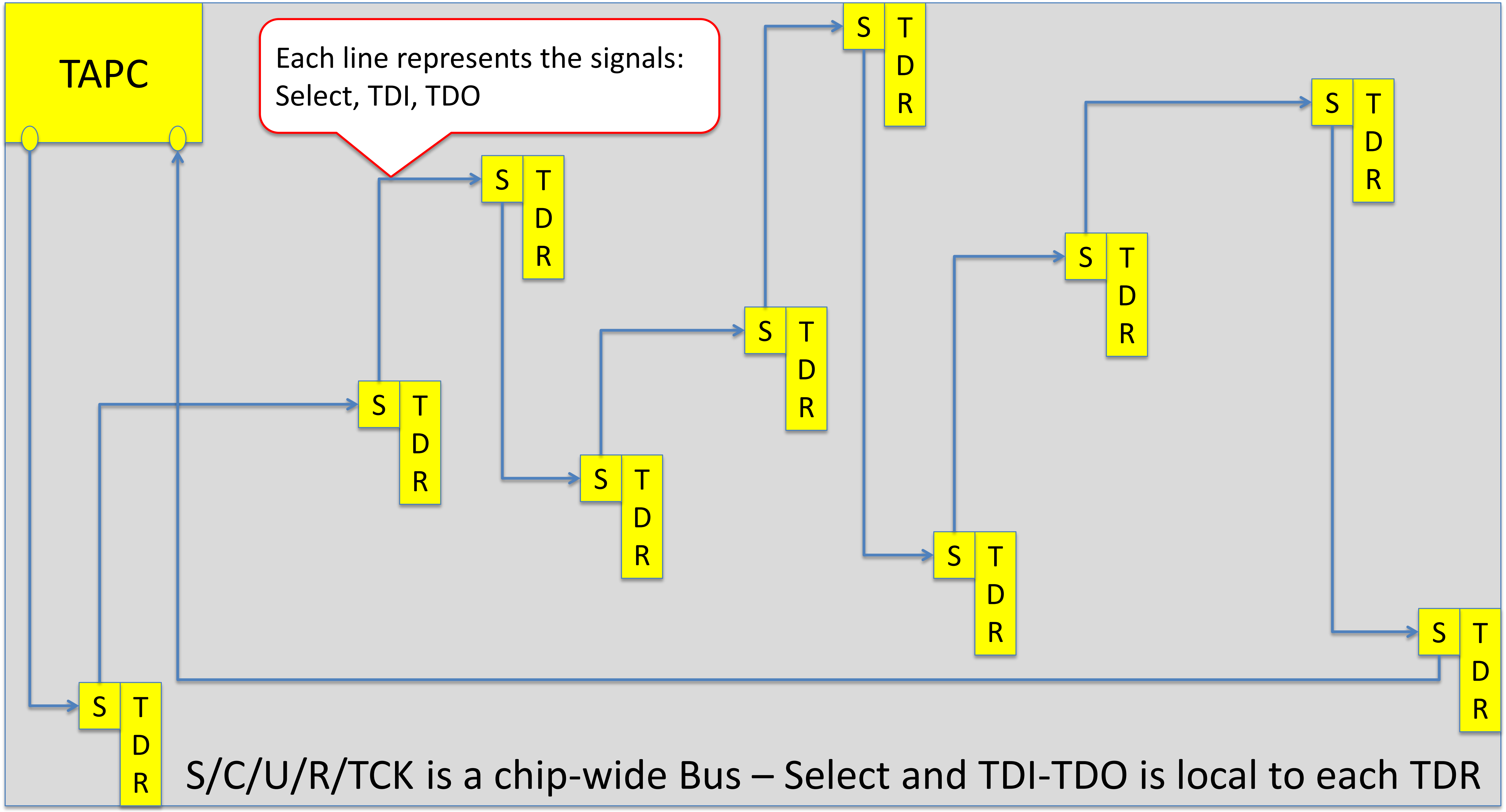
P1687 Changes the Equation



P1687 Changes the Equation


All outside the box is described by ICL/PDL
toScanIn
toSelect
fromScanOut
S h i f t
U p d a t e
M B I S T
PDL Vectors Applied Here
SC
1
All inside the box described in the 1149.1 BSDL
Once the P1687 Instruction is installed, then the rest of the controller is not relevant
ShiftEn
CaptureEn
UpdateEn
ResetN
Select
TCK
IDCode
BSR
BYP
1149.1-IR
1 6 8 7
TDI
TDO
AccessLink

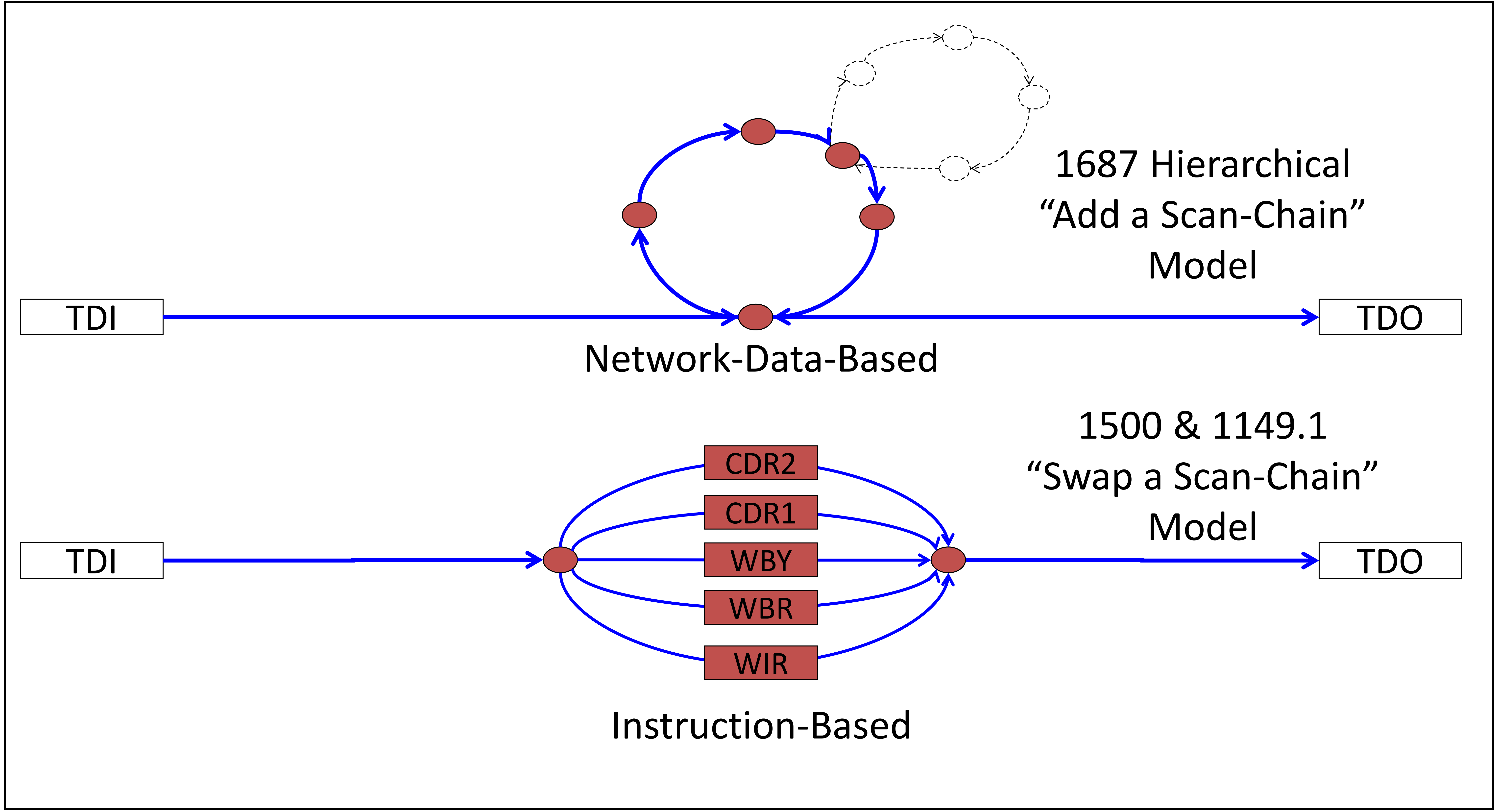
Physical Impact of Distributed Instructions



Each line represents the signals: Select, TDI, TDO

S/C/U/R/TCK is a chip-wide Bus – Select and TDI-TDO is local to each TDR

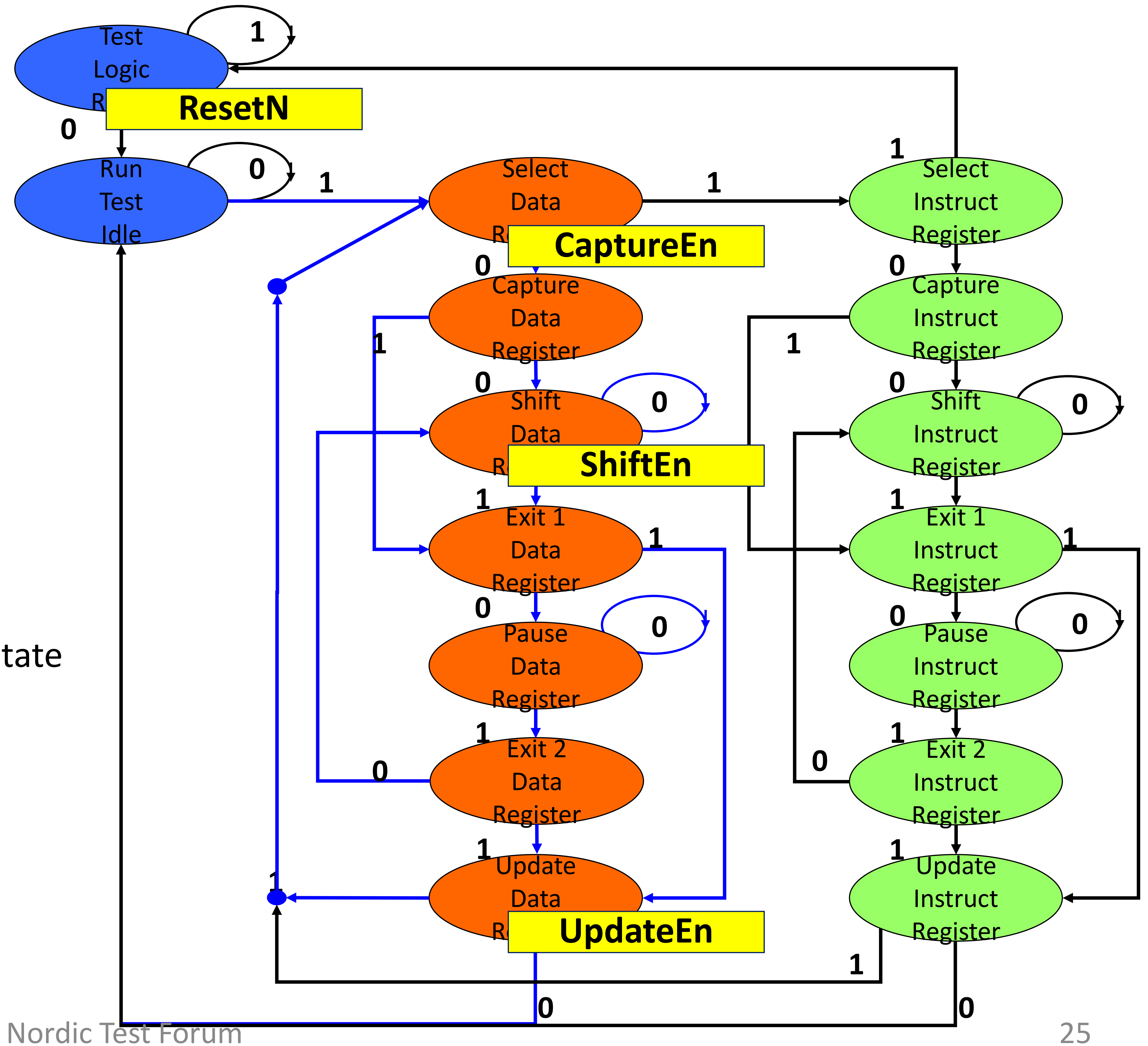
IEEE P1687 -vs- 1149.1/1500 Scan Paths



The 1149.1 Operation Sequence Generator

- Legal Sequences: those allowed by the compliant 1149.1 TAP FSM

1. Normal Event Order: a ScanDR
   [Operation = Capture-Shift-Update]
   [Zero-Bit-Scan = Capture-E1-Update]
2. All State Changes on Rising-TCK & TMS
3. Five 1’s on TMS goes to TLR from anywhere in the State Machine
4. All “Inputs and Samples” on Rising-TCK
5. All “Outputs and Updates” on Falling-TCK



The 1149.1 Operation Sequence Generator

- Legal Sequences: those allowed by the compliant 1149.1 TAP FSM

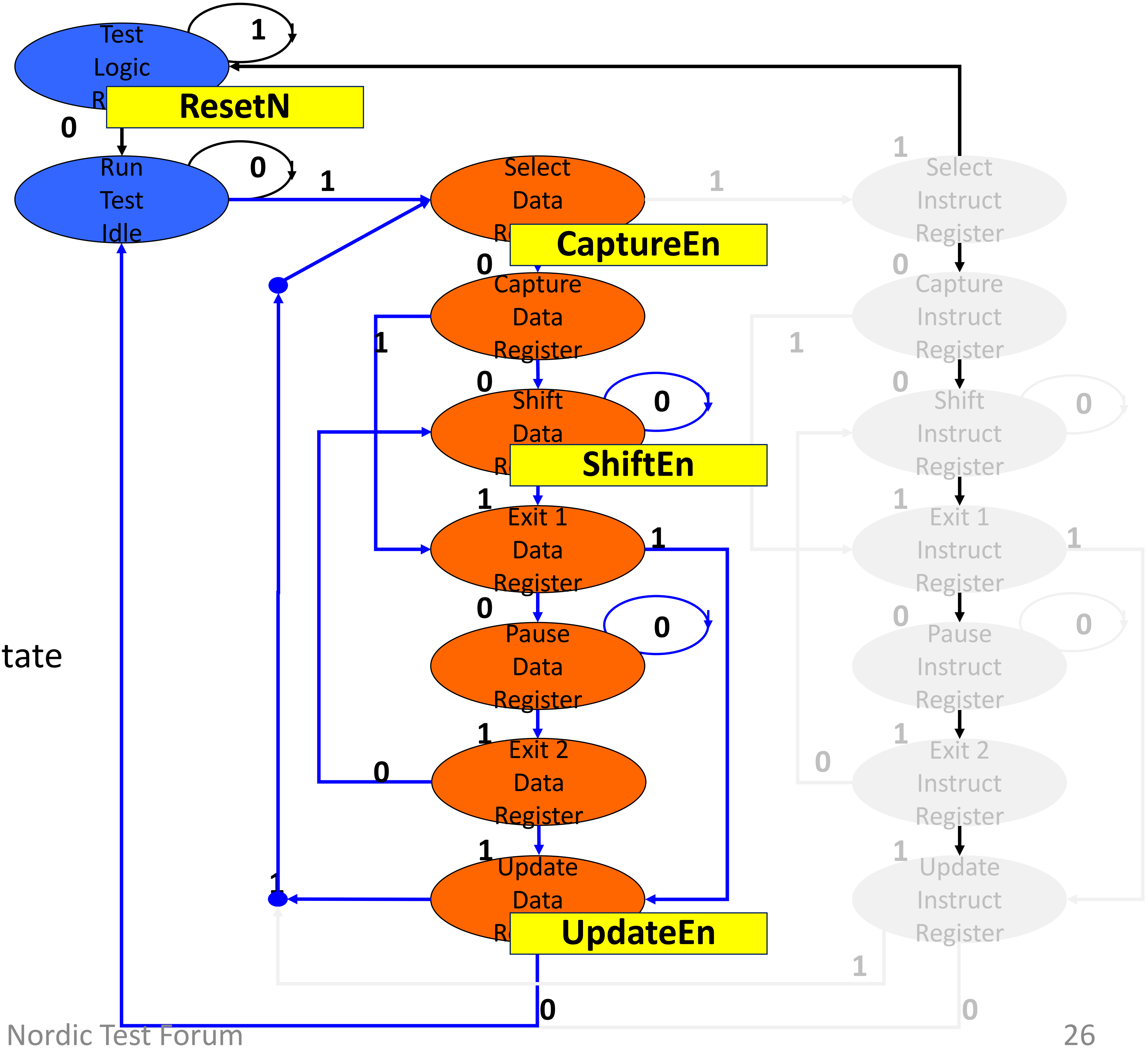
1. Normal Event Order: a ScanDR
[Operation = Capture-Shift-Update]
[Zero-Bit-Scan = Capture-E1-Update]

2. All State Changes on Rising-TCK & TMS

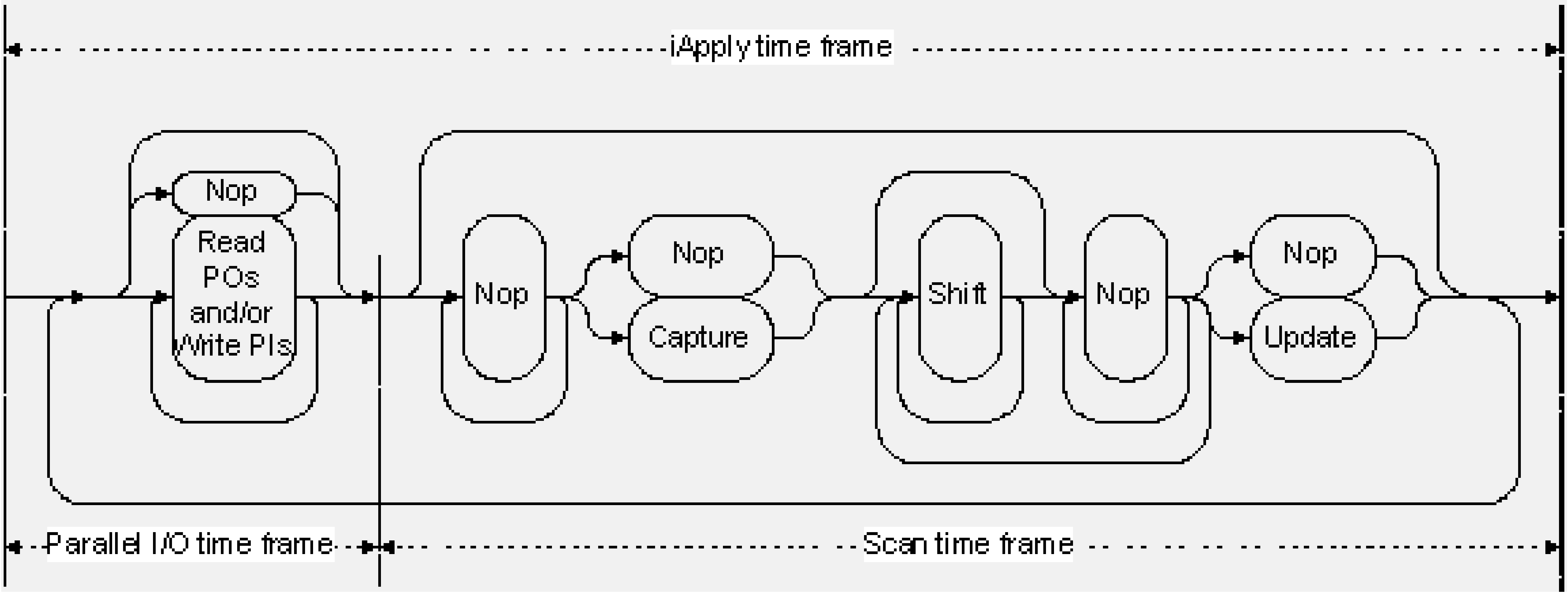
3. Five 1's on TMS goes to TLR from anywhere in the State Machine

4. All "Inputs and Samples" on Rising-TCK

5. All "Outputs and Updates" on Falling-TCK



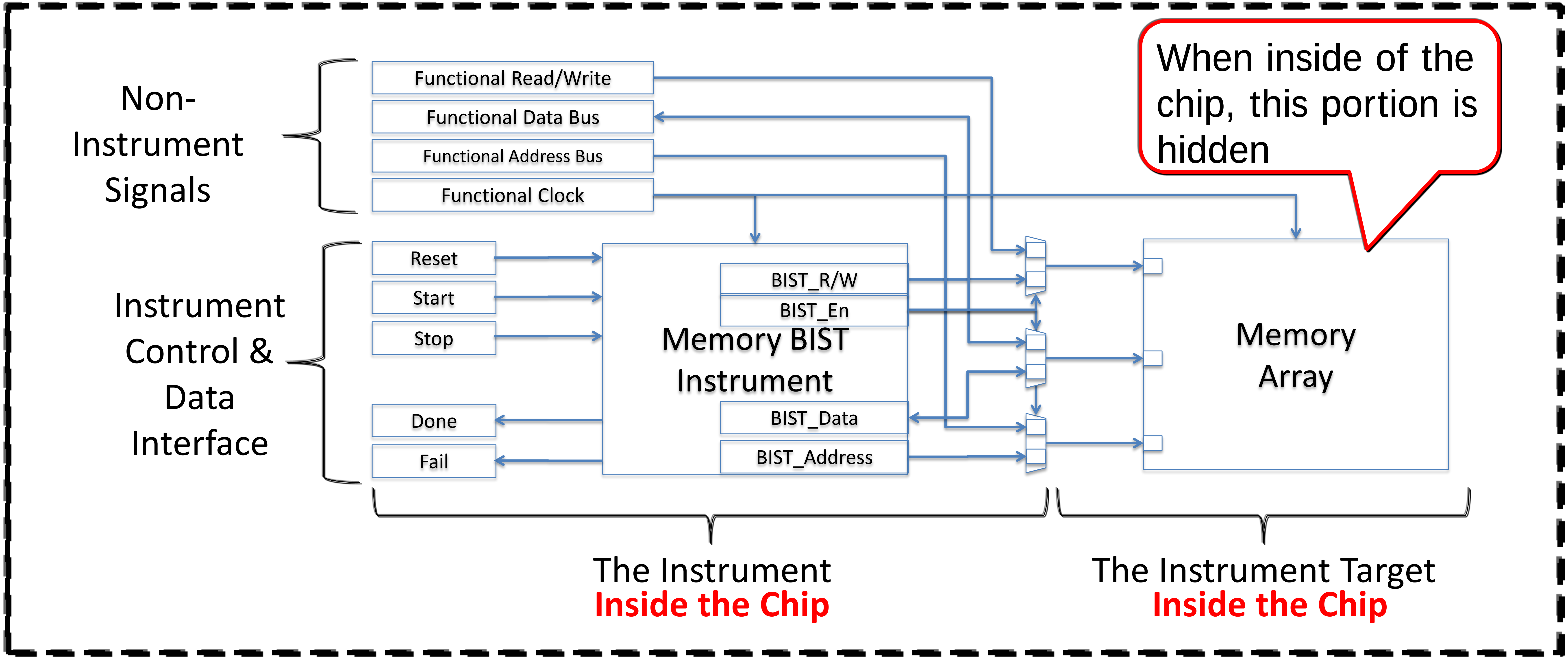
P1687 Operation Sequence



Like IEEE 1500 – the separable interface can be operated directly by ATE or other controllers
ATE sequence shown above with deny-capture/deny-update allowed

So, let’s talk about P1687 in the context of embedded instruments…

What is an Embedded Instrument?

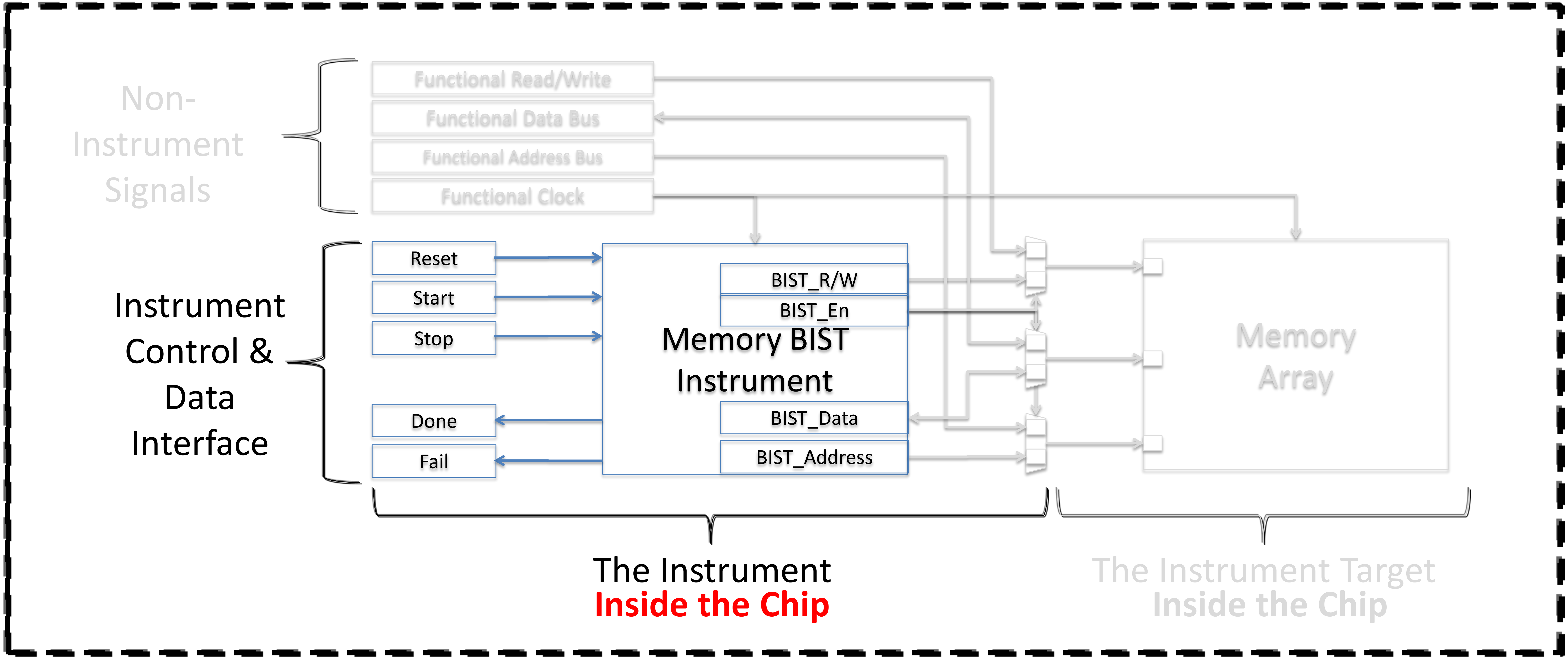


When inside of the chip, this portion is hidden

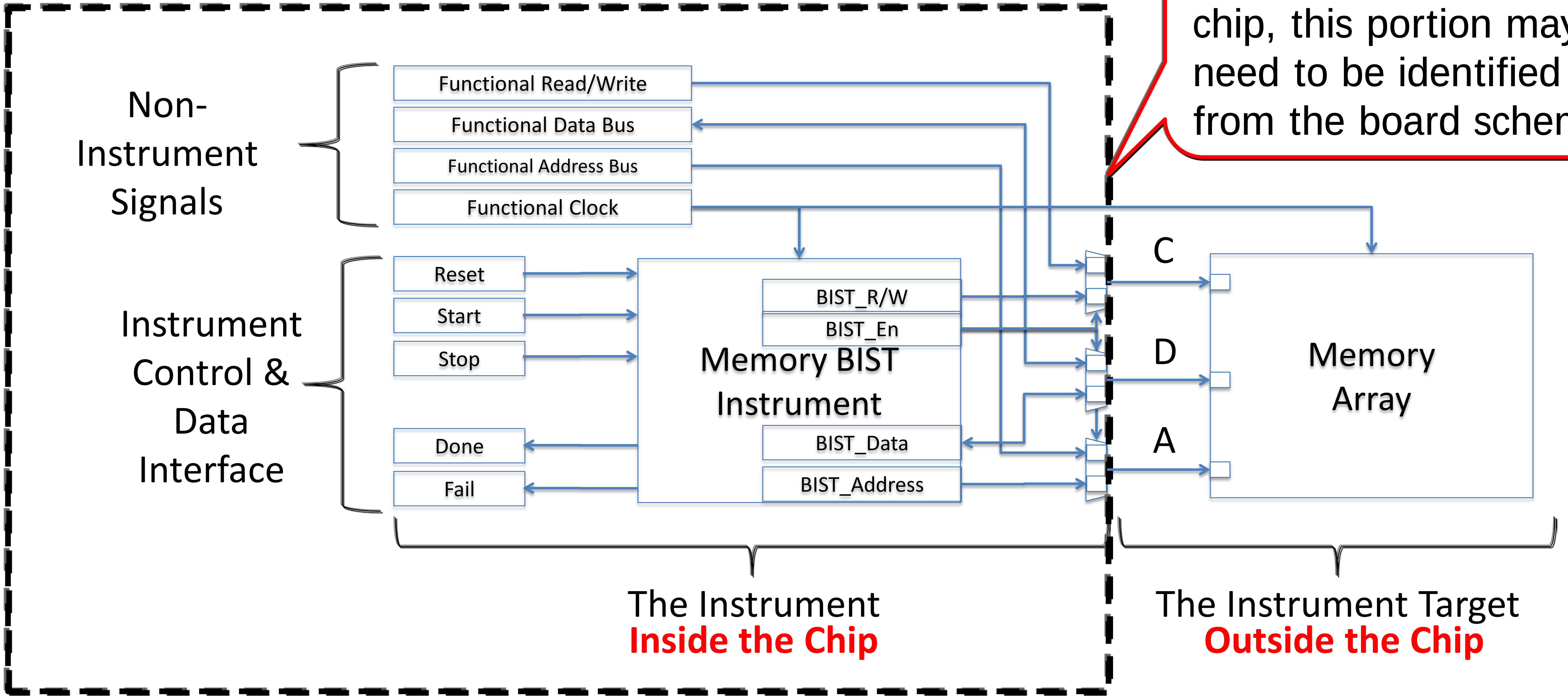
The Instrument
Inside the Chip

The Instrument Target
Inside the Chip

What is an Embedded Instrument?

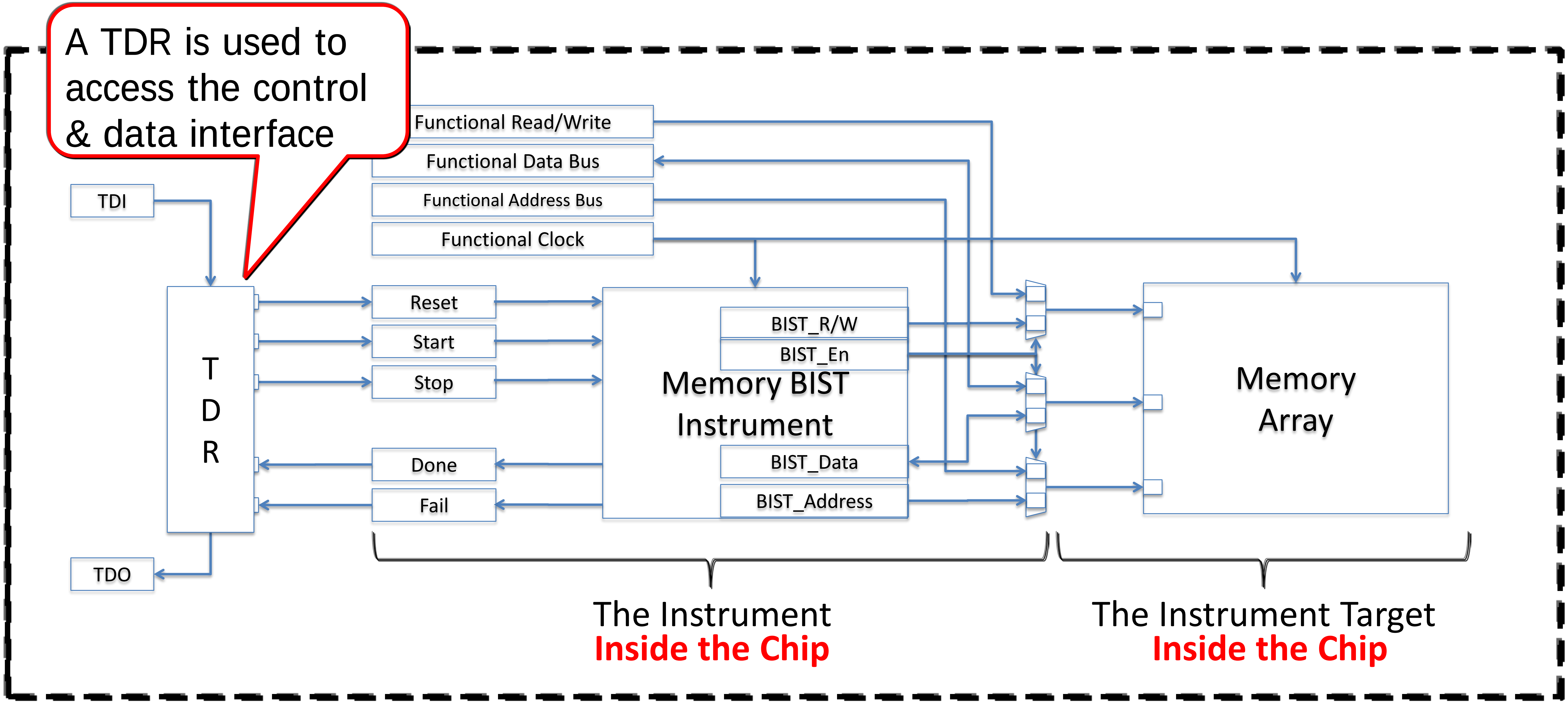


What is an Embedded Instrument?



When outside of the chip, this portion may need to be identified from the board schematic

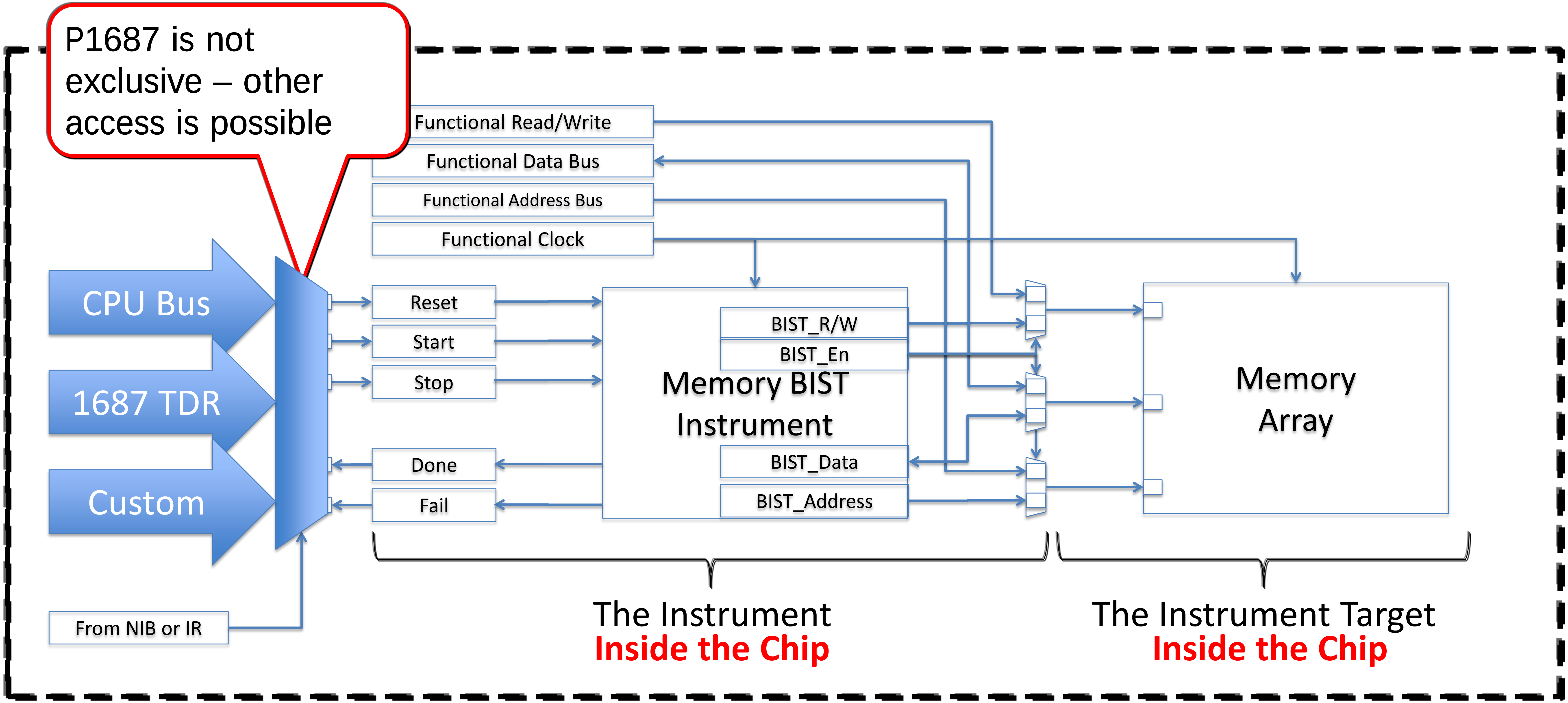
One way to Access an Embedded Instrument



A TDR is used to access the control & data interface

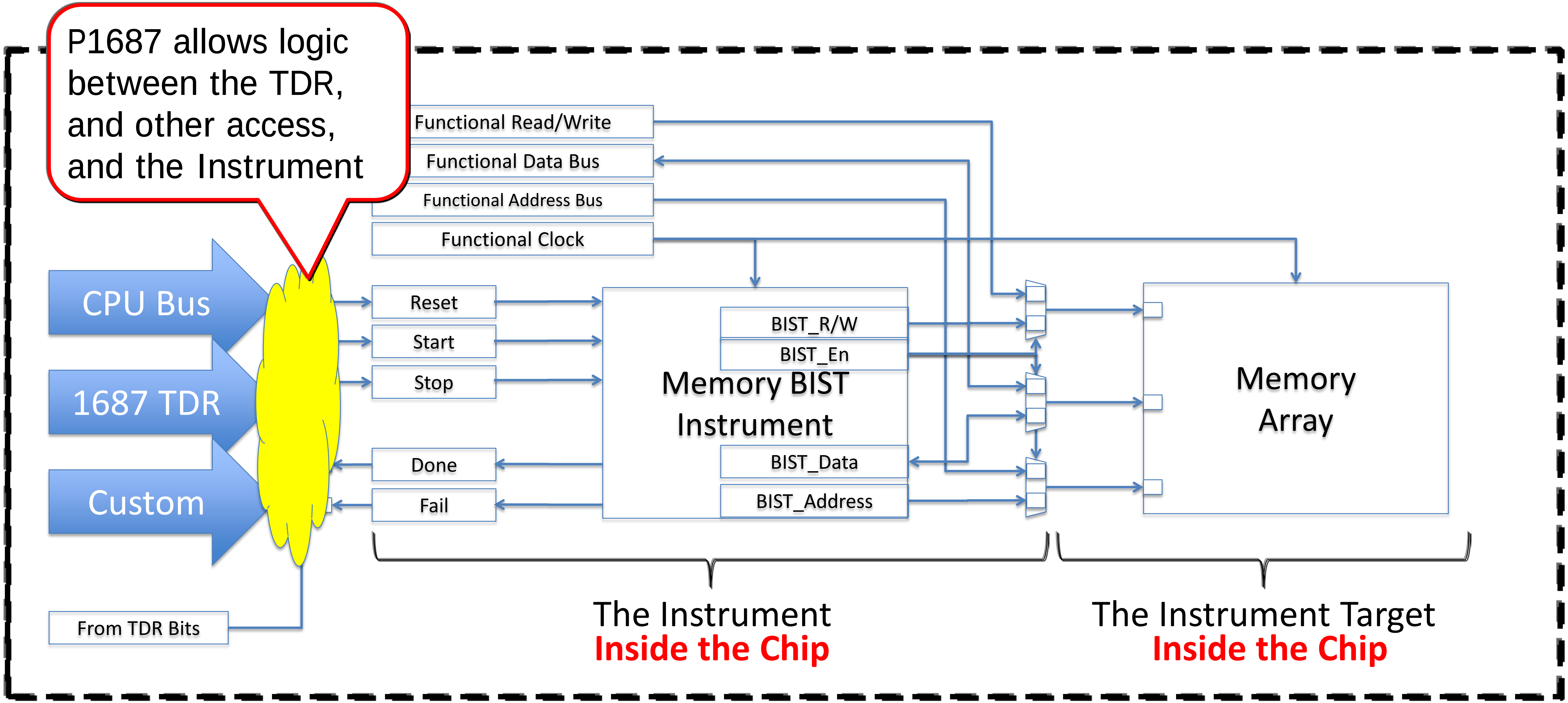
Both 1149.1-2013 and P1687 may access the embedded instrument with a TDR.

Other Access to an Embedded Instrument

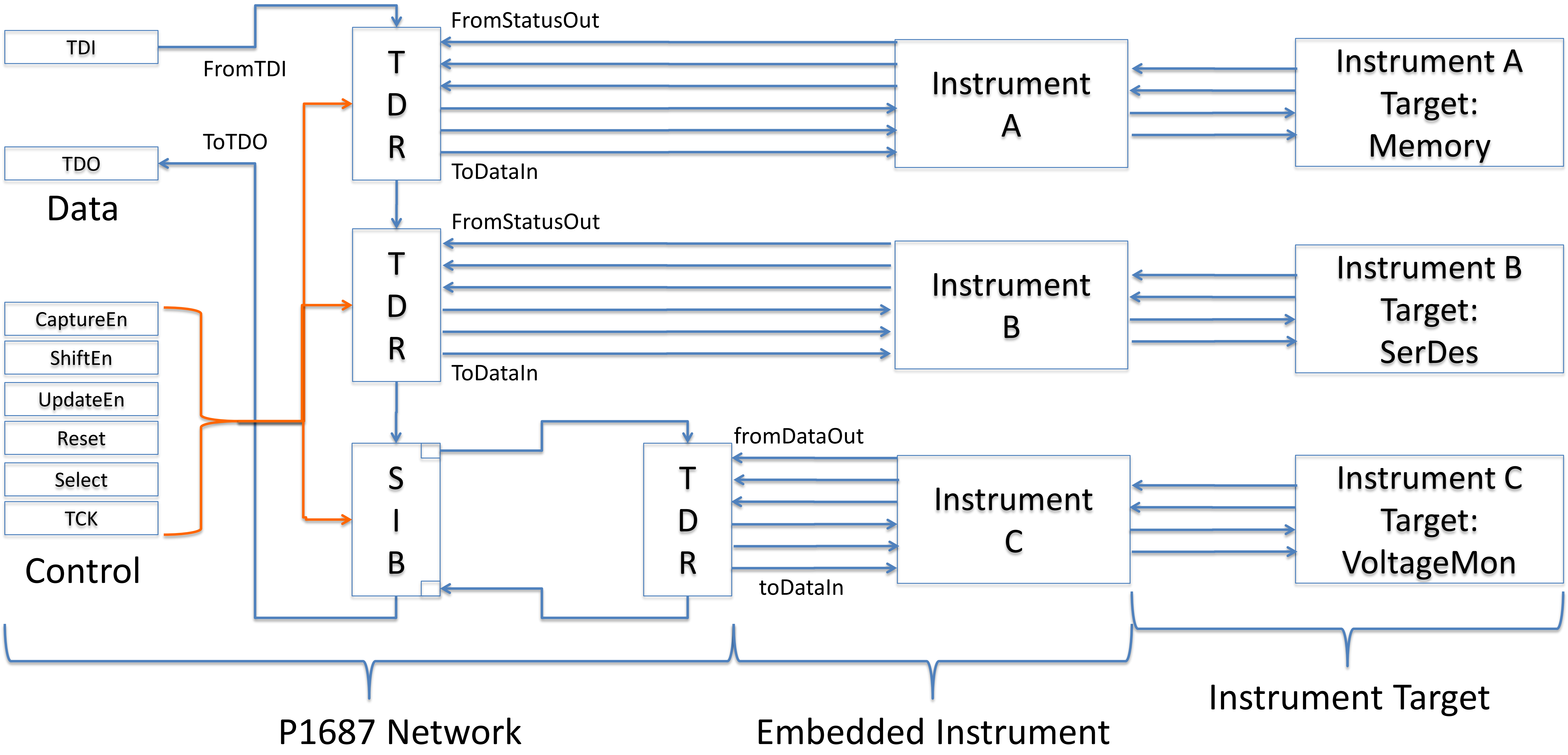


P1687 is not exclusive – other access is possible

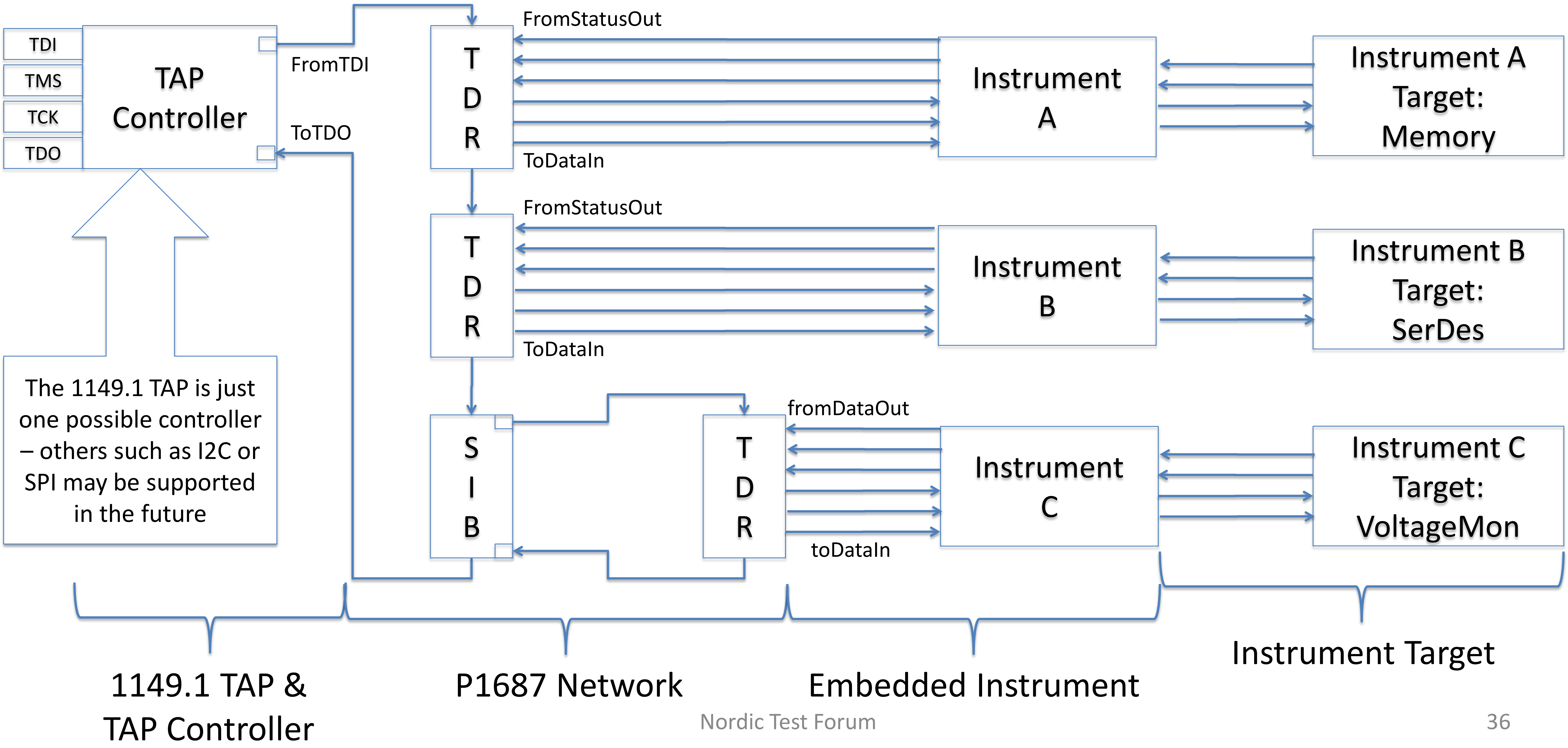
Other Access to an Embedded Instrument



1687 Network with Instruments



1687 Network with Instruments & Controller



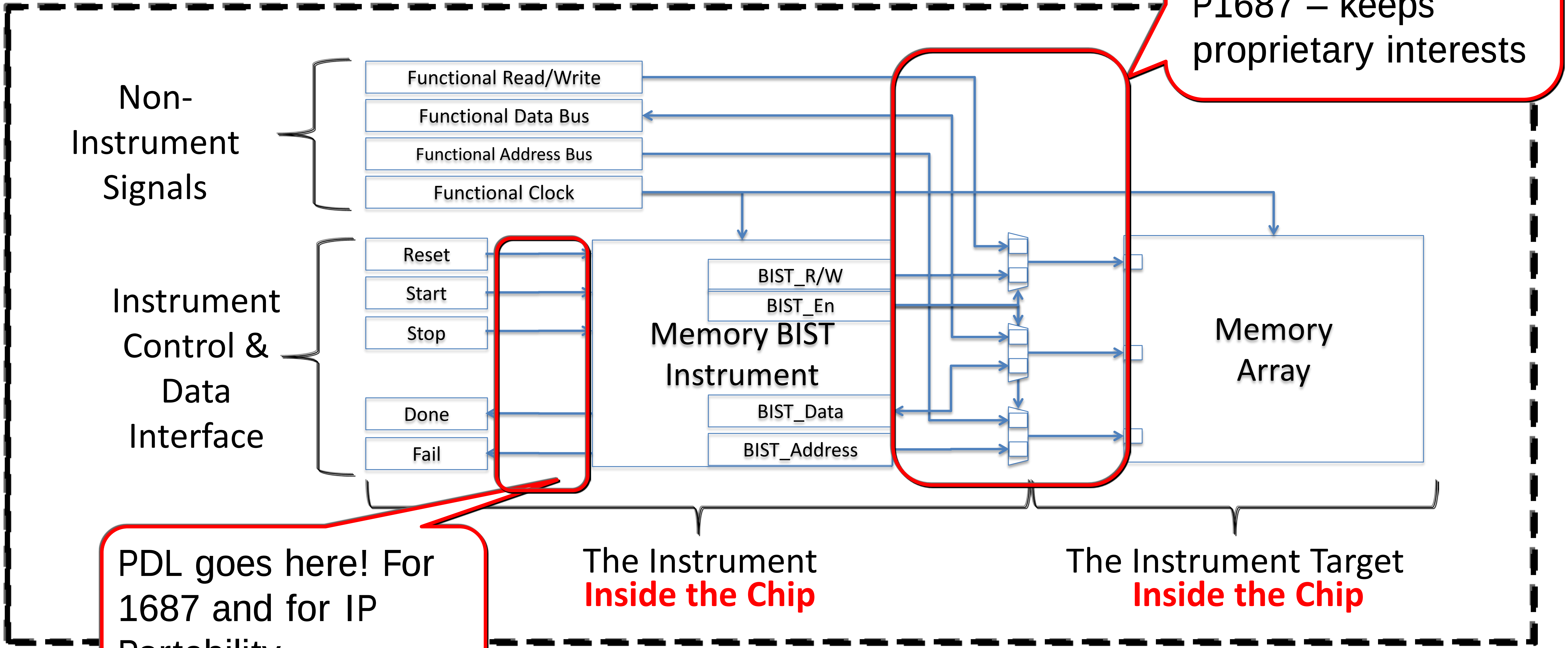
The 1149.1 TAP is just one possible controller – others such as I2C or SPI may be supported in the future

The ICL and PDL portion of 1687

- P1687 is not just about the Architecture
  - Any 1149.1 or 1500 architectures are also legal 1687
  - Purpose is to “ReTarget Instrument Vectors”

- What makes P1687 unique is the documentation
  - **ICL** to describe complex, tradeoff-driven, 1687 networks that may include variable-length scan paths
  - **PDL** to describe vectors or procedures associated with the instrument (making the instrument portable)

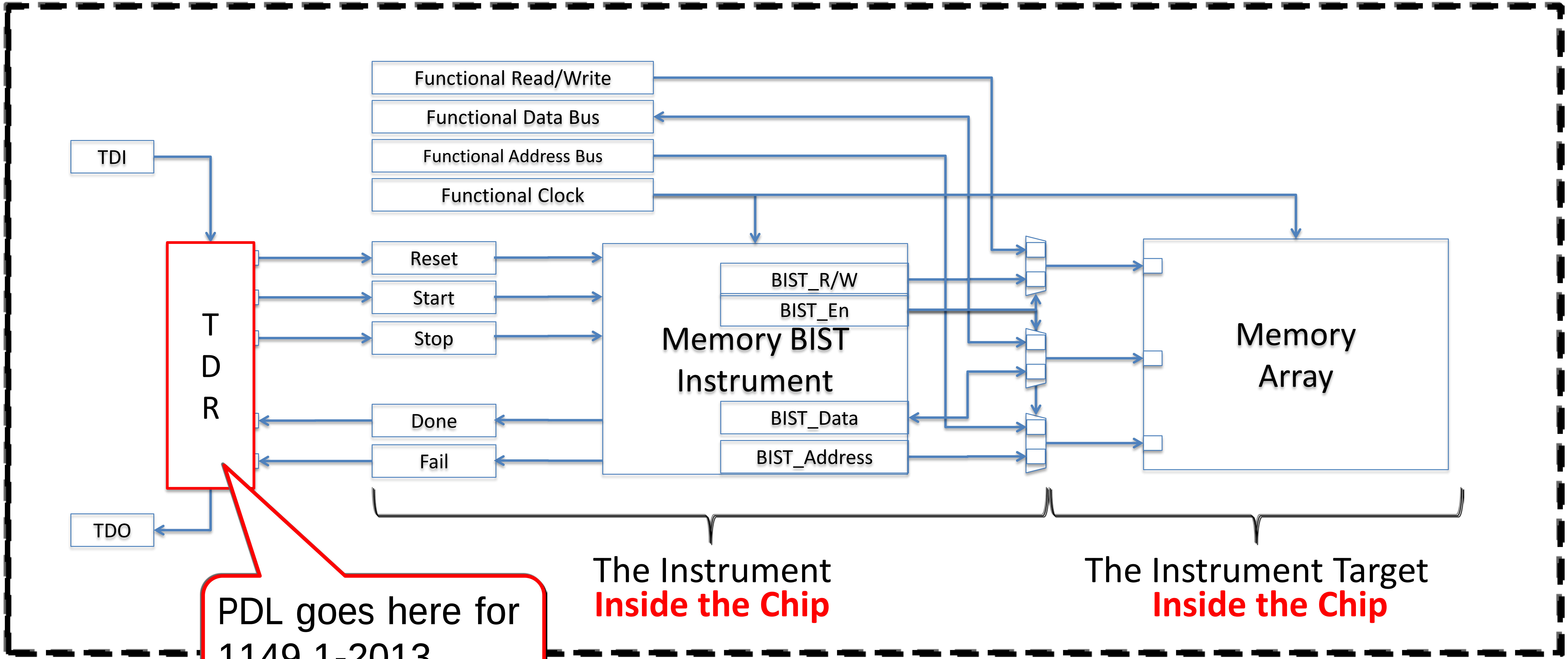
What is an Instrumen



Not part of the Documentation of P1687 – keeps proprietary interests

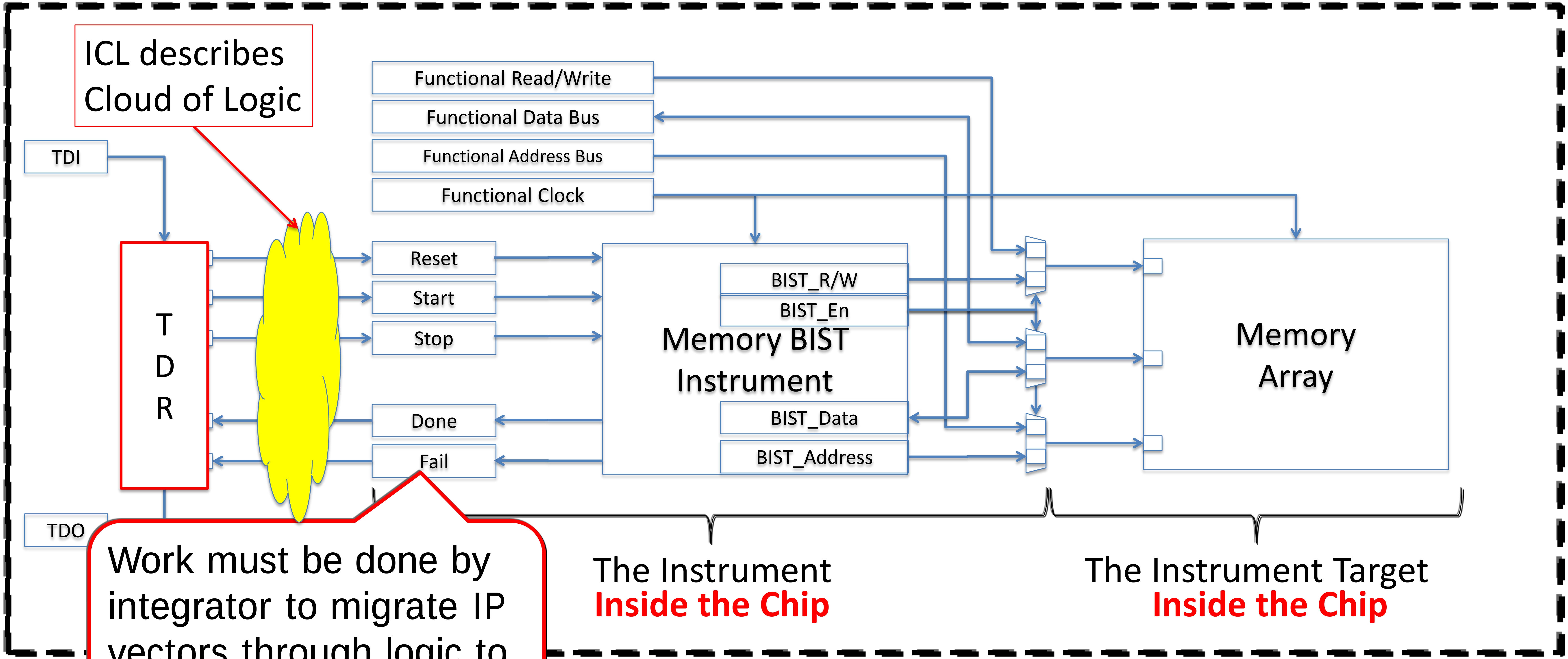
PDL goes here! For 1687 and for IP Portability

What is an Instrument?



PDL goes here for 1149.1-2013

What is an Instrument?



ICL describes Cloud of Logic

Work must be done by integrator to migrate IP vectors through logic to interface TDR

The Instrument
Inside the Chip

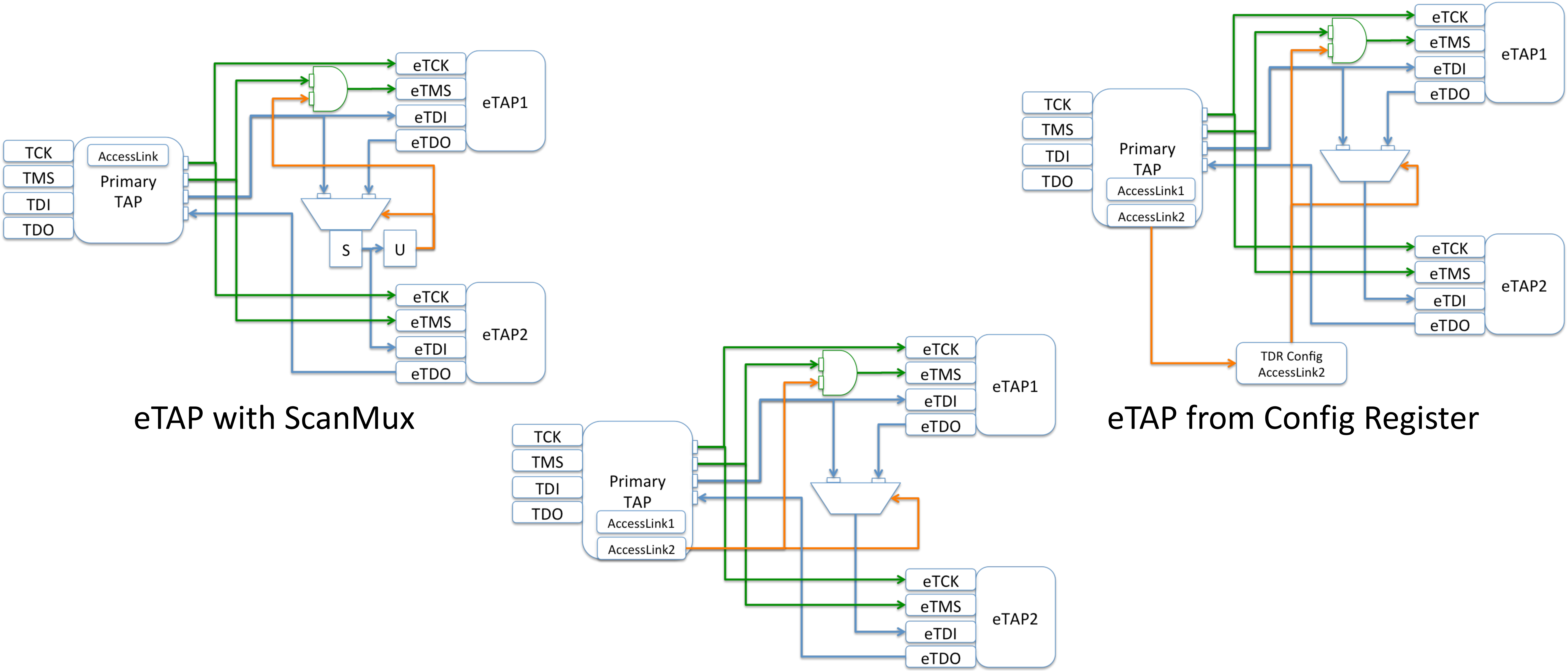
The Instrument Target
Inside the Chip

The Embedded TAP Controller

- P1687 allows and describes how to use an embedded TAP (eTAP) and TAP Controller (eTAPC)
  - This allows multiple TAPs embedded within a device (which is normal to many designs using IP Cores)
  - ICL describes the use and gating of the eTMS signal

- 1149.1 still mandates that there is only one TAP and TAP Controller
  - It does not teach how to embed multiple TAPs and TAP Controllers

eTAP Architectures



eTAP with ScanMux

eTAP from Config Register

eTAP from JTAG Instruction

The Separable Interface

- Both P1687 and 1500 use a Separable Interface
  - The interface is made of signals ShiftEn, CaptureEn, UpdateEn, Reset, TCK, TDI, TDO and Select (or SelectWIR in 1500’s case)
    - TMS and the FSM are replaced with ShiftEn, CaptureEn, UpdateEn
  - This separable interface allows direct operation (for example by ATE) – and some new 2.5D and 3D memory designs have these interfaces at the chip level
  - For P1687, this separable interface allows (but is not taught in the P1687.0 standard) alternate controllers to be designed and used (for example, a SPI, I2C, or Memory Mapped Bus – P1687.x?)

Do 1149.1 and P1687 use the same PDL

- 1149.1 can use some of the same PDL commands
- 1149.1 must deal with the IR and Controller
- 1149.1 has a restricted, specific, operation

- P1687 is just about the Instrument and…
- P1687 is about the configuration of the Scan Path
- P1687 is about local control (scan config, local reset, deny-capture, deny-update)

PDL Comparison

P#	P1687 PDL Command	15 P1687 Shared Command Comments
P#	P1687 Unique Command	5 Unique P1687 Commands
J#	1149.1 PDL Command	15 1149.1 Shared Command Comments
J#	1149.1 Unique Command	12 Unique 1149.1 Commands

Level-0 PDL

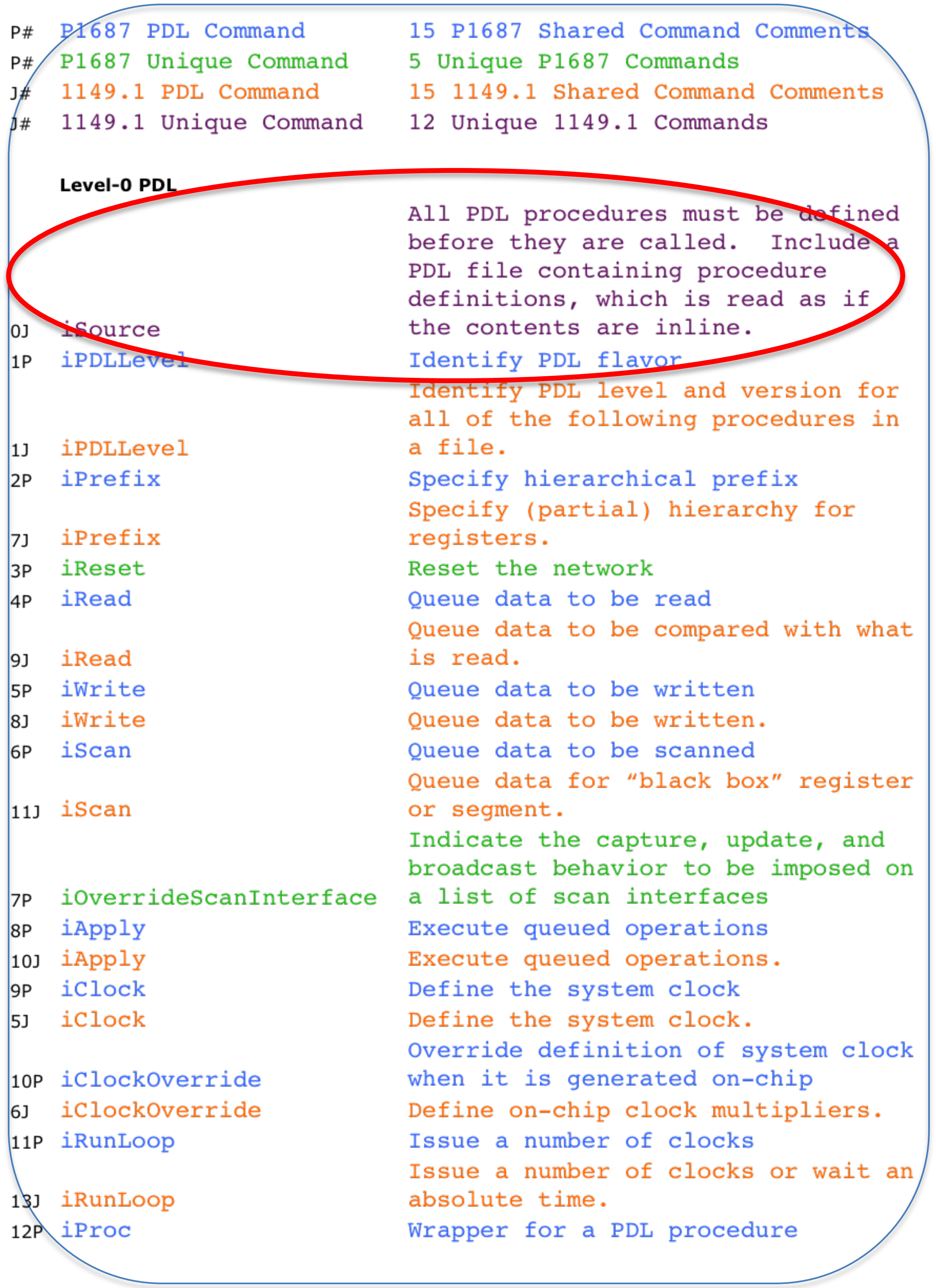
0J	iSource	All PDL procedures must be defined before they are called. Include a PDL file containing procedure definitions, which is read as if the contents are inline.
1P	iPDLLevel	Identify PDL flavor
1J	iPDLLevel	Identify PDL level and version for all of the following procedures in a file.
2P	iPrefix	Specify hierarchical prefix
7J	iPrefix	Specify (partial) hierarchy for registers.
3P	iReset	Reset the network
4P	iRead	Queue data to be read
9J	iRead	Queue data to be compared with what is read.
5P	iWrite	Queue data to be written
8J	iWrite	Queue data to be written.
6P	iScan	Queue data to be scanned
11J	iScan	Queue data for “black box” register or segment.
7P	iOverrideScanInterface	Indicate the capture, update, and broadcast behavior to be imposed on a list of scan interfaces
8P	iApply	Execute queued operations
10J	iApply	Execute queued operations.
9P	iClock	Define the system clock
5J	iClock	Define the system clock.
10P	iClockOverride	Override definition of system clock when it is generated on-chip
6J	iClockOverride	Define on-chip clock multipliers.
11P	iRunLoop	Issue a number of clocks
13J	iRunLoop	Issue a number of clocks or wait an absolute time.
12P	iProc	Wrapper for a PDL procedure

3J	iProc	A PDL procedure.
13P	iProcsForModule	Identify the module in the ICL with which subsequent iProcs are associated
2J	iProcGroup	Identify the object or instance with which the following PDL iProc procedures are associated.
14P	iUseProcNameSpace	Use namespace for subsequent iCalls
15P	iCall	Invoke a PDL procedure
12J	iCall	Transfer execution to a PDL procedure associated with the object associated with the named or current instance; or associated directly with the instance.
16P	iNote	Send text to runtime
23J	iNote	Creates a tool identifiable comment intended to pass either detailed annotation information to the output vectors (-comment) or execution status information to the system (-status).
17P	iMerge	Allow merging (concurrent evaluation) of iCalls
19J	iMerge	Provides guidance on where tools can optimize multiple PDL procedures.
18P	iTake	Disallow other merge threads from modifying a model resource
20J	iTake	Tag an object to be ‘in-use’ to provide guidance where tools can optimize PDL.
19P	iRelease	Re-allow other merge threads to modify a model resource
21J	iRelease	Release an object from ‘in-use’ to provide guidance where tools can optimize PDL.
20P	iState	Document the current state of the network

4J	iSetInstruction	Select instruction for a register accessed by multiple instructions.
14J	iLoop	Mark the beginning of a conditional loop.
15J	iUntil	Loop until any iApply -nofail in the loop meets the specified condition or the maximum loop count is reached.
16J	ifTrue	Execute commands based on last iApply —nofail.
17J	ifFalse	Execute commands based on last iApply —nofail.
18J	ifEnd	End of conditional execution.
22J	iSetFail	Set the status to “FAIL”; stop the test if the -quit parameter is specified, and output the <text>.
24J	iTRST	Assert or de-assert the TAP TRST* pin, and remain in TLR state after TRST* is off. Only executed at the top level of the current unit under test.
25J	iTMSreset	Enter TLR state via TMS. Only executed at the top level of the current unit under test.
26J	iTMSidle	Enter RTI state via TMS.

All Red and Blue commands are shared by both Standards

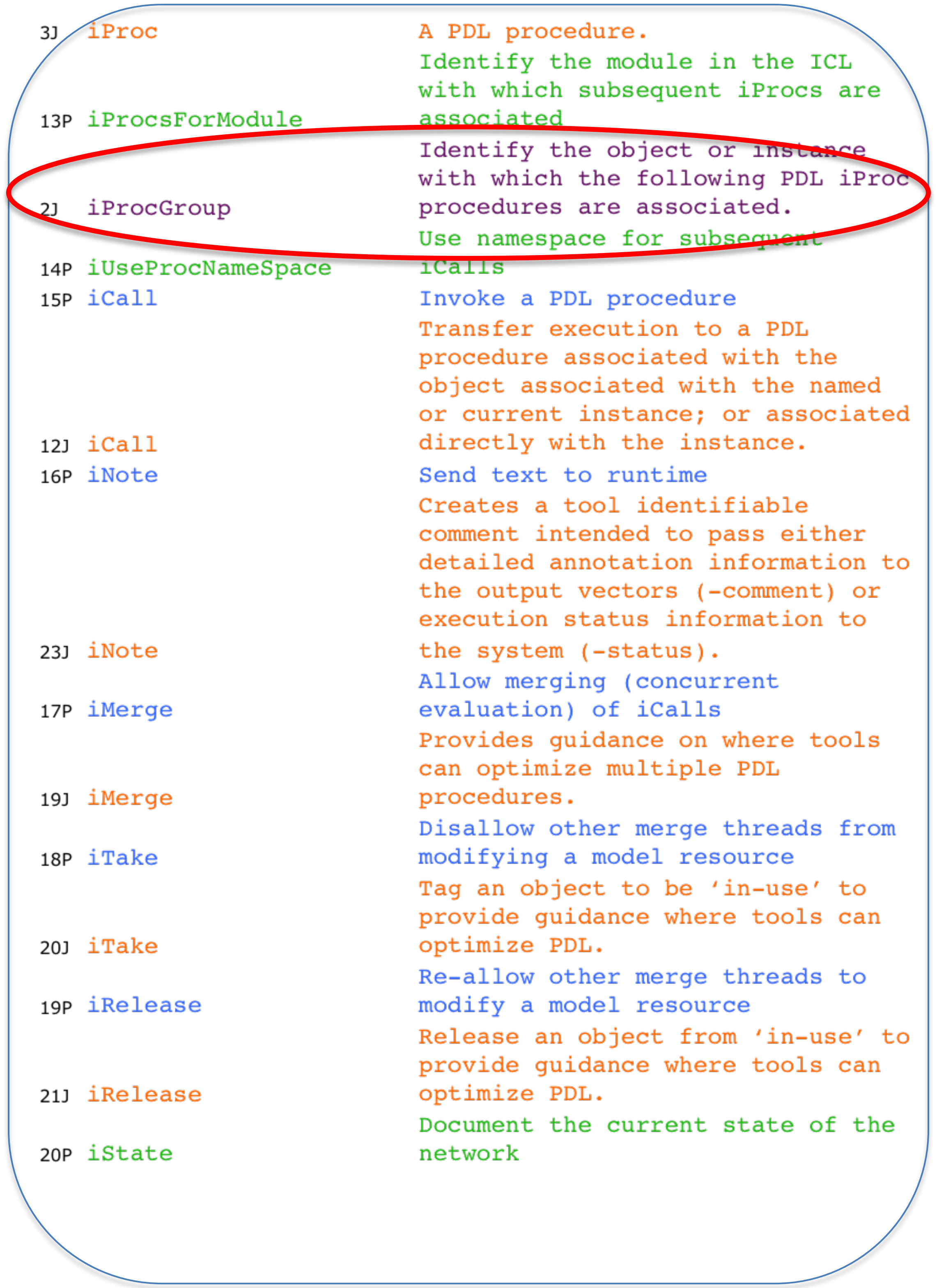
PDL Comparison: JTAG Unique



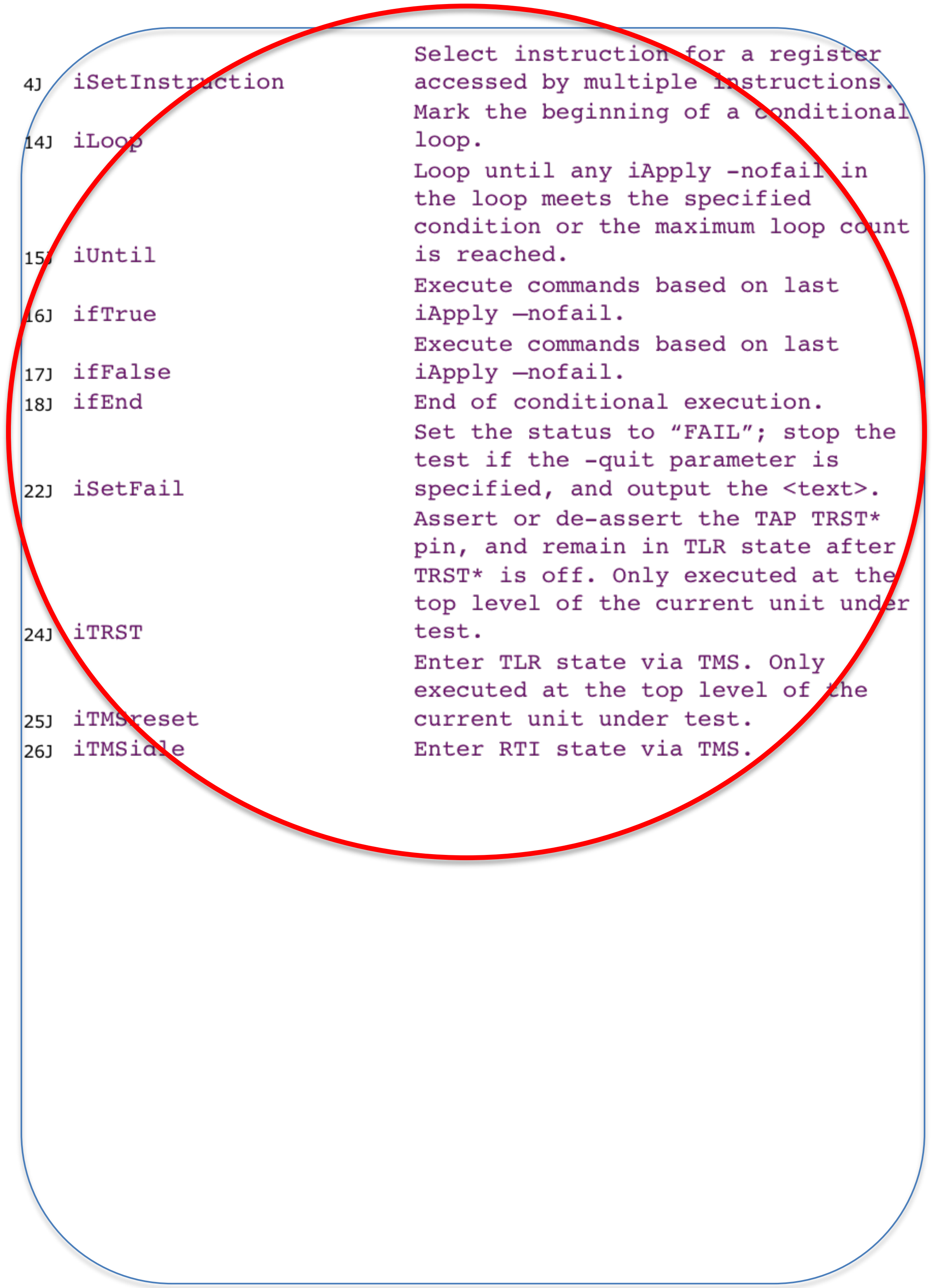
P#	P1687 PDL Command	15 P1687 Shared Command Comments
P#	P1687 Unique Command	5 Unique P1687 Commands
J#	1149.1 PDL Command	15 1149.1 Shared Command Comments
J#	1149.1 Unique Command	12 Unique 1149.1 Commands

Level-0 PDL

0J	iSource	All PDL procedures must be defined before they are called. Include a PDL file containing procedure definitions, which is read as if the contents are inline.
1P	iPDLLevel	Identify PDL flavor
1J	iPDLLevel	Identify PDL level and version for all of the following procedures in a file.
2P	iPrefix	Specify hierarchical prefix
7J	iPrefix	Specify (partial) hierarchy for registers.
3P	iReset	Reset the network
4P	iRead	Queue data to be read
9J	iRead	Queue data to be compared with what is read.
5P	iWrite	Queue data to be written
8J	iWrite	Queue data to be written.
6P	iScan	Queue data to be scanned
11J	iScan	Queue data for “black box” register or segment.
7P	iOverrideScanInterface	Indicate the capture, update, and broadcast behavior to be imposed on a list of scan interfaces
8P	iApply	Execute queued operations
10J	iApply	Execute queued operations.
9P	iClock	Define the system clock
5J	iClock	Define the system clock.
10P	iClockOverride	Override definition of system clock when it is generated on-chip
6J	iClockOverride	Define on-chip clock multipliers.
11P	iRunLoop	Issue a number of clocks
13J	iRunLoop	Issue a number of clocks or wait an absolute time.
12P	iProc	Wrapper for a PDL procedure



3J	iProc	A PDL procedure.
13P	iProcsForModule	Identify the module in the ICL with which subsequent iProcs are associated
2J	iProcGroup	Identify the object or instance with which the following PDL iProc procedures are associated.
14P	iUseProcNameSpace	Use namespace for subsequent iCalls
15P	iCall	Invoke a PDL procedure
12J	iCall	Transfer execution to a PDL procedure associated with the object associated with the named or current instance; or associated directly with the instance.
16P	iNote	Send text to runtime
23J	iNote	Creates a tool identifiable comment intended to pass either detailed annotation information to the output vectors (-comment) or execution status information to the system (-status).
17P	iMerge	Allow merging (concurrent evaluation) of iCalls
19J	iMerge	Provides guidance on where tools can optimize multiple PDL procedures.
18P	iTake	Disallow other merge threads from modifying a model resource
20J	iTake	Tag an object to be ‘in-use’ to provide guidance where tools can optimize PDL.
19P	iRelease	Re-allow other merge threads to modify a model resource
21J	iRelease	Release an object from ‘in-use’ to provide guidance where tools can optimize PDL.
20P	iState	Document the current state of the network



4J	iSetInstruction	Select instruction for a register accessed by multiple instructions.
14J	iLoop	Mark the beginning of a conditional loop.
15J	iUntil	Loop until any iApply -nofail in the loop meets the specified condition or the maximum loop count is reached.
16J	ifTrue	Execute commands based on last iApply –nofail.
17J	ifFalse	Execute commands based on last iApply –nofail.
18J	ifEnd	End of conditional execution.
22J	iSetFail	Set the status to “FAIL”; stop the test if the -quit parameter is specified, and output the <text>.
24J	iTRST	Assert or de-assert the TAP TRST* pin, and remain in TLR state after TRST* is off. Only executed at the top level of the current unit under test.
25J	iTMSreset	Enter TLR state via TMS. Only executed at the top level of the current unit under test.
26J	iTMSidle	Enter RTI state via TMS.

PDL Comparison: IJTAG Unique

P#	P1687 PDL Command	15 P1687 Shared Command Comments
P#	P1687 Unique Command	5 Unique P1687 Commands
J#	1149.1 PDL Command	15 1149.1 Shared Command Comments
J#	1149.1 Unique Command	12 Unique 1149.1 Commands

Level-0 PDL

0J	iSource	All PDL procedures must be defined before they are called. Include a PDL file containing procedure definitions, which is read as if the contents are inline.
1P	iPDLLevel	Identify PDL flavor
1J	iPDLLevel	Identify PDL level and version for all of the following procedures in a file.
2P	iPrefix	Specify hierarchical prefix
7J	iPrefix	Specify (partial) hierarchy for registers.
3P	iReset	Reset the network
4P	iRead	Queue data to be read
9J	iRead	Queue data to be compared with what is read.
5P	iWrite	Queue data to be written
8J	iWrite	Queue data to be written.
6P	iScan	Queue data to be scanned
11J	iScan	Queue data for “black box” register or segment.
7P	iOverrideScanInterface	Indicate the capture, update, and broadcast behavior to be imposed on a list of scan interfaces
8P	iApply	Execute queued operations
10J	iApply	Execute queued operations.
9P	iClock	Define the system clock
5J	iClock	Define the system clock.
10P	iClockOverride	Override definition of system clock when it is generated on-chip
6J	iClockOverride	Define on-chip clock multipliers.
11P	iRunLoop	Issue a number of clocks
13J	iRunLoop	Issue a number of clocks or wait an absolute time.
12P	iProc	Wrapper for a PDL procedure

3J	iProc	A PDL procedure.
13P	iProcsForModule	Identify the module in the ICL with which subsequent iProcs are associated
2J	iProcGroup	Identify the object or instance with which the following PDL iProc procedures are associated.
14P	iUseProcNameSpace	Use namespace for subsequent iCalls
15P	iCall	Invoke a PDL procedure
12J	iCall	Transfer execution to a PDL procedure associated with the object associated with the named or current instance; or associated directly with the instance.
16P	iNote	Send text to runtime
23J	iNote	Creates a tool identifiable comment intended to pass either detailed annotation information to the output vectors (-comment) or execution status information to the system (-status).
17P	iMerge	Allow merging (concurrent evaluation) of iCalls
19J	iMerge	Provides guidance on where tools can optimize multiple PDL procedures.
18P	iTake	Disallow other merge threads from modifying a model resource
20J	iTake	Tag an object to be ‘in-use’ to provide guidance where tools can optimize PDL.
19P	iRelease	Re-allow other merge threads to modify a model resource
21J	iRelease	Release an object from ‘in-use’ to provide guidance where tools can optimize PDL.
20P	iState	Document the current state of the network

4J	iSetInstruction	Select instruction for a register accessed by multiple instructions.
14J	iLoop	Mark the beginning of a conditional loop.
15J	iUntil	Loop until any iApply -nofail in the loop meets the specified condition or the maximum loop count is reached.
16J	ifTrue	Execute commands based on last iApply –nofail.
17J	ifFalse	Execute commands based on last iApply –nofail.
18J	ifEnd	End of conditional execution.
22J	iSetFail	Set the status to “FAIL”; stop the test if the -quit parameter is specified, and output the <text>.
24J	iTRST	Assert or de-assert the TAP TRST* pin, and remain in TLR state after TRST* is off. Only executed at the top level of the current unit under test.
25J	iTMSreset	Enter TLR state via TMS. Only executed at the top level of the current unit under test.
26J	iTMSidle	Enter RTI state via TMS.

High-level PDL comparison

Criterion	IEEE P1687	IEEE 1149.1-2013
# of PDL-0 commands	20	27
# of PDL-1 commands	4	2
PDL-0 command types	2: Setup Action	7: Procedure definition Test setup Test execution Flow control Optimization Miscellaneous Low-level

Remaining slides in this section drawn from Annex D in IEEE P1687, by Alan Bair

Unique PDL-0 commands: IEEE 1149.1-2013

#	Command(s)	Comment
1	**iSource**	Functions like a C #include to allow nested PDL.
2	**iSetInstruction**	Does not apply to P1687, since it is related to BSDL.
3	**iLoop, iUntil, ifTrue, ifFalse, ifEnd,* iSetFail**	Conditional control for PDL execution. iSetFail is a P1687 Level-1 command.
4	**iTRST, iTMSreset, iTMSidle**	Performs direct control of TAP reset and state; a related P1687 command is iReset.
5	**iProcGroup**	Performs a similar function to the P1687 iProcsForModule, but against BSDL constructs.

P1687 does not have a Controller or IR of its own, so doesn't support these

* These 5 commands provide functionality in 1149.1-2013 PDL-0 which is NOT available in P1687 PDL-0, which chose to use Tcl flow control via PDL-1. Otherwise, no major incompatibilities.

Unique PDL-0 commands: IEEE P1687

#	Commands(s)	Comment
1	iUseProcNameSpace	Does not apply to 1149.1, related to ICL namespaces.
2	iReset	Broad based ICL network reset. Similar 1149.1 commands are iTRST, iTMSreset.
3	iProcsForModule	Performs a similar function to the 1149.1 iProcGroup, but against ICL constructs.

1149.1-2013 does not have ICL, so doesn't support these

There are no major incompatibilities here.

So, is this a Problem?

- No – each is applied in context
  - 1149.1-2013 is used when JTAG is needed
  - P1687 is used when the AccessLink is active
  - So, jPDL and iPDL can be treated as two different languages

- The problem is vector delivery with instruments
  - Will instruments be delivered with jPDL, iPDL, or STIL?
  - Will portable network sections be delivered with BSDL, ICL, or CTL?

Summary and Conclusions

- P1687 and 1149.1 are different
- 1149.1-2013 has the controller
  - Still Instruction based
  - Still focused on the TDR (not the instrument)
  - Still focused on describing instruction features (i.e. Extest, RunBIST, HighZ, etc.)

- P1687 has the network
  - Focused on tradeoff driven network
  - Focused on instrument and instrument vectors
  - Purpose is to enable automated retargeting of vectors

Summary and Conclusions

- The key differences between the two standards
  - 1149.1 is still based on Instructions – using the new persistence controller to schedule multiple instruments is perceived as being inefficient
  - 1149.1 has the controller, which is needed for both 1500 and P1687 – so 1149.1 PDL that operates the controller is unique to 1149.1
  - 1149.1 PDL is focused on the TDR, not the instrument and the purpose of 1149.1 is more about selecting “instructions” than in retargeting vectors

  - P1687 allows data to configure the network which is good for scheduling, concurrence, and instrument management
  - P1687 also allows multiple TAPs (embedded)
  - P1687/1500 has a separable interface, and so allows direct operation or alternate controllers to access embedded instruments
  - P1687 PDL is focused on the embedded instrument and on documenting how to configure, operate, and use the instrument – which makes instruments portable