Intel On-Chip System Fabric (IOSF) DFx

High-level Architectural Specification (HAS)

July 1604205052,1604230640,1604248484,1405676622,1604371697,1604383018 2016

Revision 1.3_rc3

Intel Top Secret

Document Number: XXXXXX

Contents

1 Introduction 12
1.1 IOSF DFT Interface Overview....13
2 Landing Zone 14
2.1 TAP interface and support features....14
2.2 Scan interface and support features 14
2.3 Array test interface and support features 14
2.4 Miscellaneous test interface....14
2.5 Debug and validation interface and support features 14
2.6 General IOSF DFx information 15
3 JTAG TAP Interface 16
3.1 Hierarchical TAP network 17
3.2 TAP Link network 19
3.3 Slave TAP FSM Encoding 21
3.4 Slave TAP IR Address Support 21
3.5 Slave TAP Reset Behavior 33
3.15 Slave TAP Rules 34
3.16 Wrapper Serial Port Architecture 38
3.16.1 WTAP IR Opcodes 40
3.16.2 WTAP Reset Behavior 41
3.16.3 WTAP Rules....41
3.16.4 Network Use Models for WTAPs 43
3.17 DFx secure policy plug-in for TAPs....45
3.17.1 DFx Secure Plug-in Parameters for TAP 46
3.17.2 Policy Matrix for TAP 47
3.17.3 Security applied to TAPs in the Select register 49
3.17.4 Security applied to opcodes example....50
3.18 Other TAP Support Signals 52
3.18.1 Agent-Sourced TAP Interface Signals for TAP Network....53
3.18.2 Secondary Slave TAP Interface 53
3.18.3 TDO Enable....54
3.19 TAP Signal Interface Description 54
3.20 Transaction Cycle/Data Flows 60
3.21 Ordering/Coherency Rules....60
3.22 Performance/Bandwidth Analysis 60
3.23 Exception List Requirements....60
3.24 Programming Model 61
3.25 Power Management Capabilities 62
3.26 Security Feature Requirements 62
3.27 DFx Requirements....62
3.27.1 DFV/DFD Requirements 62
3.27.2 DFT Requirements 63
3.27.3 DFM Requirements....63
4 HVM Scan Interface....64
4.1 Active Scan Control Signals 68
4.2 Scan Data Chains....69
4.3 Asynchronous Scan Control Signals 69

4.4 Scan Reset Control Signals........70
4.5 Scan Static Control Signals........71
4.5.1 Clock Generator Override Control Signals........71
4.6 Array Shadow Logic for Scan Testing with Arrays........72
4.6.1 Proposed: Array initialization for scan........76
4.7 Functional Test Support........76
4.7.1 AMT/EAMT methodology........77
4.7.2 High speed bypass methodology........78
4.8 Scan Signal Interface Description........80
4.8.1 Scan signal parameter summary........86
4.9 Transaction Cycle/Data Flows........88
4.10 Ordering/Coherency Rules........88
4.11 Performance/Bandwidth Analysis........88
4.12 Exception List Requirements........88
4.13 Programming Model........88
4.14 Power Management Capabilities........88
4.15 Security Feature Requirements........88
4.16 DFx Requirements........88
4.16.1 DFV/DFD Requirements........88
4.16.2 DFT Requirements........89
4.16.3 DFM Requirements........89

5 Array Test Interface........90
5.1 Existing Array Test Methods........91
5.2 Array/Freeze/Dump module........91
5.3 Array testing with embedded power gate control........91
5.4 MBIST diagnostic results accumulation for HVM........94
5.5 Array Signal Interface Description........95
5.6 Transaction Cycle/Data Flows........106
5.7 Ordering/Coherency Rules........106
5.8 Performance/Bandwidth Analysis........106
5.9 Exception List Requirements........106
5.10 Programming Model........106
5.11 Power Management Capabilities........106
5.12 Security Feature Requirements........106
5.13 DFx Requirements........106
5.13.1 DFV/DFD Requirements........107
5.13.2 DFT Requirements........107
5.13.3 DFM Requirements........107

6 Miscellaneous DFT Interface........108
6.1.1 Boundary Scan Support........108
6.1.2 IDV Test Data Register Support........109
6.1.3 Fuse Data Support........111
6.1.4 High Volume Manufacturing Control Signals........111
6.1.5 Test Support Control Signals........112
6.1.6 Access to Control and Status Registers for DFT........113
6.2 Power Gate Common Block (PGCB) DFx interface signals........113
6.3 Miscellaneous DFT Signal Interface Description........114
6.4 Transaction Cycle/Data Flows........123
6.5 Ordering/Coherency Rules........123
6.6 Performance/Bandwidth Analysis........123
6.7 Exception List Requirements........124

6.8 Programming Model 124
6.9 Power Management Capabilities 124
6.10 Security Feature Requirements 124
6.11 DFx Requirements 124
6.11.1 DFV/DFD Requirements 124
6.11.2 DFT Requirements 124
6.11.3 DFM Requirements 125

7 Debug and Validation (DFV) 126
7.1 VISA Debug Bus for Observability 127
7.1.1 VISA Register Access 128
7.1.2 VISA Mux Widths 131
7.1.3 VISA security 133
7.1.4 VISA reset control 135
7.1.5 VISA start and end ID description 136
7.1.6 VISA rules 136
7.2 Trigger sources and events 137
7.2.1 DFx actions 137
7.3 Access to Control and Status Registers for DFV 138
7.4 DFx Secure Policy Interface 138
7.4.1 DFx secure plug-in IP-block 140
7.4.2 DFx secure plugin signal list 141
7.4.3 DFx secure plug-in parameters 142
7.4.4 DFx secure policy example 143
7.4.5 Agent/IP-block example with IP and TAP plug-in 145
7.5 ISM override for debug 146
7.5.1 IOSF Primary override signal description 148
7.5.2 IOSF Sideband override signals 150
7.6 Microprocessor debug control 154
7.7 Sideband parity error defeature 154
7.8 DFD Signal Interface Description 155
7.9 Transaction Cycle/Data Flows 169
7.10 Ordering/Coherency Rules 169
7.11 Performance/Bandwidth Analysis 169
7.12 Exception List Requirements 169
7.13 Programming Model 169
7.14 Power Management Capabilities 169
7.15 Security Feature Requirements 169
7.16 DFx Requirements 169
7.16.1 DFV/DFD Requirements 170
7.16.2 DFT Requirements 170
7.16.3 DFM Requirements 170
7.17 Legacy or obsolete use models 170
7.17.1 DFx secure plugin 170

8 Register Description 174

9 Implementation Details 175

10 Signal Interface 176
10.1 TAP Signal Interface Description 176
10.2 Scan Signal Interface Description 182
10.2.1 Scan signal parameter summary 188
10.3 Array Signal Interface Description 190

10.4 Miscellaneous DFT Signal Interface Description 200
10.5 DFV Signal Interface Description 209

11 Feature Use Models 225

12 Re-Use Collateral & Information.. 226

13 Validation Strategy 227

14 Prior Work 228

15 Open Issues 229
15.1 List of Open Issues.. 229

16 Feature Wish List 230

17 Glossary and Reference 231
17.1 Terminology 231
17.2 Reference Documents 232
17.3 Author / Acknowledgement List 233
17.4 Web Links 233
17.4.1 Chassis/SoC DFx HAS repository: 233
17.4.2 IOSF DFx: 233
17.4.3 SoC TAP: 234
17.4.4 Scan components: 234
17.4.5 SoC DFT Handbook: 234
17.4.6 SoC Scan Requirements: 234
17.4.7 MBIST: 235
17.4.8 VISA: 235
17.4.9 DFx Security Framework: 235
17.4.10 Soft IP-block DFx: 235
17.4.11 Hard IP-block DFx: 236
17.4.12 Chassis Test Controller and functional test: 236
17.4.13 North Peak: 236
17.4.14 Cluster Trigger Block Architecture: 236

Figures

Figure 3-1. Abstract view of a generalize SoC TAP placement 18
Figure 3-2. TAP link example that replaces a hierarchical TAP network 20
Figure 3-3. Graphical view of agent WTAP 38
Figure 3-4. WTAP block diagram 39
Figure 3-5. TAP controller glue logic for WTAP 40
Figure 3-6. Option 1a: sTAP Controlled Single WTAP 44
Figure 3-7. Option 1b: sTAP Controlled Multiple WTAP Network 44
Figure 3-8. Option 1c: sTAP to WTAP in Series Network 44
Figure 3-9. Slave TAP drives both sTAP and WTAP Networks 45
Figure 3-10. TAP DFx security plug-in block diagram 46
Figure 3-11. Application of DFx secure policy to SELECT register............................50
Figure 3-12. Application of DFx secure policy to IR Opcodes 52
Figure 3-13. Primary/Secondary TAP source interfaces 53
Figure 3-14. Secondary slave TAP interface example 54
Figure 3-15. TAP signal interface graphical view 1 55
Figure 3-16. TAP signal interface graphical view 2 56
Figure 3-17. TAP.7 normal mode use model block diagram 61

Figure 4-1. Generalized view of scan implementation 66
Figure 4-2. Scan example of a zero depth pattern 67
Figure 4-3. Scan example of a two capture depth pattern 67
Figure 4-4. Pre/Post clock use models for scan control 68
Figure 4-5. Verification use models for internally derived clocks 69
Figure 4-6. Scan reset bypass control 70
Figure 4-7. Clock Generator Control Example 1 71
Figure 4-8. Clock Generator Control Example 2 72
Figure 4-9. RAM Sequential Mode 73
Figure 4-10. Asynchronous Write Thru (AWT) Mode 74
Figure 4-11. Synchronous Bypass Mode 75
Figure 4-12. Output Disable Feature 75
Figure 4-13. Array scan initialization control 76
Figure 4-14. Loopback mechanism in the PCIe agent 77
Figure 4-15. Digital near end loopback in serial IO 78
Figure 4-16. High speed bypass block diagram 79
Figure 4-17. Agent specific scan signal interface block diagram 81
Figure 5-1. AFD module block diagram 91
Figure 5-2. EBB Override for HVM 93
Figure 5-3. MBIST results accumulation block diagram 95
Figure 5-4. Register file array test support signals 96
Figure 5-5. SRAM array support signals continued 96
Figure 5-6. Read-only memory (ROM) array support signals continued 97
Figure 5-7. LYA and miscellaneous array test signal support 97
Figure 6-1. Component level IDV use model 110
Figure 6-2. Typical IDV fublet 111
Figure 6-3. Functional pin reuse example block diagram 113
Figure 6-4. PGCB DFx block diagram 114
Figure 6-5. Miscellaneous test signal diagram 115
Figure 6-6. Boundary Scan control signals 116
Figure 7-1. VISA unit-level Mux block diagram 128
Figure 7-2. VISA register access network 129
Figure 7-3. VISA serial interface register controller 130
Figure 7-4. VISA register serial protocol 131
Figure 7-5. IP-blocks with VISA wide muxes to a PLM with byte lane selection 132
Figure 7-6. VISA ULM with security color code overlay 133
Figure 7-7. Another view of VISA security color codes on a ULM 134
Figure 7-8. Applying security parameters to a VISA ULM 135
Figure 7-9. VISA reset connected to Chassis DFx reset 136
Figure 7-10. DFx security plug-in block diagram 141
Figure 7-11. DFx secure policy use model example 1 144
Figure 7-12. Computing a policy matrix value for agent 1 145
Figure 7-13. Agent/IP-block with security plug-in and TAP 146
Figure 7-14. TAP TDR connectivity example for IOSF ISM override signals 147
Figure 7-15. Graphical diagram of DFV signal interface 155
Figure 7-16. Graphical view of DFV signals continued 156
Figure 7-17. DFx security plug-in with optional Sideband override 171
Figure 7-18. Sideband override for targeted OEM enabling 172
Figure 7-19. DFx secure plug-in example with Sideband override 173
Figure 10-1. TAP signal interface graphical view 1 177
Figure 10-2. TAP signal interface graphical view 2 178
Figure 10-3. Agent specific scan signal interface block diagram 183
Figure 10-4. Register file array test support signals 190
Figure 10-5. SRAM array support signals continued 191
Figure 10-6. Read-only memory (ROM) array support signals continued 191
Figure 10-7. LYA and miscellaneous array test signal support 192

Figure 10-8. Miscellaneous test signal diagram 201
Figure 10-9. Boundary Scan control signals 202
Figure 10-10. Graphical diagram of DFV signal interface 210
Figure 10-11. Graphical view of DFV signals continued 211

Tables

Table 3-1. Chapter revision history 16
Table 3-2. TAP FSM state encodings 21
Table 3-3. TAP IR opcode support 22
Table 3-4. Public TAP instruction definitions 25
Table 3-5. sTAP reset behavior 33
Table 3-6. WTAP IR address support 40
Table 3-7. WTAP reset behavior 41
Table 3-8. DFx secure plug-in IP parameters 46
Table 3-9. Policy matrix table for TAPs 48
Table 3-10. TAP interface signal description 56
Table 4-1. Chapter revision history 64
Table 4-2. Example of SIP HBP IOSF compliant interface signals 79
Table 4-3. Scan interface signal descriptions 82
Table 4-4. Scan parameter summary table 86
Table 5-1. Chapter revision history 90
Table 5-2. Register file array test signal table 98
Table 5-3. SRAM Array test signal table 100
Table 5-4. Read-only memory (ROM) array test signal table 102
Table 5-5. Miscellaneous array test signal table 104
Table 6-1. Chapter revision history 108
Table 6-2. Boundary scan test mode control signals 109
Table 6-3. Boundary scan support signals 116
Table 6-4. IDV control signals 118
Table 6-5. Parallel fuse data interface 121
Table 6-6. SoC level IO test control signals 121
Table 6-7. Test controller signals for content transport to/from IP-blocks 122
Table 6-8. Miscellaneous DFx support signals 122
Table 7-1. Chapter revision history 126
Table 7-2. VISA mux style parameter reference 131
Table 7-3. DFx secure policy bus encoding 139
Table 7-4. DFx secure policy plug-in IP signal list 142
Table 7-5. DFx secure plug-in IP parameters 142
Table 7-6. Use model examples 153
Table 7-7. VISA DFV signal descriptions 156
Table 7-8. Trigger DFV signals 160
Table 7-9. DFV security signals 162
Table 7-10. ISM override DFV signals 163
Table 7-11. General DFV signals 167
Table 7-12. DFx secure plugin obsolete signal list 171
Table 7-13. DFx secure plugin obsolete parameter list 172
Table 10-1. TAP interface signal description 178
Table 10-2. Scan interface signal descriptions 184
Table 10-3. Scan parameter summary table 188
Table 10-4. Register file array test signal table 192
Table 10-5. SRAM Array test signal table 194
Table 10-6. Read-only memory (ROM) array test signal table 197

Table 10-7. Miscellaneous array test signal table 199
Table 10-8. IDV control signals 204
Table 10-9. Parallel fuse data interface 207
Table 10-10. SoC level IO test control signals 207
Table 10-11. Test controller signals for content transport to/from IP-blocks 208
Table 10-12. Miscellaneous DFx support signals 208
Table 10-13. VISA DFV signal descriptions 211
Table 10-14. Trigger DFV signals 215
Table 10-15. DFV security signals 217
Table 10-16. ISM override DFV signals 218
Table 10-17. General DFV signals 222
Table 17-1. Terminology 231
Table 17-2. Reference Documents 232
Table 17-3. Author - Acknowledgement List 233

§

Revision History

Revision Number	Description	Revision Date
rev1.2_rc1	Initial release.	Jan 2012
rev1.2_rc2	• Updated scan interface signals — Updated the DFx security group of signals to finalized architecture	Feb 1, 2012
rev1.2_rc3	• Added new opcodes for IEEE1149.1_2012 and IEEE1149.8.1 for future use. The user defined opcodes now start at 0x30.	Feb 13, 2012
rev1.2_rc4b	• All edits were accepted, all strikeout text was deleted. • Move content into new SoC HAS template Updated scan interface signals to be organized into the following buckets: all signals, signals for IP, signals for SCC, SASC, SRC. • Added DFx security • Added TAP security	March 10, 2012
rev1.2_rc5	• Release was intended as a final draft but this was premature. There have been several updates to fix errors in diagrams. The signal interface has remained static however, there have been changes to the array test interface.	March 11, 2012
rev1.2_rc6	• Updated the array test section. • Added text description of the functional test mode for soft-IP blocks.	May 2012
rev1.2_rc7b	• This version has all the requirements, rules and permission formatted into XML tagged comments for extraction into an excel spreadsheet. • For other update refer to each chapter.	July 13, 2012
rev1.2	• Final edits based on feedback of rev1.2_rc7b	July 20, 2012
rev1.2.1	• HSD and ECN issues for Chassis DFx Gen2 rev0.9.	February, 26, 2013
rev1.2.2	• Updated with all known HSD and ECN issues for Chassis DFx Gen2 rev1.0	February 8, 2014
rev1.3_rc2	• Removed the timing correlation signals since they were never used and there are 2-3 competing methodologies to replace this feature. A future Chassis revision will address this gap. • Added Sideband ISM override signals missing from the rev1.2.2 spec. • Changed the Sideband clock override back to the original definition •	January, 2016
rev1.3_rc2	• Added array initialization signals for scan control of arrays.	January, 2016

	• Added Sideband parity defeature signal • Removed a list of VISA rules and requirements since this is now driven by the IPDS requirements list given to IP developers. • Removed the VISA startID and endID diagrams and text. Added text to refer to the VISA auto-ID tool to manage the ID assignment. Only HIPs can use the endID for this generation of Chassis. A future version of Chassis DFx and this spec will most likely remove the endID signal.	

1 Introduction

This chapter specifies the DFx requirements and features supported by the IOSF specification. DFx is defined as "design for x", where x is a variable that can be substituted for a character that designates the service being offered, such as Design For Test (DFT), Design For Validation (DFV), and other acronyms. An IOSF DFx interface is equivalent to either the primary or sideband interface that an agent may implement independently and still be IOSF compliant. The DFx interface is composed of several sub-interfaces for DFT or DFV applications. Each of these interfaces has signals that may or may not be applicable to the type of agent that instantiated on the fabric, and therefore many of them are declared as optional. However, a minimum set of signals is required to support debugging and test operations.

An agent is a general term used to describe a functional unit that connects to the IOSF fabric. It may be composed of other IP blocks like a FIFO memory structure and an IOSF sideband endpoint. Agent and IP-block terms have been used interchangeability, but we specifically differentiate the word agent for those units on the IOSF interface.

A soft-IP agent is defined as synthesizable RTL code that may have a parameterized feature list that can be configured upon compilation. A hard-IP agent is a circuit block that is designed to specific process library. A high-speed serial IO (HSIO) link (PCI Express physical layer) is an example of this type of agent. It is further subdivided into a synthesizable block and an analog block. This is an arbitrary dividing line but the theory is that changes to the analog sub-block will cause changes in a digital control block that supports these circuits. Therefore, according to this design philosophy, it is better to restrict the changes to the digital phy block rather than re-synthesizing the entire IOSF agent. Another key attribute to the IO hard-IP is the ability to debug analog quickly, and by having a self-contained direct fabric access is a key attribute in making post-silicon debug successful. The IOSF DFx interface is defined to meet this challenge.

The goal of this chapter is to provide a clearly defined DFx interface to enable developers to deliver re-usable IP-blocks and component integration teams to expect consistency between Intel-provided agents when incorporating them into their designs. Some DFT features such as scan are incorporated during a post place and route activity and are not part of the RTL code itself. However, we will still define these interfaces for situations where a hard-IP contains a DFx interface directly to the IOSF itself. Externally purchased IP-blocks that connect to the IOSF fabric will require a bridge or shim (that is, a thin veneer of glue logic) to translate other industry standard interfaces into IOSF-compliant transactions. The DFx requirements are expected to be met on the IOSF fabric side of the bridge with minimal logic within the block to handle the needs of the external IP-block.

The IOSF DFx signals are defined with format such that the signal can be easily identified. There are some variations but the basic premise is the following:

{f,a} feature set_(underscore)feature group_(underscore)signal name

1) F = fabric, a= agent:
    a) Indicates from where the signal originates.
    b) The term fabric is the DFx fabric composed of Modular DFx Unit as defined in the Chassis DFx HAS.
    c) Agent usually refers a functional IP that is connect to an IOSF primary and/or a sideband network. However, it may also be simply referred to as an IP or PI-block.
2) Feature sets (not all listed here):

a) tap = Test Access Port (TAP). Many of these signals were developed several years prior to this release and the feature set and feature group were merged without an underscore. Example: The secondary slave TAP clock is ftapsslv_tck but should have been: ftap_sslv_tck.
   i) Example: ftap_tck
b) scan = Automated Test Pattern Generation (ATPG)- based stuck-at and at-speed test technique. This does not mean 1149.1 based boundary scan.
   i) Example: A fabric signal, in the scan feature set, in the ram feature group and the signal is awt_mode: fscan_ram_awt_mode
c) Array = array based testing which usually includes a Built-In Self Test (BIST) engine of some type. Arrays include register files, FIFO, SRAMs etc. which are constructed as a 6-transistor cell (there are other configurations). In this context, they are not arrays composed of flip-flops which can be tested with scan.
   i) Example: The array test signals are a little more elaborate with a fabric source, array feature set, Logic Vision signal group, signal name, and memory type: fary_LV_EnableWR_rf.
d) bscan = the 1149.1 based boundary scan testing.
   i) Example: fbscan_tdi or abscan_tdo.

3) Feature group: This is optional and may be necessary for feature sets. This helps to group similar signals together for ease of integration with the Collage tool which requires groupings of signals of similar names to integrate IPs more quickly.
4) Examples of where it wasn't followed that closely:
   a) Example: fdfx_secure_policy, fdfx_policy_update and fdfx_earlyboot_exit these should have been named: fdfx_secure_*.

1.1 IOSF DFT Interface Overview

The following three sections describe the DFT signals that will appear on the IOSF interface to enable High Volume Manufacturing (HVM) test capabilities within the IOSF agent. This specification will present architectural illustrations and descriptions for their use in order to understand the intended purpose. This specification will not describe in detail exactly how an implementation will use the signals. For more information, refer to other SoC DFx architectural specifications.

§

2 Landing Zone

This chapter provides a high level list of the DFx expectations for IP-blocks and their adherence to a common signal interface. It is only meant to be a reference and not all features are captured here.

2.1 TAP interface and support features

- The TAP network is defined in the SoC TAP HAS rev0.90.x and compliant to IEEE 1149.7 T0.
    - The SoC TAP specification extends the concept of a hierarchy of TAPs with designated slave TAPs implementing a SELECT register that enables a group of child TAPs. These "parent" TAPs are part of the Chassis DFx Gen2 specification and apply to the Master DFx Unit (MDU) TAP, Region DFx Unit (RDU) TAP and Cluster DFx Unit (CDU) TAP.
- The TAP network supports a secondary TAP port that is outside of the scope of the IEEE 1149.7 standard. However, its use does not hinder any functionality of the standard.
- All TAPs have a slave ID code with a specific encoding that forms a linked list to uniquely identify any slave TAP on the TAP network.

2.2 Scan interface and support features

- A minimum standard set of scan control signals on the IP interface. A majority of the scan control logic is in the Cluster DFx Unit (CDU).
- Many scan controls are conditional, meaning that if the IP implements the design feature then the scan control becomes required. For example, if the IP uses latches then the fscan_latchopen, fscan_latchclosed_b must be part of the IP DFx interface.

2.3 Array test interface and support features

- All SoCs have converged on the use of Logic Vision's MBIST tool and flow for testing register files, arrays, and FIFOs. In the past, this The array test interface that is included in this revision to assist the Collage tool to incorporate IP-blocks without additional steps in wrapping the IP-block with the signals necessary for integration.
    - Logic Vision is now owned by Mentor Graphics.

2.4 Miscellaneous test interface

- Boundary interface for SoC connectivity between IP-blocks as well as managing "fused off" IP-blocks for market segment product skews.
- An IDV interface for IP-blocks to be integrated and stitched at the SoC level. This interface is targeting the hard IP-blocks since they are delivered as a complete IP.
- There are other miscellaneous signals for trigger actions, parallel fuse distribution, test content delivery, leakage testing, etc.

2.5 Debug and validation interface and support features

- Visualization of Internal Signals Architecture (VISA) interface.

- Trigger events inputs and trigger source outputs to support a DFx fabric interconnection of trigger sources to trigger engine for local processing. These trigger engines can pre-process the events for local (regional) consumption or passed to Lakemore for trace operations.
- A timestamp toggle bit interface for IP-blocks that have their own local timestamp counter. This is used by the North Peak Global Trace Hub to correlate other timestamp domains to the global timestamp counter for hardware/software trace capabilities.
- A DFx security policy signal group that indicates the current policy. It is a fan out bus from the central security aggregator.
- The IOSF Sideband ISM override signals have been granted waivers for soft IP-blocks in past revisions of the IOSF DFx spec. Several options have been review and the conclusion is to simply expose the interface signals and let the SoC integration teams drive them with a TAP TDR (or TDRs) in the modular-DFx fabric. Both the primary and sideband ISM signals are expected to be available so that preliminary power measurements can be controlled with the clock gate override signal.
- A Power Gate Control Block (PGCB) was placed as part of the debug interface but it really only applies to DFT HVM to force the critical PGCB signals into powered up/down condition.

2.6 General IOSF DFx information

The IOSF DFx signals names as defined in this specification must be used as the root name of the signal. Agent/IP-block developers and integration teams have the freedom to assign any reasonable number of characters as a prefix or suffix as applied to these root names in accordance to the naming convention of the design teams.
The use of agent and IP-block can be used interchangeably. The term agent refers to a functional IP-block with a primary and sideband interface.

§

3 JTAG TAP Interface

Table 3-1. Chapter revision history

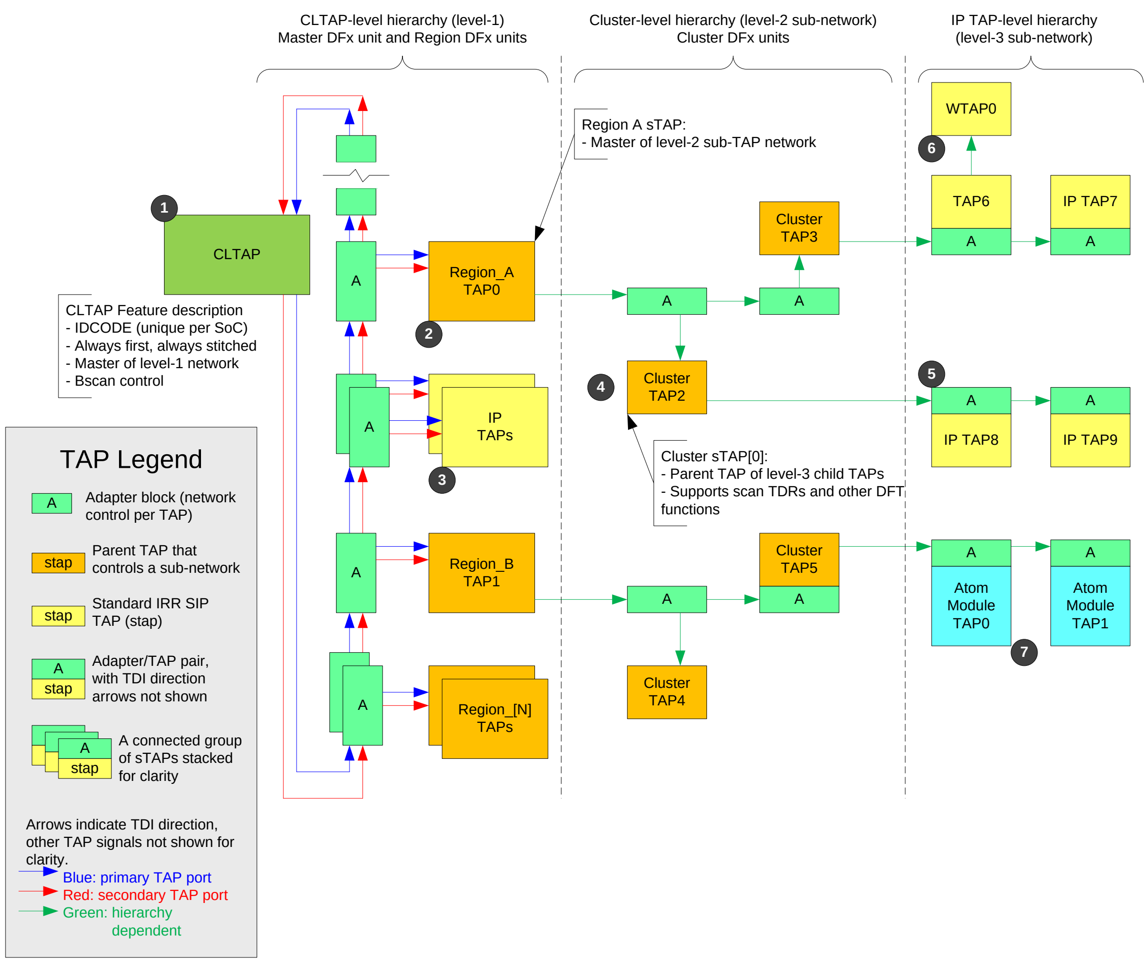
Revision Number	Description	Revision Date
rev1.2_rc1	• Initial release.	Jan 2012
rev1.2_rc2	• Added DFx security group of signals to TAP, however, it is defined in the debug/validation section of this specification	Feb 1, 2012
rev1.2_rc3	• Added new opcodes for IEEE1149.1_2012 and IEEE1149.8.1 for future use. • The user defined opcodes now start at 0x30.	Feb 13, 2012
rev1.2_rc4b	• All edits were accepted, all strikeout text was deleted. • Move content into new SoC HAS template • Updated IR opcode table with security color code (**Error! Reference source not found.**) • Edit text to remove old descriptions or obsolete language and rules. • Added TAP security section. • Added <tapname> prefix to the slave ID code signal name.	March 10, 2012
rev1.2_rc6	• Moved the latch from the output of the lookup table to the secure policy bus input. • The TAP IR table showed an incorrect value for the starting user defined opcodes (Table 3-4). • Updated the TAP security Visio diagrams because they had text to describe the TAP as an agent or IP-block when in reality it is the TAP.	May, 2012
rev1.2_rc7 (final)	• Added the security plug-in rules for the WTAP • Added the programmable reset option for WTAP in the permission section of the rules. • General clean of the rules, requirements and permissions based on the XML tags use model. In other words, for the XML tag to capture a rule it needs to be worded such that when the rule is extracted it needs to read like a rule.	July 20, 2012
rev1.2.2_rc1	• Added re-time TAP information	Feb 2014
rev1.3	• Added TAP link top level diagram	Jan 2016

The IEEE 1149.1 JTAG TAP specification is the industry standard for package pin connectivity testing between two components mounted on a board. It has a serial interface protocol to read and write the boundary scan register that connects all of the IO cells in the pad ring to apply the continuity tests. The same serial protocol is used to access other test data registers to provide a backdoor means of reading and writing registers for test and debug. Test Access Port (TAP) is an effective protocol for this because it has its own clock and reset domains to provide a guaranteed access during reset for debug when the component is hung and cannot communicate with otherwise. The SoC TAP High level Architectural spec HAS (rev 1.0 or later) defines the slave TAP, the Chip Level TAP and the TAP network in which these components operate. There are two TAP networks, one is the Chassis DFx Gen2.0 compliant sometimes referred to as the SoC TAP network or hierarchical network. The other is the TAP link network which is compliant to Chassis DFx gen2.1. It is defined in a separate HAS (TAP Link HAS rev.90). The SoC TAP network is a superset of the IEEE 1149.7 T0 (T0 capability level) to organize the TAPs into a network.

3.1 Hierarchical TAP network

Shown in Figure 3-1 is a simplified block diagram of a TAP network based on the legacy SoC hierarchical TAP network with a number of slave TAPs at several different levels of hierarchy. The modular DFx fabric (specifically the TAP network in this case), is implemented as a separate entity from the IOSF of SoC System Agent fabrics. This allows for a flexible TAP network organization that can be assembled at the full chip level independently of the various functional fabrics of the SoC. There are two topologies that an integration team may choose between when assembling the TAP network. To reduce clutter the diagram we will only show a TDI path with colored arrows to indicate a TAP port association. Also, we enumerate the cluster and IP-block TAPs for reference. The diagram refers to the use of Region DFx Units (RDUs) and Cluster DFx units (CDUs) which is from the Chassis DFx Gen 2.0 generation. Chassis DFx Gen2.1 has not defined a clear architecture as of this release. Server SoC uses a sub-system as a replacement to the RDU and client SoCs are composed of a sea of clusters without any RDUs.

Figure 3-1. Abstract view of a generalize SoC TAP placement



(1) CLTAP: There is one designated Chip-Level TAP (CLTAP) in a SoC component connected to their assigned package pins. CLTAP controls the mode in which all of the slave TAPs at this level 1 of the network interact with the CLTAP. The CLTAP is also responsible for boundary scan functions for the SoC.

CLTAP-level hierarchy (level-1): The network adapter is graphically represented by a green box with the letter 'A'. Each TAP has an associated adapter logic block to show its connection to the network but this is for illustration purposes only. The actual logic implementation is a single module with a number of ports connecting individual TAPs. The CLTAP and any parent TAP has a select register that determines the mode of operation for each TAP for its level of hierarchy. The modes are defined as, Normal, Exclusive, Isolated or Shadow. When a TAP is selected as "Normal" on the network this means that the selected TAP is in series with the CLTAP (or any parent TAP). Each TAP at this level of hierarchy (level-1) may be selected to operate on the primary (in series with the CLTAP) or secondary TAP port with CLTAP's secondary select register. A secondary TAP port is most likely re-used from other functional pins. TAPs in level-1 hierarchy are the only ones that can be selected on to the secondary TAP port. Adapters are composed of muxes to serially connect a selected TAP with other selected TAP. The path through all adapter blocks is combinational logic. Although the logic is contained in one module the routing to and from the TAPs will add wire load delay that may be greater than the gate delay. In this topology it is important to balance and limit the number of TAPs in a given hierarchy to maintain a high frequency of operation. It is SoC dependent to choose the number of TAPs at each level and the number of regional TAPs to control the cluster TAPs.

(2) Region_A TAP0: Region DFx Unit (RDU) with a slave TAP to control a group of cluster DFx units (CDU) at the level-2 sub-network which in this diagram is composed of cluster TAPs 2 and 3. This separates the network into modular pieces to enable quicker integration and ease of creating derivative SoCs from base SoCs. This is the concept of the modular-DFx fabric that the DFx Chassis (Gen2.0) is based upon. In this diagram, all the slave TAP masters of a sub-network are colored in orange (except for the CLTAP). The child TAPs controlled by Region A sTAP are in series with it.

(3) Master DFx unit and specific IP TAPs: The level-1 hierarchy is intended to support the DFX TAP fabric. In the Chassis DFx gen2.0 timeframe there are SoCs with the Atom core TAPs connected at CLTAP level of hierarchy. However, since the TAP frequency could not be supported some SoCs moved them to the IP level under a CDU TAP. It design dependent to manage the TAP network but there may be cases where an IP must be at the CLTAP's level of control.

(4) Cluster TAP2 at level-2: This CDU TAP is a master of level-3 sub-network where it controls access to a number of IP-block TAPs. A large IP may have its own TAP network in which case the level-3 TAP would be a parent TAP to yet another level of hierarchy (sub-tap network at level-4) to its own internal TAPs. This may exist within a complex IP-block which contains a number of TAPs. Of course, the number of levels of hierarchy has trade-offs just like the number of TAPs at a given level hierarchy.

(5) IP-block TAPs: This illustrates that the cluster TAP2 is the parent controlling a group of IP-blocks (IP TAP8 and 9) within this partition. There may be any number of IP TAPs but this block diagram only shows two to reduce clutter.

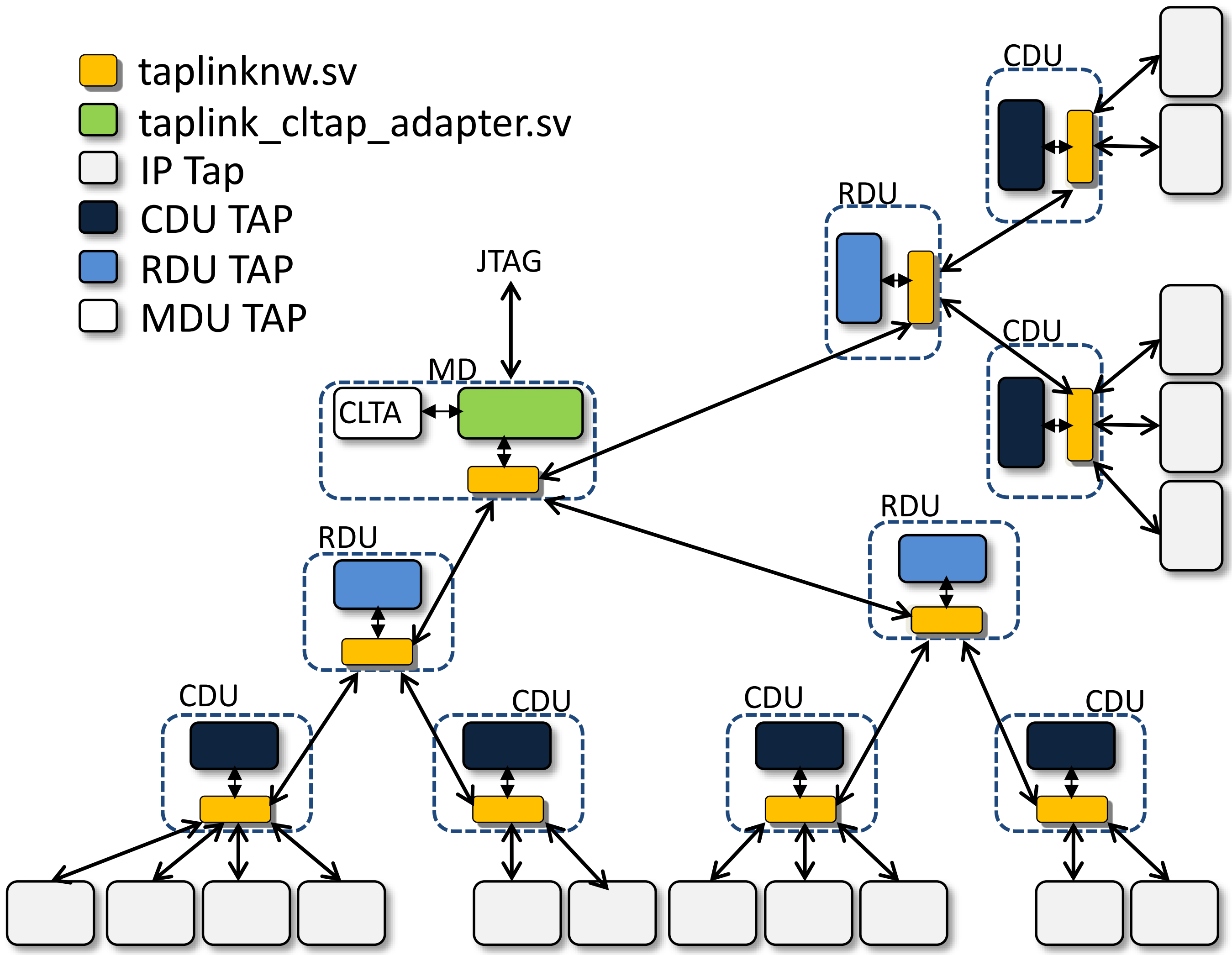
(6) TAP6 with WTAP control: Slave TAPs have the option to control a WTAP and is shown here for completeness. This spec discourages the use of WTAPs because of their inherent limitations except in specific use models which only apply to a small number of hard-IP blocks.

(7) Other Regions and other Atom (IA-cores): In the past, the IA-core have been placed at the CLTAP level of hierarchy but recent SoCs have moved them to the Cluster DFx Unit's level of hierarchy to satisfy timing closure at higher TAP frequencies (above 50MHz). It is the integration team's decision on where to place IP TAPs in general to meet the expected TAP timing closure at the required frequency.

3.2 TAP Link network

The TAP link network is shown in Figure 3-2 is an illustration of a TAP link network in the Chassis DFx 2.0 form of a Master DFx unit with a CLTAP and a layer of Region DFx Unit (RDU) TAPs and Cluster DFx Unit (CDU). The TAP link network is based on a protocol that performs indirect addressing within a set of two IR/DR scans. The first phase of the IR-DR / IR-DR sequence protocol is TAP link network IP's IR opcode then the DR would contain the target TAP's IR opcode. In the second phase, the IR would be the TAP link network IP's DR address that points to the TAP of interest. Then the DR of the second phase would be the TDR payload that is shifted in the register. The TAP link network IPs are parameterized to handle N number of child TAPs and M number of next network IPs.

Figure 3-2. TAP link example that replaces a hierarchical TAP network



3.3 Slave TAP FSM Encoding

Table 3-2 provides the expected TAP Finite State Machine (FSM) encodings for all master and slave TAPs in the component. Two options are available; a four bit binary encoding and a 16-bit one-hot encoding. The hex value is the bit position for the one-hot encoding. Defining the state values will provide consistency between agents that may have been developed from different IP providers. This allows development of pre-silicon checkers and test benches that can be universally shared among the SoC product divisions.

Table 3-2. TAP FSM state encodings

TAP State	One-hot Encoding	Hex Encoding	Binary Encoding
Test-Logic-Reset	0000_0000_0000_0001b	0001h	0000b
Run-Test/Idle	0000_0000_0000_0010b	0002h	0001b
Select-DR-Scan	0000_0000_0000_0100b	0004h	0010b
Capture-DR	0000_0000_0000_1000b	0008h	0011b
Shift-DR	0000_0000_0001_0000b	0010h	0100b
Exit1-DR	0000_0000_0010_0000b	0020h	0101b
Pause-DR	0000_0000_0100_0000b	0040h	0110b
Exit2-DR	0000_0000_1000_0000b	0080h	0111b
Update-DR	0000_0001_0000_0000b	0100h	1000b
Select-IR-Scan	0000_0010_0000_0000b	0200h	1001b
Capture-IR	0000_0100_0000_0000b	0400h	1010b
Shift-IR	0000_1000_0000_0000b	0800h	1011b
Exit1-IR	0001_0000_0000_0000b	1000h	1100b
Pause-IR	0010_0000_0000_0000b	2000h	1101b
Exit2-IR	0100_0000_0000_0000b	4000h	1110b
Update-IR	1000_0000_0000_0000b	8000h	1111b

3.4 Slave TAP IR Address Support

Table 3-3 lists the TAP instructions and opcode assignments for each slave TAP implementation. The Chip-Level TAP (CLTAP) is expected to support the minimum set of boundary scan instructions necessary to perform package level IO interconnect testing. However, it is optional for an IO agent slave TAP to control its own boundary scan signals. If enabled, then all the rules and guidelines that govern the operation of the boundary scan chain apply to this slave TAP. A hard-IP agent may override the existing boundary scan control signals for general HVM test requirements, but this outside of the 1149.1-defined behavior and cannot interfere with it. Any decoded unsupported or reserved instruction will point to the Bypass register. Embedded TAPs in externally purchased IP-blocks may not be compliant to these IR opcode requirements unless the IP-block is designed to the IOSF specification. It is required that all IOSF agent designs use the SIP TAP IP-block on the Intel Re-use Repository which is compliant to the SoC TAP HAS rev0.90 (or later) specification. Refer to Table 3-4 for more details about the IR instructions.

Table 3-3. TAP IR opcode support

TAP IR	Security level	Opcode ValueError! Reference source not found.	CLTAP	Slave TAP w/o bscan	Slave TAP w/bscan[6]
Common opcodes for both CLTAP and slave TAP					
SAMPLE/PRELOAD	green	0x01	Mandatory	Reserved	Mandatory
IDCODE	green	0x02	Mandatory	Reserved	Reserved
PRELOAD	green	0x03	Optional (See Note **Error! Reference source not found.**)	Reserved	Optional (See Note **Error! Reference source not found.**)
CLAMP	green	0x04	Optional	Reserved	Optional
USERCODE	green	0x05	Optional (See Note **Error! Reference source not found.**)	Reserved	Reserved
INTEST	green	0x06	Optional (See Note **Error! Reference source not found.**)	Reserved	Optional (See Note **Error! Reference source not found.**)
RUNBIST	green	0x07	Optional (See Note **Error! Reference source not found.**)	Reserved	Optional (See Note **Error! Reference source not found.**)
HIGHZ	green	0x08	Mandatory	Reserved	Mandatory
EXTEST	green	0x09	Mandatory	Reserved	Mandatory
TOGGLE_SETUP	green	0x0A	Optional	Reserved	Optional
SELECTIVE_TOGGLE	green	0x0B	Optional	Reserved	Optional
SLVIDCODE	green	0x0C	Reserved	Mandatory	Mandatory
EXTEST_TOGGLE	green	0x0D	Optional	Reserved	Optional

TAP IR	Security level	Opcode ValueError! Reference source not found.	CLTAP	Slave TAP w/o bscan	Slave TAP w/bscan[6]
EXTEST_PULSE	green	0x0E	IO dependent **Error! Reference source not found.**	Reserved	IO dependent**Error! Reference source not found.**
EXTEST_TRAIN	green	0x0F	IO dependent**Error! Reference source not found.**	Reserved	IO dependent**Error! Reference source not found.**
INIT_SETUP	green	0x18	Optional	Reserved	Optional
INIT_RUN	green	0x19	Optional	Reserved	Optional
CLAMP_HOLD	green	0x1A	Optional	Reserved	Optional
CLAMP_RELEASE	green	0x1B	Optional	Reserved	Optional
IC_RESET	green	0x1C	Optional	Reserved	Optional
BYPASS	green	0xFF[5]	Mandatory	Mandatory	Mandatory
CLTAP opcodes					
ECIDCODE	green	0x00	Optional	Reserved	Reserved
CLTAPC_SEC_SEL	green	0x10	Optional	NA	NA
CLTAPC_SELECT	green	0x11	Mandatory	NA	NA
CLTAPC_SELECT_OVR	green	0x12	Mandatory	NA	NA
Reserved	green	0x13	Reserved		
CLTAPC_REMOVE	green	0x14	Optional	NA	NA
CLTAPC_TDRRSTEN	green	0x15	Optional	NA	NA
CLTAPC_ITDRRSTSEL	green	0x16	Optional	NA	NA
CLTAPC_RTDRRSTSEL	green	0x17	Optional	NA	NA
CLTAPC reserved opcode space for future TAP HAS specification use.	green	0x1D-0x2F	Reserved	NA	NA
CLTAPC SoC defined opcodes	various (IP-block dependent)	0x30-0xFE[5]	SoC defined	NA	NA
Slave TAP opcodes					
TAPC_SEC_SEL	orange	0x10	NA	Optional	Optional

TAP IR	Security level	Opcode ValueError! Reference source not found.	CLTAP	Slave TAP w/o bscan	Slave TAP w/bscan[6]
TAPC_SELECT	green	0x11	NA	Conditional**Error! Reference source not found.**	Conditional**Error! Reference source not found.**
Reserved	green	0x12	NA	Reserved	Reserved
TAPC_WTAP_SEL	orange	0x13	NA	Optional	Optional
TAPC_REMOVE	green	0x14	NA	Optional	Optional
TAPC_TDRRSTEN	green	0x15	NA	Optional	Optional
TAPC_ITDRRSTSEL	green	0x16	NA	Optional	Optional
TAPC_RTDRRSTSEL	green	0x17	NA	Optional	Optional
TAPC reserved opcode space for future TAP HAS specification use.	green	0x1D-0x2F	NA	Reserved	Reserved
TAPC Agent defined opcodes	various (IP-block dependent)	0x30-0xFE[5]	NA	Agent (IP-block) defined	Agent (IP-block) defined
Note 1	Unused or reserved opcodes must be decoded to point to the Bypass register.				
Note 2	These boundary scan instructions are implemented with an opcode but no other support logic since these are generally not used with Intel TAPs for boundary scan operations.				
Note 3	If the agent instantiates a modular Phy (high speed serial differential IOs) that supports 1149.6 IO test protocol, then the TAP is required to implement the control signals necessary to operate the boundary scan chain in a dot 6 compliant mode.				
Note 4	If this slave TAP's parameter (STAP_NUMBER_OF_TAPS_IN_TAP_NETWORK >=1) indicates at least one TAP that it controls on a sub-TAP network then a Select register is required.				
Note 5	The length of the slave TAP IR register may be N-bits in length (recommended minimum of 8-bits). The Bypass instruction is decoded as {111…11}, where a logic 1 is in every bit position of the instruction register. The value shown is for an 8-bit IR length and it is for illustration purposes only.				
Note 6	The boundary scan opcodes and RTL logic will be enabled for slave TAPs when the parameter is set, otherwise, the opcodes are reserved and cannot be used. STAP_ENABLE_BSCAN = 0, opcodes 0x1-0xB and 0xD-0x1C are reserved and point to the BYPASS register. STAP_ENABLE_BSCAN = 1, enables opcodes 0x1-0xB and 0xD-0x1C to be active for boundary scan and the associated RTL logic is enabled for this sTAP.				

1

Table 3-4. Public TAP instruction definitions

Instruction	Encoding N-bit IR (hex)	Data Register Selected	Description
Common opcodes for CLTAP and slave TAP			
SAMPLE/PRELOAD	0x1	Boundary Scan	The SAMPLE/PRELOAD instruction is used to allow scanning of the boundary-scan register without causing interference to the normal operation of the device. Two functions can be performed by a single instruction. SAMPLE: provides a snapshot of the data on the input and output pins without affecting the normal operation of the device. PRELOAD: provides an initial pattern to be placed into the boundary-scan register cells. This allows initial known data to be present prior to the selection of another boundary-scan test operation. This is a public instruction and public data register with a green level of security access.
IDCODE	0x2	IDCODE	The IDCODE instruction is forced into the parallel output latches (flip-flops) of the instruction register during the Test-Logic-Reset TAP state. This allows the device identification data register to be selected immediately when entering the Shift-DR state. This is a public instruction and public data register with a green level of security access.
PRELOAD	0x3	Boundary Scan	It has been an historical Intel practice to combine the SAMPLE and PRELOAD instructions together. If they need to be separated for any reason, then SAMPLE will be assigned 0x1 and PRELOAD will be assigned 0x3. This is a public instruction and public data register with a green level of security access.
CLAMP	0x4	Bypass	This instruction allows static "guarding values" to be set onto components that are not specifically being tested while maintaining the Bypass register as the serial path through the device. This is a public instruction and public data register with a green level of security access.

Instruction	Encoding N-bit IR (hex)	Data Register Selected	Description
USERCODE	0x5	USERCODE	This opcode is for FPGAs with a separate flash memory connected to the FPGA for normal operation. The TAP instruction allows identification of the flash memory component through the FPGA master TAP. Refer to the IEEE1149.1-2001 specification for more details. This is a public instruction and public data register with a green level of security access.
INTEST	0x6	Boundary Scan	The INTEST instruction allows static (slow-speed) testing of the on-chip system logic, with each test pattern and response being shifted through the boundary-scan register. The INTEST instruction requires that the on-chip system logic be operated in a single-step mode, where the circuitry moves one step forward in its operation each time shifting of the boundary-scan register is completed. This is a public instruction and public data register with a green level of security access.
RUNBIST	0x7	BIST	This instruction connects the DR shift register of a RAM/Array BIST engine on the die. It is implementation-specific for what function is accessed or initiated with this IR address. This is a public instruction and public data register with a green level of security access.
HIGHZ	0x8	Bypass	The HIGHZ instruction is used to force all outputs of the device (except TDO) into a high impedance state. This instruction shall select the Bypass Register to be connected between TDI and TDO in the Shift-DR controller state. This is a public instruction and public data register with a green level of security access.
EXTEST	0x9	Boundary Scan	The EXTEST instruction provides a means to test the connection between this component and another on the board or off the board through a connector by toggling the output Boundary-Scan Register Cells. Once enabled, the outputs toggle once every falling edge of Test Clock (TCK) in the Run-Test/Idle state. This is a public instruction and public data register with a green level of security access.

Instruction	Encoding N-bit IR (hex)	Data Register Selected	Description
TOGGLE_SETUP	0x0A	TOGGLECTRL	This opcode uses the Toggle Control register. • It sets the frequency out the I/O pin, based on TCK frequency in. • Sets relationship of TCK frequency to I/O response in RTI. • The binary value(s) for the TOGGLE_SETUP instruction may be selected by the device designer. This is a public instruction and public data register with a green level of security access. This opcode is not implemented as part of the CLTAP/sTAP RTL rev1.5.x.
SELECTIVE_TOGGLE	0x0B	Boundary Scan	This opcode uses the boundary register to enable selective toggling of a designated IO. • Boundary register initialized prior to SELECTIVE_TOGGLE. • I/O can drive '0' or '1' out, no restriction from starting state of toggle. • A '1' in the boundary cell for each I/O selects the I/O to toggle in RTI. • Instruction doesn't move bits to the update flop in Update-DR. • I/O maintains same state as previously initialized prior to SELECTIVE_TOGGLE. • The binary value(s) for the SELECTIVE_TOGGLE instruction may be selected by the device designer. This is a public instruction and public data register with a green level of security access. This opcode is not implemented as part of the CLTAP/sTAP RTL rev1.5.x.
SLVIDCODE	0x0C	SLVIDCODE	This instruction is used to identify a slave TAP controller in an agent. All slave TAPs are expected to load this opcode in the instruction register after a TRST_b or exit from TLR state. This is a public instruction and public data register with a green level of security access.
EXTEST_TOGGLE	0x0D	Boundary Scan	This instruction enables the Extest toggle feature for IOs. This function toggles the IO at the same frequency that the TCK pin is clocked at. This is a public instruction and public data register with a green level of security access.

Instruction	Encoding N-bit IR (hex)	Data Register Selected	Description
EXTEST_PULSE	0x0E	Boundary Scan	This instruction enables the Extest pulse feature for IOs that support the 1149.6 boundary scan feature. Refer to the IEEE1149.6 specification for more information. This is a public instruction and public data register with a green level of security access.
EXTEST_TRAIN	0x0F	Boundary Scan	This instruction enables the Extest train feature for IOs that support the 1149.6 boundary scan feature. Refer to the IEEE1149.6 specification for more information. This is a public instruction and public data register with a green level of security access.
INIT_SETUP	0x18	INITSETUP	A TDR contains the binary for configuring the component. • The I/O analog characteristics are not changed while the I/O is controlled by the system logic. • Instruction like EXTEST will make the TDR binary take effect on the I/O. • Controlling PLL or power domains can be immediate with Update-DR • Can check the state of important that relate to external pins or special internal functions or conditions. • The binary value(s) for the instruction may be selected by the component designer. This is a public instruction and public data register with a green level of security access. This opcode is reserved for future use and not implemented in the CLTAP/sTAP RTL rev1.5.x.

Instruction	Encoding N-bit IR (hex)	Data Register Selected	Description
INIT_RUN	0x19	INITRUN_STAT	Used if sequential initialization is required by waiting in Run-Test/Idle for a period of time. • I/O response is EXTEST like, takes control of the I/O. • Boundary register should be initialized before execution. • Can work in conjunction with the INIT_SETUP TDR. • Can have a status TDR to monitor progress or when completed. • The binary value(s) for the instruction may be selected by the component designer. This is a public instruction and public data register with a green level of security access. This opcode is reserved for future use and not implemented in the CLTAP/sTAP RTL rev1.5.x.
CLAMP_HOLD	0x1A	CLAMP_CTRL	Control the state of the optional Test-Mode Persistence (TMP) controller. • Hold: The boundary register maintains control of I/O regardless of the instruction currently loaded. • Release: Boundary register not in control with non-boundary register commands loaded. • Sticky bit. • May want to control some I/O signals while running an external BIST for example. • The binary value(s) for the instruction may be selected by the component designer. This is a public instruction and public data register with a green level of security access. This opcode is reserved for future use and not implemented in the CLTAP/sTAP RTL rev1.5.x.
CLAMP_RELEASE	0x1B	CLAMP_CTRL	Refer to the CLAMP_HOLD opcode description. This is a public instruction and public data register with a green level of security access. This opcode is reserved for future use and not implemented in the CLTAP/sTAP RTL rev1.5.x.

Instruction	Encoding N-bit IR (hex)	Data Register Selected	Description
IC_RESET	0x1C	IC_RESET	This register can block external resets and generate internal mission mode resets. • Can be made sticky to hold some logic in reset, while the instruction is not loaded. • The binary logic value(s) for the *IC_RESET* instruction should not include the all zeros value. • The binary logic value(s) for the *IC_RESET* instruction may be selected by the component designer. This is a public instruction and public data register with a green level of security access. This opcode is reserved for future use and not implemented in the CLTAP/sTAP RTL rev1.5.x.
BYPASS	0xFF*	Bypass	The BYPASS command selects the Bypass Register, a single bit register connected between the TDI and TDO pins. This register provides a pass-through for this TAP so that other TAPs in the serial chain can be accessed. This is a public instruction and public data register with a green level of security access. **Note:* The Bypass instruction is dependent on the length of the IR register.
CLTAP specific opcodes			
ECIDCODE	0x0	ECIDCODE	This opcode enables an internal TDR for Electronic chip identification (ECID) is a “unique embedded serial number” for the component. The length of the TDR is parameterized and the value is captured during Capture-DR state. This opcode is reserved for future use and it is not implemented in TAP rev1.5.x It possible uses are: Component lifetime history, tracking and possibly identify counterfeit components. This is a public instruction and public data register with a green level of security access.
CLTAPC_SEC_SEL	0x10	CLTAPC_SEC_SEL	This instruction enables a test data register in the CLTAPC to control which TAP on the network will be connected to the secondary TAP port. This is a private instruction and private data register.

Instruction	Encoding N-bit IR (hex)	Data Register Selected	Description
CLTAPC_SELECT	0x11	CLTAPC_SELECT	This instruction is from the 1149.7 T0 specification for connecting TAPs within an SoC. This TDR is used to exclude, decouple, or allow normal scan operations with other TAPs in the fabric or within an agent. This register is mandatory. This is a public instruction and public data register with a green level of security access.
CLTAPC_SELECT_OVR	0x12	CLTAPC_SELECT_ OVR	This instruction will access the SELECT override data register. The output of this override register is OR'd with the output of the CLTAPC_SELECT register. The SELECT_OVR register is cleared only by powergood reset and not the TRST_b or the Test-Logic/Reset state. This is a public instruction and public data register with a green level of security access.
Reserved	0x13	BYPASS	For future use.
CLTAPC_REMOVE	0x14	CLTAPC_REMOVE	This instruction and data register removes a master TAP from the primary TAP pins. The register is a single bit in length if it is set to logic 1. Then in the Run/Test-idle state, a mux will disconnect the master TAP from the primary TAP port pins. Implementation of this opcode and register is optional.
CLTAPC_TDRRSTEN	0x15	CLTAPC_TDRRSTE N	This instruction enables which reset type will reset the internal and remote test data registers. The selection is powergood reset (default), TRST bar, or a software write. When the reset is selected then the ITDRRSTSEL and RTDRRSTSEL determine which TDRs are cleared by that selected reset.
CLTAPC_ITDRRSTSEL	0x16	CLTAPC_ITDRRST SEL	This instruction enables access to the ITDRRSTSEL register. Each bit is assigned to a TDR which enables this TDR to be reset based on the type selected by CLTAPC_TDRRSTEN.
CLTAPC_RTDRRSTSEL	0x17	CLTAPC_RTDRRST SEL	This instruction enables access to the ITDRRSTSEL register. Each bit is assigned to a TDR which enables this TDR to be reset based on the type selected by CLTAPC_TDRRSTEN.
Reserved for future use by this spec	0x1D – 0x2F	Bypass	These opcodes are reserved for future use in convergence issues.
SoC or Agent defined opcodes	0x30-0xFE	Bypass	These opcodes are available to the IP-block developer.

Instruction	Encoding N-bit IR (hex)	Data Register Selected	Description
Slave TAP specific opcodes			
TAPC_SEC_SEL	0x10	TAPC_SEC_SEL	This instruction enables the TAPC_SEC_SEL data register in an agent (TAPC) to control which TAPs within this agent that will be connected to the secondary TAP port. This requires the inclusion of the optional secondary slave TAP interface. This is a private instruction and private data register.
TAPC_SELECT	0x11	TAPC_SELECT	This TDR is used to exclude, decouple, or allow normal IR/DR scan operations with other TAPs within this agent. This register is required for agents that have three or more TAPs within this agent including the agent slave TAP. This is a private instruction and private data register.
Reserved	0x12	Bypass	For all slave TAPs, 0x12 is reserved to maintain opcode consistency with the master TAP. This is a private instruction.
TAPC_WTAP_SEL	0x13	TAPC_WTAP_SEL	This instruction and data register will enable a WTAP to be placed in series with the master TAP. Generally, the WTAPs are arranged in parallel and only one WTAP is access at a time. A series stitching of WTAPs is supported. This is a private instruction and private data register.
TAPC_REMOVE	0x14	CLTAPC_REMOVE	This instruction and data register removes a slave TAP from its primary TAP pins. The register is a single bit in length. If it is set to logic 1, then in the Run/Test-idle state, a mux will disconnect the master TAP from the primary TAP port pins. Implementation of this opcode and register is optional.
TAPC_TDRRSTEN	0x15	TAPC_TDRRSTEN	This instruction enables which reset type will reset the internal and remote test data registers. The selection is powergood reset (default), TRST bar, or a software write. When the reset is selected then the ITDRRSTSEL and RTDRRSTSEL determine which TDRs are cleared by that selected reset.
TAPC_ITDRRSTSEL	0x16	TAPC_ITDRRSTSE L	This instruction enables access to the ITDRRSTSEL register. Each bit is assigned to a TDR which enables this TDR to be reset based on the type selected by CLTAPC_TDRRSTEN.

Instruction	Encoding N-bit IR (hex)	Data Register Selected	Description
TAPC_RTDRRSTSEL	0x17	TAPC_RTDRRSTSE L	This instruction enables access to the ITDRRSTSEL register. Each bit is assigned to a TDR which enables this TDR to be reset based on the type selected by CLTAPC_TDRRSTEN.
Reserved for future use by this spec	0x1D – 0x2F	Bypass	These opcodes are reserved for future use in convergence issues.
SoC or Agent defined opcodes	0x30-0xFE	Bypass	These opcodes are available to the IP-block developer.

3.5 Slave TAP Reset Behavior

Table 3-5 lists the expected behaviors of the TAPs within the agents with the reset configuration or initial conditions.

Table 3-5. sTAP reset behavior

Configuration condition	Powergood reset (powergood_rst_b)	TRST_b or Test-Logic-Reset state
Slave TAP IR	IR = SLVIDCODE	IR = SLVIDCODE
TAPC_SEC_SEL	TAPC_SEC_SEL = 0x0	Retain previous state. This register is only cleared with powergood reset.
TAPC_SELECT	TAPC_SELECT= 0x0	Retain previous state. This register is only cleared with powergood reset.
TAPC_WTAP_SEL	TAPC_WTAP_SEL = 0x0	Retain previous state. This register is only cleared with powergood reset.
TAPC_REMOVE	TAPC_REMOVE = 0	Retain previous state. This register is only cleared with powergood reset.
TAPC_TDRRSTEN	TAPC_TDRRSTEN = 0	Retain previous state. This register is only cleared with powergood reset.
TAPC_ITDRRSTSEL	TAPC_ITDRRSTSEL = 0	Retain previous state. This register is only cleared with powergood reset.
TAPC_RTDRRSTSEL	TAPC_RTDRRSTSEL = 0	Retain previous state. This register is only cleared with powergood reset.
TAPC reserved data registers (0x1D – 0x2F)	Reset to logic 0	Retain previous state. This register is only cleared with powergood reset.

Configuration condition	Powergood reset (powergood_rst_b)	TRST_b or Test-Logic-Reset state
TAPC private data registers (0x30-0xFE)	Reset to default state	Default is power good reset. A per register selection of reset type is an optional parameter. Selectable reset is power good reset, software reset or Test-Logic-Reset state including TRST_B.

3.15 Slave TAP Rules

Rules:

1. If an agent requires a serial test/debug register then a TAP interface with a TAP controller is required and the slave TAP is the preferred architecture. A Wrapper Serial Port (WSP but commonly referred to as WTAP) option is available if necessary.
    a. A slave TAP is based on the SoC TAP HAS and is compliant to IEEE1149.1-2001 with a couple exceptions (for example, SLVIDCODE opcode and TDR).
        i. Each slave TAP shall implement the full set of five interface signals: ftap_tck, ftap_tms, ftap_tdi, atap_tdo, and ftap_trst_b.
        ii. For legacy TAPs embedded in an agent prior to the existence of IOSF, the ftap_trst_b signal may not be connected internally but must be included as part of the TAP interface. The same is true for external IPs that are wrapped for the IOSF interface.
        iii. From an interface point of view the agent may be a soft-IP or a hard-IP block.
    b. This slave TAP must operate with a TAP.7 network as defined in the SoC TAP HAS.
        i. In essence, this requirement stipulates that the agent TAP must be passive with respect to the operation of the network and cannot hinder its operation.
    c. An agent TAP must provide TDO enable output signal as part of the signal list.
        i. This rule applies to Intel-developed TAP agents. The usage of this signal is determined by the TAP network and CLTAP IP-blocks.
        ii. TDO enable is defined as: TDOen = Shift-DR OR Shift-IR and the output is flopped on the falling edge of TCK.
    d. The TDO and TDOen output signals must be clocked on the falling edge of TCK.
        i. There is one exception to this rule for a re-time TAP. A re-time TAP has the option to clock the TDO output on the rising or falling edge of TCK.
        ii. A re-time TAP is a slave TAP that is designated with this function or a specifically designed re-time TAP IP.
        iii. IP-blocks or DFx fabric TAPs (TAPs that are part of the Modular-DFx architecture) cannot be re-time TAPs and must not enable the posedge TDO parameter.
        iv. A re-time TAP used at the CLTAP's level of hierarchy can ONLY be a negative edge clocked TDO.
        v. Generally, the last TAP at Region DFx Unit or the Cluster DFx Unit's level of hierarchy may be a positive edge TDO re-time TAP.

            1. The decision to use a positive edge or negative edge TDO parameter is up to the SoC integration team.
    e. When entering into the Capture-IR state the Least Significant Bits[1:0] of the instruction shift register must be "01" and the Most Significant Bits [NumOfIRbits-1:2] must be logic 0.
        i. For example, if the IR register is 8 bits, then the instruction shift register must capture 8'b0000_0001.
    f. All TCK driven flops must not be part of the scan chain for HVM testing.
        i. This may be an obvious rule since TAPs are used for controlling test operations but it is possible to identify specific TAPs to be separated from scan while leaving others.
2. The slave TAP must implement a slave ID code (SLVIDCODE).
    a. A slave TAP's instruction register will reset to the default opcode value of 0xC and connect the SLVIDCODE register between TDI and TDO after the Update-IR state.
        i. This is a known variant on the 1149.1 rules.
    b. All Intel developed agents that implements a slave TAP must implement a SLVIDCODE and not an IDCODE. Atom cores are expected to implement both starting with Goldmont.
    c. Any external IP that requires an IDCODE will get an assigned value from the SLVIDCODE Assignment Tool.
    d. The ID code generated from the tool is applied to the IDCODE of the vendor TAP.
    e. If the vendor TAP gets an ID assignment from the insertion flow, this value should *not* be used as the ID for the feature but it will be stored as part the database generated for the project. In place of the vendor's ID will be the SLVIDCODE. The reason is because the SLVIDCODE has information about which TAP is controlling the network adapter that is connecting this vendor TAP to the network.
3. All opcodes in **Error! Reference source not found.** must be supported. Unsupported or reserved codes must be decoded to point to the Bypass register.
    a. The minimum IR length must be no less than 8 bits.
    b. The IR length may be greater than 8 bits (but less than 16-bits) if necessary to support a larger sparse matrix arrangement of opcodes.
    c. Opcodes 0x0 through 0x2F are reserved for boundary scan, commonly used opcodes for all TAPs like the programmable reset IRs and future expansion by the SoC TAP HAS and this IOSF specification.
    d. The agent-specific opcodes begin at 0x30.
4. The test data registers (TDRs) must implement the following.
    a. The TDR must shift from the Most Significant Bit (MSB) to the Least Significant Bit (LSB). The first data bit to appear on TDO comes from the LSB of the data register.
    b. All private test data registers are constructed to be sticky and must be cleared with only power good reset (fdfx_powergood). This is a minimum requirement with the optional ability to clear the registers with a TRST_b or a software clear bit. Refer to the permission section item 7.
    c. A remote data register is not allowed at the IOSF DFx agent interface.
5. IP-blocks with a TAP must connect the DFx secure policy signal group (fdfx_secure_policy, fdfx_policy_update, and fdfx_earlyboot_exit) to this TAP and to the DFx Secure Plugin (DSP) instance. The DSP is mandatory for all IRR derived slave TAPs and all IOSF DFx compliant SIPs/HIPs. DFx security is applied to the TAP and TAP network in two ways. First, security allows or disallows a TAP to be accessed on the network and second security is applied to opcodes within any given TAP so a particular security level determines which registers are accessible.
    a. The DFx secure policy cannot break the operation of the TAP network. Therefore, the minimum set of opcodes that must always be available

regardless of the current security policy are the SLVIDCODE and Bypass instructions. A TAP may be orange or red on the network which means that the green opcodes will be inaccessible but this doesn't violate the rule since the security is applied to the parent's select TDR and forces the TAP to appear isolated when security clearance is not granted.

   b. The IR shift register in a TAP must not be affected by any bit value of the DFx secure policy bus.
   c. The SoC TAP HAS defines opcodes between 0x0 and 0x2F and their security color code levels defined in **Error! Reference source not found.**. In general, most are green level of access since many of them are public instructions.

6. An IOSF DFx agent is defined as having one standard slave TAP interface. If the number of TAPs within an agent is greater than two then the soft-IP block must implement a TAP network. The secondary slave TAP interface is intended only for supporting expanded networking capabilities such as the hierarchical-hybrid or the tertiary TAP port. However, it doesn't make sense to require a TAP network to support only one more TAP within the IP (see Permissions section).

Recommendations:

1. Refer to the SoC TAP HAS rev0.90 or later for more information about the CLTAP, slave TAPs, WTAPs, and network control features.

Permissions:

1. An agent may have a secondary slave TAP interface. This can only be used by the network to implement the hierarchical-hybrid or the tertiary TAP port capabilities. A second slave TAP cannot be connected to this interface.
2. An IOSF DFx agent may have up to two slave TAPs on the agent/IP-block interface. The IOSF DFx spec will not define the TAP as a bus. Therefore, the IP-block must include a unique pre-fix name on the ports. For example, abc_ftap_tck and def_ftap_tck where "abc" and "def" are functional block notations describing the sub-unit names.
   a. This sub-TAP network within an agent is recommended to be a hierarchical topology.
   b. If the subordinate TAPs are of the WTAP type, then WTAP network must be used to control the WTAPs. The Logic Vision MBIST WTAP is not included as part of this requirement since the MBIST insertion flow keeps the MBIST WTAPs connected separately to its own 1149.1 TAP.
3. A slave TAP that is driving a WTAP or group of WTAPs will have the option to drive the capture and shift signals on either the rising or falling edge of TCK.
   a. Note: The fwtap_updatewr is always flopped on the rising edge of TCK.
4. An agent slave TAP is allowed to implement the boundary scan opcodes and logic for local control over its section of the boundary scan chain.
   a. It is expected, but not required, that the SoC CLTAP implement the boundary scan public instructions to control the bscan test register for all of the IO blocks in the pad ring.
   b. An agent that implements the opcodes and logic for a local boundary scan register must be a 1149.1 compliant slave TAP and follow the supported instructions shown in **Error! Reference source not found.** for boundary scan operations.
   c. An agent must deliver a BSDL file as part of its collateral.
5. It is optional to remove the slave TAP from its TAP port pins on the network.
   a. The opcode for TAPC_REMOVE is 0x14.
   b. TAPC_REMOVE register is only a single bit.
   c. The taprmv signal will enable the TAPC TDI input to "pass-thru" mTAP to the TAP.7 network.

   d. The TAPC_REMOVE register bit is cleared by power good reset (fdfx_powergood).

6. A slave TAP may act as a master to control a sub-network of other slave TAPs.

   a. A slave TAP must implement the TAPC_SELECT instruction to control which mode the slave TAPs will function on the sub-network controlled by this sTAP.

   b. This instruction connects the TAPC_SELECT data register to TDI and TDO. After entry into the Run-Test/Idle state the outputs of the register forces the TAP or TAPs to the selected mode. The modes are normal, excluded, or isolated.

      i. Normal mode: For all IR and DR scans (capture, shift, update and others) the slave TAP is stitched serially with the CLTAP.

      ii. Excluded mode: For all IR scans, the slave TAP is in series with the CLTAP. For all DR scans the slave TAP is excluded. In other words, the DR register pointed to by the Instruction Register is not connected between TDI and TDO of the network.

      iii. Isolated mode: For all IR and DR scans the TAP is idle and does not exist between TDI and TDO as part of the TAP network.

      iv. Shadow mode: The sTAP receives IR/DR scans on TDI but the TDO output is blocked.

   c. When a sTAP isolates another slave TAP on the next level of hierarchy the adapter logic will enforce isolation by gating TCK, force TMS to logic 1 and force TRST_b to logic 1. When a slave TAP is isolated, the private TDRs are persistent can retain their programming state as long as power is applied.

7. A private test data register may be cleared with either a power good or programmatically set/cleared with Test-Logic-Reset (TLR) state or a software write.

   a. The TAPC_TDRRSTEN will enable the type of reset to be used: power good reset (fdfx_powergood), TRST_b pin or Test-Logic-Reset state (TLR), or a software write to an encoding within the reset type field.

   b. TAPC_ITDRRSTSEL selects which internal TDR (TDRs) applies the reset type. If a register is not selected the power good reset is applied.

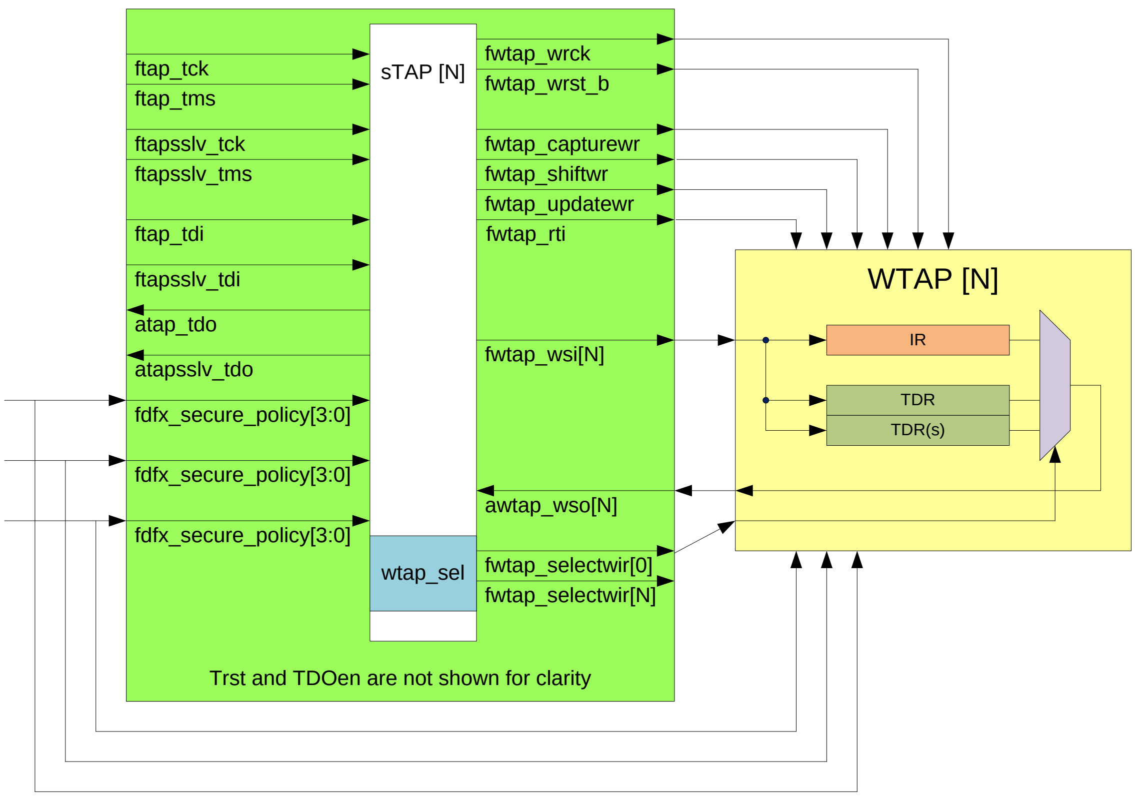
   c. TAPC_RTDRRSTSEL selects which remote TDR (TDRs) applies the reset type. If a register is not selected the power good reset is applied.

8. A SoC integration team or a large IP-block may place re-time TAPs according to the Chassis DFx Gen2 guidelines or as needed to maintain the expected frequency of operation. The sTAP acting as a re-time TAP has an optional compile time parameter to allow this TAP to flop the TDO signal on the rising edge of TCK. A cost reduced TAP specifically designed as a re-time TAP will replace the sTAP in future SoCs based on Chassis DFx Gen2 architecture. A negedge re-time may be placed anywhere within the network, however, a posedge re-time edge TDO may NOT be used as the last re-time TAP at the CLTAP's level hierarchy. IP-blocks or DFx fabric TAPs must implement the posedge TDO parameter option.

3.16 Wrapper Serial Port Architecture

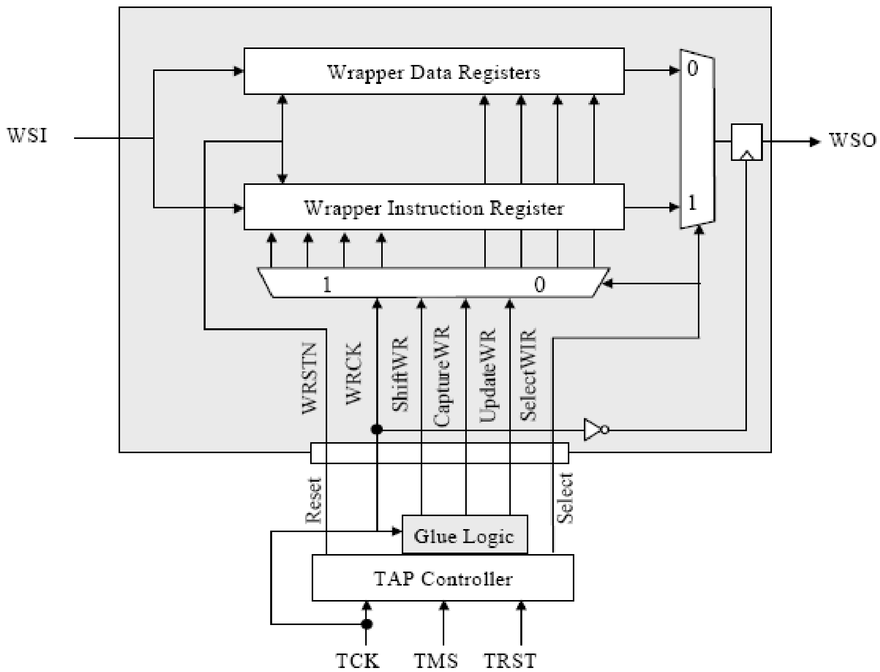
A wrapper serial port (WTAP) is an available option for accessing test and debug registers rather than the slave TAP described in the previous section, however, its use is discouraged. Shown in Figure 3-3 is a graphical view of the WTAP's interface signals to the network or fabric. Choosing a WTAP over a slave TAP is the decision of the IP-block developer. In general, for larger, more complex agents it is more desirable to instantiate a slave TAP since there are a number of advantages to this, namely, the use of the TAP.7 network for selecting the TAP onto either primary or secondary TAP ports.

Figure 3-3. Graphical view of agent WTAP



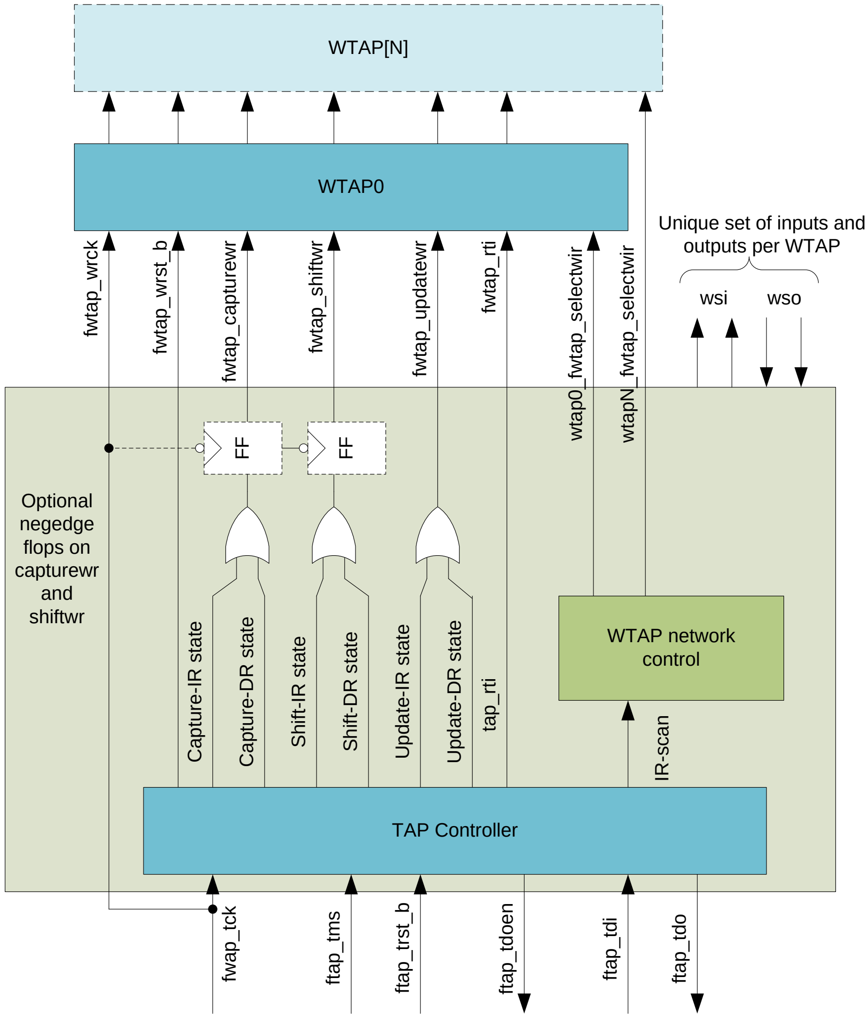
WTAP construction details are shown in Figure 3-4. This figure is taken from the IEEE1500 spec where its basic behavior is outlined but there are addition rules so that the WTAP doesn't disrupt the operation of the TAP.7 network and other slave TAPs. It is essentially an access port that uses control signals similar to those used on individual test data registers except that there isn't finite state machine controlling the overall operation. An instruction register within the WTAP contains a shift register portion and a shadow register portion. The IR shadow register decodes the updated opcode value to control which TDR is selected between the WSI input and the WSO output. The selected test/debug register is then available for Scan-DR operations. The primary difference between the WTAP and a slave TAP is that the control signals must be provided by a slave TAP elsewhere. This indicates the essential trade-off between a slave TAP and a WTAP in terms of gate count and but more importantly will not exist o the TAP.7 network. The WTAP is selected to be in series with the controlling slave TAP but any TAP on a network controlled by this slave cannot be enabled with the WTAP. This is an important consideration for choosing to use a WTAP.

Figure 3-4. WTAP block diagram



An example of how the WTAP is controlled by a slave TAP as shown in Figure 3-5. It requires a minimum amount of logic to generate the capture, shift, and update signals for both the data and instruction registers. The capture and shift signals (fwtap_capturewr and fwtap_shiftwr) can be optionally driven off of the positive or negative edge of TCK by setting a parameter. This allows the integration team to decide if they need more setup and hold time to drive the WTAP control signals from the designated slave TAP. The update signal (fwtap_updatewr) is flopped on the rising edge of TCK (no option to change this). Refer to the SoC TAP HAS rev0.90 or later for more detailed information.

Figure 3-5. TAP controller glue logic for WTAP



3.16.1 WTAP IR Opcodes

Table 3-6 lists the required opcode value for each instruction in the WTAP. **Error! Reference source not found.**

Table 3-6. WTAP IR address support

TAP IR	Opcode Value1	WTAP requirements
SLVIDCODE	0x0C	Optional, but this opcode cannot be used for any other TDR.

TAP IR	Opcode Value1	WTAP requirements
TAPC_TDRRSTEN	0x15	Optional (parameter based). This instruction enables which reset type will reset the internal and remote test data registers. The selection is powergood reset (default), TRST bar, or a software write. When the reset is selected then the ITDRRSTSEL and RTDRRSTSEL determine which TDRs are cleared by that selected reset.
TAPC_ITDRRSTSEL	0x16	Optional (parameter based). This instruction enables access to the ITDRRSTSEL register. Each bit is assigned to a TDR which enables this TDR to be reset based on the type selected by TAPC_TDRRSTEN
TAPC_RTDRRSTSEL	0x17	Optional (parameter based). This instruction enables access to the RTDRRSTSEL register. Each bit is assigned to a TDR which enables this TDR to be reset based on the type selected by TAPC_TDRRSTEN.
Agent/IP-block defined	0x00- 0x0B 0x0D-0xFE	User defined
BYPASS	0xFF	Mandatory

NOTES:
1. The WTAP IR is fixed at eight bits in length.

3.16.2 WTAP Reset Behavior

Table 3-7 lists the expected behaviors of the TAPs within the agents with the reset configuration or initial conditions.

Table 3-7. WTAP reset behavior

Configuration condition	Power_good_b reset	TRST_b or Test-Logic-Reset state
WTAP IR shift register (Capture-IR)	IR = 0x01	IR = 0x01
WTAP IR shadow register	IR = 0xFF	IR = 0xFF
WTAP Bypass	Bypass = 0	Bypass = 0

3.16.3 WTAP Rules

Rules:

1. If a TAP test and debug register (TDR) is required for the agent/IP-block, then an agent must implement a TAP to access the TDR. A WTAP is an optional implementation to satisfy this requirement but it is not a preferred solution.
    a. A slave TAP is the preferred solution over a WTAP.
    b. The WTAP is originally based on the IEEE1500 specification, but it has been slightly altered in this specification to work effectively with other slave TAPs.
    c. A remote data register is not allowed at the IOSF DFx agent interface.
2. The WTAP must implement these attributes to operate correctly with a slave TAP controlling this WTAP.
    a. The IR length of a WTAP is fixed at 8 bits.
    b. The agent's user-defined opcodes are available in this range: 0x0 – 0x0B and 0x0D through 0xFE. Refer to **Error! Reference source not found.**.
    c. Unsupported or reserved codes must be decoded to point to the BYPASS register.
    d. When entering into the Capture-IR state, the value captured in the IR shift register is 8'b0000_0001.
    e. A Bypass register of one bit in length must be included as part of the WTAP.
    f. The BYPASS instruction register opcode will be 0xFF.
    g. Upon reset, the instruction that is loaded in the IR shadow register will be BYPASS.
    h. The WSO output signal must be clocked on the falling edge of TCK.
3. All TCK driven flops must not be part of the scan chain for HVM testing.
4. The TDR must shift from the Most Significant Bit (MSB) to the Least Significant Bit (LSB). The first data bit to appear on TDO comes from the LSB of the data register.
    a. All WTAP private test data registers are constructed to be cleared with power good reset (fdfx_powergood).
5. A WTAP cannot drive a slave TAP.
    a. A WTAP does not have the protocol or signaling necessary to drive a slave TAP. It is recommended that an agent instantiates a slave TAP if it is expected to control other TAPs (slave TAPs or WTAPs) within the agent.
6. The updatewr (fwtap_updatewr) from the sTAP must be clocked on the positive edge of TCK.
7. A WTAP must not be used to accessing a local boundary scan chain.
8. The DFx secure policy signal group (fdfx_secure_policy, fdfx_policy_update, and fdfx_earlyboot_exit) is mandatory for all WTAPs. DFx security is applied to the opcodes of the WTAP. There is no WTAP network option for security. The designated slave TAP that drives the WTAP may have network access security applied as necessary.
    a. The DFx secure policy cannot break the operation of the TAP or the WTAP network. Therefore, the minimum set of opcodes that must always be available regardless of the current security policy. Opcodes for the SLVIDCODE and Bypass instructions must always be green. A TAP may be orange or red on the network which means that the green opcodes will be inaccessible but this doesn't violate the rule since the security is applied to the parent's select TDR and forces the TAP to appear isolated when security clearance is not granted.
    b. The IR shift register in a TAP must not be affected by any bit value of the DFx secure policy bus.
    c. The SoC TAP HAS defines opcodes between 0x0 and 0x2F and their security color code levels defined in **Error! Reference source not found.**. If the WTAP uses any of these it follow a similar application of the spec defined opcodes with the appropriate security level. For example, the slave ID code (SLVIDCODE), the bypass register or the programmable reset opcode are all labeled as green. In general, most are green level of access since many of them are public instructions.

Recommendations:

1. In most cases a slave TAP is a preferred solution over a WTAP.
2. However, for very small hard IP-blocks where the size of a slave TAP (approximately 800 gates) is a significant amount of logic compared to the function logic in which it is expected to interface, a WTAP make sense.

Permissions:

1. It is optional to implement a SLVIDCODE for a WTAP.
    a. If a SLVIDCODE is implemented for a WTAP then the IR opcode must be 0xC.
2. A Run-Test/Idle control signal is optionally available to the WTAP to execute an operation after the test data registers have been updated.
    a. Use of this signal is implementation dependent. One possible use model would be to align execution a test state machine after all the test data registers have been written to.
3. The WTAP's capturewr and shiftwr control signals (fwtap_capturewr and fwtap_shiftwr) are optionally driven on either the positive or negative edge of TCK from the slave TAP.
4. A private test data register may be cleared with either a power good or programmatically set/cleared with Test-Logic-Reset (TLR) state or a software write.
    a. The WTAP_TDRRSTEN (0x15) will enable the type of reset to be used: power good reset (fdfx_powergood), TRST_b pin or Test-Logic-Reset state (TLR), or a software write to an encoding within the reset type field.
    b. WTAP_ITDRRSTSEL(0x16) selects which internal TDR (TDRs) applies the reset type. If a register is not selected the power good reset is applied.
    c. WTAP_RTDRRSTSEL(0x17) selects which remote TDR (TDRs) applies the reset type. If a register is not selected the power good reset is applied.

3.16.4 Network Use Models for WTAPs

A WTAP select register (TAPC_WTAP_SEL) in a designated slave TAP will drive a set of WTAPs. A bit per WTAP determines which WTAP is active in series with this slave TAP. A sTAP can communicate with one or more WTAPs connected to its WTAP network which is illustrated in diagrams in Figure 3-6 through Figure 3-8. For larger IP-blocks or logic partitions, it may be necessary to control a variety of other slave TAPs and WTAPs as shown in Figure 3-9. Since the WTAP network is separated from the TAP.7 network, it enforces a strict behavior when the sTAP is communicating with other sTAPs or the WTAPs. This allows an auto-discovery process to handle the TAP.7 network without interference from a WTAP. The term "network" is used to describe the connectivity to and from the WTAPs but in reality this network is only a collection of wires. This is different from the TAP.7 network which has a small amount of logic in each adapter to control the slave TAP. Refer to the SoC TAP HAS rev0.90 or later for details on how the sTAP controls the WTAPs.

Figure 3-6. Option 1a: sTAP Controlled Single WTAP

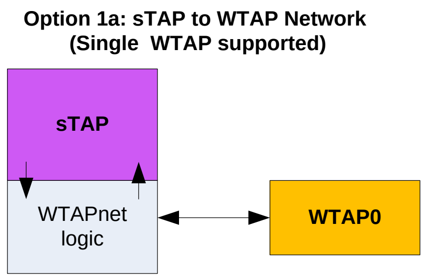


Figure 3-7. Option 1b: sTAP Controlled Multiple WTAP Network

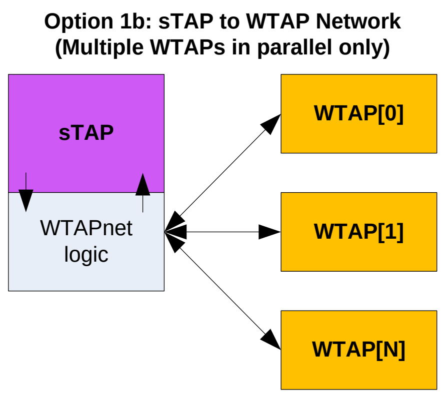


Figure 3-8. Option 1c: sTAP to WTAP in Series Network

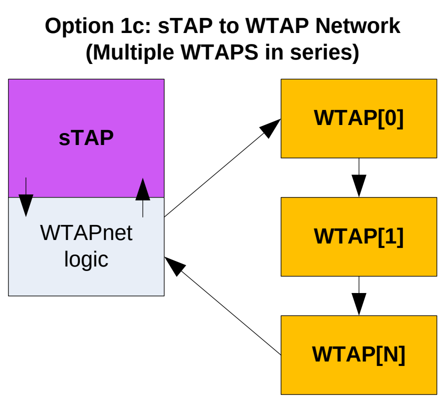
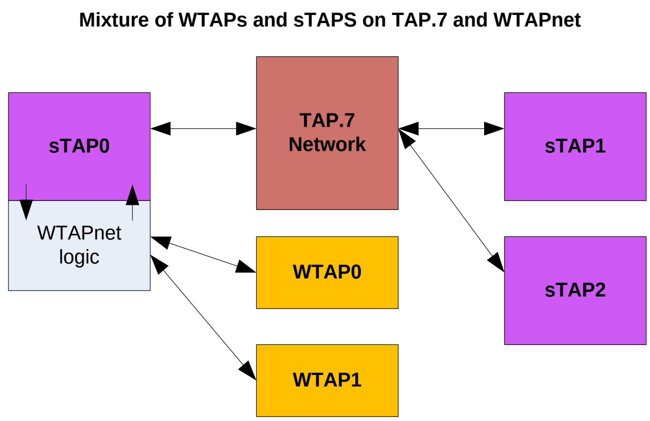


Figure 3-9. Slave TAP drives both sTAP and WTAP Networks

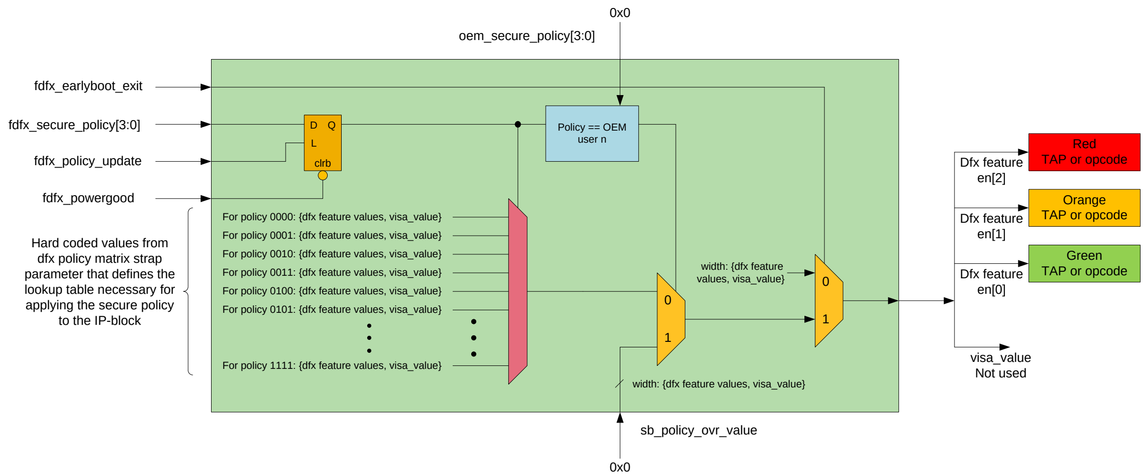


3.17 DFx secure policy plug-in for TAPs

The DFx security plug-in IP-block is shown in Figure 3-10 is required for all slave TAPs, WTAPs and the Chip-Level TAP. There are three defined IOSF DFx fabric facing interface signals groups and two supporting IP-facing signals buses. Several parameters allow the IP-block to provide the security enabling coverage required by each agent or IP-block. The left side of the block shows the IOSF DFx defined signals are listed in section 7.4. They are labeled as as fdfx_earlyboot_exit, fdfx_secure_policy[3:0] and fdfx_policy_update in the diagram. The policy is broadcasted to all IP-blocks on the secure policy bus and latched with the policy update signal. The policy value is an input to a lookup table that is defined by the policy matrix parameter. A latch is used rather than a flop for two reasons, one is to latch the value and hold the policy constant after the boot window closes when the level of DFx access is known. The second has to do with potential clock attacks on the clock line disrupting the policy value. More information is available in the SoC TAP HAS rev0.90 and the Chassis DFx security framework HAS.

The DFx secure plug-in IP is re-used for the TAP but it doesn't use all of its capabilities. The oem_secure_policy, sb_policy_ovr_value and VISA signals are not used for the TAP security plug-in.

Figure 3-10. TAP DFx security plug-in block diagram



3.17.1 DFx Secure Plug-in Parameters for TAP

Table 3-8 shows the policy plug-in parameters used by the TAPs. The secure width is set to 4 for this revision of spec which is compliant to IOSF DFx rev1.2 and the number of features to secure is fixed to three (3). TAPs will only support a green, orange and red level of security so we assign one DFx feature enable per color code. The policy matrix is configured such that each color code is accessible based on the policy definition. For example, a red opcode or red TAP access (via the Select register) should only be enabled with policy 2 or 4. The policy select register parameter assigns an enumerated color code for each TAP that defines if it accessible under the current policy value. The last parameter is the secure policy opcode which assigns a color code to each opcode within a TAP. This protects test data registers within the TAP. If an orange TAP is active, it may still contain registers that are only accessible with red access. This provides a second level of protection for the SoC so that it doesn't have to duplicate TAPs to segregate sensitive registers.

Table 3-8. DFx secure plug-in IP parameters

Parameter name	Parameter value(s)	Comments
User defined parameters		
STAP_DFX_SECURE_ POLICY_SELECTREG	taptuple(dec), colorcode(enum)	A two dimensional array of values. taptuple: the decimal number that indicate the position in the Select register assign to a particular TAP. enum colorcode = {green, orange, red, orange1, orange2, etc.}
STAP_DFX_SECURE_ POLICY_OPCODE	opcode(hex), colorcode(enum)	A two dimensional array of values. opcode: a hex value per opcode definition. Unused opcode point to the bypass register and are defaulted to green. enum colorcode = {green, orange, red, orange1, orange2, etc.}

Local TAP parameters		
STP_DFX_SECURE_WIDTH	4	This parameter is fixed at 4 for this revision of the spec. (Defined for this SoC HAS revision, the IOSF DFx rev1.2 and in the Chassis mod-dfx Gen3 HAS)
STP_NUM_OF_FEATURES _TO_SECURE	3	Fixed for sTAP use models.
STAP_DFX_SECURE_ POLICY_MATRIX	[DFX_SECURE_WIDTH-1:0][NUM_OF_FEATURES_ TO _SECURE+1:0]	This parameter determines the lookup table necessary to assign the policy with the DFx feature(s) including VISA access. This parameter is fixed for TAPs refer to **Error! Reference source not found.**.
STAP_DFX_EARLYBOOT_F EATURE_ENABLE	[STAP_DFX_NUM_OF_FEAT URES_TO _SECURE+1:0]	This parameter sets the hard coded value for the early debug window for this agent/IP-block. Early boot DFx feature enable for the TAPs must be SECURE_GREEN. STAP_DFX_EARLYBOOT_FEATURE_ ENABLE [4:2] = {3'b001 , STAP_DFX_VISA_BLACK}
USE_SB_OVR	0	The TAP does not use the Sideband override feature.
SECURE_GREEN	0	Assigned local parameter value for enumerating the color codes.
SECURE_ORANGE	1	Assigned local parameter value for enumerating the color codes.
SECURE_RED	2	Assigned local parameter value for enumerating the color codes.
USER1_ORANGE	7	User 1 defined unlocked
USER2_ORANGE	8	User 2 defined unlocked
USER3_ORANGE	9	User 3 defined unlocked
USER4_ORANGE	10	User 4 defined unlocked
USER5_ORANGE	11	User 5 defined unlocked
USER6_ORANGE	12	User 6 defined unlocked
USER7_ORANGE	13	User 7 defined unlocked
USER8_ORANGE	14	User 8 defined unlocked

3.17.2 Policy Matrix for TAP

Table 3-9 list all of the policies and the associated settings for the DFx feature enables for each color code. The table in general should be fixed for the TAPs because it is known set of features, namely, TAP enabling and opcode enabling. Another reason we have a fixed table is because the TAP will only support three values of green, orange, and red. Based on this table the policy matrix is assigned as the following:

1. STAP_DFX_SECURE_ POLICY_MATRIX[6:0]= {00111, 01011, 10011, 00111, 10011, 00111, 00111}
2. STAP_DFX_SECURE_ POLICY_MATRIX[14:7] = {01011}
3. STAP_DFX_SECURE_ POLICY_MATRIX[15] = {00111}

Table 3-9. Policy matrix table for TAPs

Policy Name	DFx Secure Policy Bus Encode	DFx feature en[2] red	DFx feature en[1] orange	DFx feature en[0] green	VISA not used	Description with TAP examples
Security Locked	0000	0	0	1	11	Public/Locked, Green TAP
Functionality Locked	0001	0	0	1	11	Public/Locked, Green TAP
Security Unlocked	0010	1	0	0	11	Intel-Only/ Private/Unlocked, Red TAP
Reserved	0011	0	0	1	11	Public/Locked, Green TAP
Intel Unlocked	0100	1	0	0	11	Intel-Only/Private/Unlocked, Red TAP
OEM Unlocked	0101	0	1	0	11	OEM/Partial Unlock, Orange TAP
Revoked (Reserved)	0110	0	0	1	11	Public/Locked, Green TAP
"User 1" Unlocked	0111	0	1	0	11	OEM/Partial Unlock, Orange TAP
"User 2" Unlocked	1000	0	1	0	11	OEM/Partial Unlock, Orange TAP
"User 3" Unlocked	1001	0	1	0	11	OEM/Partial Unlock, Orange TAP
"User 4" Unlocked	1010	0	1	0	11	OEM/Partial Unlock, Orange TAP
"User 5" Unlocked	1011	0	1	0	11	OEM/Partial Unlock, Orange TAP
"User 6" Unlocked	1100	0	1	0	11	OEM/Partial Unlock, Orange TAP
"User 7" Unlocked	1101	0	1	0	11	OEM/Partial Unlock, Orange TAP
"User 8" Unlocked	1110	0	1	0	11	OEM/Partial Unlock, Orange TAP

Policy Name	DFx Secure Policy Bus Encode	DFx feature en[2] red	DFx feature en[1] orange	DFx feature en[0] green	VISA not used	Description with TAP examples
Part Disabled	1111	0	0	1	11	Public/Locked, Green TAP

3.17.3 Security applied to TAPs in the Select register

Figure 3-11 shows an example of applying the secure policy to the SELECT data register. This allows the SoC integration team to manage access to groups of TAPs on the network. We use the same set of DFx feature enables as described previously. When dfxsecure_feature_en[0] is logic 1 then the 2-bit Select register values will enable access to those TAPs that are designated as green security level is allowed to pass through the mux to control the TAP on the network as needed (meaning Normal, Exclusive, Isolated, or Shadow mode). If this TAP is not allowed to be accessed based on the security policy then output bits are forced to "00" which makes the TAP isolated from the network.

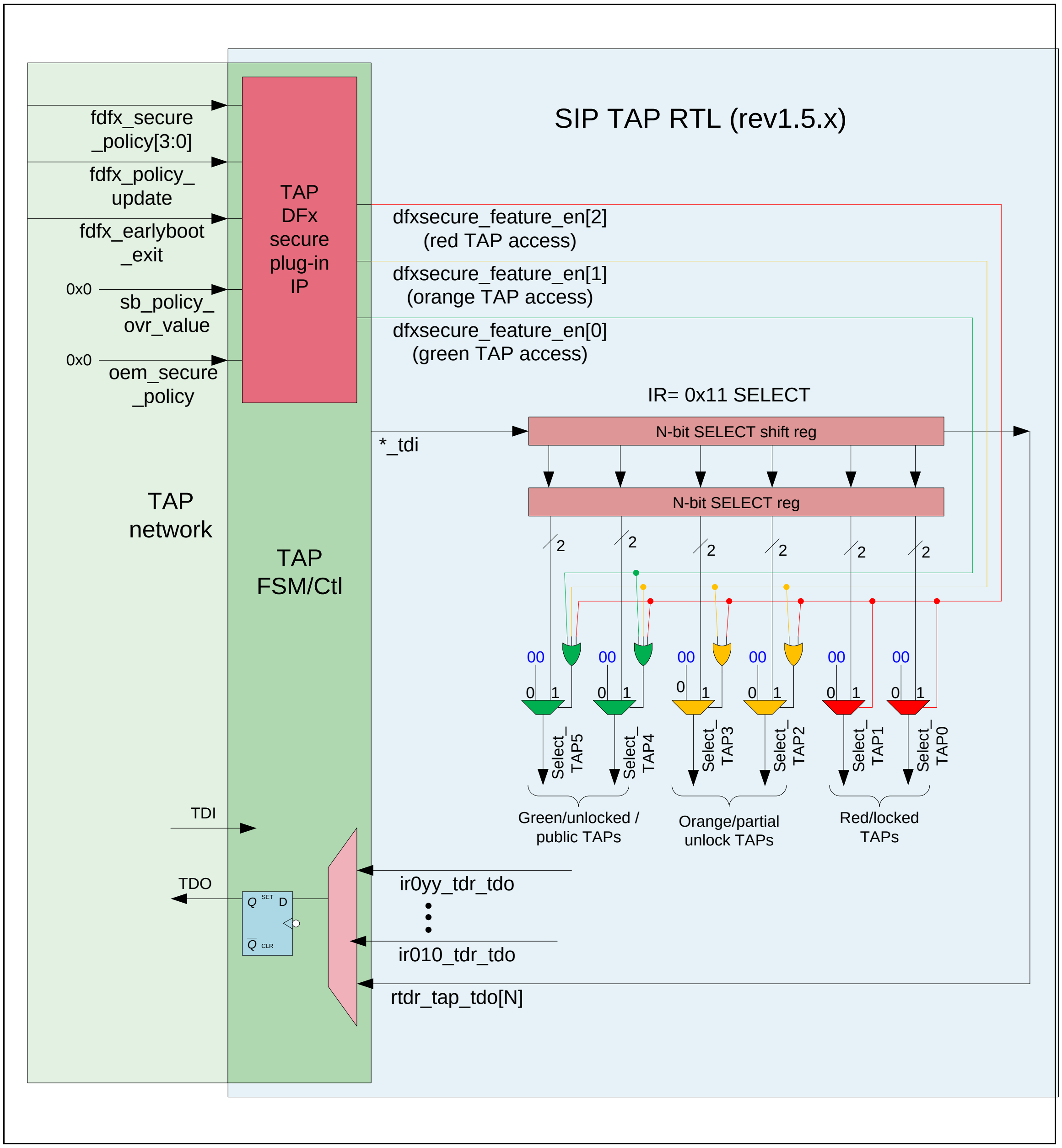
If the dfxsecure_feature_en[1] signal is logic 1 then those assigned TAPs with an orange/partial unlocked condition can be accessed. Any other groups of TAPs that are designated as green are also accessible.

If the dfxsecure_feature_en[2] signal is logic 1 then assigned TAPs with red/locked security level is be accessible. Any other groups of TAPs that are designated as green or orange are also accessible. This diagram is for illustration purposes only and the actual implementation may differ.

We use the six TAPs shown in Figure 3-11 and define the select register policy. Each TAP in the Select register requires two bits. A decimal value represents the enumerated values are green = 0, orange = 1 and red = 2. The value in the parameter is {taptuple, colorcode}.

```
STAP_DFX_SECURE_ POLICY_SELECTREG = {
0, SECURE_RED,
1, SECURE_RED,
2, SECURE_ORANGE,
3, SECURE_ORANGE,
4, SECURE_GREEN,
5, SECURE_GREEN }
```

Figure 3-11. Application of DFx secure policy to SELECT register



3.17.4 Security applied to opcodes example

Figure 3-12 is an example of how to apply the DFx secure policy is applied to IR opcodes to protect access to the test data registers (TDRs). They can be either internal or a remote TDR. Since security is applied to the IR opcode decoder itself it can manage access to any TDR. This diagram is for illustration only and may not reflect the actual implementation. The basic idea is that we assign one DFx feature enable per color (level of security access). When the green level is active the dfxsecure_feature_ en[0] is logic 1 and any other TDR with an security level of orange or red the instruction decoder will point to the bypass register and no access is permitted. Each sTAP will have a parameter array to define which opcode is applied with the desired security level.

An orange or partially unlocked set of opcodes allow access to TDRs that have been assigned the dfxsecure_feature_en[1]. This means both the green and the orange security levels can access these TDRs. If a TDR is designated as red or locked then that opcode will decode to the bypass instruction.

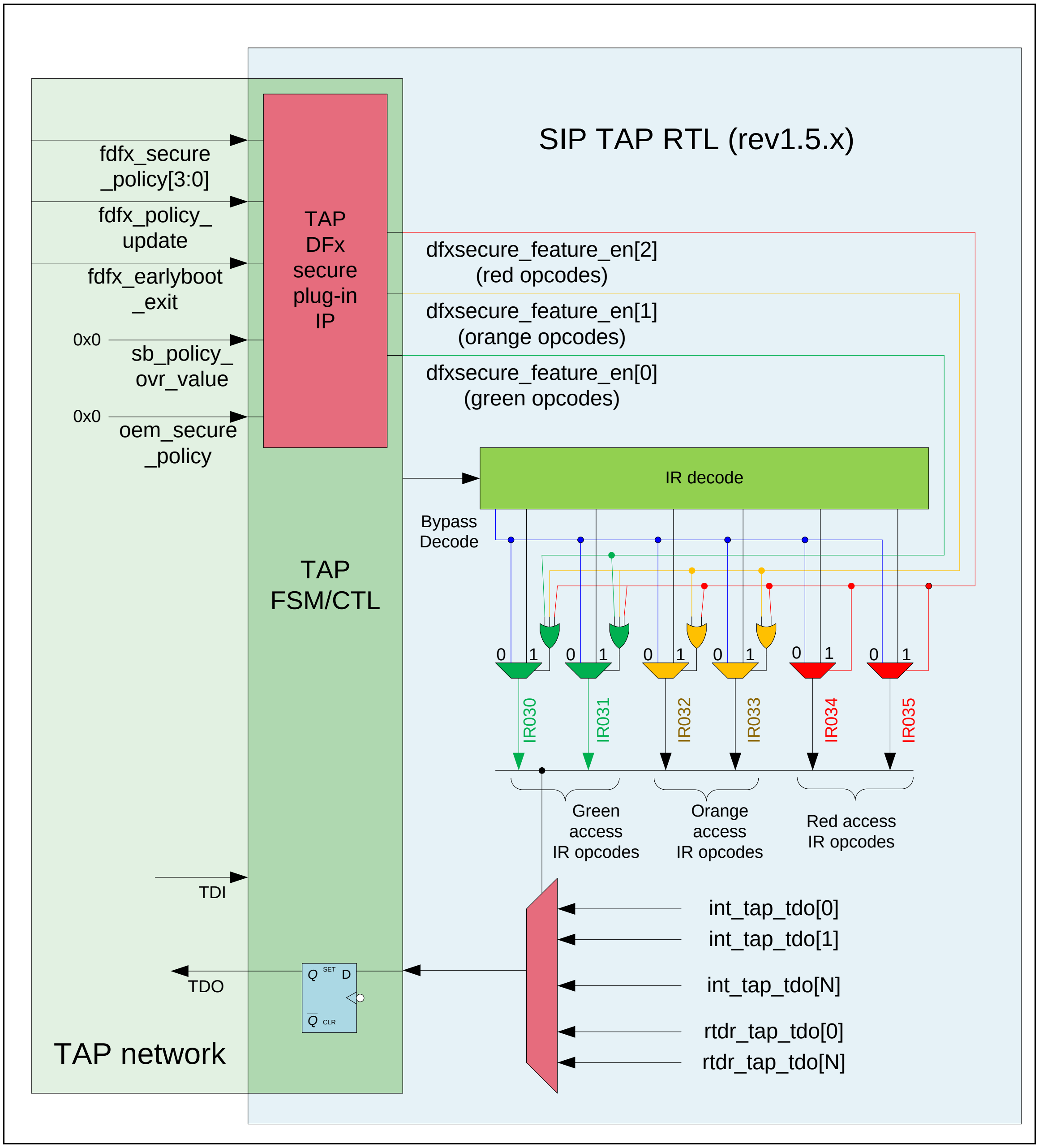
A red or locked set of opcodes will only allow access to TDRs when the dfxsecure_feature_en[2] is logic 1 enabled based on the policy value. The diagram simply decodes the opcodes and uses a mux to select between the decoder values or the bypass value. The security levels control the mux selections between the two. The actual implementation is expected merged within the instruction decoder.

There are decodings on the secure policy bus where green is not available either. This would be considered black, however, it may be possible to have a TAP network access that is green but a particular TAP's opcode based on the security level encodings are black. To prevent network corruption any slave TAP must provide access to the slave ID code and the bypass opcode regardless of the security level. Meaning the SLVIDCODE and BYPASS are available in black (disabled), green, (unlocked/public), orange (partially unlocked) or red (locked, private Intel only) security levels.

In example, we assume that this is one of the TAPs from the previous example, it doesn't matter which one so we pick orange TAP2. The secure policy opcode parameter is set to:

```
STAP_DFX_SECURE_ POLICY_OPCODE = {
0x30, SECURE_GREEN,
0x31, SECURE_GREEN,
0x32, SECURE_ORANGE,
0x33, SECURE_ORANGE,
0x34, SECURE_RED,
0x35, SECURE_RED
}
```

Figure 3-12. Application of DFx secure policy to IR Opcodes



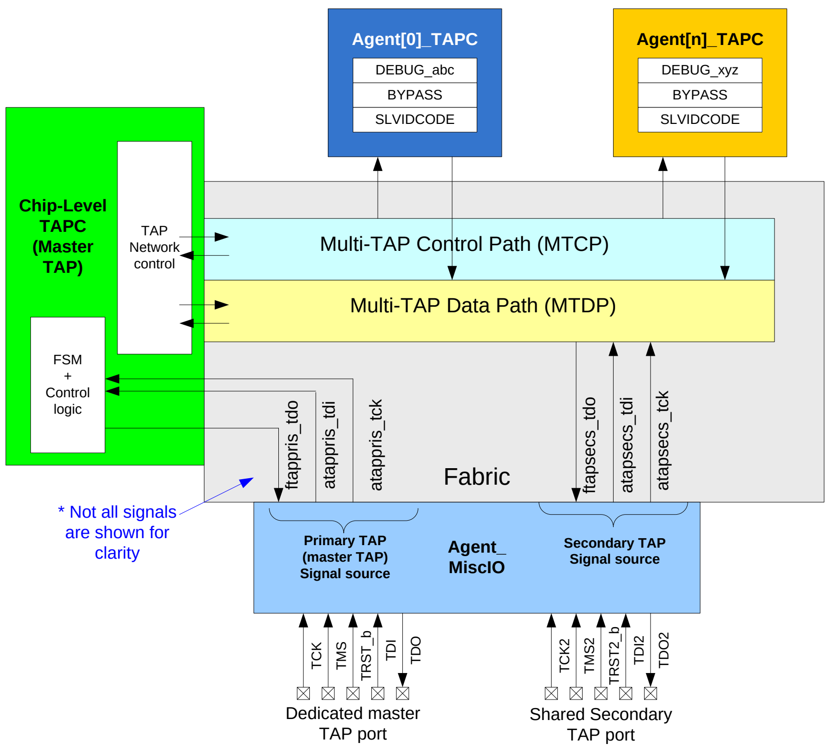
3.18 Other TAP Support Signals

This section describes other TAP signals on the interface that assist the agent's controller.

3.18.1 Agent-Sourced TAP Interface Signals for TAP Network

A secondary TAP port on the SoC package has been discussed earlier in this specification. It has two use models: one is for improving HVM test time through parallel test delivery and the other for portability of IA-core test content. Figure 3-13 shows an IO hard-IP agent with the TAP source signals connecting to the CLTAP for distribution throughout the TAP network. The signal names are defined on the IOSF DFx interface to maintain consistency across SoC divisions. A fixed signal name enables auto-generation and connection to the CLTAP and its associated TAP.7 network.

Figure 3-13. Primary/Secondary TAP source interfaces

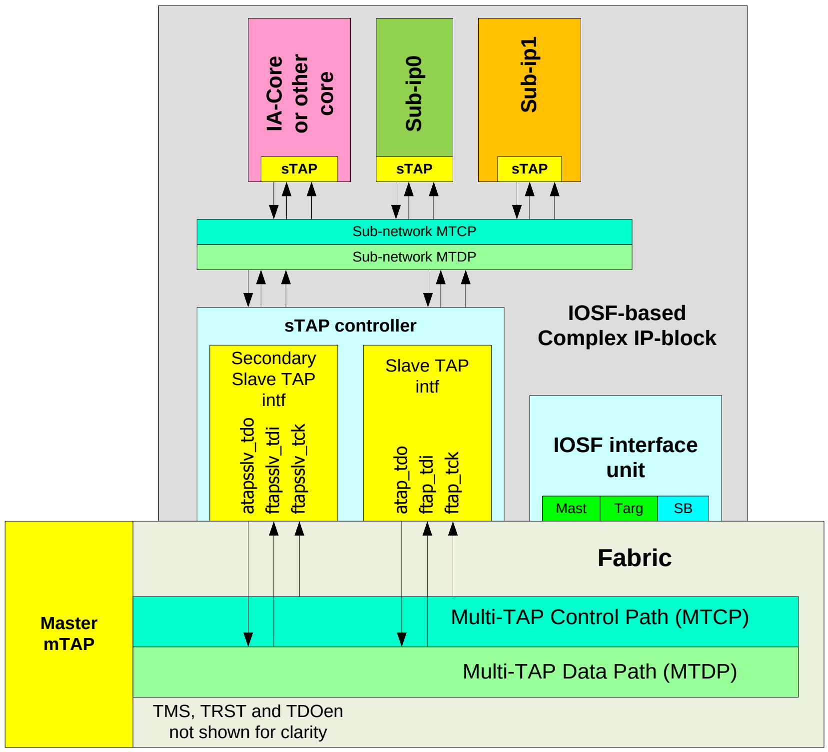


3.18.2 Secondary Slave TAP Interface

There are conditions where a single slave TAP interface on an IOSF agent is not sufficient. A secondary slave TAP could provide parallel HVM test content to an agent, or it may be used as a tertiary TAP port where routing from a local set of miscellaneous IO pins is advantageous to the floorplan in a purely hierarchical topology.

For illustration purposes we assume a complex agent, such as the one shown in Figure 3-14, it is connected to the IOSF fabric to provide capabilities associated with an accelerator services unit. Since it contains an IA-core a more effective test strategy to apply parallel test content to the core while other parts of this agent or the SoC to be tested.

Figure 3-14. Secondary slave TAP interface example



3.18.3 TDO Enable

A TDO enable signal is required with all TAPs. Its purpose is to drive a CMOS pad to to be compliant to the 1149.1 specification. According to the spec, the TDO package pin is driven only when shifting the DR or IR registers, otherwise it is tri-stated. It is possible that a secondary TAP port is required and the only available package pin is to reuse an existing functional general purpose CMOS IO (GPIO), which would require the use of the TDOen signal. This may apply to the CLTAP TAP as well, however. in most cases, it will be assigned an open-drain driver allowing it to be compliant without the signal.

3.19 TAP Signal Interface Description

A graphical view of the TAP interface signals are shown in Figure 3-15 and Figure 3-16. This diagram summarizes the available TAP signals that support the usage models described previously. Table 3-10 presents a complete list of signal names. The integration team has the freedom to assign any reasonable number of characters as a prefix or suffix as applied to the root name to identify the signal in a full chip SoC.

Figure 3-15. TAP signal interface graphical view 1

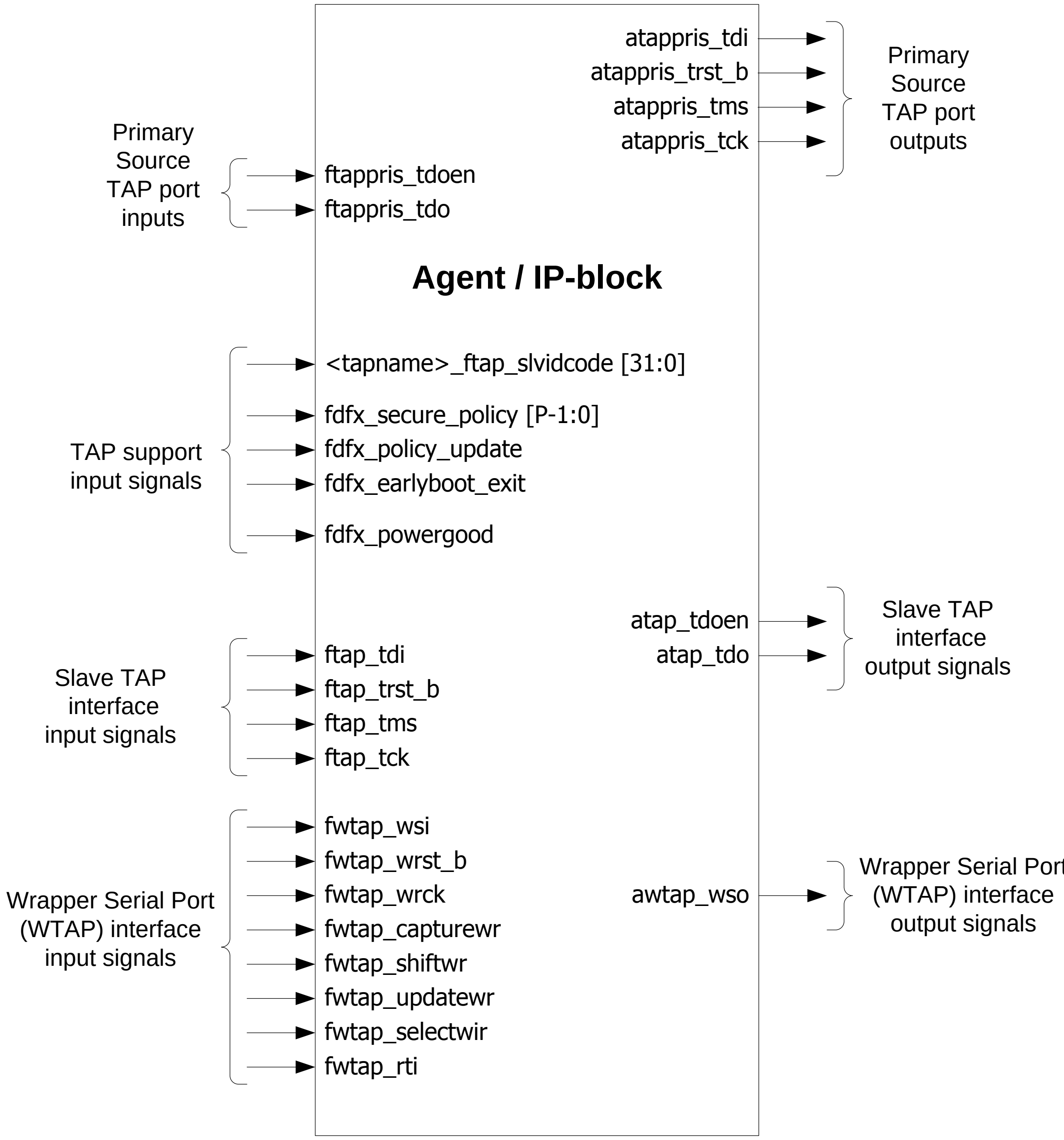


Figure 3-16. TAP signal interface graphical view 2

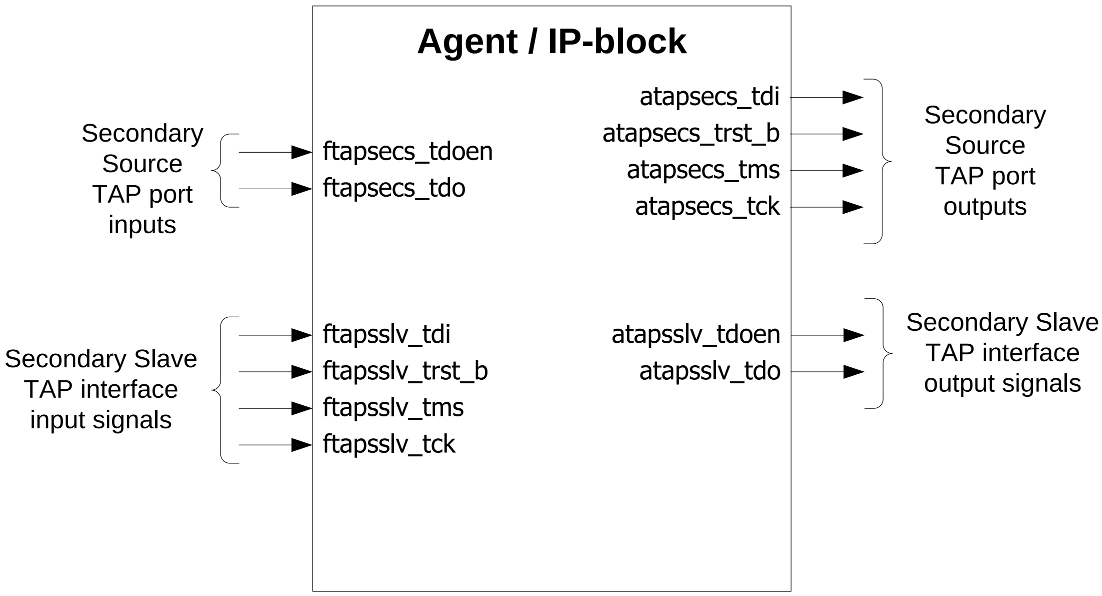


Table 3-10. TAP interface signal description

Signal	I/O	R/O/C[1]	Description
Slave TAP signals			
ftap_tck[2]	I	C	**Fabric TAP clock:** This is the TAP clock from the fabric that originates from package pins connected to the TAP network.
ftap_tms	I	C	**Fabric TAP test mode:** This is the TAP finite state machine test mode control signal from the fabric that originates from package pins connected to the TAP network.
ftap_trst_b	I	C	**Fabric TAP reset bar:** This is the TAP reset from the fabric that may originate from a package pin connecting to the master TAP controller and the TAP network.
ftap_tdi	I	C	**Fabric Test Data In:** This is a test data in (TDI) from the fabric that originates from the master TAP controller through the TAP network.
atap_tdo	O	C	**Agent Test Data Out:** This is the test data out (TDO) from this agent.
atap_tdoen	O	C	**Agent Test Data Out Enable:** This is the test data out enable to drive the tristate enable on the TDO pad control. atap_tdoen = Shift-DR OR Shift_IR

Signal	I/O	R/O/C[1]	Description
Agent TAP support signals			
<tapname>_ftap_slvidcode[31:0]	I	R	**Fabric Slave ID Code:** This is a signal bus port provides the slave ID code for the sTAP. Note1: This signal bus is required if a TAP interface is required. Note2: Use of the tapname is optional. It is useful for agents/IP-blocks with more than one TAP within the module to uniquely identify the codes. The SIP TAP RTL IP-block will not provide a prefix.
fdfx_powergood	I	R	Refer to section 6.2. Note: This signal is equivalent to **dfx_powergood_rst_b**.
fdfx_secure_policy [DFXSECURE_WIDTH-1:0]	I	R	**Fabric DFx security policy.** This bus is a binary encoded value of the security policy that is the current state of the SoC. The parameter value is constant for a given SoC generation and aligned with a process node. All IP-blocks must have the same width. For this revision of the IOSF DFx HAS: DFXSECURE_WIDTH = 4 Note: An agent/IP-block may have two DFx secure policy plug-in blocks one for sTAP and one for the agent/IP-block. It is the IP-block's responsibility to connect both together. Note2:
fdfx_policy_update	I	R	**Fabric DFx policy update.** This signal is the latch enable to capture the policy value to prevent glitches. 0: Latch values 1: Update to new policy value Note: An agent/IP-block may have two DFx secure policy plug-in blocks one for sTAP and one for the agent/IP-block. It is the IP-block's responsibility to connect both together.
fdfx_earlyboot_exit	I	R	**Fabric DFx early boot exit.** This signal indicates when the early boot debug window is closed. 0: Debug capabilities are available during this phase of the boot flow 1: DFx security policy must be used Note: An agent/IP-block may have two DFx secure policy plug-in blocks one for sTAP and one for the agent/IP-block. It is the IP-block's responsibility to connect both together.
WTAP signals			
awtap_wso	O	O	**Agent WTAP Serial port Output:** This is the Wrapper Serial Port (WTAP) data output signal.
fwtap_wsi	I	O	**Fabric WTAP Serial Port Input:** This is the Wrapper Serial Port (WTAP) data input signal.
fwtap_wrst_b	I	O	**Fabric WTAP Reset Bar:** This is the Wrapper Serial Port (WTAP) reset input signal. This signal is active low.
fwtap_wrck[2]	I	O	**Fabric WSP Clock:** This is the Wrapper Serial Port (WTAP) clock input signal.

Signal	I/O	R/O/C[1]	Description
fwtap_capturewr	I	O	**Fabric WTAP Capture Wrapper Register control:** This is the Wrapper Serial Port (WTAP) capture control signal to enable capturing input data from WIR or test data registers.
fwtap_shiftwr	I	O	**Fabric WTAP Shift Wrapper Register control:** This is the Wrapper Serial Port (WTAP) shift control signal to enable shifting of the WIR and test data registers.
fwtap_updatewr	I	O	**Fabric WTAP Update Wrapper Register control:** This is the Wrapper Serial Port (WTAP) update control signal to enable updating a shadow WIR or test register from the contents of the shift register.
fwtap_selectwir	I	O	**Fabric WTAP Select Wrapper Instruction Register:** This is the Wrapper Serial Port (WTAP) select control signal to select either the IR register or a test data register the actions by the capture, shift, and update control signals. 0: Any decoded data register will be active in the DR-Scan branch from the controlling sTAP/mTAP FSM that is driving this WTAP's control signals. 1: The WTAP's instruction register will be active in the IR-Scan branch from the controlling sTAP/mTAP FSM that is driving this WTAP's control signals.
fwtap_rti	I	O	**Fabric WTAP Run-Test/Idle:** This is the Wrapper Serial Port (WTAP) control signal to indicate that the driving TAP state machine is in the Run-Test/Idle state. One possible use model is to execute an operation after the test data register was updated, for example, start a BIST engine after all the registers are updated.
Primary source for TAP signals			
atappris_tck[2]	O	O	**Agent Primary Source TAP TCK:** This is the signal source of Test Clock (TCK) from this agent to the CLTAP. This signal is available for those situations where the CLTAP package pins are located in this agent to control the TAP network.
atappris_tms	O	O	**Agent Primary Source TAP TMS:** This is the signal source of Test Mode Select from this agent to the CLTAP. Refer to atappris_tck for more description of the primary source TAP interface.
atappris_trst_b	O	O	**Agent Primary Source TAP TRST_b:** This is the signal source of Test Reset bar from this agent to the CLTAP. Refer to atappris_tck for more description of the primary source TAP interface.
atappris_tdi	O	O	**Agent Primary Source TAP TDI:** This is the signal source of Test Data In from this agent to the CLTAP. Refer to atappris_tck for more description of the primary source TAP interface.
ftappris_tdo	I	O	**Fabric Primary Source TAP TDO:** This is the primary signal sink of Test Data Out from the CLTAP to this agent where the master TAP package pins are located. Refer to atappris_tck for more description of the primary source TAP interface.

Signal	I/O	R/O/C[1]	Description
ftappris_tdoen	I	O	**Fabric Primary Source TAP TDOen:** This is the primary signal sink of TDO enable from the CLTAP to this agent to control the tri-state enable of the physical layer that drive the master TAP TDO. Refer to atappris_tck for more description of the primary source TAP interface.
Secondary source for TAP signals			
atapsecs_tck[2]	O	O	**Agent Secondary Source TAP TCK:** This is the secondary signal source of Test Clock from this agent to the TAP network. This agent is supplying an additional TAP for reusing test vectors to the core or for increased HVM throughput from the tester.
atapsecs_tms	O	O	**Agent Secondary Source TAP TMS:** This is the secondary signal source of Test Mode Select from this agent to the TAP network. Refer to atapsecs_tck for more description of the primary source TAP interface.
atapsecs_trst_b	O	O	**Agent Secondary Source TAP TRST_b:** This is the secondary signal source of Test Reset bar from this agent to the TAP network. Refer to atapsecs_tck for more description of the primary source TAP interface.
atapsecs_tdi	O	O	**Agent Secondary Source TAP TDI:** This is the secondary signal source of Test Data In from this agent to the TAP network. Refer to atapsecs_tck for more description of the primary source TAP interface.
ftapsecs_tdo	I	O	**Fabric Secondary Source TAP TDO:** This is the secondary signal sink of Test Data Out from the fabric to this agent where the secondary TAP package pins are located.
ftapsecs_tdoen	I	O	**Fabric Secondary Source TAP TDOen:** This is the secondary signal sink of TDO Enable from the fabric to this agent to control the tri-state enable of the physical layer that drives the secondary TAP TDO.
Secondary slave TAP signals			
ftapsslv_tck[2]	I	C	**Fabric Secondary Slave TAP clock:** This is the TAP clock from the fabric that originates from package pins connected to the master TAP controller elsewhere in the component. Note: The secondary slave TAP port is required for the slave TAP version that is available in the Intel Reuse Repository. It is optional for a hard-IP TAP. Use of the secondary slave TAP port is SoC dependent. One use model supports a tertiary TAP port to an assigned set of GPIO pins.
ftapsslv_tms	I	C	**Fabric Secondary Slave TAP test mode:** This is the TAP finite state machine test mode control signal from the fabric that originates from package pins connected to the master TAP controller elsewhere in the component. Note: Refer to the explanation note in the ftapslv_tck signal description.

Signal	I/O	R/O/C[1]	Description
ftapsslv_trst_b	I	C	**Fabric Secondary Slave TAP reset bar:** This is the TAP reset from the fabric that may originate from a package pin connecting to the master TAP controller and the TAP network. Note: Refer to the explanation note in the ftapslv_tck signal description.
ftapsslv_tdi	I	C	**Fabric Secondary Slave Test Data In (other TDI):** This is a test data in (TDI) from the master TAP controller elsewhere in the component. Note: Refer to the explanation note in the ftapslv_tck signal description.
atapsslv_tdo	O	C	**Agent Secondary Slave Test Data Out:** This is the test data out (TDO) from this agent. Note: Refer to the explanation note in the ftapslv_tck signal description.
atapsslv_tdoen	O	C	**Agent Test Data Out Enable:** This is the test data out enable to drive the tristate enable on the TDO pad control. Note: Refer to the explanation note in the ftapslv_tck signal description. Also, this signal is defined as: atapslv_tdoen = Shift-DR OR Shift-IR.
NOTE: [1]Note1: R = required, O = optional, C = conditional. Conditional means that if the feature is used then the signals labeled as conditional become required. [2]Note 2: The TCK clock signals are expected to be all synchronous and in phase with respect to the driving TCK source. The source may be either the primary or secondary port. Note 3: The symbol M is the number of agent ports on the package when the extended platform TAP port is supported (M = MTAP_EXI_NUM_OF_TAP_ AGENTS_ON_PLATFORM) Note3: The slave TAP IP-block from IRR is required to support the secondary slave interface and it is optional for a hard IP-block. It is SoC dependent on how this interface will be used. A secondary slave TAP supports a hierarchical-hybrid topology or a local tertiary TAP port.			

3.20 Transaction Cycle/Data Flows

Not applicable.

3.21 Ordering/Coherency Rules

Not applicable.

3.22 Performance/Bandwidth Analysis

Not applicable.

3.23 Exception List Requirements

Not applicable.

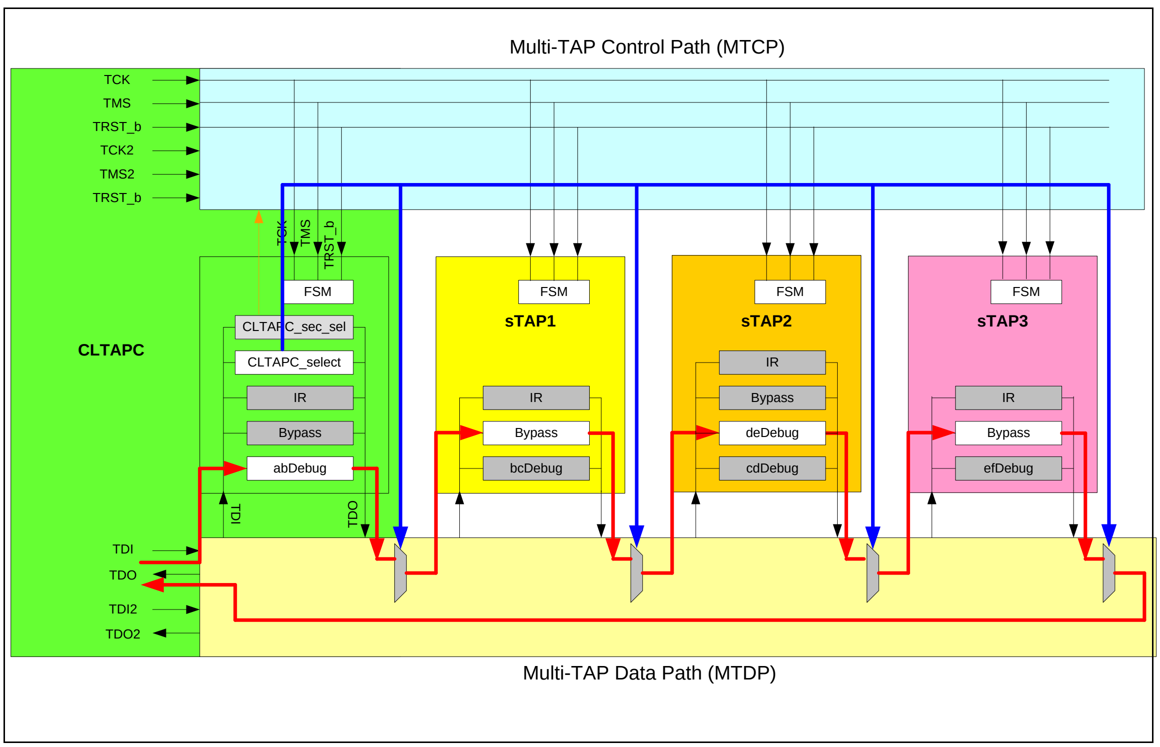
3.24 Programming Model

A simple example is presented here but the reader should refer to the SoC TAP HAS rev0.90.x (or later) for a more complete use model description.

Normal mode is the same use model as the 1149.1 spec has done for many years. Meaning that if N number of TAP were connected in series (historically this would be a collection of components on a motherboard) then all of the IR registers are shifted together to configure which TAP data registers are expected to be accessed. Any TAP not being accessed was placed in a Bypass mode. This required each TAP to be active in the serial chain (unless some form of logic and/or electrical bypass was incorporated). The same technique is employed with a TAP.7 network operating in normal mode.

Figure 3-17 shows the DR shift portion of the use model. In this use model all agents (Agents[3:1]) are in series with the CLTAP. The CLTAPC_SELECT register configures the network to enables all the TAPs to be made normal. All of the instruction registers must be accessed and configured to select a data register or its bypass register. The CLTAP's abDebug register is the target for access and the other TAP must be placed in Bypass. This usage model assumes the agent of interest is not power gated.

Figure 3-17. TAP.7 normal mode use model block diagram



Register assumptions:

CLTAPC_SELECT:

- Bits[5:4]: Agent[3]_TAPC
  - Bit field definition is the same as Bits[1:0]
- Bits[3:2]: Agent[2]_TAPC
  - Bit field definition is the same as Bits[1:0]
- Bits[1:0]: Agent[1]_TAPC

- 00: Isolated
- 01: Normal
- 10: Excluded
- 11: Shadow mode

IR assumptions:

- CLTAPC:
  - abDebug IR = abDebugIR (this value is not hardcoded but should be in a header file)
  - abDebug = 0x25 // desired TDR data to be written
- All TAPs:
  - bp = bypass (0xFF, assuming an 8-bit IR register length)

IR format: TAP.IR instruction mnemonic.(optional hex code)

DR format: {binary value (if in bypass), (comma) tap name.DR register name.(optional hex value to be written), (comma) other values}

The following list enumerates the steps for this use model.

1. Assert TRST_b.
2. Execute required SoC initialization.
3. Write the CLTAP select control register to remove all slave TAPs except the one of interest.
   a. IR = cltapc.cltapc_select
   b. DR = cltapc.cltapc_select.0x15   //agent[3:1]_tapc=normal
4. The TAP network and desired agent are now configured to accept read and write operations to its test registers.
   a. Access cltapc abDebug register
      i. IR shift = cltapc.abDebugIR, agent1_tapc.bp, agent2_tapc.bp, agent3_tapc.bp,
      ii. DR shift = {cltapc,abDebug, 1, 1, 1}

3.25 Power Management Capabilities

Refer to the Chassis DFx HAS for information about the TAP and TAP network for low power debug with the TAPs.

3.26 Security Feature Requirements

The slave TAPs are required to implement the DFx secure policy plug-in IP-block. This allows the TAP and TAP network provide security color coded access levels to each TAP on the network and each opcode. Refer to section 3.17.

3.27 DFx Requirements

3.27.1 DFV/DFD Requirements

There are no specific DFV requirements for TAP. For post-silicon debug there are a few features that help in identifying which TAP is active on the network:

1. The Slave ID codes form a linked list to identify each child TAP and their associated parent TAP.
2. The IR shift register captures 0x01 which allows the host TAP controller to parse the logic 1s to determine an IR length and potentially an abnormal behaving TAP.

3. A host controller can enable one TAP at time and read each SLVIDCODE. The TAP host can progress through each branch of the network tree and identify all the TAPs and any potential SLVIDCODE mismatched.

3.27.2 DFT Requirements

The CLTAP, slave TAPs and the TAP network cannot be scanned. They are used to enable the DFT features and would be default of their use validate that the TAP and TAP network function properly. Generally, a functional test that dumps all of the Slave ID codes will exercise most of the logic within the FSM and connectivity between the TAP and TAP network.

Rule 1.f (from section 3.15.): All TCK driven flops must not be part of the scan chain for HVM testing.

3.27.2.1 Burn-in Specific Requirements

The CLTAP, slave TAP and TAP network provides the serial access control for burn-in of the SoC.

3.27.3 DFM Requirements

None specifically for TAP.

§

4 HVM Scan Interface

This chapter focuses on the scan interface for testing the logic in an agent (IP-block).

Table 4-1. Chapter revision history

Revision Number	Description	Revision Date
rev1.2_rc1	• Initial release.	Jan 2012
rev1.2_rc2	• Updated scan interface signals • Updated the DFx security group of signals to finalized architecture	Feb 1, 2012
rev1.2_rc3	• Added new opcodes for IEEE1149.1_2012 and IEEE1149.8.1 for future use. • The user defined opcodes now start at 0x30.	Feb 13, 2012
rev1.2_rc4b	• All edits were accepted, all strikeout text was deleted. • Move content into new SoC HAS template • Updated scan interface signals to be organized into the following buckets: all signals, signals for IP, signals for SCC, SASC, SRC. • Removed the scan source signals. This is an obsolete use model, generally, the scan control signals originate from the Narrow Test Interface (NTI).	March 10, 2012
rev1.2_rc5	• Removed the fscan_edtclk and *_edtupdate from the SCC diagram and scan signal list.	March 11, 2012
rev1.2_rc6	• Fixed the heading numbers. Moved them up a level • Added the function test support • Remove the SCC, SRC, and SASC signals from the IOSF DFx HAS. The SCC is no longer required for hard-IP blocks and the DFx fabric will contain these scan components.	May, 2012
rev1.2_rc7b	• Removed fscan_cdi/fscan_cdo because these only apply to the scan controllers (SCC/SRC/SASC). • Removed old text and re-arranged the content within the scan section to highlight the requirements, rules and permissions. • Updated the scan controls for the arrays. — The fscan_ram_awt_wen and fscan_ram_bysel is based on the number of writeable arrays (ROMs not included) —	July, 2012
rev1.2 (final)	• Added back the parameter for the number of arrays for the *ram_bypsel. • Updated the generalized view of the scan control system	July 20, 2012

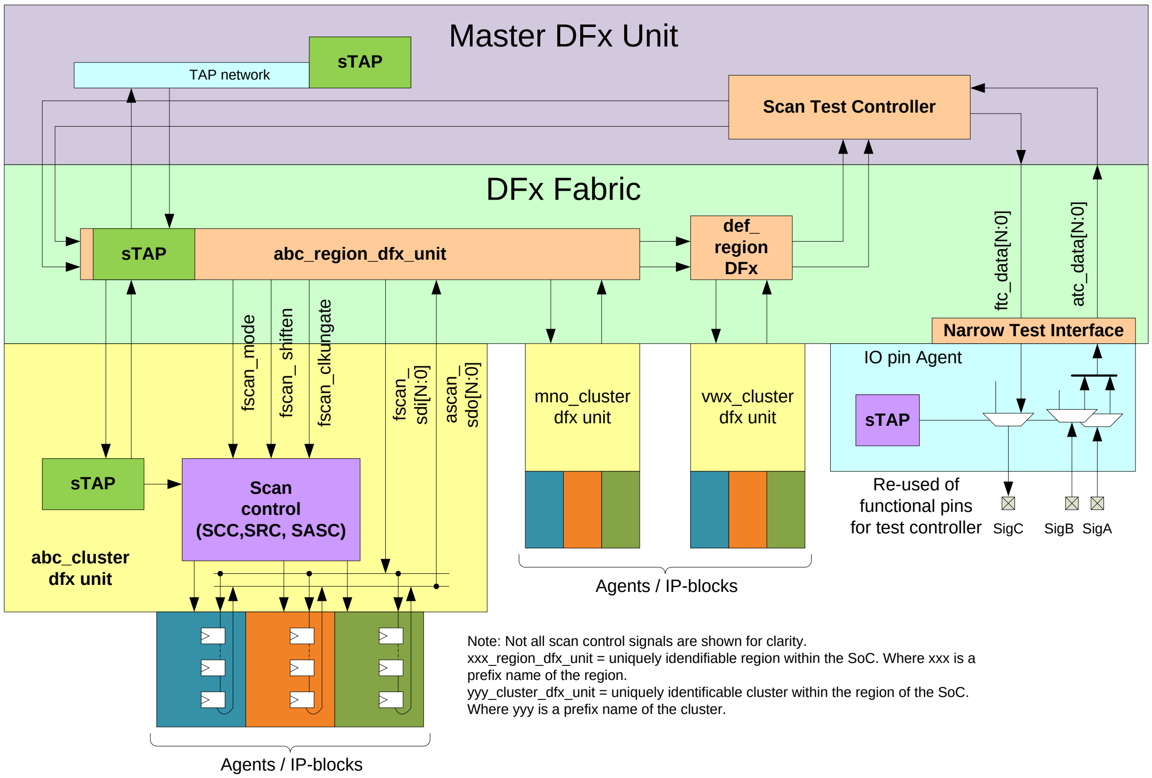
Scan-based testing is the best-known method for determining gross defects in a majority of digital logic devices for High Volume Manufacturing (HVM). Scan is complemented with other types of testing as part of an overall test methodology that targets sections of the SoC component with specific content for improved test coverage. For example, RAM-based (transistor cell based) register files or arrays, such as queues and FIFOs, use algorithmic tests from a BIST engine that provide better coverage than what a stuck-at fault model could do. The stuck-at fault model alone is insufficient for modern SoC designs due to the fault models associated with extremely small nanometer sized transistor and wire feature sets. Therefore, a more comprehensive at-speed test paradigm is required. Coverage gaps between any of these methodologies, it is usually expected that a functional test is applied to the agents. However, there is an effort to reduce or eliminate functional testing altogether.

Scan-based testing has been improved over the years to include other fault models such as at-speed, transition, and bridging faults. Some of these methodologies require additional logic either in the fabric or partition to support it. The stitching of flip-flops that form a scan chain are inserted after synthesis in the place and route flow, and therefore does not natively exist in the RTL code. However, the agent must be designed ahead of time to be scan-friendly which may require additional logic or signal interface pins. Because of this, both soft-IP and hard-IP agents will require a different set of scan interface signals depending on the types of logic structures within the IP. For this reason, a defined scan interface as a superset of all the possible conditions in which scan may be used on an agent must be included in this specification. There are very few required scan signals with others that are conditional in the general sense to cover the rest.

The scan interface section is divided into subsections that match the signal list table. The first section describes the active scan control signals that are usually driven by the tester and distributed throughout the SoC. This distribution structure is implementation dependent but the resulting control signals are presented to the partition or hard-IP agent through a test wrapper. The next two sections, Agents with Internally Derived Clocks and Segregated Scan Control Chains, describe the connections of the inputs and outputs of the chains to the scan distribution network. The fourth section is the Asynchronous Control Signals that remain static during the course of a scan test segment. For each segment, they may change due to different testing conditions. These signals may be driven from a TAP test data register for the hard-IP agents, or they may appear as part of a soft-IP agent that is driven by the partition scan control logic. The assignment of the functional pins is SoC dependent and these are usually the DDR IOs in a bypass mode (if available) and/or other miscellaneous general purpose IOs.

A generalized scan implementation is shown in Figure 4-1. The purpose of this diagram is to show how the signals are used together to form an overall methodology from the tester to a test controller block and then to each agent or partition. This topology is for illustration purposes only and not intended to provide specific architectural requirements; however, it is based on the Chassis concept of hierarchical layers of modular DFx units. Not all of the signals are shown to simplify the diagram. The test controller's test data inputs and outputs are sinked and sourced on the ftc_data and atc_data buses defined in the miscellaneous test bus section. This test controller bus is connected to SoC specific set of general purpose IOs or other functional pins that can accept test content from the tester. The test controller re-distributes the content back to the agents or partitions. In this example, a region DFx unit forms a daisy chain to deliver the scan content among the regional units. Another daisy chain is formed within the cluster DFx unit to delivery scan content to the IP-blocks. The scan control TAP will drive all of the scan controllers and any other agent specific scan attributes that may exist for this block. A Scan Clock Controller (SCC), a Scan Reset Controller (SRC) and a Scan Asynchronous Scan Controller (SASC) control the scan signals on the agent or IP-block's interface to manage scan test operations. It is not the intention of this specification to define the scan methodology in sufficient detail for integration teams to implement. Section 4.16 lists the references to obtain more information about scan based testing.

Figure 4-1. Generalized view of scan implementation



A brief description of a typical scan application is shown in **Error! Reference source not found.** and **Error! Reference source not found.**. The only signals in this diagram that are seen by agent/IP-block are the functional clock and the shiften. The other signals are being manipulated by the Scan Clock Controller but it is important to display them together to illustrate the overall use model. Scan control signals are driven from a cluster DFx unit with either static controls (from a TAP register) or actively as the test content is being applied. The active scan components are the Scan Clock Controller (SCC), the Scan Reset Controller (SRC) and the Scan Asynchronous Scan Control (SASC). **Error! Reference source not found.** is a zero depth pattern and **Error! Reference source not found.** is a programmable sequential depth of two but it can deeper depending on the complexity of the logic being tested.

Figure 4-2. Scan example of a zero depth pattern

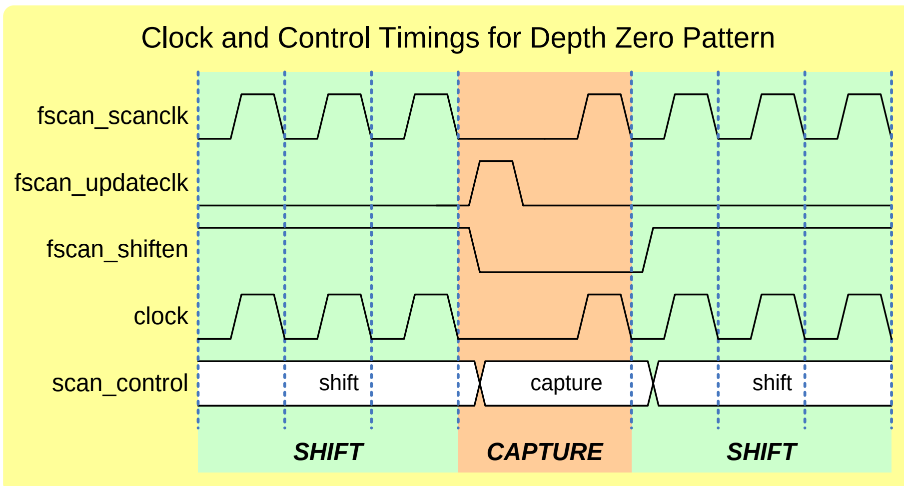
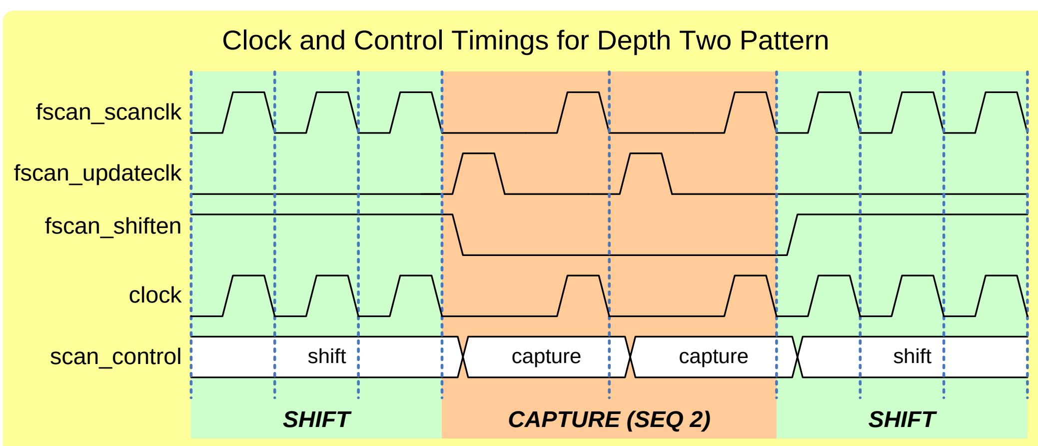


Figure 4-3. Scan example of a two capture depth pattern



4.1 Active Scan Control Signals

Scan signals described in this section:

- ascan_preclk_<clockname>
- fscan_postclk_< clockname >

If an IP-block generates an internal clock that is derived off the functional clock input then the IP-block must place the pre-clock signal (before the divider) and post- clock signal (after the divider) (fscan_postclk_<clockname> and ascan_preclk<clockname>) at the top level of the IP as part of the scan control override signals. If an IP-block has embedded clock then the IP-block must implement both sets of scan controls, meaning both ascan_preclk* and fscan_postclk* must be present on the interface. It is considered an agent bug if only one of the two signals are present. Most agents' functional clock or clocks are available at the interface so that a scan clock controller may directly manipulate them for at-speed testing. To prevent coverage loss for this scenario, a provision must be made for bringing the internal clock out of the agent so that it can be manipulated by the scan control logic and sent back to drive the functional logic. The internal derived clocked is made available by connecting it to ascan_preclk<clockname> with the specific clock's name appended to the signal to uniquely identify it from other potential derived clocks. The manipulated post clock signal (fscan_postclk*) is routed back to the agent where it is then connected to all of the agent's clock destination modules. Figure 4-4 shows an abstract agent with an internal derived clock controller and the minimum interface signals needed to illustrate the use model. The pre/post clock signals are presented to fabric where a few options exist to manipulate the functional clock for at-speed testing. These are labeled as SCC, bypass, and other for an implementation-specific scan control logic block. It is expected that the DFx fabric will use the the Scan Clock Controller (SCC) as defined in the SoC Scan Requirements Handbook rev0.70 or later.

Figure 4-4. Pre/Post clock use models for scan control

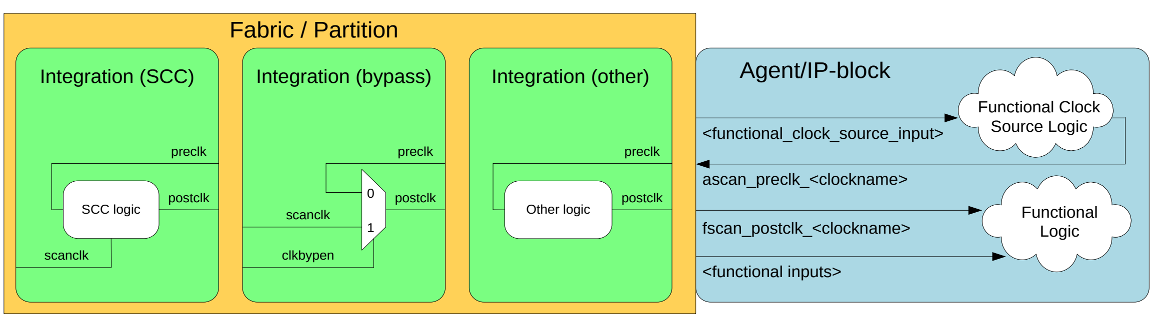
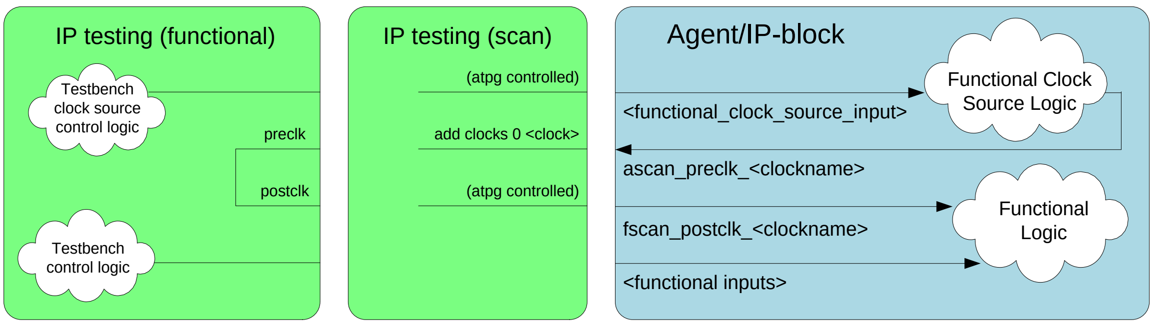


Figure 4-5 shows options for verification testing on these signals at the partition or top-level of the SoC.

Figure 4-5. Verification use models for internally derived clocks



4.2 Scan Data Chains

Scan signals described in this section:

- fscan_sdi
- ascan_sdo

It is optional for an agent to include the scan data inputs and outputs on its DFx interface. Scan chains are not usually part of the RTL of IP-block.

4.3 Asynchronous Scan Control Signals

Scan signals described in this section:

- fscan_ret_ctrl
- fscan_shiften
- fscan_latchopen
- fscan_latchclosed_b
- fscan_clkungate
- fscan_clkungate_syn

There are a number of asynchronous control signals that assist the scan controllers during test. They enable attributes within the agent to make the logic more "scan friendly" to achieve the best possible test coverage. The signals are asynchronous due to the fact that these signals are driven per scan data set but are not required to arrive synchronously with the active scan signals. They may be static for the duration of the particular test but may change per scan set. They are captured with the fscan_updateclk signal for use by the scan controller. The most commonly used scan control signals are listed in the table.

The fscan_ret_ctrl is controlled from a Cluster DFx unit's scan control unit (most likely a TAP register bit) to set or clear the sleep signal to enable HVM testing of the retention flops. The mux that enables this signal is the fdfx_pgcb_bypass signal that is internal to the Power Gate

Common Block (PGCB). Refer to the PGCB HAS document or the SoC Scan Requirements HAS for more information.

The fscan_shiften is a required scan control signal that enables the flops within an agent to shift the test content through the chains. The clock ungate control signals are required for all IP-blocks (fscan_clkungate and fscan_clkungate_syn). This allows the scan controller to force ungating of the clocks to conduct the scan operation necessary for testing. The fscan_clkungate_syn are for installed clock gates by the synthesis tool. The signal is placed on the interface and later connected in the post auto-place and route (APR) flow. If an IP-block is constructed with latches then the fscan_latchopen and fscan_latchclosed_b signals are required on the interface and logically combined with the functional logic to cause the appropriate action. For example, the fscan_latchopen forces the latches to appear transparent during scan shift operations which allow them to work with the existing Mux-D scan architecture.

4.4 Scan Reset Control Signals

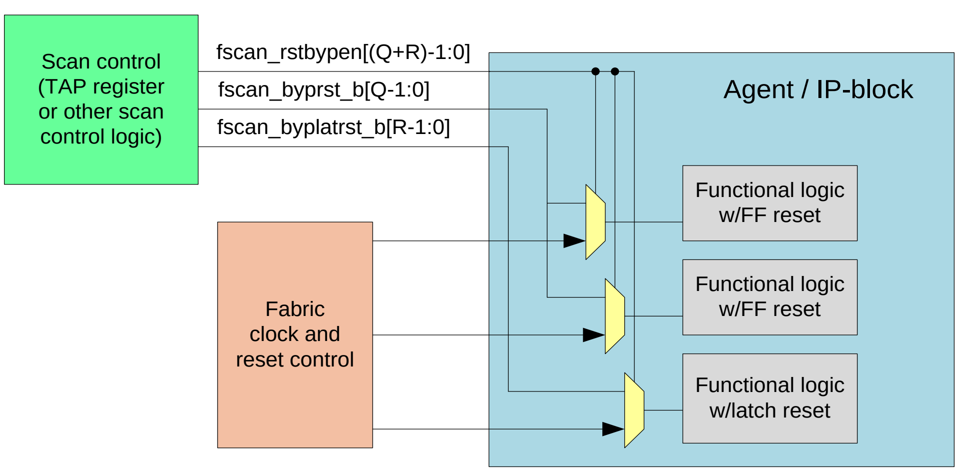
Scan signals described in this section:

- fscan_rstbypen
- fscan_byprst_b
- fscan_byplatrst_b

If the IP-block generates internal resets then it must implement the scan bypass control signals on the interface so the scan controllers can effectively drive them during scan operations. These signals are fscan_rstbypen, fscan_byprst_b and fscan_byplatrst_b.

The IP-block developer is required to separate each reset signal and control separately to allow the SoC integration team the flexibility to organize their scan control infrastructure. This means a separate reset enable and reset value signal for each reset domain and separate set of enables and reset signals for flops and latches. An example is shown in Figure 4-6. The reset bypass enable signal bus than is the summation of the two reset bus types. An SoC integration team may then choose to group the signal types together and drive together.

Figure 4-6. Scan reset bypass control



4.5 Scan Static Control Signals

Scan signals described in this section:

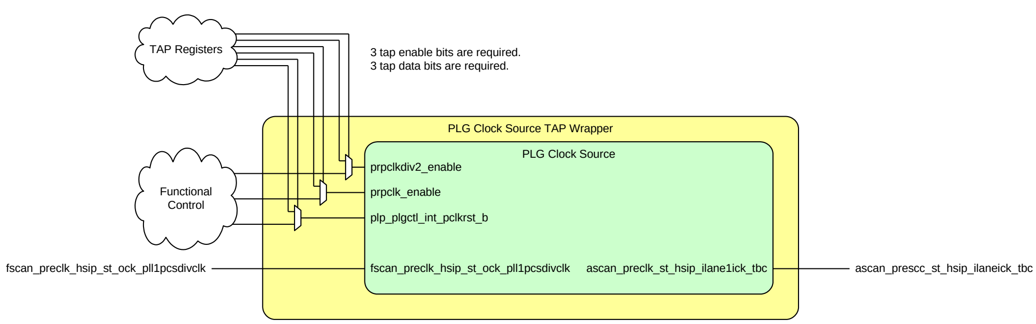
- fscan_mode
- fscan_mode_atspeed
- fscan_clkgenctrl
- fscan_clkgenctrlen

The scan mode signal (fscan_mode) is required on all IP-blocks. The at-speed signal is conditionally required depending on the implementation of internal logic that requires overriding to support the at-speed mode. Since these are statically controlled and stay set for the length of that particular test segment (stuck-at verses at-speed), these signals may be controlled in the cluster DFx unit slave TAP's test data registers.

4.5.1 Clock Generator Override Control Signals

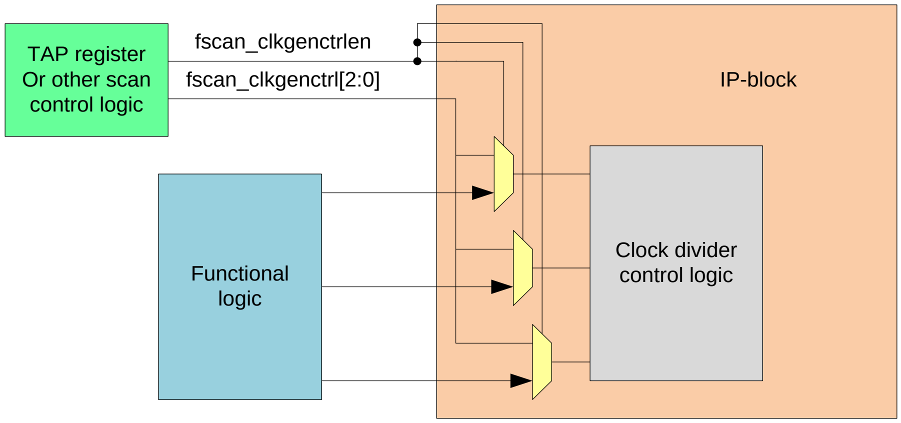
If an IP-block (agent) has logic to control clock dividers or internal clock then the agent must implement a bypass enable (fscan_clkgenctrlen) and override control (fscan_clkgenctrl) on the interface for scan operations. Each control and mux enable must be individually brought out as a signal. Figure 4-7 shows an example clock source block that has one parent clock and three controls. In this example, each control has its own bypass mux and its own control signal. The interface appears on the IOSF DFT interface as fscan_clkgenctrl[2:0] and fscan_clkgenctrlen[2:0]. These control and data values are driven from a TAP register within the DFx fabric for this partition.

Figure 4-7. Clock Generator Control Example 1



Another example is shown in Figure 4-8, where one enable signal controls all of the clock generator control muxes. It is up to the integration team to appropriately drive the signals for each signal group. It is a burden on the IP-block developer to allow for any combination of data inputs and mux control overrides. In other words, an IP-block developer must provide equal numbers of control and enable signals to satisfy both cases.

Figure 4-8. Clock Generator Control Example 2



4.6 Array Shadow Logic for Scan Testing with Arrays

Scan signals described in this section:

- fscan_ram_wrdis_b
- fscan_ram_rddis_b
- fscan_ram_odis_b
- fscan_ram_awt_mode
- fscan_ram_awt_ren
- fscan_ram_awt_wen
- fscan_ram_bypsel

If an agent/IP-block has an array, register file, FIFO or similar structures then it is required to include the fscan_ram_*dis*, fscan_ram_awt* and fscan_ram_bypsel signals. It is up to the SoC integration team to choose which array test methodology to use. The first group supports a sequential mode where the RAM is used as part of the cone of logic between flops in the scan path. A second group provides asynchronous write thru (AWT) mode while a third option is a synchronous path around the array to bypass it completely. The fourth group is a single signal to gate the output. This signal can be applied to any of the other modes.

All seven signals must be present on the wrapper but a parameter selects which set of signals are actually used for that particular type shadow of logic. Therefore, if an agent contains one or more arrays then the IP-develop must provide all the array shadow control signals with a parameter to select which one type is used. The SoC integration team has a choice of which three modes to implement with an optional output disable feature.

The RAM Sequential Mode is shown in Figure 4-9. The memory is a register file or array used by the function logic for a variety of reasons but usually they hold data (transactions) temporary until some traffic control logic grants permission to enter the fabric. These memories are considered large signal arrays which will be tested with the MBIST vendor tool. By contrast a large memory that is greater than 1024 entries will use a small signal array (SRAM) which requires a Programmable BIST (PBIST) for HVM testing.

The basic idea for RAM Sequential Mode is use the array as part of the scan test. An address and data value is set from scan flops further upstream that are controlling the function inputs. The array is used during the scan test to latch on to the binary result after progressing through the cone of logic up to and through the array. It is a multi-cycle scan test. The green muxes and signal inputs are required by the MBIST and outside of the scope of this document. The red block indicates the functional signal inputs to the agent. The orange colored logic must be added to the array wrapper for each agent. The signal names are the IOSF defined name to control this feature. An EXOR gate tree will observe the functional values for the address, read and write enables.

Figure 4-9. RAM Sequential Mode

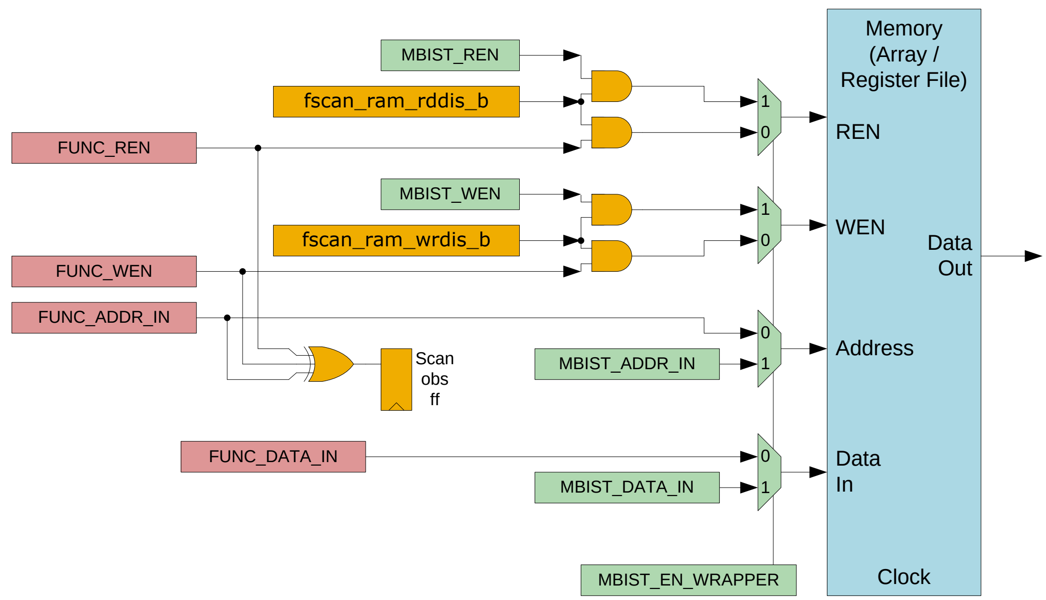
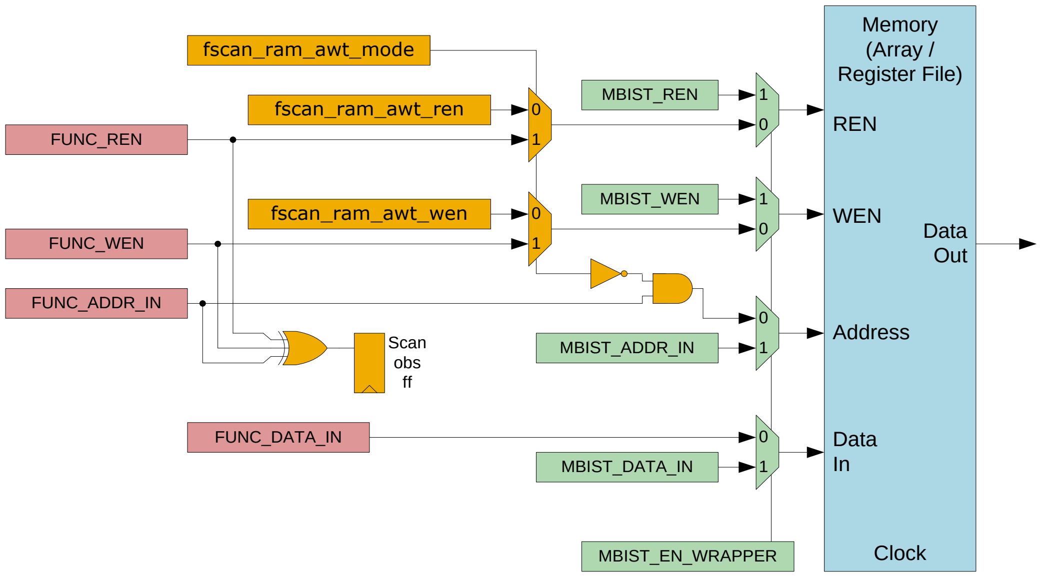


Figure 4-10 shows the Asynchronous Write-Thru mode. This is different than the RAM Sequential Mode in that the scan test doesn't care what address you write to because only the address at logic zero is used. The array becomes a multi-cycle path between the last scan flop before the array and the first scan flop after the array.

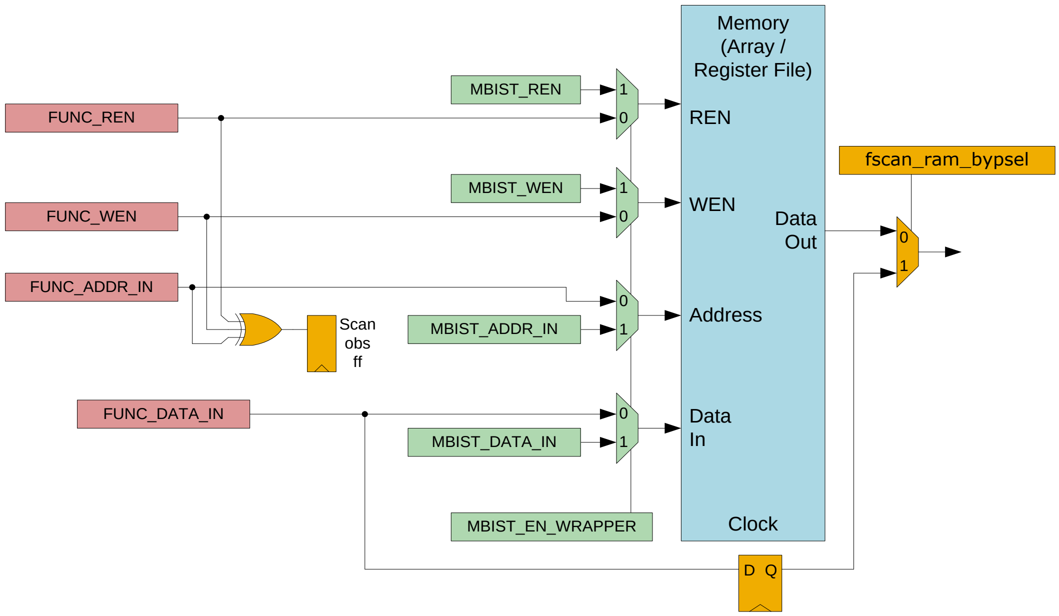
Figure 4-10. Asynchronous Write Thru (AWT) Mode



The Synchronous Bypass Mode is shown in Figure 4-11. The Synchronous Bypass Mode passes the values on the data bus around the array to the next scan flop. The array itself is not used like it is in the AWT mode. The path is still multi-cycle because the cone of logic from the last scan flop (on the data bus) passes through a flop stage in the array wrapper and then it is captured by the first scan flop after the array. The cone of logic for the address, read and write enable signals are captured with an EXOR tree and a scan observability flop.

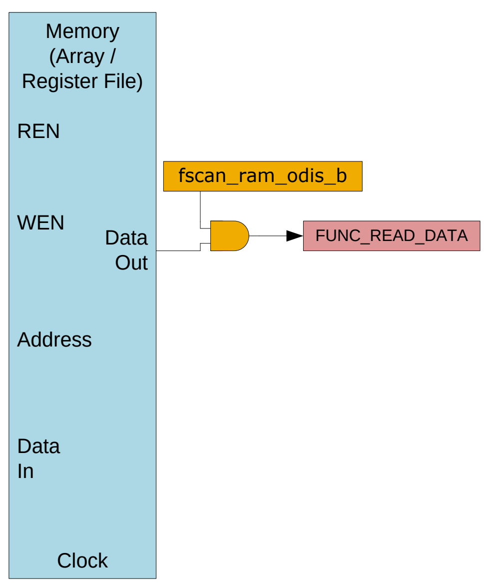
Note: The synchronous bypass mode cannot be used for arrays that are enabled for Array/Freeze/Dump (AFD) debug feature.

Figure 4-11. Synchronous Bypass Mode



An optional output disable feature is shown in Figure 4-12. The fscan_ram_odis_b is available for any flavor of array shadow logic for scan testing that was previous described (RAM Sequential, AWT or Sync-bypass).

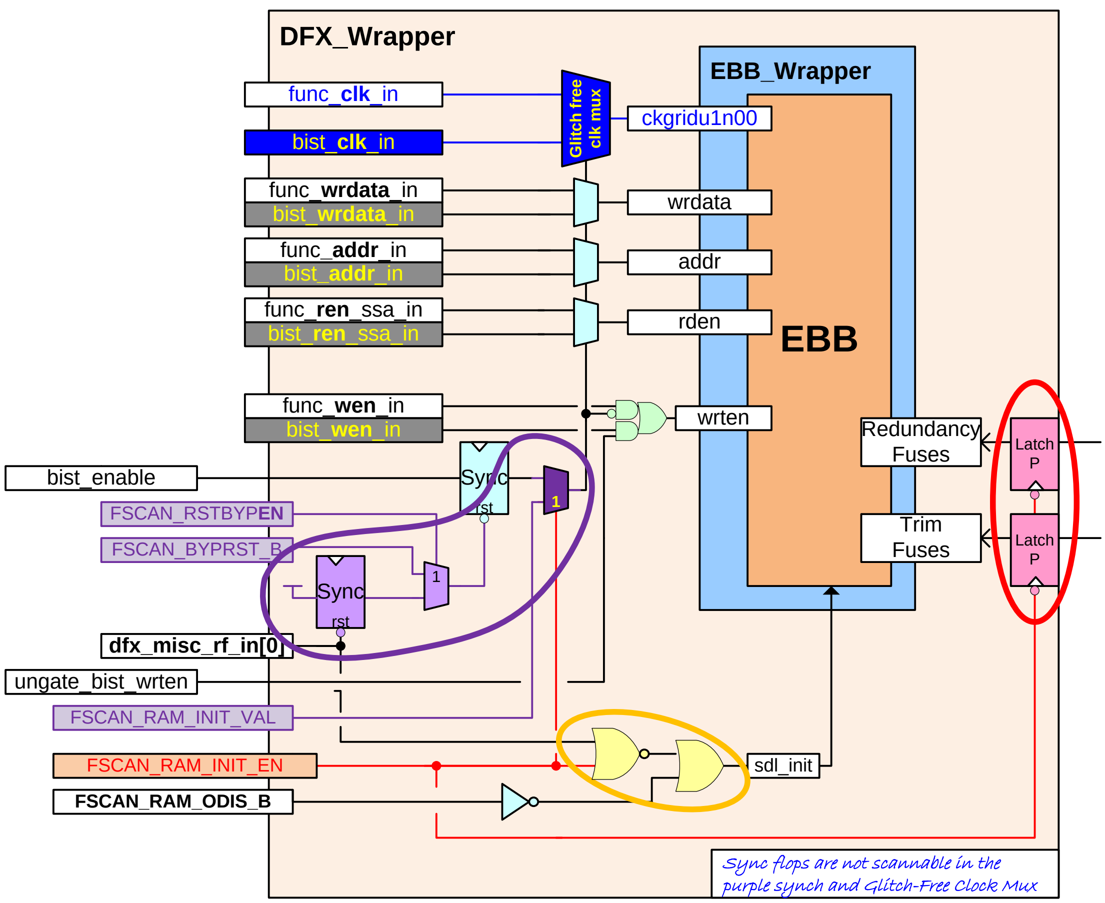
Figure 4-12. Output Disable Feature



4.6.1 Proposed: Array initialization for scan

To improve scan converge around the arrays these scan control signals will initialize the array to remove X pollution and allow scanning of the synchronizer. Shown in Figure 4-13 is a block diagram an array wrapper with the DFT control signals. The fscan_ram_init_val is a dynamic control signal that allows scan test to control the DFx muxes to select between the functional control signals or the memory BIST control signals. The fscan_ram_init_en is a static control signal to initialize the array for scan operations. This control has three functions, one is to enable the use of the fscan_ram_init_val signal, a second function is to remove X's from appear out of the array and the third is to latch the redundancy and trim values from the fuses.

Figure 4-13. Array scan initialization control



4.7 Functional Test Support

Although this is a structural test chapter, functional testing has been used for test hole reduction. Previous SoC's has used the Test Access Mechanism (TAM) that is located in the Pondicherry SoC System Agent's A-unit as the source of transactions that target the soft IP-blocks. Within the soft-IP blocks, a near end digital loopback path is created where downstream transactions are altered with Address Translation Mode (ATM) to generate upstream transactions that are consumed by a MISR in the TAM. The MISR is an event-based signature accumulator to eliminate X contamination of the final pass/fail signature that determines if the test hole is covered. This spec strong recommends that soft IP-blocks implement the near-end digital loopback with ATM.

4.7.1 AMT/EAMT methodology

The AMT/EAMT methodology is applicable only to the PCIe controller for functional test. If the IP-block is a PCIe controller then the IP should implement both AMT/EAMT and the high speed bypass feature to allow the integration team to select which provides the best coverage for their market segment. The transaction layer shown below is responsible for generation of the out bound TLP (Transaction Layer Packets) traffic and the reception of the inbound TLP traffic. Digital near end loop (DNELB) is implemented almost at the beginning of the physical layer fabric and this DFX infrastructure is capable of looping back the downstream traffic into the upstream without the need for external devices.

Figure 4-14. Loopback mechanism in the PCIe agent

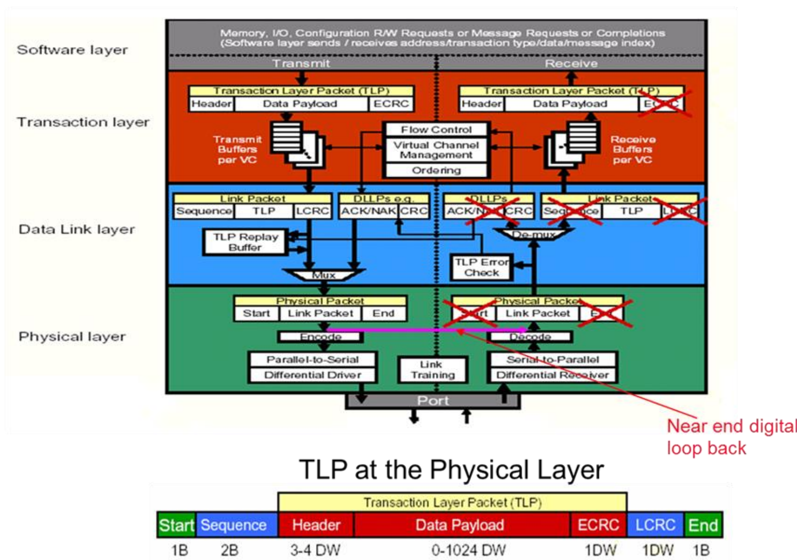
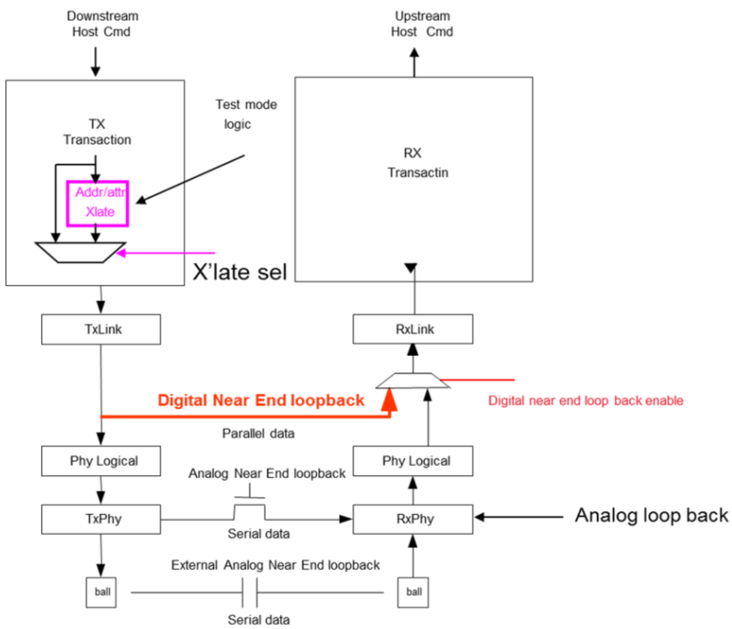


Figure 4-15 shows transaction being hacked in the downstream before looped back. In general it could be hacked either before or after being looped back and the shown implementation is a matter of choice in each product. For completeness, the other loopback mechanisms that exist in PCIE are also shown, namely analog on the die and analog off-the die loop backs.

Figure 4-15. Digital near end loopback in serial IO



4.7.2 High speed bypass methodology

All soft-IP serial interface controller agents (PCIe, USB, SATA, and others) must implement the high speed bypass for functional test coverage. Figure 4-16 shows high speed bypass mechanism. A two-step process is followed to send and receive the transactions using this structure.

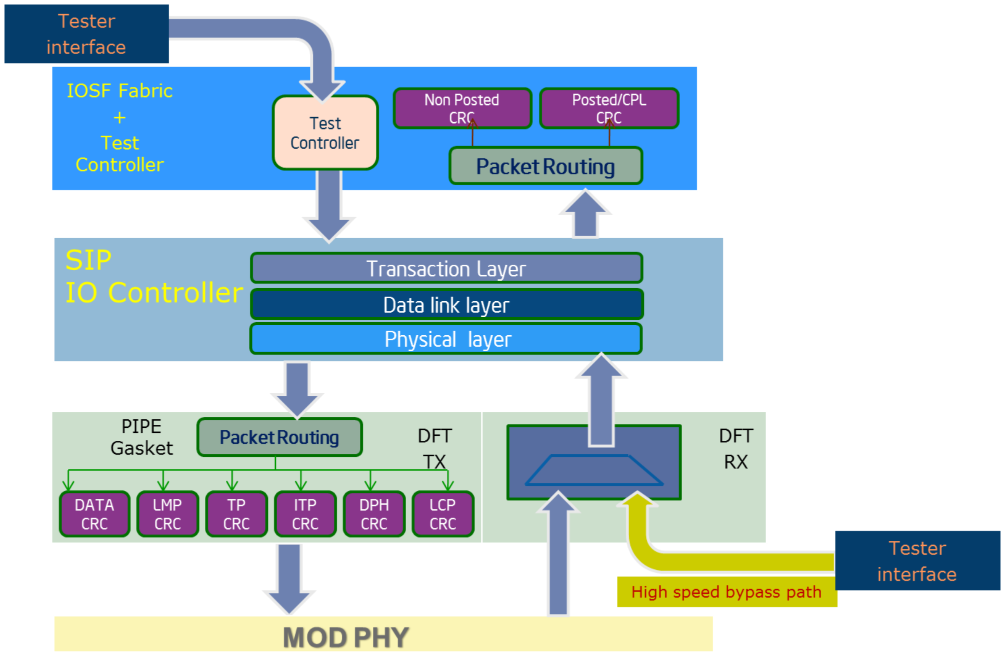
Step1: Down Stream transactions

- Tester loads the Test controller and triggers start command
- Transactions flow through the IOSF fabric to the IP under testing
- Signature is collected in the IP using different MISRs
  - Collected signature should be agnostic to determinism

Step2: Upstream transactions

- Tester responds with data/completions using the "high-speed bypass path" with over the ftc_data and optionally the ftc_clk pin.
- This tester sent data is muxed with the incoming data from Modphy. All the incoming signals in the upstream path are forced by the tester.
- Test controller collects the incoming transactions into various MISRS.

Figure 4-16. High speed bypass block diagram



SOC/PCH should have provision to "stage" the HBP path to meet transaction timing. It is recommended that these HBP pins be piggy-backed onto the Scan pins. Scan routing reaches every corner and IP of the chip and hence with this approach, there is no additional cost for routing the HBP wires.

The agent/IP-block must use the ftc_data and ftc_clk signals to provide the IOSF compliant interface for consuming the transactions out-of-band and sending it back to the test controller. Any particular SIP controller will not need more than 24 inputs but the width is a parameter (TestCtrlIn_WIDTH) so it can be set based on the needs for the agent/IP-block. The IP-block must parameterized (HBP_DOUBLE_DATA_RATE_ENABLE=1) to handle data arriving on both edges of the clock to satisfy performance requirements of delivering content to the device to provide the stimulus necessary for the expected coverage. A future revision of the IOSF DFx HAS will re-examine this for a more formalize signal interface.

Table 4-2. Example of SIP HBP IOSF compliant interface signals

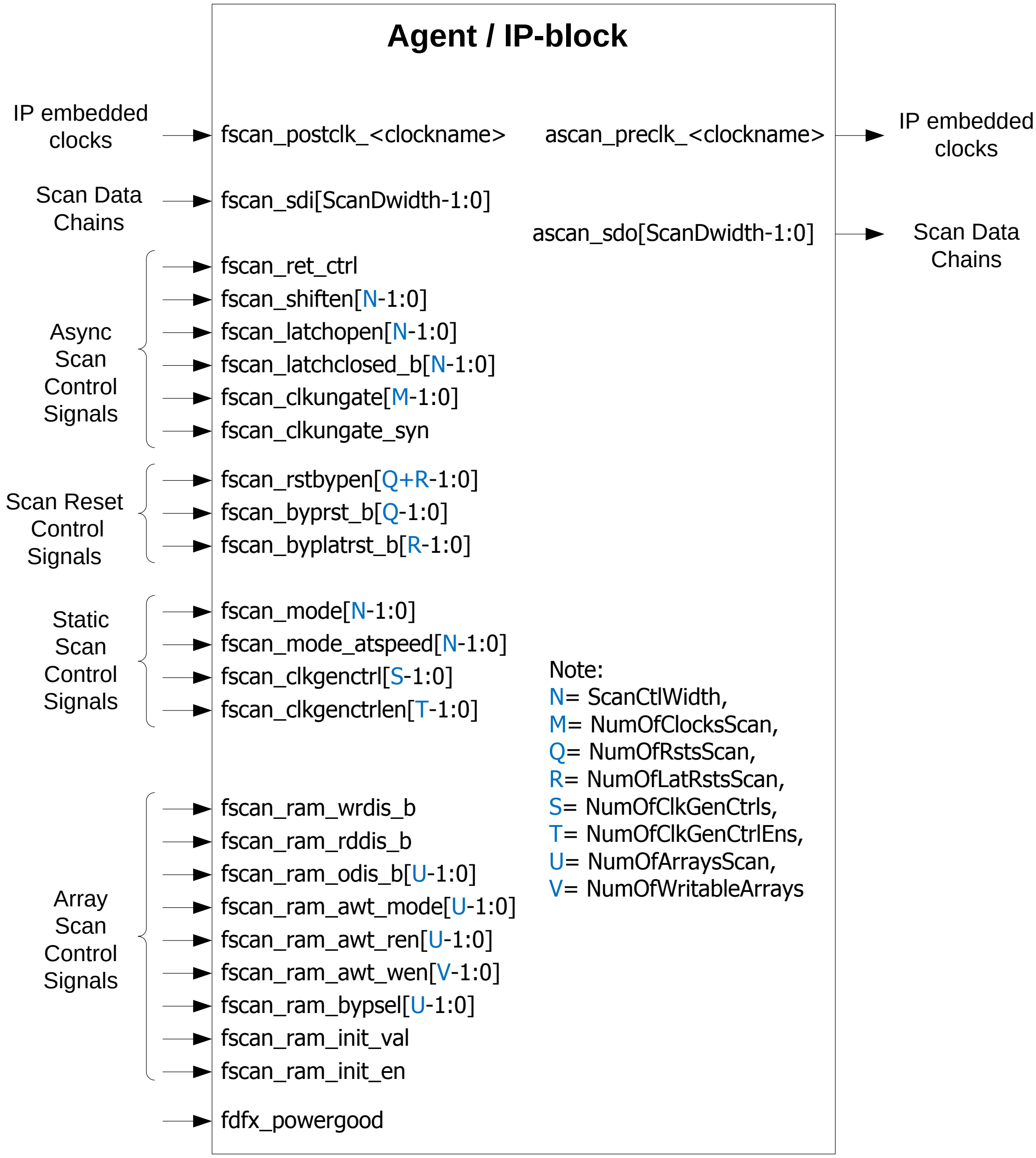
SIP IP-block with HBP bypass	Internal HBP Pin	IOSF DFx pin
USB2 HBP Rx Bypass TestCtrlIn_WIDTH = 10	dt_ux_usb2hbp_data[9:0]	ftc_data[9:0]
	Common HBP CRC reset (dt_ux_hbp_crcrst_b)	ftc_data[10]
	Common HBP clock (dt_ux_hbp_clk)	ftc_clk
HSIC HBP Rx Bypass TestCtrlIn_WIDTH = 10	dt_ux_hsichbp_data[9:0]	ftc_data[9:0]
	Common HBP CRC reset (dt_ux_hbp_crcrst_b)	ftc_data[10]

	Common HBP clock (dt_ux_hbp_clk)	ftc_clk
USB3 HBP Rx Bypass TestCtrlIn_WIDTH = 24	dt_ux_hbpdata[15:0]	ftc_data[15:0]
	dt_ux_hbpframe[1:0]	ftc_data[17:16]
	dt_ux_hbp_rx_status[1:0]	ftc_data[19:18]
	dt_ux_hbp_phystatus	ftc_data[20]
	dt_ux_hbp_rxelecidle	ftc_data[21]
	dt_ux_hbp_rxvalid	ftc_data[22]
	Common HBP CRC reset (dt_ux_hbp_crcrst_b)	ftc_data[23]
	Common HBP clock (dt_ux_hbp_clk)	ftc_clk
SSIC HBP Rx Bypass	dt_ux_rmmi_rx_phy_do_rdy[1:0]	ftc_data[1:0]
TestCtrlIn_WIDTH = 24	dt_ux_rmmi_rx_symbol_err[1:0]	ftc_data[3:2]
	dt_ux_rmmi_rx_datan_ctrl[1:0]	ftc_data[5:4]
	dt_ux_rmmi_rx_symbol[15:0]	ftc_data[21:6]
	dt_ux_rmmi_rx_burst	ftc_data[22]
	Common HBP CRC reset (dt_ux_hbp_crcrst_b)	ftc_data[23]
	Common HBP clock (dt_ux_hbp_clk)	ftc_clk

4.8 Scan Signal Interface Description

A graphical view of the scan interface signal set is shown in Figure 4-17. The signal details are listed in Table 4-3. The integration team has the freedom to assign any reasonable number of characters as a prefix or suffix as applied to the root name to identify the signal in a full chip SoC. The parameter summary is list in Table 4-4.

Figure 4-17. Agent specific scan signal interface block diagram



1 **Table 4-3. Scan interface signal descriptions**

Signal	I/O	R/O/C[1]	Description
Embedded clock control			
ascan_preclk_ <clockname>	O	C	**Fabric Pre-clock <source clock name>:** This signal, and its accompanying postclk signal, is optionally available for those logic blocks that produce internally generated clocks for their logic. This port allows IP-blocks to export these derived clocks for control by scan clock control logic outside the IP-block (wrapper, partition, or full chip). These internally generated clocks should be directly sent out, prior to functional use i.e. "pre" scan control. Note: The pre and post must exist as pairs. If there is need for ascan_preclk_<clockname> then interface must also have fscan_postclk_<clockname>
fscan_postclk_ <clockname>	I	C	**Agent Post-clock <source clock name>:** This input signal is associated with accompanying preclk output signal. It allows IP-blocks to receive the "post" scan clock control version of internally derived clocks. This version of the clock connects to all modules within this agent/IP-block that were originally connected to the internally-generated derived functional clock. Note: The pre and post must exist as pairs.
Scan data chain signals			
fscan_sdi [ScanDwidth-1:0]	I	O	**Fabric Scan Data In:** This signal bus is the scan data inputs for all of the serially-stitched scan flops/latches within this IP-agent.
ascan_sdo [ScanDwidth-1:0]	O	O	**Agent Scan Data Out:** This signal bus is the scan data outputs for all of the serially-stitched scan flops/latches within this agent.
Asynchronous scan control signals			
fscan_ret_ctrl	I	C	**Fabric scan retention control:** This signal determines the state of the retention cell within a retention flop for scan operations. A mux in the Power Gate Common Block (PGCB) is controlled by fscan_mode. When enabled, the signal is controlled from an asynchronous scan controller (SASC) in the DFx fabric. **Note:** This is signal is required if the IP-block supports retention cells.
fscan_shiften [ScanCtlWidth-1:0]	I	R	**Fabric Scan Shift Enable:** This signal determines whether the data chains are enabled for shifting this does not apply to the control chains. This signal is bused to support hard IP-block modular physical layer that requires scan control per lane. Use of a bit vector for this signal name is optional. The value of ScanCtlWidth is implementation-dependent and may vary per-agent or per-lane for a hard-IP IO agent.

Signal	I/O	R/O/C[1]	Description
fscan_latchopen [ScanCtlWidth-1:0]	I	C	**Fabric Scan Latch Open Enable:** This signal controls the latch open during scan operations. This signal is bused to support hard-IP block modular physical layer that requires scan control per lane. Use of a bit vector for this signal name is optional. The value of ScanCtlWidth is implementation-dependent and may vary per-agent or per-lane for a hard-IP IO agent. Note: If this IP-block contains latches then this signal must be used to control them.
fscan_latchclosed_b [ScanCtlWidth-1:0]	I	C	**Fabric Scan Latch Closed bar:** This signal controls the latch closed during scan operations. This signal is bused to support hard-IP block modular physical layer that requires scan control per lane. Use of a bit vector for this signal name is optional. The value of ScanCtlWidth is implementation-dependent and may vary per-agent or per-lane for a hard-IP IO agent. Note: If this IP-block contains latches then this signal must be used to control them.
fscan_clkungate	I	R	**Fabric Scan Clock Ungate:** This signal controls the clock gating logic during scan operations. This signal is bused to support hard-IP block modular physical layer that requires scan control per lane.
fscan_clkungate_syn	I	R	**Fabric Scan Clock Ungate for Synthesis Inserted Clock Gates:** This signal controls the clock gating logic inserted during synthesis. This signal cannot be used interchangeably with the fscan_clkungate signal that is used exclusively to control clock gating logic that exists in the pre-synthesis design. This signal controls the clock gating logic that is added after synthesis for scan operations.
Scan reset control signals			
fscan_rstbypen [(NumOfRstsScan+ NumOfLatRstsScan) -1:0]	I	C	**Fabric Scan Reset Bypass Enable:** This signal will enable the ability for the bypass reset signals to be active. The reset override signal group must be implemented for IP-blocks with embedded or derived internal reset signals. • 0: Reset bypass and Latch reset bypass are ignored. • 1: Reset bypass and Latch reset bypass are active. Use of a bit vector for this signal name is optional. The value of NumOfRstsScan is implementation-dependent and may vary per-agent or per-lane. This signal is bused to support hard-IP block modular physical layer that requires scan control per lane. Note: It is expected that a soft-IP agent will only use a single control wire for enabling the reset bypasses. A hard-IP agent that implements scan on a per lane basis will require a vector set of enable signals.

Signal	I/O	R/O/C[1]	Description
fscan_byprst_b [NumOfRstsScan-1:0]	I	C	**Fabric Scan Bypass Reset bar:** This signal is a reset input for scan operations that bypasses the internal agent reset logic and applies a reset directly to the agent. The reset override signal group must be implemented for IP-blocks with embedded or derived internal reset signals. Note1: Use of a bit vector for this signal name is optional. The value of NumOfRstsScan is implementation-dependent and may vary per-agent or per-lane. Note2: This signal is enabled with fscan_rstbypen.
fscan_byplatrst_b [NumOfLatRstsScan-1:0]	I	C	**Fabric Scan Bypass Latch Reset bar:** This signal is a reset input for scan operations that bypasses the internal agent reset logic and applies a reset directly to the latches within the agent. This signal is bused to support hard-IP block modular physical layer that requires scan control per lane. Note1: Use of a bit vector for this signal name is optional. The value of NumOfLatRstsScan is implementation-dependent and may vary per-agent or per-lane. Note2: This signal is enabled with fscan_rstbypen.
Static scan control signals			
fscan_mode [ScanCtlWidth-1:0]	I	R	**Fabric Scan Mode:** This signal enables modes within this agent for scan operations. This signal is bused to support a hard IP-block's physical layer that requires scan control per lane. Use of a bit vector for this signal is optional. The value ScanCtlWidth is implementation dependent and may vary per-agent or per-lane. Soft-IP use model: This signal may or may not be used depending on the attributes within the IP-block that need to be made scan friendly. However, it is still required on the interface. Hard-IP use model: Its primary use is to enable the SCC/SCRC controller for this hard-IP partition. Other scan enabling features should be controlled by the asynchronous control signal group. If a scan attribute is unique to this partition and a corresponding control signal is not available than a local TAP test/debug register will enable its actions.
fscan_mode_atspeed [ScanCtlWidth-1:0]	I	C	**Fabric Scan At-speed Mode:** This signal enables the at-speed mode for this agent. This signal is bused to support hard-IP block modular physical layer that requires scan control per lane. Use of a bit vector for this signal name is optional. The value of ScanCtlWidth is implementation-dependent and may vary per-agent or per-lane.

Signal	I/O	R/O/C[1]	Description
fscan_clkgenctrl [NumOfClkGenCtrls-1:0]	I	C	**Fabric Scan Clock Generator Control:** This signal bus overrides clock control values within the agent. The override value is enabled with fscan_clkgenstrlen. This bus may be composed of clock select override and other miscellaneous control signals used for conditioning the clock selects. For agents with a TAP, these signals would be connected to the output of an assigned test data register. For agents without a TAP, this signal bus that is connected to a scan control logic block (SCC/SCRC) within the DFx fabric. Note: This signal may be a single bit.
fscan_clkgenctrlen [NumOfClkGenCtrlEns-1:0]	I	C	**Fabric Scan Clock Generator Control Enable:** This signal (or signal group) is the enable for the fscan_clkgenctrl override control bus. A mux override control can manipulate the signal only during scan operations. For agents with a TAP, these signals would be connected to the output of an assigned test data register. For agents without a TAP, this signal group is a bus that is connected to and by controlled by the scan control logic block (SCC). If this signal is a bus then bit[0] of the interface is assigned to the SCC the fscan_clkgenctrlen[0]: This bit may be assigned to select between internal functional clocks and external SCC clocks. It is implementation dependent to bus this signal or use it as a single bit.
Array shadow logic for scan control signals			
fscan_ram_wrdis_b	I	C	**Fabric Scan RAM Write Disable bar:** This signal controls the write enable on the agent's array during scan operations. Note: This is signal is required for any IP-block's memory wrapper that contains an array, RAM, FIFO, etc.
fscan_ram_rddis_b	I	C	**Fabric Scan RAM Read Disable bar:** This signal controls the read enable on the agent's array during scan operations. Note: This is signal is required for any IP-block's memory wrapper that contains an array, RAM, FIFO, etc.
fscan_ram_odis_b [NumOfArraysScan-1:0]	I	C	**Fabric Scan RAM Output Disable bar:** This signal controls masking output of the agent's array during scan operations. Note: This is signal is required for any IP-block's memory wrapper that contains an array, RAM, FIFO, etc.
fscan_ram_awt_mode [NumOfArraysScan-1:0]	I	C	**Fabric Scan RAM AWT Mode:** This signal enables the mode to conduct scan operations on an Array Write Through (AWT) testing model for arrays. Note: This is signal is required for any IP-block's memory wrapper that contains an array, RAM, FIFO, etc.

Signal	I/O	R/O/C[1]	Description
fscan_ram_awt_ren [NumOfArraysScan -1:0]	I	C	**Fabric Scan RAM AWT Read Enable:** This signal is the read enable for scan operations using an Array Write Through (AWT) testing model for arrays. Note: This is signal is required for any IP-block's memory wrapper that contains an array, RAM, FIFO, etc.
fscan_ram_awt_wen [NumOfWritableArrays -1:0]	I	C	**Fabric Scan RAM AWT Write Enable:** This signal is the write enable for scan operations using an Array Write Through (AWT) testing model for arrays. Note: This is signal is required for any IP-block's memory wrapper that contains an array, RAM, FIFO, etc.
fscan_ram_bypsel [NumOfArraysScan -1:0]	I	C	**Fabric Scan RAM Bypass Select:** This signal selects the bypass path around the array to conduct scan operations on this type of array test configuration. Note: This is signal is required for any IP-block's memory wrapper that contains an array, RAM, FIFO, etc.
fscan_ram_init_val	I	C	**Fabric Scan RAM initialization value.** This signal is the RAM initialization value to control the DFx muxes with in the array DFX_Wrapper. 0: Mux points to functional array controls. Also, this signal has no effect on all other initialization logic. 1: Mux points to DFT array controls (from BIST)
fscan_ram_init_en	I	C	**Fabric Scan RAM initialization enable.** This signal controls the array initialization for scan operations. 0: Normal operation 1: Enable initialization and the logic value currently driven on fscan_ram_init_val is active.
Reset input signals			
fdfx_powergood	I	R	Refer to section 6.2. Note: This signal is equivalent to **dfx_powergood_rst_b**.
NOTES: [1]Note1: R = required, O = optional, C = conditional. If the IP-block contains logic that requires specific controls to manage that logic during test operations then the signal is required. For example, if an IP-block has an array then the scan test controls for arrays are required.			

4.8.1 Scan signal parameter summary

Table 4-4. Scan parameter summary table

Parameter Name	Letter designation from diagram	Description
ScanCtlWidth	N	**Scan Control Width:** This strap is primarily used for hard-IP agents where the scan control segregated per lane. However, soft-IP agents can certainly take advantages of this feature.

Parameter Name	Letter designation from diagram	Description
ScanDwidth	-	**Scan Data chain width:** This strap value determines the number of scan data chains.
NumOfClocksScan	M	**Number of Clocks for Scan:** This value determines the number of clocks that are supported by this agent that require scan override control. Soft-IP agents: This value is used for the number of clocks that require bypassing. Hard-IP agents: This value may be set to the same value as ScanCtlWidth to control the scan logic on a per lane basis.
NumOfRstsScan	Q	**Number of Resets for Scan:** This value determines the number of resets that are supported by this agent that require scan override control. Soft-IP agents: This value is used for the number of resets that require bypassing. Hard-IP agents: This value may be set to the same value as ScanCtlWidth to control the scan logic on a per lane basis.
NumOfLatRstsScan	R	**Number of Resets with Latch-based design for Scan:** This value determines the number of resets associated with latches that are supported by this agent that require scan override control. Soft-IP agents: This value is used for the number of latched based resets that require bypassing. Hard-IP agents: This value may be set to the same value as ScanCtlWidth to control the scan logic on a per lane basis.
NumOfClkGenCtrls	S	**Number of Clock Generate Control Overrides:** This value determines the number of clock control signal overrides for the fscan_clkgenctrl signal group.
NumOfClkGenCtrlEns	T	**Number of Clock Generate Control Enables:** This value determines the number of clock control signal overrides for the fscan_clkgenctrl signal group.
NumOfArraysScan	U	**Number of Arrays for Scan:** This value determines the number of arrays that require scan control overrides. NumOfArraysScan = NumberOf_RFarrays + NumOf_SRAMarrays + NumOf_ROMarrays
NumOfWritableArrays	V	**Number of Arrays for RF and SRAM:** This value is the number of arrays that do not include ROMs. NumOfArraysScan = NumberOf_RFarrays + NumOf_SRAMarrays
NumOf_RFarrays	-	**Number of Arrays for RF array type**
NumOf_SRAMarrays	-	**Number of Arrays for SRAM array type**
NumOf_ROMarrays	-	**Number of Arrays for ROM array type**

1

4.9 Transaction Cycle/Data Flows

Not applicable

4.10 Ordering/Coherency Rules

Not applicable

4.11 Performance/Bandwidth Analysis

Not applicable

4.12 Exception List Requirements

Not applicable

4.13 Programming Model

Refer to the SoC DFT Handbook.

4.14 Power Management Capabilities

Not applicable

4.15 Security Feature Requirements

The SIP TAP rev1.5.x supports the DFx secure policy plug-in IP-block. This plug-in will allow the user to specify the security requirements for access to TAP on a network (security controlled via Select register) or specific opcodes within a TAP.

4.16 DFx Requirements

The SoC DFT Handbook should be referenced a general overview of SoC test methodologies.

Link:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC%20DFx/Forms/AllItems.aspx

Directory:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC DFx/DFx Overview Docs/SoC DFT Handbook

4.16.1 DFV/DFD Requirements

Refer to the SoC DFT Handbook.

4.16.2 DFT Requirements

Not applicable

4.16.2.1 Burn-in Specific Requirements

Not applicable

4.16.3 DFM Requirements

Not applicable

§

5 Array Test Interface

This chapter describes the array test interface as defined by the MBIST tool. It is included as part of the IOSF DFx definition.

Table 5-1. Chapter revision history

Revision Number	Description	Revision Date
rev1.2_rc1	• Initial release.	Jan 2012
rev1.2_rc3	• Added this chapter	Feb 13, 2012
rev1.2_rc4	• Removed fary_pwren_b and added a power enable per array/ram/rom type. — fary_pwren_b_rf, fary_pwren_b_sram, fary_pwren_b_rom • Added an output for the power enable signal so that it can be stitched serially between the array/ram/roms. — aary_pwren_b_rf, aary_pwren_b_sram, aary_pwren_b_rom • Added a LYA bus for both high and low inputs.	April, 2012
rev1.2_rc7	• Added new LYA signals • Added fuse bus from miscellaneous signal list but added a prefix to uniqufy the fuses to each memory type. • Added wakeup and firewall enable signals for each memory type.	July 7, 2012
rev1.2_rc7a	• An urgent fix to the wakeup and firewall signals. Only the SRAM needs them so the “sram” name was dropped.	July 11, 2012
rev1.2 (final)	• In rc7 the ffuse_data bus had a prefix for each array type. It actually doesn’t make sense for ROMs. So we changed it to be generic. However, further investigation into the memory use models showed all three memory types used the fuses so it was included in this revision and moved to the end of the signal name as part of the suffix • Included the fuse data valid for all mem types but it is a common signal for the agent/IP-block •	July 20, 2012
Rev1.2.1	• Updated AFD section • Added array testing for arrays with embedded power gate control logic. • Added new section for MBIST diagnostic and done signal assertion.	Feb 2013
rev1.2.2 rc1	• Added new power gate mux select signal. This signal control the mux power gate override signals for an autonomously power gated array.	Jan 2014

The array test DFT signals are required if an agent/IP-block has one of the following memory types; a register file, a SRAM based small signal array, or a read-only memory (ROM).

5.1 Existing Array Test Methods

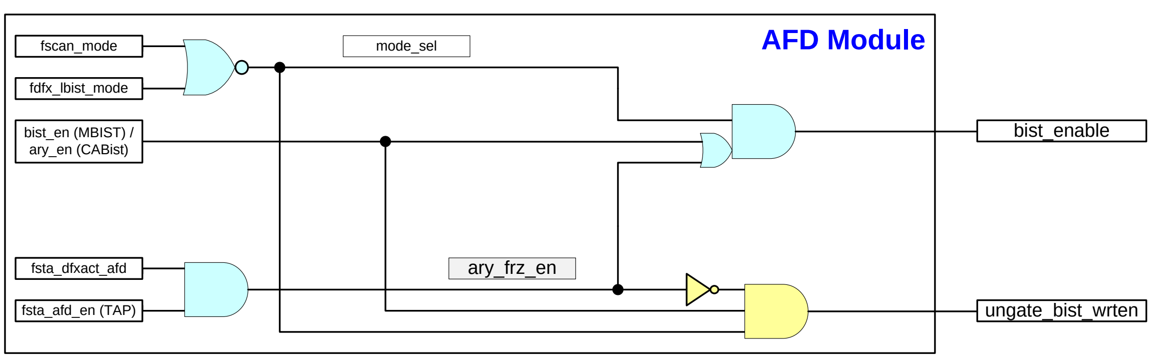
It has been established through other convergence work groups that the Logic Vision MBIST will be the methodology of choice for testing register files and Read-Only Memories (ROMs). Testing of the SRAM small signal based memory structures have not converged but the memory wrappers for this revision have and the interface defined in Table 5-2 through Table 5-4. This specification in the past has not included vendor based interfaces because it is outside of our control when signals change or new ones are being required. However, this particular interface happens to be more stable than other IOSF DFx interfaces has been in the past due to integration team feedback and feature enhancements. Since SoCs have recently migrated to a Collage-based tool flow for integrating IP-blocks, it is now a more compelling reason to include the MBIST interface.

5.2 Array/Freeze/Dump module

The array/freeze/dump DFx feature has been an effective debug use model in the client computer segment. Any IP that contains a SRAM or register file must comply by providing the capability to perform array/freeze/dump by providing the signal interface and connecting it to the array DFx wrapper. A block diagram view is shown in Figure 5-1 is for illustration purposes only and may not reflect the actual implementation. The trigger assertion forces the BIST collar DFT muxes to be switch from the functional path to the MBIST path. The AFD trigger input is logically combined with the MBIST enable and the other scan controls. The enabled signal is synchronized to the array's write clock domain within the memory wrapper.

This module is included as part of every MBIST wrapper. It is the responsibility of the SoC to protect arrays from unauthorized access from dumping their contents. This is easily accomplished in the DFx fabric with the TAP security feature that enables or disables a TAP on the network. In other words, the MBIST TAP will be designated as a red level of access in the region DFx unit's sTAP Select register. For more information about security for the TAPs refer to the SoC TAP HAS (section 17.4.3). Also, more detailed information about the use of AFD with agents/IP-blocks refer to the SIP DFX requirements specification (section 17.4.10).

Figure 5-1. AFD module block diagram



5.3 Array testing with embedded power gate control

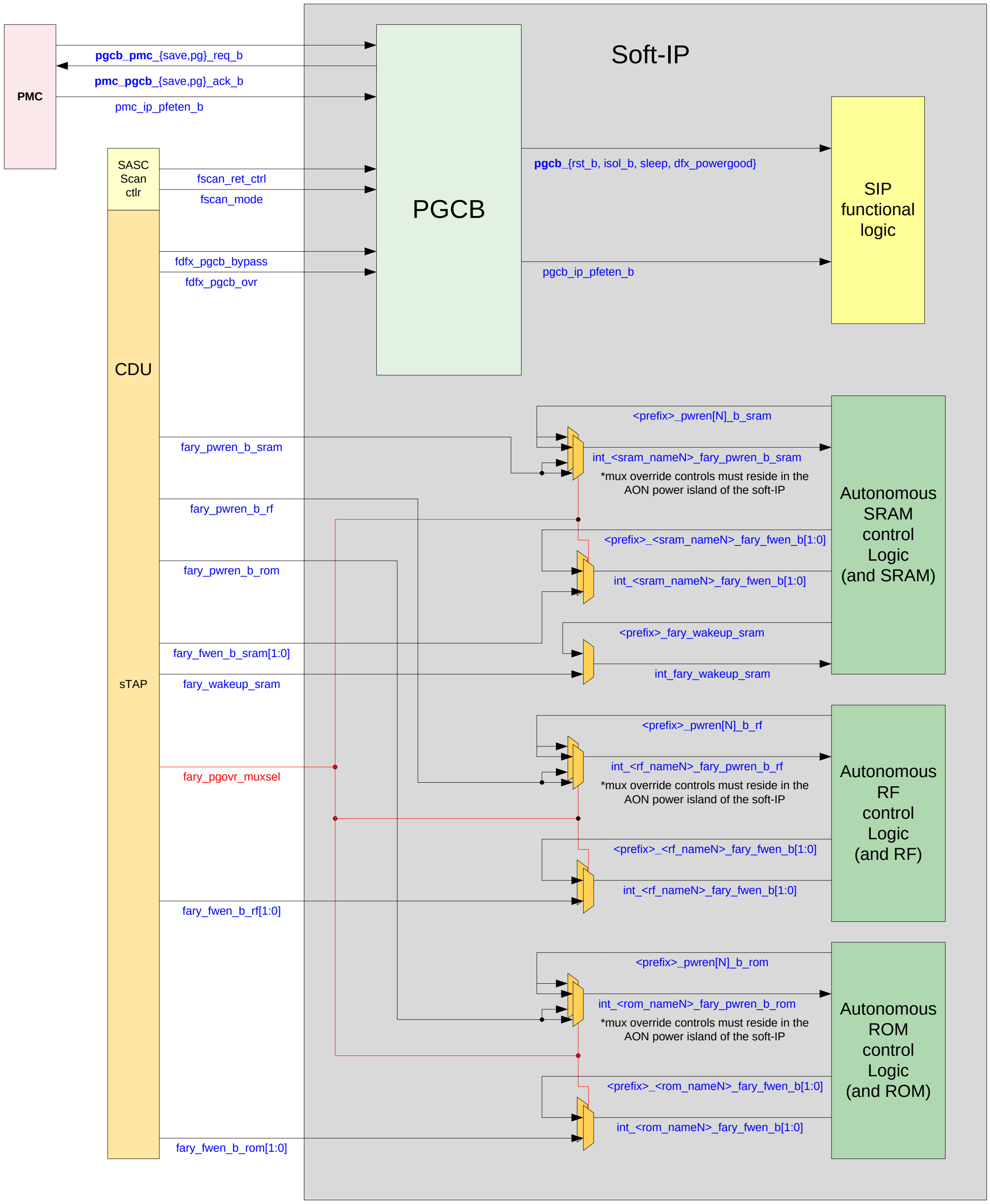
If soft IPs are designed with independent power gate controls for the SRAM, Register File and Read-Only Memory array types then they are required to provide the DFx override muxes for HVM testing. In other words, if the array memory types have internal logic to separately control power gate and firewalls from the Power Gate Common Block (PGCB) then a mux override must be implemented by soft IP-block. It is more likely that Register Files (RF) and ROMs will be power gated with the bulk control logic of the IP which is controlled by the Power Gate Common Block (PGCB) however, the option is available.

The block diagram shown in Figure 5-2 shows the full range of options for each memory array type and the intended implementation. The mux overrides is controlled by one IOSF DFT signal and fanned out to all SRAM banks such that one control enables all them for HVM testing. If a soft-IP implements RF or ROMs with independent power control then these must also implement the mux overrides. For example, in most cases the RF and ROM power gate controls are managed with the control logic, meaning, if the soft-IP logic is powered up so is the RF and ROMs. In this case, the Power Gate Common Block (PGCB) controls the power sequencing for these two array types and the independent mux overrides are not necessary.

The control for array mux overrides is the fary_pgovr_muxsel signal. If an IP-block is composed of several PGCBs then the IP-block may add a prefix to uniquely identify them. The SoC integration team will assign a CDU TAP bit to drive one or all the mux select overrides

The IOSF array DFT power enable pin interface does not change but the array DFx interface is updated for the firewall enables for each memory type. It is responsibility of the soft-IP to implement the mux overrides internally and to name the signals according to their naming convention. The group of pins on the interface is conditional depending on the type of array implemented and the implementation of separate control requirements from the IP’s PGCB enabling requirements.

Figure 5-2. EBB Override for HVM



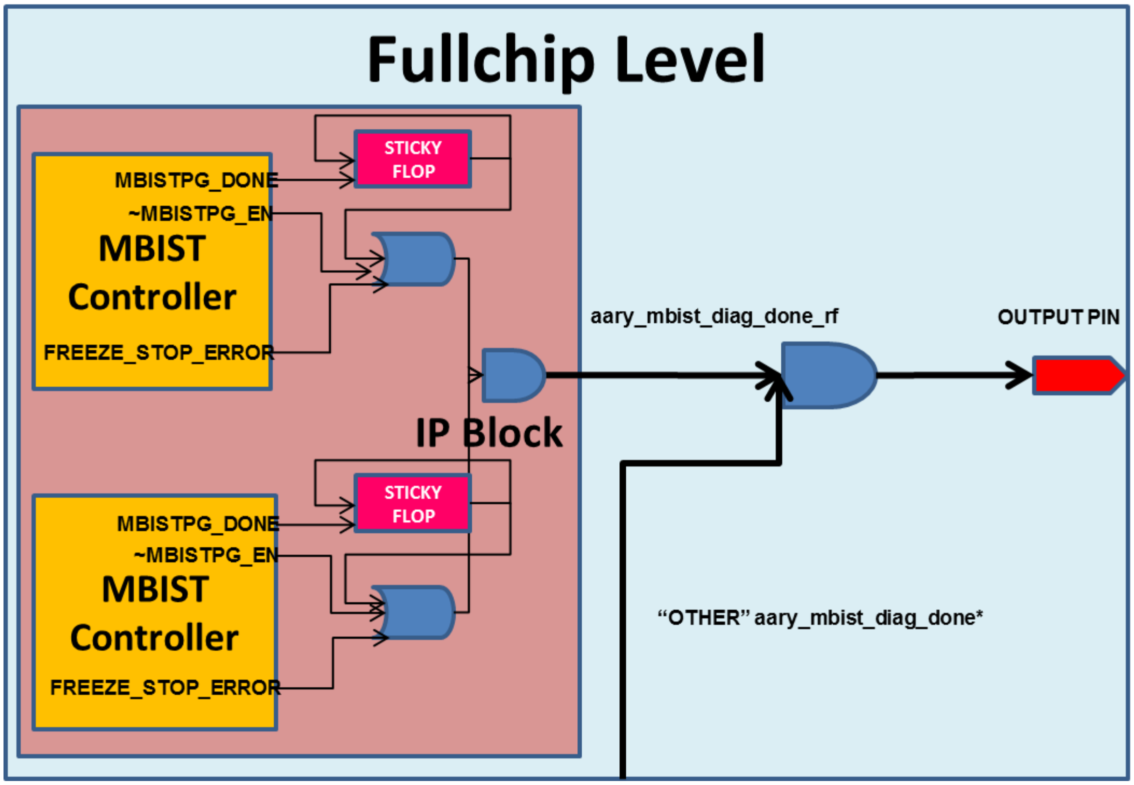
5.4 MBIST diagnostic results accumulation for HVM

To expedite HVM array test operations the MBIST wrapper contains logic within IP blocks to communicate when the controller has stopped due to its failure limit being reached. The output of each controller is accumulated in a reduction AND gate with other controllers within the IP-block. New soft IP-block developments based on this version of the spec are required to implement the signal interface. The names of the pins on the IP boundary will be aary_mbist_diag_done_rf for RF controllers and aary_mbist_diag_done_sram for SRAM controllers. With this hardware in place, products can choose to AND together all IP level *mbist_diag_done* signals and bring them out to a chip pin where they can be monitored by the tester. The benefit is realized by the test program when the chip level pin is asserted high indicating all controllers have stopped allowing it to move to the next phase of the test suite without having to wait a predefined wait time. The next phase usually consists of dumping out of the controller failure data to TDO to analyze which arrays have defects. Depending on the algorithm being run and where the array defects are located, this can result in approx. a 50% reduction in diagnostic test time per array defect collected, which is substantial especially if hundreds of defects are being collected.

Figure 5-3 is a high level block diagram showing how the MBIST wrapper logic works. During the first diagnostic pass (failure limit set to 1), the pattern does a full run using the predefined pattern wait time. It then scans out data for any controllers that have stopped on the first failure. In the next pass, controllers without failures in the first pass will have their “DONE” status captured by a sticky flop which will contribute a “1” to the AND gate driving the IP level *mbist_diag_done* port. The result is that these controllers will no longer gate the chip level output pin being monitored. For controllers that still have failures, their mbist_diag_done signals will go high as soon as the controller stops at the desired failure count. When all other controllers have stopped at their failure limits, the chip level output pin will go high and all controller failure data will be scanned out. This process then repeats multiple times until all failures have been shifted out or all controllers have reached the DONE state. By synchronizing the controller scanout in this manner, we ensure that all controllers are ready to be scanned out (stopped and not still running) as soon as the chip level pin being monitored goes high.

Note: This requirement targets new SoC developments.

Figure 5-3. MBIST results accumulation block diagram



5.5 Array Signal Interface Description

A graphical view of the array test interface is divided among the three memory types and shown in **Error! Reference source not found.** through Figure 5-7. A more detailed signal description is listed in Table 5-2 through Table 5-5. The integration team has the freedom to assign any reasonable number of characters as a prefix or suffix as applied to the root name to identify the signal in a full chip SoC.

Figure 5-4. Register file array test support signals

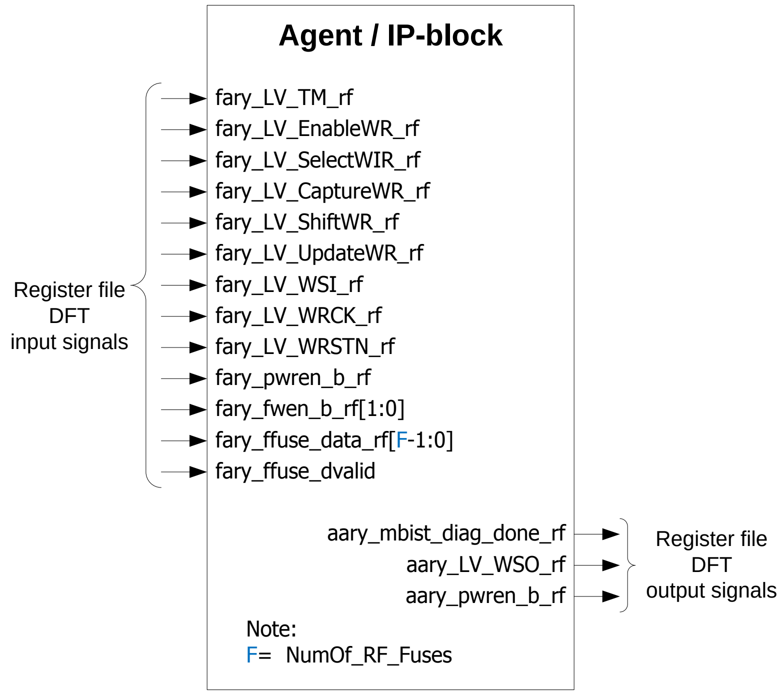


Figure 5-5. SRAM array support signals continued

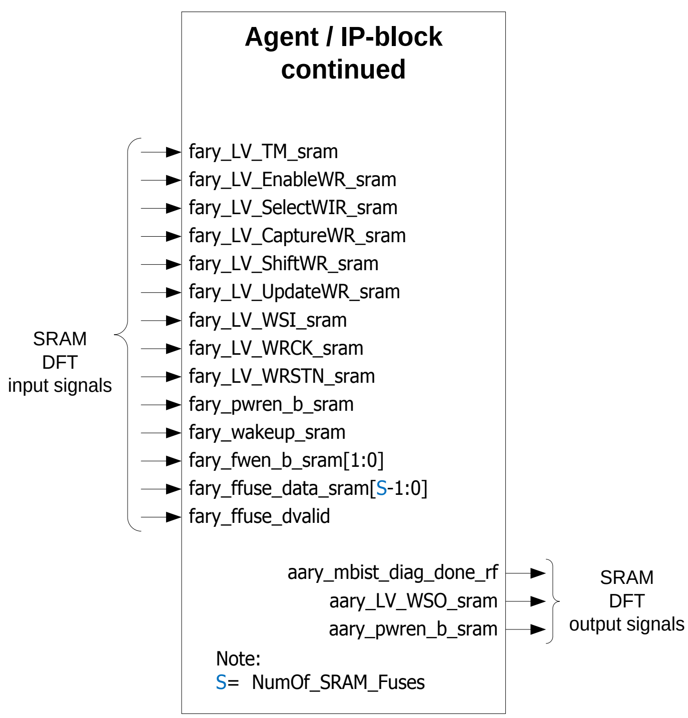


Figure 5-6. Read-only memory (ROM) array support signals continued

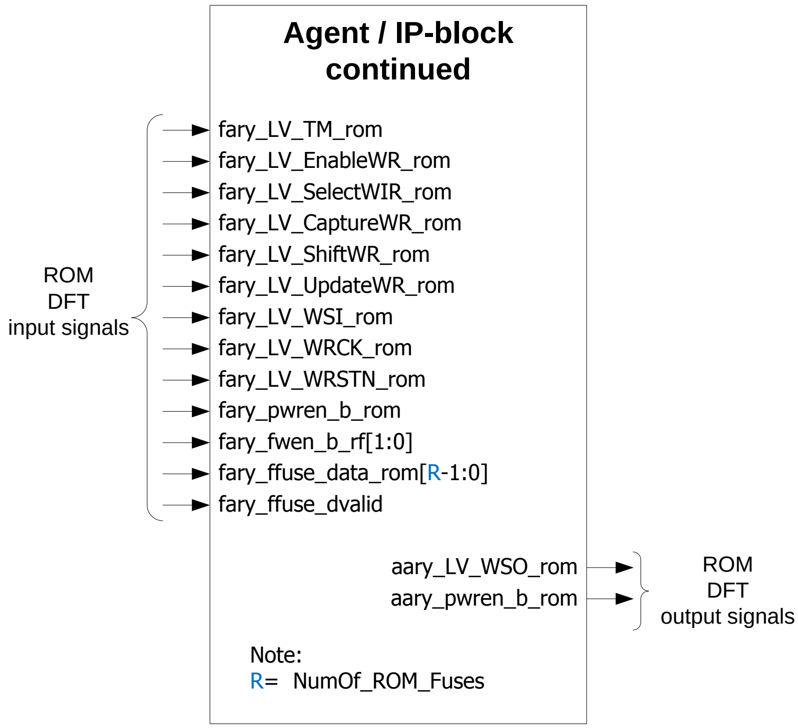


Figure 5-7. LYA and miscellaneous array test signal support

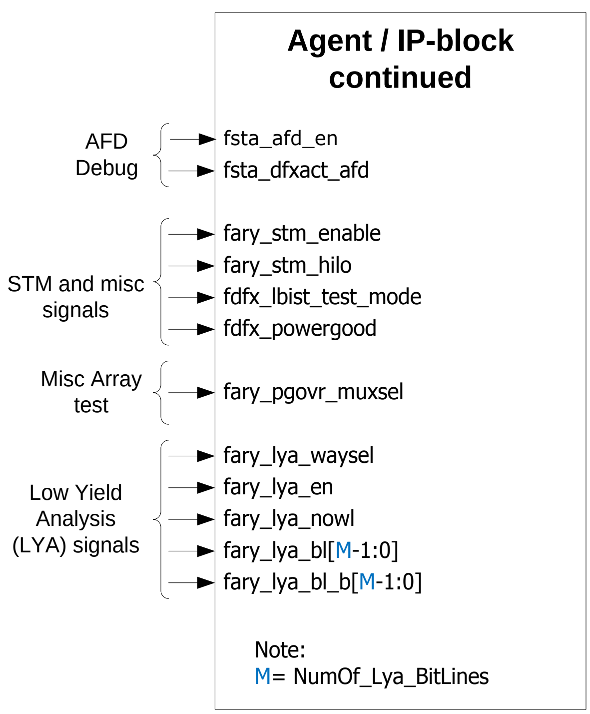


Table 5-2. Register file array test signal table

Signal	I/O	R/C[1]	Description
MBIST test signals for register files			
fary_LV_TM_rf	I	C	**Fabric array Logic Vision (MBIST) Test Mode for RFs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal informs the BIST collar that is in test mode for register file associated at the top level of the agent (IP-block). Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_LV_EnableWR_rf	I	C	**Fabric array Logic Vision (MBIST) Enable WTAP Register for RFs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal enables the embedded WTAP register in the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_LV_SelectWIR_rf	I	C	**Fabric array Logic Vision (MBIST) Select WTAP IR for RFs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_LV_CaptureWR_rf	I	C	**Fabric array Logic Vision (MBIST) Capture WTAP register for RFs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_LV_ShiftWR_rf	I	C	**Fabric array Logic Vision (MBIST) Shift WTAP register for RFs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_LV_UpdateWR_rf	I	C	**Fabric array Logic Vision (MBIST) Update WTAP register for RFs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.

Signal	I/O	R/C[1]	Description
fary_LV_WSI_rf	I	C	**Fabric array Logic Vision (MBIST) WTAP serial input for RFs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_LV_WRCK_rf	I	C	**Fabric array Logic Vision (MBIST) WTAP clock for RFs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_LV_WRSTN_rf	I	C	**Fabric array Logic Vision (MBIST) WTAP reset (bar) for RFs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
aary_LV_WSO_rf	O	C	**Fabric array Logic Vision (MBIST) WTAP serial output for RFs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_pwren_b_rf	I	C	**Fabric array power enable bar for RFs.** This signal enables the power for array test operations.
aary_pwren_b_rf	O	C	**Agent array power enable bar for RFs.** This signal enables the power for array test operations. This signal is available as an output so the integration team can serially stitch the control signal for several similar arrays.
fary_fwen_b_rf [1:0]	I	C	**Fabric array firewall enable bar for RFs.** This signal is active low and it forces the firewall to be enabled. [0]: Enables the firewall all the inputs to EBB which includes data in, address, enables, STM i/ps, clock, lya* signals,wakeup_ms01h,clock etc. [1]: Enables firewall the data out from EBB
aary_mbist_diag_done _rf	O	C	**Agent array MBIST diagnostic done signal for RF type.** This signal indicates that either the MBIST controller is done or an error occurred such that the controller stopped.
fary_ffuse_data_rf [NumOf_RF_Fuses-1:0]	I	C	**Fabric array for RFs using IOSF fuse data bus.** This signal bus is re-used from IOSF fuse data bus defined in the miscellaneous DFx signal interface. The fary_ffuse_data_rf signal bus is generic and assigned to the FUSE_MISC_RF_IN signal bus on the memory. If an agent/IP-block has no RF memories then the parameter is set to NumOf_RF_Fuses =1.

Signal	I/O	R/C[1]	Description
fary_ffuse_dvalid	I	C	**Fabric array for RFs using IOSF fuse data valid.** This signal indicates that the fuse data is valid. There is only one fuse data valid per agent/IP-block.
[1]Note1: R = required, C = conditional. If an IP-block has arrays, FIFOs, ROMs or SRAM memories then these signals are required.			

Table 5-3. SRAM Array test signal table

Signal	I/O	R/C[1]	Description
MBIST test signals for SRAMs			
fary_LV_TM_sram	I	C	**Fabric array Logic Vision (MBIST) Test Mode for SRAMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal informs the BIST collar that is in test mode for register file associated at the top level of the agent (IP-block). Note: If an agent/IP-block requires an SRAM then this signal is required.
fary_LV_EnableWR_ sram	I	C	**Fabric array Logic Vision (MBIST) Enable WTAP Register for SRAMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal enables the embedded WTAP register in the BIST collar. Note: If an agent/IP-block requires an SRAM then this signal is required.
fary_LV_SelectWIR_ sram	I	C	**Fabric array Logic Vision (MBIST) Select WTAP IR for SRAMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires an SRAM then this signal is required.
fary_LV_CaptureWR_ sram	I	C	**Fabric array Logic Vision (MBIST) Capture WTAP register for SRAMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires an SRAM then this signal is required.
fary_LV_ShiftWR_ sram	I	C	**Fabric array Logic Vision (MBIST) Shift WTAP register for SRAMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. **Note:** If an agent/IP-block requires an SRAM then this signal is required.

Signal	I/O	R/C[1]	Description
fary_LV_UpdateWR_ sram	I	C	**Fabric array Logic Vision (MBIST) Update WTAP register for SRAMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires an SRAM then this signal is required.
fary_LV_WSI_ sram	I	C	**Fabric array Logic Vision (MBIST) WTAP serial input for SRAMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires an SRAM then this signal is required.
fary_LV_WRCK_ sram	I	C	**Fabric array Logic Vision (MBIST) WTAP clock for SRAMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires an SRAM then this signal is required.
fary_LV_WRSTN_ sram	I	C	**Fabric array Logic Vision (MBIST) WTAP reset (bar) for SRAMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires an SRAM then this signal is required.
aary_LV_WSO_ sram	O	C	**Fabric array Logic Vision (MBIST) WTAP serial output for SRAMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires an SRAM then this signal is required.
fary_pwren_b_sram	I	C	**Fabric array power enable bar for SRAMs.** This signal enables the power for array test operations.
aary_pwren_b_sram	O	C	**Agent array power enable bar for SRAMs.** This signal enables the power for array test operations. This signal is available as an output so the integration team can serially stitch the control signal for several similar arrays.
fary_wakeup_sram	I	C	**Fabric wakeup for SRAM.** This signal forces the memory to wake up from a sleep mode. 0: Memory goes into sleep mode with many power saving features enabled. 1: Assertion of on this signal causes the SRAM to come out of sleep and get ready for access.

Signal	I/O	R/C[1]	Description
fary_fwen_b_sram [1:0]	I	C	**Fabric array firewall enable bar for SRAM.** This signal is active low and it forces the firewall to be enabled. [0]: Enables the firewall all the inputs to EBB which includes data in, address, enables, STM i/ps, clock, lya* signals,wakeup_ms01h,clock etc. [1]: Enables firewall the data out from EBB
aary_mbist_diag_ done_sram	O	C	**Agent array MBIST diagnostic done signal for RF type.** This signal indicates that either the MBIST controller is done or an error occurred such that the controller stopped
fary_ffuse_data_sram [NumOf_SRAM_Fuses - 1:0]	I	C	**Fabric array for SRAMs using IOSF fuse data bus.** This signal bus is re-used from IOSF fuse data bus defined in the miscellaneous DFx signal interface. The fary_ffuse_data_sram signal bus is generic and assigned to the FUSE_MISC_SSA_IN signal bus on the memory. If an agent/IP-block has no RF memories then the parameter is set to NumOf_SRAM_Fuses =1.
fary_ffuse_dvalid	I	C	**Fabric array for RFs using IOSF fuse data valid.** This signal indicates that the fuse data is valid. There is only one fuse data valid per agent/IP-block.

[1]Note1: R = required, C = conditional. If an IP-block has arrays, FIFOs, ROMs or SRAM memories then these signals are required.

Table 5-4. Read-only memory (ROM) array test signal table

Signal	I/O	R/C[1]	Description
MBIST test signals for ROMs			
fary_LV_TM_rom	I	C	**Fabric array Logic Vision (MBIST) Test Mode for ROMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal informs the BIST collar that is in test mode for register file associated at the top level of the agent (IP-block). Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_LV_EnableWR_ro m	I	C	**Fabric array Logic Vision (MBIST) Enable WTAP Register for ROMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal enables the embedded WTAP register in the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_LV_SelectWIR_ro m	I	C	**Fabric array Logic Vision (MBIST) Select WTAP IR for ROMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.

Signal	I/O	R/C[1]	Description
fary_LV_CaptureWR_rom	I	C	**Fabric array Logic Vision (MBIST) Capture WTAP register for ROMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_LV_ShiftWR_rom	I	C	**Fabric array Logic Vision (MBIST) Shift WTAP register for ROMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_LV_UpdateWR_rom	I	C	**Fabric array Logic Vision (MBIST) Update WTAP register for ROMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_LV_WSI_rom	I	C	**Fabric array Logic Vision (MBIST) WTAP serial input for ROMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_LV_WRCK_rom	I	C	**Fabric array Logic Vision (MBIST) WTAP clock for ROMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_LV_WRSTN_rom	I	C	**Fabric array Logic Vision (MBIST) WTAP reset (bar) for ROMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
aary_LV_WSO_rom	O	C	**Fabric array Logic Vision (MBIST) WTAP serial output for ROMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_pwren_b_rom	I	C	**Fabric array power enable bar for ROMs.** This signal enables the power for array test operations.

Signal	I/O	R/C[1]	Description
aary_pwren_b_rom	O	C	**Agent array power enable bar for ROMs.** This signal enables the power for array test operations. This signal is available as an output so the integration team can serially stitch the control signal for several similar arrays.
fary_fwen_b_rom [1:0]	I	C	**Fabric array firewall enable bar for ROMs.** This signal is active low and it forces the firewall to be enabled. [0]: Enables the firewall all the inputs to EBB which includes data in, address, enables, STM i/ps, clock, lya* signals,wakeup_ms01h,clock etc. [1]: Enables firewall the data out from EBB
fary_ffuse_data_rom [NumOf_RF_Fuses - 1:0]	I	C	**Fabric array for ROMs using IOSF fuse data bus.** This signal bus is re-used from IOSF fuse data bus defined in the miscellaneous DFx signal interface. The fary_ffuse_data_rom signal bus is generic and assigned to the FUSE_MISC_ROM_IN signal bus on the memory. If an agent/IP-block has no RF memories then the parameter is set to NumOf_ROM_Fuses =1.
fary_ffuse_dvalid	I	C	**Fabric array for RFs using IOSF fuse data valid.** This signal indicates that the fuse data is valid. There is only one fuse data valid per agent/IP-block.
[1]Note1: R = required, C = conditional. If an IP-block has arrays, FIFOs, ROMs or SRAM memories then these signals are required.			

Table 5-5. Miscellaneous array test signal table

Signal	I/O	R/O/C[1]	Description
Low Yield Analysis (LYA) DFT signals			
fary_lya_waysel	I	C	**Fabric array, LYA signal group, slice select.** This signal select a LYA data slice. fary_lya_waysel connect to: lyawaysel_ms00h internally.
fary_lya_dataen	I	C	**Fabric array, LYA signal group, data enable.** This signal is the LYA data enable. fary_lya_dataen connects to: lyaen_ms00h internally.
fary_lya_nowl	I	C	**Fabric array, LYA signal group, no write line.** This signal indicates a read or write to a line of memory cells. 0: write 1: read fary_lya_nowl connects to: lyanowl_ms00h internally.
fary_lya_nowrysel	I	C	**Fabric array, LYA signal group, no write ysel.** This signal indicates a no write with 'y' select. fary_lay_nowrysel connects to lyanowrysel_ms00h internally.
fary_lya_bl[M-1:0]	I	C	**Fabric array, LYA signal group bit line input.** This signal is used for low yield analysis of the arrays. It is parameterized for M number of inputs. The minimum and default is set to 2. fary_lya_bl[1:0] connects to: lyabl_ms00l [1:0]

fary_lya_bl_b[M-1:0]	I	C	**Fabric low yield analysis bit line bar input.** This signal is used for low yield analysis of the arrays. It is parameterized for M number of inputs. It is parameterized for M number of inputs. The minimum and default is set to 2. fary_lya_bl_b[1:0] connects to: lyabl_ms00_bl [1:0]
Miscellaneous array DFT signals			
fary_pgovr_muxsel	I	C	**Fabric array power gate override mux select.** This signal overrides the internal mux to selects between the internal power gate controls to the arrays/RF/ROM and the TAP test data register. **0:** Normal operation, internal power gate controls **1:** Force EBBs to use TAP TDR bits to drive values
fary_stm_enable	I	C	**Fabric STM enable.** This signal enables the Stability Test Mode (STM) for arrays. 0: STM disabled 1: STM enabled and block the normal write driver signal.
fary_stm_hilo	I	C	**Fabric STB high/low value.** This signal determines if the bitline or bit line bar is floating to the SRAM. 0: Bitline_b is floating 1: Bitline is floating
fdfx_lbist_test_mode	I	C	**Fabric DFx LBIST test mode:** This signal supports LBIST operations in the presence of arrays. This signal is conditional based on the SoC decision to use the LBIST flow.
fdfx_powergood	I	C	Refer to section 6.2. Note: This signal is equivalent to **dfx_powergood_rst_b**.
Array debug signals			
fsta_dfxact_afd	I	C	**Fabric SoC trigger arch DFx action for array/freeze/dump:** This signal forces the array's (or register file's) DFx mux inputs to select the BIST engine to extract it contents. 0: Normal array (RF) operation. The functional path to the array is active. 1: Force the DFx mux to select the MBIST. Note: This signal is sourced from the Cluster Trigger Block in the regional DFx unit.
fsta_afd_en	I	C	**Fabric SoC trigger arch DFx action enable:** This signal allows the silicon validation team to select which arrays can be enabled for AFD. This signal is connected to a cluster DFx unit TAP bit.
NOTE: [1]R = required, O = optional, C = conditional. Conditional means that if the feature is used then the signals labeled as conditional become required.			

1

5.6 Transaction Cycle/Data Flows

Not applicable

5.7 Ordering/Coherency Rules

Not applicable

5.8 Performance/Bandwidth Analysis

Not applicable

5.9 Exception List Requirements

Not applicable

5.10 Programming Model

Refer to the MBIST Flow and Debug Handbook rev200.

Link:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC%20DFx/Forms/AllItems.aspx

Directory:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC DFx/DFx Methodologies (MBIST, LBIST, VISA, etc.)/Memory BIST/10_Spec Releases/Rev200

5.11 Power Management Capabilities

Not applicable

5.12 Security Feature Requirements

Access is controlled through the TAP network with the embedded DFx secure policy plug-in IP-block.

5.13 DFx Requirements

The SoC DFT Handbook should be referenced a general overview of SoC test methodologies.

Link:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC%20DFx/Forms/AllItems.aspx

Directory:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC DFx/DFx Overview Docs/SoC DFT Handbook

5.13.1 DFV/DFD Requirements

The MBIST array wrapper must include an Array/Freeze/Dump (AFD) module. Refer to the Chassis DFx HAS for more information.

5.13.2 DFT Requirements

Refer to the SoC DFT Handbook

5.13.2.1 Burn-in Specific Requirements

Refer to the SoC DFT Handbook

5.13.3 DFM Requirements

Refer to the SoC DFT Handbook

§

6 Miscellaneous DFT Interface

The miscellaneous test interface is a collection of HVM test control functions that are feature-specific and cannot be readily absorbed into other existing IOSF DFx interfaces (TAP, Scan, or DFV). This specification discourages projects from adding sideband wires that cannot be readily transferred between other SoC components or business groups. Every attempt has been made to include all known signals as a superset so that the DFx requirements for the agent are met.

Table 6-1. Chapter revision history

Revision Number	Description	Revision Date
rev1.2_rc1	• Initial release.	Jan 2012
rev1.2_rc2	• updated IDV	Feb 1, 2012
rev1.2_rc3	• Added the miscellaneous DFx trigger action bus	Feb 13, 2012
rev1.2_rc4b	• Updated and edited text where needed	March 10, 2012
rev1.2_rc7b	• Updated the parameter names of the ftc_data and atc_data signals since they were named incorrectly. — OLD: [TCOWIDTH-1:0], NEW: [TestCtrlIn_WIDTH-1:0] — OLD: [TCIWIDTH-1:0], NEW: [TestCtrlOut_WIDTH-1:0]	July 13, 2012
rev1.2 (final)	• Added the *_LV_TM test mode signal for LBIST support	July 20, 2012
rev1.2.2_rc1	• Moved the PGCB DFx section here since it is for DFT override and control. It was originally for debug and HVM but the debug features were dismissed over time as unworkable with the development of autonomous power gating IPs. • Separated the signal description table into several sub-tables. • Removed fdfx_rst_b from this section. It only applies to the DFD section and only to the VISA ULM.	February 7, 2014

6.1.1 Boundary Scan Support

The most common use model for the boundary scan implementation would be the component's Chip-Level TAP providing the control wires necessary for the boundary scan chain. Soft-IP or hard-IP agents that support boundary scan must provide the signals listed in Table 6-2. Usually, this chain is stitched through the hard-IP agents (IO physical layers) as an abutment in the IO pad ring. It is difficult to foresee every possible integration scenario, so the IOSF specification defines the boundary scan interface and allows the fabric to complete the stitch path back to the master TAP if necessary. Refer to the IEEE1149.1, IEEE1149.6, and the SoC TAP HAS rev0.88 or later for more information about the usage model for the boundary scan interface.

Table 6-2 shows the relationship between the control signal assertions and the most commonly used boundary scan instructions. This table is only for reference. Refer to the previously mentioned specification source for more information. Note that the common control signals such as, *_tck, *_tdi, *_capturedr, *_shiftdr, *_updatedr, *_tdo are not listed in the table below since they are better explained in the IEEE1149.1 specification.

Table 6-2. Boundary scan test mode control signals

Instruction	Boundary Scan Test Mode Signals[1]								
	mode	intest _mode	highz	chainen	extogen	extogsig_b	d6select	d6init	d6actestsig _b
SAMPLE/ PRELOAD	0	0	0	1	0	1	0	0	1
EXTEST	1	0	0	1	0	1	0	see note 4	1
EXTEST_ TOGGLE	1	0	0	1	1	toggling[3]	1	pulse[2]	toggling[3]
EXTEST_ TRAIN	1	0	0	1	0	1	1	pulse[2]	toggle
EXTEST_ PULSE	1	0	0	1	0	1	1	pulse[2]	pulse
CLAMP	1	0	0	1	0	1	0	0	1
HIGHZ	1	0	1	1	0	1	0	0	1
INTEST	0	1	0	1	0	1	0	0	1
BYPASS	0	0	0	0	0	1	0	0	1

Notes:

[1]Note: The "fbscan_" was removed to reduce the number of characters in the header cells. The mode, highz, chainen and extogen must be deglitched in the mTAP (or controlling sTAP). The reader should refer to the SoC TAP HAS rev0.88 or later for detailed information about the boundary-scan implementation.

[2]Note: The fbscan_d6init is pulsed with TCK in the Exit1-DR or Exit2-DR states

[3]Note: Toggles at the falling edge of TCK in Run-Test/Idle

[4]Note: It is product dependent whether the d6init signal is asserted for EXTEST. The 1149.6 (page 59) states "This standard only mandates the initialization of the hysteresis for EXTEST, EXTEST_PULSE, or EXTEST_TRAIN instructions".

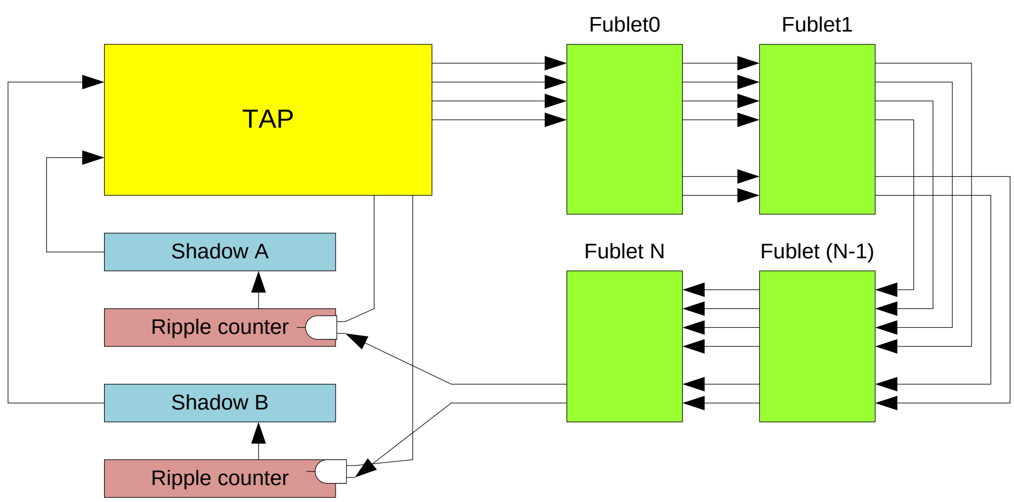
Rule 6.2.2.1 (d). Whenever a test receiver is operating in the level-detection mode on an AC input pin, the test receiver output shall be cleared of prior history on the falling edge of TCK in the Capture-DR TAP Controller state.

6.1.2 IDV Test Data Register Support

The Inter-Die Variation (IDV) devices are composed of ring oscillators constructed with specific transistor characteristics to reveal process variations from either typical silicon or purposely skewed silicon that is expected for a particular wafer lot. This information can be useful during debug to focus on areas of potential speed path timing problems and related issues. An IDV "fublet" contains a set of ring oscillators with each one design to measure a particular parametric variation. It is controlled with test data register to select which oscillator chain is active. Only one fublet can be active at a time because all of the devices are stitched serially together to form a long chain within the component. The fublets are placed at regular intervals across the SoC, and any particular IOSF agent may or may not have an instance of them. If an agent/IP-block is harden and delivered to the SoC integration without the ability to re-synthesize then the agent must provide the IDV signals on its interface as listed in Table 6-2.

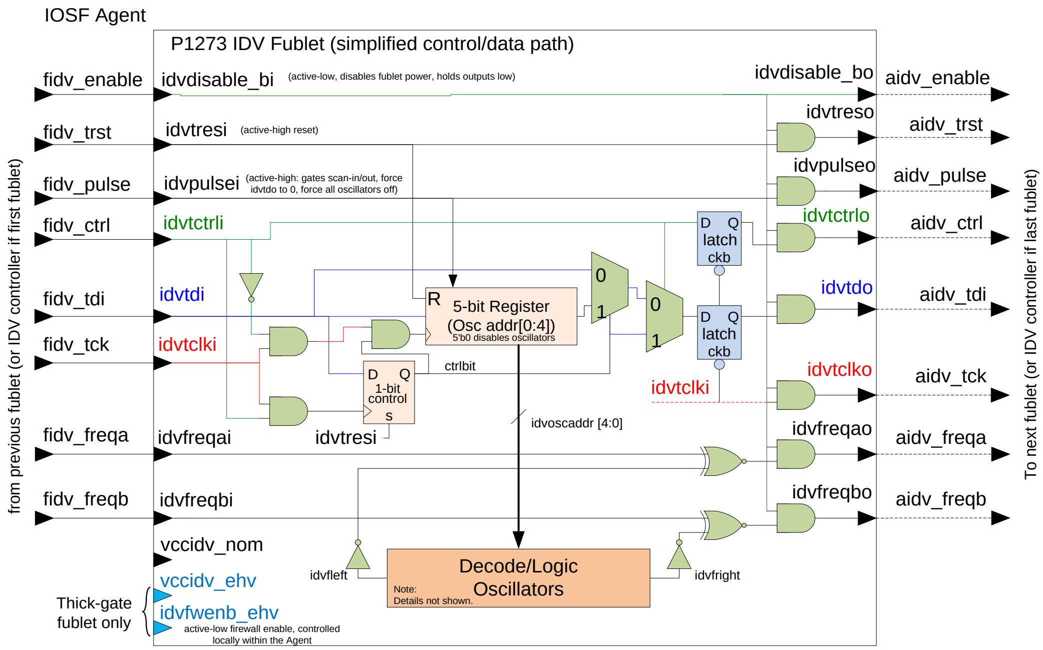
Figure 6-1 is a high level block diagram describes the overall system of IDV "fublets" connected in series to a controller with a set of counters that is interfaced to a slave TAP. The IDV controller may have one or two ripple counters that divide the frequency down coming from a selected fublet. A previous validated range of the measured frequency count indicates local skew and other process information from a particular ring oscillator within a selected fublet. Ripple counter depths are designed for the frequency of operation of the ring oscillators to provide usable signature values without saturation. Although the diagram doesn't show it a 14-bit counter its expected depth is 13 bits with a 1-bit sticky overflow. Two counters are required for the extended feature set. Other optional signals, shown in the signal list but not in the diagrams, are for other extensions of the IDV to include more capabilities that are controlled locally with a test data register. These are the standard capture, shift, and update control signals for the test data register. An event output provides an indication that an expected result has occurred. Refer to the IDV design specification for more information.

Figure 6-1. Component level IDV use model



Details of a typical IDV fublet are shown in Figure 6-2. This fublet uses only a subset of the IDV signals defined in Table 6-3. A 5-bit register is written serially to select a ring oscillator if the value is logic 0 then no oscillator is selected. It is important to only select one fublet at a time because the output of the ring oscillators are EXOR'd into the serial data stream. If the fublets are disabled (fidv_enable = 0) then the outputs are forced to logic 1 and the downstream fublets[N:1] will be in reset. The IDV controller must reset the first (fublet[0]) when the enable signal is de-asserted (logic 0),

Figure 6-2. Typical IDV fublet



6.1.3 Fuse Data Support

Data from fuse bits are included as part of the miscellaneous test bus because it is outside of the primary and sideband interface definitions, even though it has functional implications. The expected use model is to deliver fuse data on the sideband interface as messages with data to each of the agents in need of the content. Under this use model the sideband router fabric and its endpoints must be clocked with an external crystal or a 300 ppm accurate clock source with PLLs in bypass mode to operate the sideband fabric during reset. However, there are agents/IP-blocks without a sideband endpoint or hard-IP blocks that cannot make the transition yet will require a defined parallel interface for several more years.

For agents or IP-blocks that requirement configuration fuse configuration support and the IP-block cannot implement a fuse puller IP DFx then the agent must implement the parallel fuse inputs as listed in Table 6-2. In most cases, a fuse puller will be used to deliver the data to the agent.

6.1.4 High Volume Manufacturing Control Signals

The High Volume Manufacturing (HVM) control signal group provides direct control over those DFT features that are latency sensitive or require a global action that cannot be provided with individual JTAG TAP writes to test data registers.

6.1.4.1 Leakage Test Signals

This is an output status signal from the hard-IP agent’s leakage test methodology, commonly referred to as No-Touch Leakage (NTL) test. The purpose of this methodology is to charge (or discharge) a node, then disconnect the voltage rail source and allow the node to float. A resistor-capacitor (RC) decay relationship of that node will occur over a period of time. The more "leaky" the IO is, the faster it will decay. The physical layer has an operational amplifier (opAmp) comparator sensing this node against any voltage reference. A digital logic block will collect these outputs into a single status output (adft_leakstat) that is connected to a reused package pin. The tester will sample the pin when the logic value switches from a 0 to a 1 for a given time period and determine the pass/fail condition. The digital logic block in the IO contains an inverter (implemented as an exclusive-OR) as part of the overall Boolean function to generate a signal output. This inversion adjusts the leak to ground and leak to Vcc tests since the comparator only switches from a logic 0 to 1. Another test methodology, similar to NTL, is called Direct Test Leakage (DCL) for serial differential links. This methodology uses a pass transistor on the back side of a termination resistor so that the tester can apply a voltage and measure the decay directly with resistor ladder and a high impedance voltage probe. The output of the DCL is connected to the analog monitor port and not through the fabric to a miscellaneous IO pad.

All of the adft_leakstat signals are collected in the fabric and muxed to an agent signal named adft_leakres. Because there are other ways of determining the crossover point for the pass/fail, such as using internal counters to time the event, use of these DFx interface signals are optional.

6.1.5 Test Support Control Signals

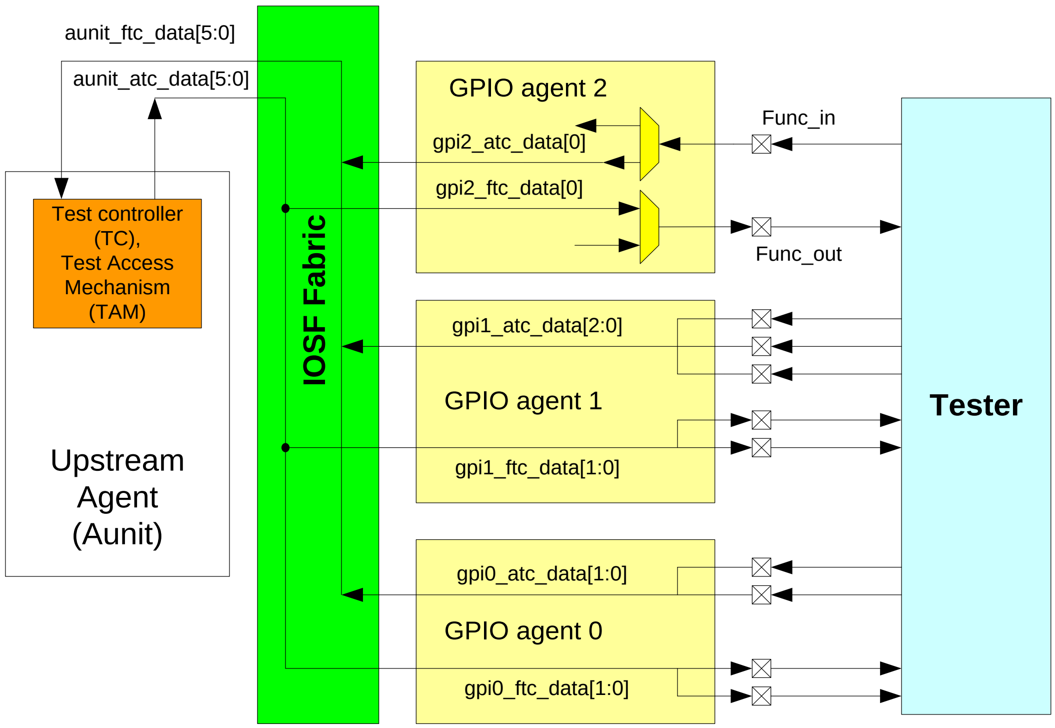
This section includes signals or buses to support delivering test content from the tester to an internal test controller, but they are not specific HVM control signals.

6.1.5.1 Functional Pin Reuse for Test Content Delivery

The Test Controller (TC) data bus (atc_data[N:0], ftc_data[N:0]) delivers test content from the tester into the Test Access Mechanism (TAM) through the IOSF agents and the fabric. Figure 6-3 is a block diagram that describes how the atc_data and ftc_data buses are gathered to form a test bus through the fabric to the TAM. The diagram shows a simplified General Purpose IO agent that contains a mux on its functional pins to redirect the input (or output) data from the physical layer to the agent boundary. Controlling the mux is accomplished with a TAP register bit setting. The other agents in the diagram are streamlined by showing only the wires, yet it is assumed that a mux structure similar to the GPIO agent 2 is implemented within each of these blocks. It is the integration team's responsibility to collect and route the atc_data / ftc_data signals in the fabric to and from the TAM.

Clock reference signals (atc_clk/ftc_clk) are optional for use by the agents or fabric to deliver the data from its clocking reference. How this may be used is agent-dependent. In general, this signal may be the same as the IOSF clock in order to reduce clock synchronization issues and potentially requiring asynchronous FIFOs in the component.

Figure 6-3. Functional pin reuse example block diagram



aunit_ftc_data[5:0] = {gpi2_atc_data[0], gpi1_atc_data[2:0], gpi0_atc_data[1:0]}

6.1.6 Access to Control and Status Registers for DFT

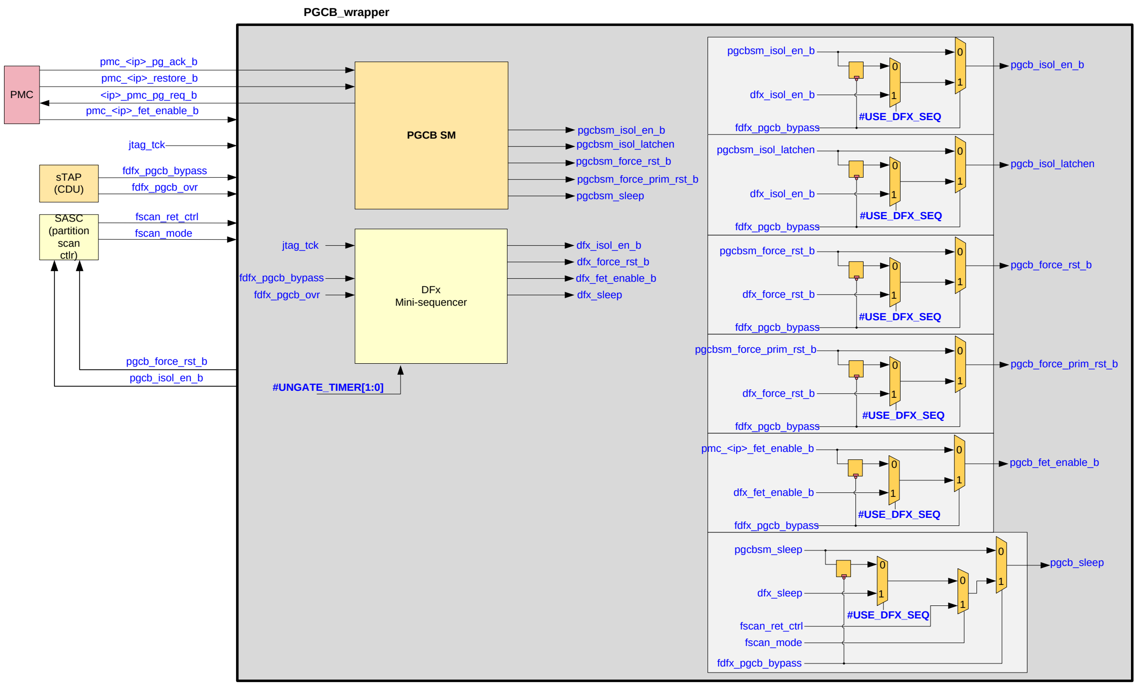
Generally, most test features are accessed with TAP data registers because of their ability to directly program the desired function without layers of protocols and it can be done during reset. However, for a function test support, these registers must be accessible on sideband with access from the TAP. For performance reasons another access point may be from a set of re-usable GPIO pins reading and writing config messages directly to sideband. A TAP to IOSF sideband interface feature is required somewhere in the SoC, refer to the SoC TAP HAS for details on this implementation.

6.2 Power Gate Common Block (PGCB) DFx interface signals

The Power Gate Common Block (PGCB) contains DFx that enables HVM testing by forcing the soft IP-block to force power up or down. The PGCB block diagram is shown in Figure 6-4. The reader should refer to the PGCB integration guide for more information.

If a soft IP-block implements the PGCB for autonomous internal power gating then the SIP must implement the fdfx_pgcb_bypass and fdfx_pgcb_ovr DFx signals.

Figure 6-4. PGCB DFx block diagram



6.3 Miscellaneous DFT Signal Interface Description

Figure 6-5 and Figure 6-6 shows a graphical view of the miscellaneous test and boundary scan control signals. Table 6-3 provides a more detailed description of the function, signal direction, signal name, and whether it’s required or not. The integration team has the freedom to assign any reasonable number of characters as a prefix or suffix as applied to the root name to identify the signal in a full chip SoC.

Figure 6-5. Miscellaneous test signal diagram

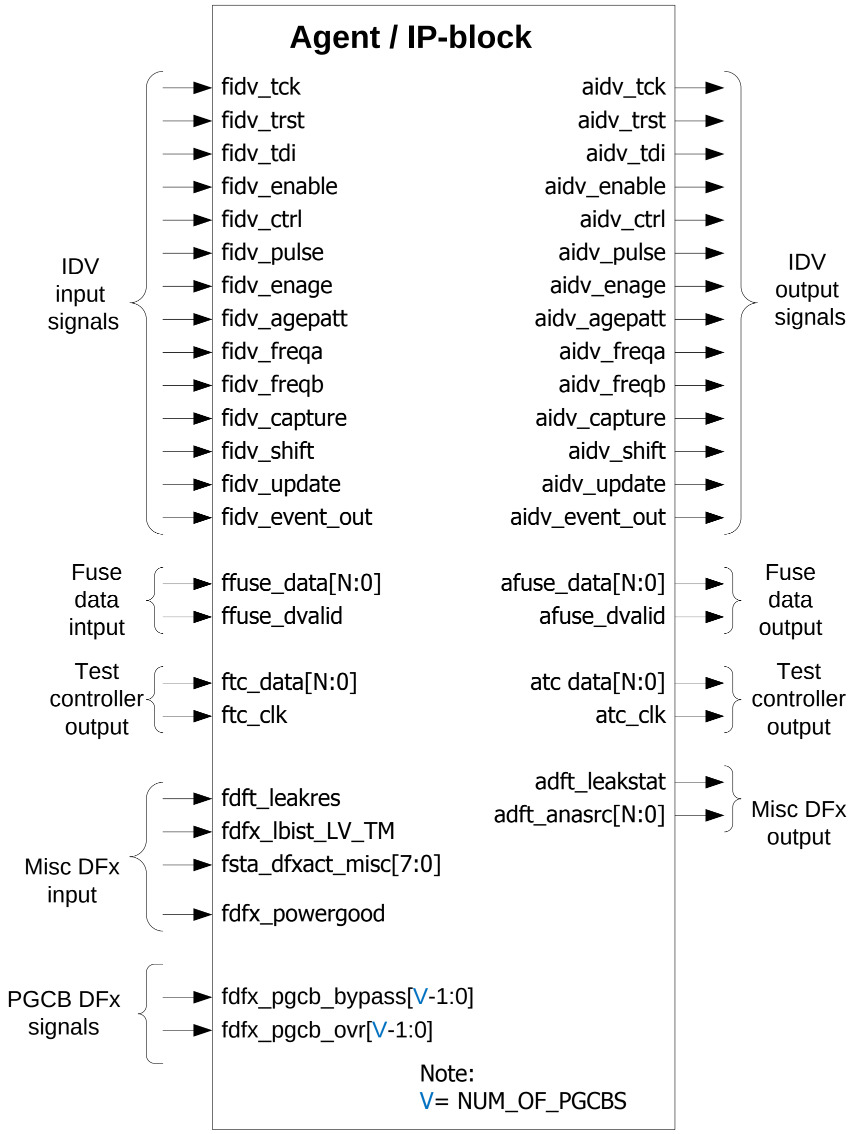


Figure 6-6. Boundary Scan control signals

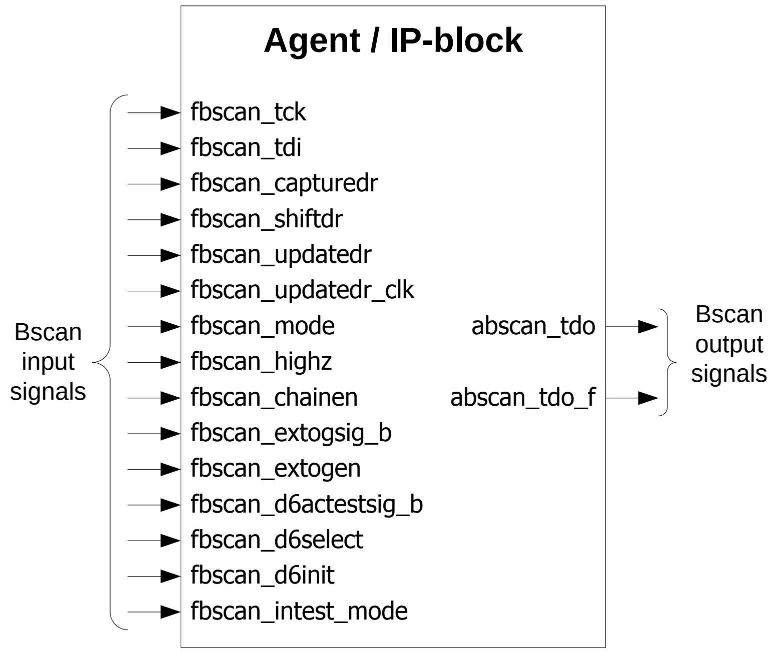


Table 6-3. Boundary scan support signals

Signal	I/O	R/O/C[1]	Description
fbscan_tck[2]	I	C	**Fabric Boundary Scan Test Clock Input.** This signal is the test clock input for *this* agent's section of the boundary scan chain stitched through the physical layer pins in the hard-IP macro. Generally, this version of the TCK signal is routed backwards with respect to the data (fbscan_tdi) signal. This fbscan_tck signal and the ftap_tck are eventually connected to the same package pin source. Its use is implementation-dependent.
fbscan_tdi	I	C	**Fabric Boundary Scan Test Data Input.** This signal is the test data input for *this* agent's section of the boundary scan chain stitched through the physical layer pins in the hard-IP macro.
fbscan_capturedr	I	C	**Fabric Boundary Scan Capture-DR Input.** This signal is the boundary scan decoded Capture-DR control signal for *this* agent's physical layer pins in the hard-IP macro.
fbscan_shiftdr	I	C	**Fabric Boundary Scan Shift-DR Input.** This signal is the boundary scan decoded Shift-DR control signal for *this* agent's physical layer pins in the hard-IP macro.
fbscan_updatedr	I	C	**Fabric Boundary Scan Update-DR Input.** This signal is the boundary scan decoded Update-DR control signal for *this* agent's physical layer pins in the hard-IP macro. There are two potential bscan cell designs; this signal supports the cell with a negative edge (TCK) clock enable flop. This signal is connected to the bscan cell's enable input.

Signal	I/O	R/O/C[1]	Description
fbscan_updatedr_clk	I	C	**Fabric Boundary Scan Update-DR Clock.** This signal is the boundary scan decoded Update-DR control signal for *this* agent's physical layer pins in the hard-IP macro. There are two potential bscan cell designs; this signal supports the cell with the updatedr directly connected to the bscan cell's clock input.
fbscan_mode	I	C	**Fabric Boundary Scan Mode Input.** This signal is the boundary scan decoded Mode control signal for *this* agent's physical layer pins in the hard-IP macro.
fbscan_highz	I	C	**Fabric Boundary Scan High-Z control.** This signal is the test data output for *this* agent's section of the boundary scan chain stitched through the physical layer pins in the hard-IP macro.
fbscan_chainen	I	C	**Fabric Boundary Scan Chain Enable.** This signal enables circuits in the physical layer to allow for boundary scan sample/preload operations to occur. It may be used to enable the proper pull ups/downs or to turn on sense amps. This signal is active when any boundary scan instruction is decoded in the master TAP.
fbscan_extogen	I	C	**Fabric Boundary Scan Extest Toggle Enable.** This signal enables the EXTEST_TOGGLE function when the instruction is valid. EXTEST_TOGGLE is a modified EXTEST boundary scan function that toggles bscan cell as an output at a period equal to the TCK frequency. This may be used for any IO type. For high speed serial IOs, this signal enables both the transmit (Tx) and receive (Rx) paths as an output for toggling logic values at the IO voltage level from the component to a test card.
fbscan_extogsig_b	I	C	**Fabric Boundary Scan Extest Toggle Signal bar.** This signal provides the toggling signal source when the EXTEST_TOGGLE instruction is enabled.
fbscan_d6init	I	C	**Fabric Boundary Scan "dot6" Initialization Signal.** This signal will initialize or clear the hysteresis memory (depending on the implement choice) in the IO circuit that captures detected values on the pins.
fbscan_d6select	I	C	**Fabric Boundary Scan "dot 6" Select:** This signal enables the test circuitry for dot6 test mode and indicates when either of the two train or pulse 1149.6 modes are active.
fbscan_d6actestsig_b	I	C	**Fabric Boundary Scan "dot 6" AC Test Signal bar:** This signal enables the 1149.6 test mode for use with IR instructions EXTEST_TRAIN and EXTEST_PULSE to perform the boundary scan test function for AC-coupled differential IOs.
fbscan_intest_mode	I	C	**Fabric Boundary Scan Intest Mode:** This signal enables an INTEST boundary-scan test mode. This is rarely used but available to SoCs.
abscan_tdo	O	C	**Agent Boundary Scan Test Data Output.** This signal is the test data output for *this* agent's section of the boundary scan chain stitched through the physical layer pins in the hard-IP macro.

Signal	I/O	R/O/C[1]	Description
abscan_tdo_f	O	C	**Agent Boundary Scan Test Data Output on Falling edge.** This signal is the test data output for *this* agent's section of the boundary scan chain. The output is asserted on the falling edge of TCK. If boundary scan is supported by this TAP then this signal is required.
fdfx_rst_b	I	C	**Fabric DFx reset bar:** This signal is sinked by the IP-block for resetting designated DFx logic. The signal is controlled within the DFx fabric, the Chassis reset unit and/or the Power Management Controller (PMC). It is the responsibility of the integration team to properly connect this signal and assert it for each IP. **Note:** Soft-IP blocks are required to include this signal on the IP interface and connect it to the VISA reset. If a hard-IP block is designated as a secure IO IP then it must implement this reset connecting to all VISA ULMs/PLMs within the IP. Otherwise, for all other hard IP-blocks it is optional.

NOTE:

[1]Note1: R = required, O = optional, C = Conditional. Conditional means that if the feature is used then the signals labeled as conditional become required. It may be obvious but if the IP-block supports boundary scan the boundary signals are required.

[2]Note2: The tck clock signals are expected to be synchronous and in phase with respect to the driving tck source. Usually, this is the primary TAP tck for most use models but may include the secondary TAP port tck signal. For example, a mux cannot be used to select between a TAP tck signal and the boundary scan tck signal within an agent.

Table 6-4. IDV control signals

Signal	I/O	R/O/C[1]	Description
fidv_tck[2]	I	C	**Fabric IDV TCK:** TAP clock from the previous IDV that originates from the master TAP or fabric TAP controller.
fidv_trst	I	C	**Fabric IDV TRST:** This is signal will reset the IDV fublet. 0: IDV normal operation 1: IDV will be in reset until the signal is logic 0.
fidv_tdi	I	C	**Fabric IDV TDI:** TAP data input.
fidv_enable	I	C	**Fabric IDV Enable:** This signal will enable the IDV fublet. The actual signal on the fublet is named as idvdisable_b but the functionality is the same. IOSF DFx spec strives to name signals with positive true logic except for active low resets.
fidv_ctrl	I	C	**Fabric IDV Control:** This signal enables a 1-bit flop datapath through the IDV.

fidv_pulse	I	C	**Fabric IDV Pulse:** This signal to gate scan in/out feature of IDV, thus disabling IDV and forcing the address to 0 and disabling all oscillators. 0: normal IDV operation. 1: When it is held high it will gate any scan in/out function and force the idvtdo output to 0. It will also force idvoscaddr[0] and idvoscaddr_b[0] low at the same time. This triggers the address decoder block to disable all the oscillators, regardless of the values of idvoscaddr[4:1] and idvoscaddr_b[4:1].
fidv_freqa	I	C	**Fabric IDV Previous Frequency Oscillator A:** Output from previous IDV oscillator is an input to this IDV. If this is the first IDV, then this input is grounded.
fidv_freqb	I	C	**Fabric IDV Previous Frequency Oscillator B:** Output from previous IDV oscillator is an input to this IDV. If this is the first IDV, then this input is grounded.
fidv_enage	I	O	**Fabric IDV Enable Aging:** This control signal will enable the aging oscillator feature of this IDV.
fidv_agepatt	I	O	**Fabric IDV Age Pattern:** This control signal will enable the age pattern logic in the IDV.
fidv_capture	I	O	**Fabric IDV Capture:** This control signal will capture status or shadow register data during the Capture-DR state. IDVs are serially linked together and this input signal is connected from the previous IDV in the chain. The purpose of this signal is for future extensions of the IDV controller where an internal TAP data register provides more sophisticated capabilities within the fublets.
fidv_shift	I	O	**Fabric IDV Shift:** This control signal will shift the TAP data register during the Shift-DR state. IDVs are serially linked together and this input signal is connected from the previous IDV in the chain. This signal is for future extensions of the IDV controller where an internal TAP data register provides more sophisticated capabilities within the fublets.
fidv_update	I	O	**Fabric IDV Update:** This control signal will update the shadow register with data from the shift register during the Update-DR state. IDVs are serially linked together and this input signal is connected from the previous IDV in the chain. This signal is for future extensions of the IDV controller where an internal TAP data register provides more sophisticated capabilities within the fublets.
fidv_event_out	I	O	**Fabric IDV Event Output:** An event output indicates expected results from a specific feature within the fublet. IDVs are serially linked together and this input signal is connected from the previous IDV in the chain. This signal is for future extensions of the IDV controller where an internal TAP data register provides more sophisticated capabilities within the fublets.
aidv_tck[2]	O	C	**Agent IDV TCK:** This control signal passes through to the next IDV or to the IDV controller. See fidv_* for the definition.
aidv_trst	O	C	**Agent IDV TRST:** IDV that originates from the IDV controller.

aidv_tdo	O	C	**Agent IDV TDO:** This control signal passes through to the next IDV or to the IDV controller. See fidv_* for the definition.
aidv_enable	O	C	**Agent IDV Enable:** This control signal passes through to the next IDV or to the IDV controller. See fidv_* for the definition.
aidv_ctrl	O	C	**Agent IDV Control:** This control signal passes through to the next IDV or to the IDV controller. See fidv_* for the definition
aidv_pulse	O	C	**Agent IDV pulse:** This control signal passes through to the next IDV or to the IDV controller. See fidv_* for the definition.
aidv_freqa	O	C	**Agent IDV Previous Frequency Oscillator A:** This control signal passes through to the next IDV or to the IDV controller. See fidv_* for the definition.
aidv_freqb	O	C	**Agent IDV Previous Frequency Oscillator B:** This control signal passes through to the next IDV or to the IDV controller. See fidv_* for the definition.
aidv_enage	O	O	**Agent IDV Enable Aging:** This control signal passes through to the next IDV or to the IDV controller. See fidv_* for the definition.
aidv_agepatt	O	O	**Agent IDV Age Pattern:** This control signal passes through to the next IDV or to the IDV controller. See fidv_* for the definition.
aidv_capture	O	O	**Agent IDV Capture:** This control signal passes through to the next IDV or to the IDV controller. See fidv_* for the definition.
aidv_shift	O	O	**Agent IDV Shift:** This control signal passes through to the next IDV or to the IDV controller. See fidv_* for the definition.
aidv_update	O	O	**Agent IDV Update:** This control signal passes through to the next IDV or to the IDV controller. See fidv_* for the definition.
aidv_event_out	O	O	**Agent IDV Event Output:** This control signal passes through to the next IDV or to the IDV controller. See fidv_* for the definition.

NOTE:

[1]Note1: R = required, O = optional, C = Conditional. Conditional means that if the feature is used then the signals labeled as conditional become required. It may be obvious but if the IP-block supports boundary scan the boundary signals are required.

1

Table 6-5. Parallel fuse data interface

Signal	I/O	R/O/C[1]	Description
ffuse_data [FFDWIDTH-1:0]	I	O	**Fabric Fuse Data[N:0]:** This bus delivers the fuse data from a centralized fuse controller through the fabric to an IOSF agent. This data is then used for specialized configuration control of features for circuits. For example, a PLL may have divider ratio settings that are fixed for a particular market segment. These values must be delivered before the deassertion of reset. Use of this bus implies a direct connection of the fuse values from the fuse block to this agent.
ffuse_dvalid	I	O	**Fabric Fuse Data Valid:** This signal indicates when the fuse values from the fabric to the IOSF agent are valid for reading.
afuse_data [AFDWIDTH-1:0]	O	O	**Agent Fuse Data[N:0]:** This bus delivers the fuse data from an IOSF agent to the fabric that is then distributed to other agents as needed. This agent may be a security block that contains the fuses rather than a DFx control block in the fabric.
afuse_dvalid	O	O	**Agent Fuse Data Valid:** This signal indicates when the fuse values from the agent to the fabric are valid for reading.
NOTE: [1]Note1: R = required, O = optional, C = Conditional. Conditional means that if the feature is used then the signals labeled as conditional become required. It may be obvious but if the IP-block supports boundary scan the boundary signals are required.			

Table 6-6. SoC level IO test control signals

Signal	I/O	R/O/C[1]	Description
adft_leakstat	O	O	**Agent DFT Leakage Status:** This is a source signal from this hard-IP agent that generated an output response from the leakage testing digital logic in the physical layer.
fdft_leakres	I	O	**Fabric DFT Leak Results:** This is the sink signal for the results of the leakage testing from one or more hard-IP agents on the fabric. This signal is intended for hard-IP agents to output the results from internal digital logic to the tester reusing general purpose IOs.
adft_anasrc [ANASRCWIDTH-1:0]	O	O	**Fabric DFT Analog Source[N:0]:** This is the source signal for analog from Small Signal Array (SSA) Low Yield Analysis testing. The fabric collects the signals from all of the agents and delivers them to a hard-IP agent with an analog pad. An analog pass-gate mux will select which of the agent’s output to select.

NOTE:
[1]Note1: R = required, O = optional, C = Conditional. Conditional means that if the feature is used then the signals labeled as conditional become required. It may be obvious but if the IP-block supports boundary scan the boundary signals are required.

Table 6-7. Test controller signals for content transport to/from IP-blocks

Signal	I/O	R/O/C[1]	Description
atc_clk	O	O	**Agent Test Controller Clock:** This is a reference clock that may be used optionally by the IOSF TAM Test Controller to deliver data sent by the TAM.
atc_data [TestCtrlOut_WIDTH-1:0]	O	O	**Agent Test Controller Data [N:0]:** This signal bus provides the test controller with data streaming from the tester. The fabric will gather all of the available atc_data[N:0] signals and deliver them as a single bus to the IOSF TAM test controller. Any agent (presumed to be a hard-IP agent) that can provide a bypass capability on its IOs is a candidate to participate in this bus.
ftc_clk	I	O	**Fabric Test Controller Clock:** This is a reference clock that may be used optionally by the IOSF TAM Test Controller to deliver data sent by the TAM.
ftc_data [TestCtrlIn_WIDTH-1:0]	I	O	**Fabric Test Controller Data [N:0]:** This signal bus provides the tester with data streaming from the IOSF TAM test controller (TAM-TC). The data content from test controller is usually in the form of compare results destined to the tester for pass/fail determination. Any agent (presumed to be a hard-IP agent) that can provide a bypass capability on its IOs is a candidate to participate in this bus.

NOTE:
[1]Note1: R = required, O = optional, C = Conditional. Conditional means that if the feature is used then the signals labeled as conditional become required. It may be obvious but if the IP-block supports boundary scan the boundary signals are required.

Table 6-8. Miscellaneous DFx support signals

Signal	I/O	R/O/C[1]	Description
fdfx_lbist_LV_TM	I	O	**Fabric DFx LBIST Logic Vision Test Mode:** This is the LV specific test mode signal from the LV TAP for use the LBIST controller logic.

fsta_dfxact_misc[7:0]	I	O	**Fabric SoC Trigger Arch DFx action for miscellaneous actions:** This signal group is available for the SoC integration team to implement an IP-block specific action. 0: Normal operation. 1: Force the assigned miscellaneous DFx action.
fdfx_pgcb_bypass [NUM_OF_PGCBS-1:0]	I	C	**Fabric DFx PGCB bypass.** This signal controls the Power Gate Common Block's bypass muxes to enable a DFx override signal to control the enabling of this IP-block's PGCB instantiation. **0:** Normal PGCB operation **1:** PGCB is bypassed and forces the override value **Note:** Refer to section 6.2 for more information.
fdfx_pgcb_ovr [NUM_OF_PGCBS-1:0]	I	C	**Fabric DFx PGCB override (value).** This signal controls the DFx sequencer inside of the PGCB block. The DFx sequencer automates the activation/deactivation of the power management control signals to power up or down the domain that this PGCB controls. **0:** Power gate device (PGD) is force on, meaning, the IP-block will be powered up **1:** Power gate device (PGD) is force off, meaning, the IP-block will be powered down **Note:** Refer to section 6.2 for more information.
fdfx_powergood	I	R	**Fabric DFx power good:** The DFx power good signal is required by all agents/IP-blocks that have DFx associated with them. Note: This signal is equivalent to **dfx_powergood_rst_b.**
NOTE: [1]Note1: R = required, O = optional, C = Conditional. Conditional means that if the feature is used then the signals labeled as conditional become required. It may be obvious but if the IP-block supports boundary scan the boundary signals are required.			

6.4 Transaction Cycle/Data Flows

Not applicable

6.5 Ordering/Coherency Rules

Not applicable

6.6 Performance/Bandwidth Analysis

Not applicable

6.7 Exception List Requirements

Not applicable

6.8 Programming Model

Not applicable

6.9 Power Management Capabilities

Not applicable

6.10 Security Feature Requirements

Boundary scan is expected to be a customer visible feature in most SoC market segments. This would be considered a DFx green level of access using the color coding example to describe it. A green level means only those DFx features that should be available to customers at all times. The DFx secure policy value of 0x0 (0000b) which is a "Security Locked" is the access level for an SoC operating in a normal functional mode.

6.11 DFx Requirements

The SoC DFT Handbook should be referenced a general overview of SoC test methodologies.

Link:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC%20DFx/Forms/AllItems.aspx

Directory:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC DFx/DFx Overview Docs/SoC DFT Handbook

6.11.1 DFV/DFD Requirements

Refer to the SoC DFT Handbook.

6.11.2 DFT Requirements

Refer to the SoC DFT Handbook.

6.11.2.1 Burn-in Specific Requirements

Refer to the SoC DFT Handbook.

6.11.3 DFM Requirements

Refer to the SoC DFT Handbook.

§

7 Debug and Validation (DFV)

This section describes the Design For Validation (DFV) debug and validation features for IOSF agents. This section will introduce the high-level concepts, define each sub-interface, and finally describe a framework to show how the sub-interfaces work together for an overall debug and validation methodology. Because of this, it is expected that SoCs implement the Lakemore architecture in a central DFx block that integrates these functions together to provide the stated goal of reusability, modular design, and consistent validation practices across various divisions. In other words, if the fabric does not provide some level of capability, then the DFV features in the agents cannot be utilized. The converse is also true: if IP developers do not implement the capabilities, then the fabric cannot take advantage of them. A comprehensive strategy to incorporate these elements together to form an overall DFV methodology with a set of software tools is the goal of Lakemore.

The following sub-sections describe each DFV component and a few methodologies that utilize these features. Later there are more detailed sections of each interface.

Table 7-1. Chapter revision history

Revision Number	Description	Revision Date
rev1.2_rc1	• Initial release.	Jan 2012
rev1.2_rc3	• Added the DFx secure policy signal group and a basic description of the feature. •	Feb 13, 2012
rev1.2_rc4b	• Added trigger source and event signals for synchronous signals.	March 10, 2012
rev1.2_rc7	• DFx secure policy example truth table and Lakemore example truth table figures had the old ftap_dfxsecure label for the policy[3:0] column. • Moved the latch from the output of the lookup table to the input secure policy bus.	July 7, 2012
rev1.2_rc7b	• Added a fdfx_rst_b signal and a requirement that the VISA reset is connected to this signal.	July, 2012
rev1.2 (final)	• No edits	July 20, 2012
rev1.2.2_rc1	• Added the wide VISA bus definitions • Added VISA reset definition with the inclusion of a PGCB. • Split the signal table into separate tables based on signal groupings.	February,2014
1.3_rc1	• Added new interfaces for Sideband ISM debug, idle counter value, endpoint clock gate over	January, 2016
1.3_rc2	• Added serial read data signal to VISA configuration bus interface Signal definition for IPs: avisa_ser_rdata for IPs. Signal definition for VISA Register Controller: fvisa_ser_rdata.	May, 2016

7.1 VISA Debug Bus for Observability

One of the most common debug features used by nearly all SoCs and chipset components is a mux-based structure to collect internal signals and funnel them down to a narrower bus for observation internally, externally, or both. Several architectures, in the past and from several divisions, have addressed this issue and implemented a variety of ways, such as Node Observation Architecture (NOA), Chipwatcher, or a Debug Ring, that select internal signals and output them onto a dedicated debug port. This information can then be correlated, with logic analyzer trace captures of the interfaces, to discover how the component is operating with the applied validation test. An IP-block must implement a VISA Unit Level Mux (ULM) and the associated interface signals as defined in Table 7-7. The IP-block must follow all rules and guidelines in this section 7.1.x.   Very small IP-blocks with their embedded debug capabilities could be granted a waiver. As of this writing only the random number generator IP has been granted a waiver.

Visualization of Internal Signals Architecture (VISA) is a software tool-driven feature that accepts a signal list and auto-inserts the RTL code inside an agent or fabric connecting the selected signals or buses found in the design to a mux structure. The tool also allows the user to select signals directly from a parsed design in a GUI-based window rather than a signal list that may be useful during the initial implementation phase of the design. Figure 7-1 shows an agent with debug signals collected by the Unit-Level Mux (ULM) that supports a bypass feature and M-number of clock domains per interface. The observability bus is expandable in increments of one byte widths with a limit of 256 input byte lanes to select from. The number of clocks to select from is symmetric with the number of input byte lanes which is also 256. Although this an unrealistically large number of clocks to choose from, it was made as a 1:1 mapping with the input byte lanes for ease of implementation and programming model. The ULM outputs are collected at a partition-level mux (PLM) where the number input bytes are selectable across a set of ULM outputs, but the mixture of unique selections narrows to reduce routing congestion. Finally, the outputs of the PLMs are connected to a Central-Level Mux (CLM) where there are two outputs: one is a multi-byte lane that is connected to the ODLA for wide on-die (on-platform) trace to DDR memory, and the other is a crossbar mux usually 16 bits wide but could support up to 32 bits. This output is connected to the SoC Trigger Processor, the SoCHAP counters, and a set of package pins for use as a debug port. Reuse of functional pins is common for a debug port and is usually unavoidable for all but the large die and package sizes.

It is optional for a PLM to implement a VISA Sync Gasket (VSG) that takes the signals through an asynchronous clock-crossing FIFO structure to retime them to the VISA clock domain. This VISA clock is generally the highest dominate clock domain for the SoC. There is a method to compensate for these IP domains that have higher clock rates. The idea is to sacrifice data width for transferring two phases at one half the frequency rate. For example, a two byte lane VISA ULM the higher frequency debug data is split into two phases with phase A on byte 0 and phase B on byte 1. The amount of content is cut in half but higher frequency can be accommodated. The sync gasket is required for implementing the Lakemore architecture components.

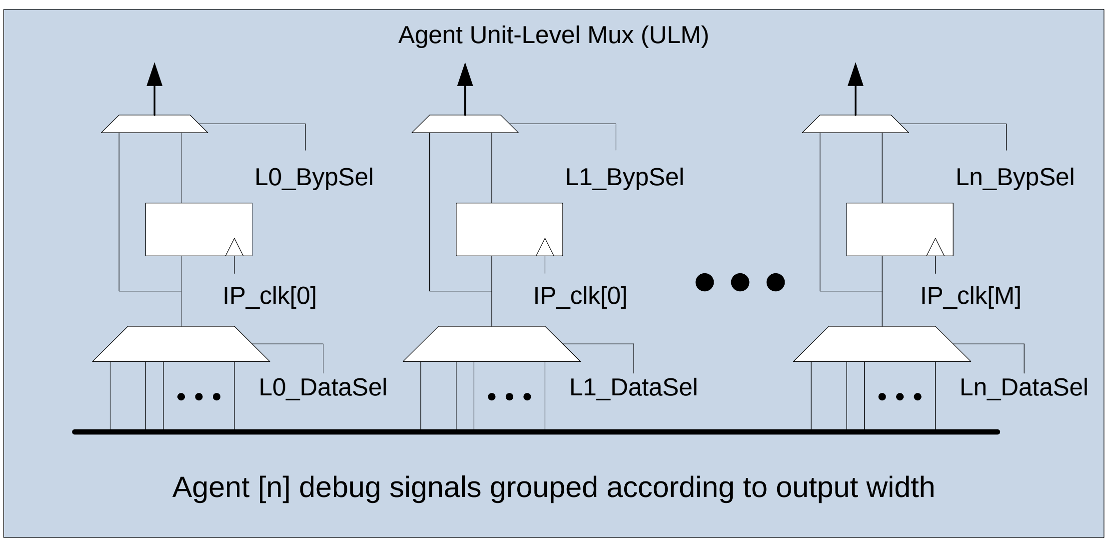
The sync gasket is composed of FIFO structure transfers a grey-coded pointer to indicate that the buffer has an entry to transfer. This also indicates a data is valid. A data valid output travels with the byte lane to the next level of muxing. This is used by the Lakemore components to know when the data is valid during a VISA clock period. This retains the signal's original domain information in the form a data signal. Most designs may choose to implement the ULM and PLM with source synchronous clocks and instantiate only one VSG at the CLM and connect it to Lakemore.

A VISA observability bus from the fabric to the agent is an optional feature. It has a specific use model to reuse the functional IO pins for higher bandwidth debug data output to an external logic analyzer. This feature was originally implemented in server chipsets and it is called the Pass-through Debug Port (PDP). The functional data path is bypassed between the transaction and link layers and the debug data is injected into that path. The link and physical layers are used in their native operation rather than bypassing the physical layer completely. It is implementation-dependent on how this is accomplished. For example, a PCIe x8 at 2.5Gb/s speed has peak bandwidth of 2.0GB/s which could handle a 40-bit wide (5-byte lane wide) debug bus at a core frequency of 400MHz. It would be uncommon to have this odd sized debug bus but at this bandwidth rate, it could easily handle a 4 byte (32-bit) wide debug bus. This use model may only be viable for very large server chipset components where the number of high speed interfaces is larger.

In general, selection of debug signals is implementation specific by the agent IP-block developer and the SoC integration team for the fabric. However, there is an effort to develop a more automated flow that starts with the specification and embeds the transaction flows directly into the document with a special template. This information can then be extracted with a tool and used as part of the debug signal list. With this combination of tools and flows it is possible to reduce the number of debug signals in a mature agent for derivative products. It is likely that a new agent will have many debug signals for post-silicon validation. In past chipsets and SoCs, it was unlikely that signals were removed from a derivative product for two reasons. First, the integration team does not know which ones to keep and which to remove. Secondly, it requires an engineering resource to remove signals and re-validate the structure. VISA will come with self-checking test benches to validate the network and a future version will strip out previous observability structures to get old designs transitioned into VISA.

Refer to the VISA HAS rev085 or later for more information about this DFV feature.

Figure 7-1. VISA unit-level Mux block diagram

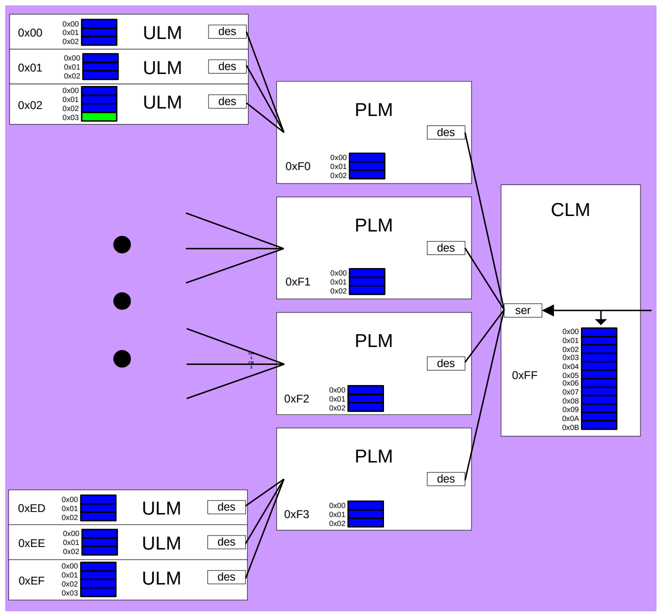


7.1.1 VISA Register Access

All VISA ULM/PLM/CLM (xLM) registers are access with a proprietary serial communication bus. The VISA Register Controller (VRC) is the source of the serial bus and connects to a CLM or PLM and distributes information in a daisy-chain star topology. The VRC has a TAP and an IOSF Sideband interface.

Figure 7-2 is a high level diagram that shows the VISA central controller with an access point to serializer that drives a simple three wire interface. This is fanned out to all PLMs throughout the fabric where a derserializer decodes the packet to determine if the register control data packet belongs to this PLM. The data is re-transmitted to all PLMs and ULMs that are lower in the hierarchy regardless if the PLM (or ULM) previously consumed the data values. Although this diagram only shows two levels of hierarchy below the CLM there may be more since VISA will support a daisy chain within an agent.

Figure 7-2. VISA register access network

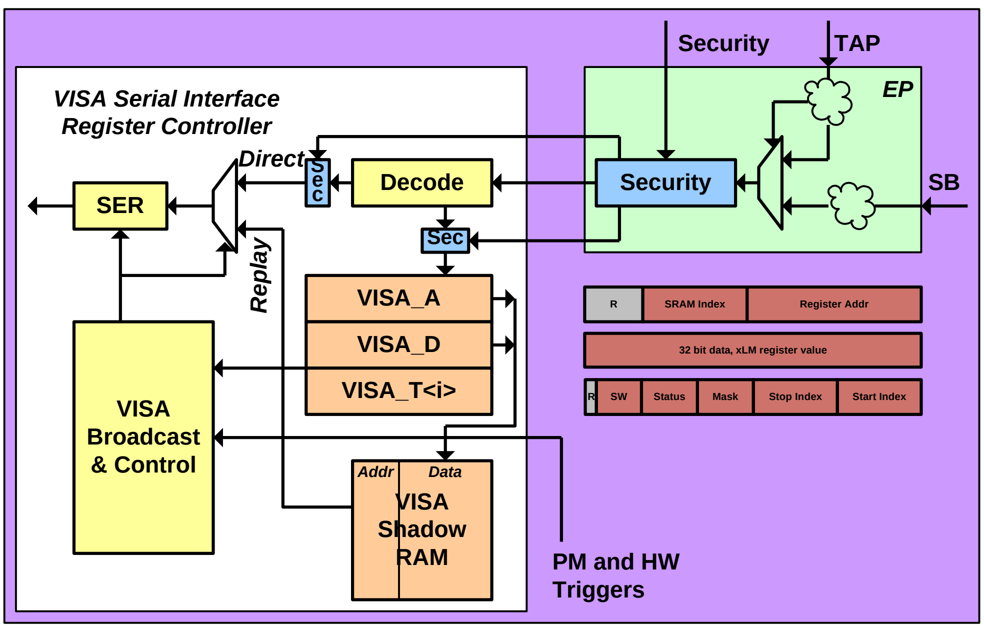


The VISA Register Controller implements a single RAM to store the register configurations for a limited set of VISA xLM instances throughout the SoC. Local registers at each xLM controls which debug byte lane are selected at each level. Although it has the appearance that this storage is a redundant duplication of the data, it is actually necessary for the complex use models involving power management and security for the observability solution.

Figure 7-3 shows a high level block diagram of the VISA register controller. An IOSF sideband endpoint block contains the endpoint logic block with a TAP test/debug register and a security block. The endpoint is standard from the Intel Reuse Repository (IRR) that includes the optional register access target agent. The TAP has test/debug register that can bypass the endpoint and be driven directly. The security block is based on SAI bits that specify access privileges for customer use. There are two methods of access for the register controller either directly from the EP block to a selected PLM/ULM mux control register or sourced from a RAM. When updating a VISA register directly, registers in the VISA control block are written with the request. A state machine serializes the data with output fanned out to all PLMs which pass the serial information on the ULMs. At each level of hierarchy the data is passed on to the last leaf of the tree. At each intermediate node the logic decodes the stream to determine if this unit is the destination of the data. The other method is to fill an SRAM with the data and let the state machine download it to all xLMs. Each SRAM entry is 48 bits, with the lower 32 designated for the VISA register data and the upper 16 bits designated for the register address (Unit_Id+Offset), though the actual register address within the IOSF-SB addressing scheme is limited to 14 bits. The SRAM is limited to 256 entries within the current register definition, which should be more than enough for most debug scenarios.

To support the re-configuration of the units with the stored values, the VISA controller provides a set of replay sequence registers and up to 8 hardware triggers. Each sequence register contains a RAM start and stop address defining a contiguous area within the RAM that contains the register configurations of interest for that particular sequence. In addition, there are individual masks for each of the 8 hardware triggers, along with a software trigger to force immediate replay. Finally, a set of status bits in each register indicates the state of the replay for that particular sequence to enable hardware to poll them as needed. Each SoC can define the size of the SRAM (up to 256 entries maximum) and as many hardware triggers (maximum 8) that it deems necessary. The SoC retains full flexibility to choose which power management events and/or other internal (presumably programmable) debug triggers to assign to the VISA hardware trigger inputs. Since the replay sequence registers can also be triggered directly by software, they can be also be used simply for fast re-configuration to speed up power-cycling validation runs. The combination of the programmable SRAM and the replay sequence registers provides a very fast, efficient, powerful and consistent mechanism to re-configure the VISA registers with relatively low hardware overhead. This re-programming is fully configurable for each run providing a dynamic and flexible means to provide debug observability through VISA. Since VISA also provides inputs to other debug infrastructure on-die, such as Lakemore that contains ODLA and SoCHAP counters, the flexibility improves the values of these features as well.

Figure 7-3. VISA serial interface register controller

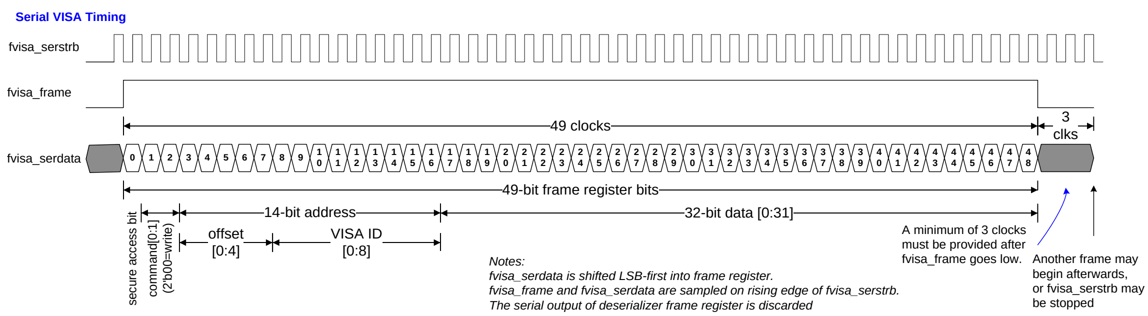


The register access mechanism also enables a single unified security access model for the full VISA infrastructure. The VISA Controller will accept two security enable bits, one each for the direct access and the RAM-program methods to program the registers. Software and/or hardware mechanisms outside the VISA controller can statically (via fuses) or dynamically (via keys) control these inputs based on the security requirements and access levels of the specific user. For the Replay-SRAM, the security enable is on the path to program (write) the SRAM as opposed to the replay (read) path. This scheme allows software to enable access to the SRAM once to program a specific configuration. The access can then be disabled while still allowing the programmed values to be replayed as necessary without software and/or security intervention.

Moreover, the addition of the hardware triggers with the replay sequence registers will enable the debug of complex power-state transition events that often happen too early for externally programmed debug hardware, which is typically much slower. The SoC Power Management Unit (PMU), along with any programmable trigger generation hardware, can drive these hardware triggers to replay any specific VISA configurations that are required for debug.

The serial interface to each PLM/ULM is shown in Figure 7-4. The fvisa_frame signal indicates the entire packet transaction. The fvisa_serdata is the remote register control information sent to each xLM to indicate control, address and the data that is stored in the local VISA register for the mux selection. The fvisa_serstrb indicates when the data bit is valid.

Figure 7-4. VISA register serial protocol



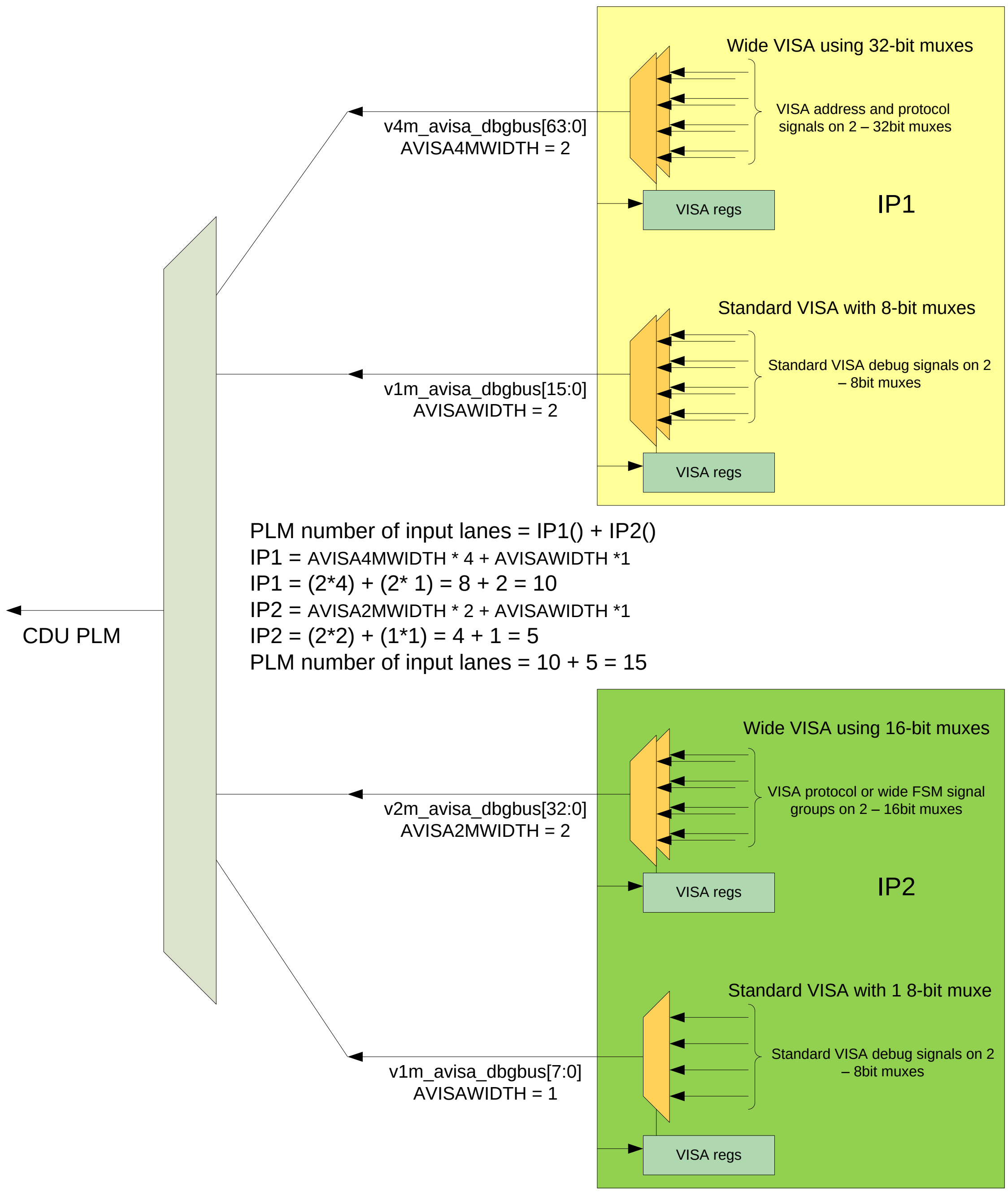
7.1.2 VISA Mux Widths

Table 7-2 is an example of two IP-blocks with an asymmetric number of lanes and choice of VISA wide mux width. IP1 is using two 32-bit wide muxes for address and protocol information. It also has two standard byte lanes wide debug buses for control signals. IP2 has two 16-bit wide muxes for protocol or concurrent state machine views. Also, it has one byte lane of standard (legacy) debug information. The PLM is expected to select individual byte lanes for maximum debug efficiency in selecting corresponding information each IP or both together. In this example, the PLM output is 6 lanes. Each VISA mux width style has its own IOSF DFx bus definition and parameters to identify the quantity of those muxes. The parameters can be used to mathematically calculate the number of input lanes needed for PLM in a Cluster DFx Unit (CDU). Each mux style has an equivalent number of byte lanes as shown in Table 7-2. This allows the VISA insertion tool to easily compute the number lanes required to connect the IP-blocks to the PLM.

Table 7-2. VISA mux style parameter reference

VISA mux style	Number of equivalent byte lanes (multiplication factor)	Parameter (IP perspective)	Comments
Standard (8-bit) byte lane	1	AVISAWIDTH	To maintain legacy with the previous revisions of the IOSF DFx HAS, this parameter is not renamed.
16-bit word lane	2	AVISA2MWIDTH	
32-bit double word lane	4	AVISA4MWIDTH	

Figure 7-5. IP-blocks with VISA wide muxes to a PLM with byte lane selection



7.1.3 VISA security

For several years, the VISA ULM has supported a set of customer visible lanes and Intel only lanes. However, this granularity is too course and we need to provide more debug information for customer platform debug then what this provides. If apply a color code analogy to the VISA debug signals then it would be ideal to provide VISA green debug signals for performance analysis and software tuning without the need to obtain a key. This means new software written for an Intel SoC doesn't need a key just to ensure it is functioning properly. However, only providing green is insufficient for hardware/software co-debug, therefore a customer will need VISA orange level signals for critical debug and validation. For obvious reasons that Intel must be able to debug their own silicon we need access to all debug signals and all DFx features. The Security Unlocked (red2) provides complete access. However for customer returns they may have secrets that are distributed to various locations on the die. These must be identified and protected with Intel Unlocked (red4). For more information, the reader is encouraged to review the Chassis Security HAS and the Chassis DFx Security HAS. An example from Avoton's P-unit (Atom core power management controller) of each of these levels is listed here. This is for illustration only and specific VISA security levels must be reviewed with that project's security architect.

1. VISA green: power states are publically available to comprehend the current state of the Atom cores.
2. VISA orange: access to debug bug message to and from the cores to understand a detailed analysis of the fine grain power state control.
3. VISA red: the instruction pointer of the 8051 microcontroller.

Shown Figure 7-6 is an example of applying the color code conditions with VISA ULM. It an attempt at apply color to the inputs of a ULM to show lanes that have different access level. If a number of input lanes are assigned as VISA green then only those lanes are accessible in the proper policy state value (refer to section 7.4 for details). When a customer has a valid (authenticated) key and we are in the OEM Unlocked state (policy=0x5) then the VISA orange debug signals and the green signals are available for debug. Finally, an authenticated Intel key will set the policy to Intel Unlocked (0x4) and all the lanes are available.

We are also maintaining the legacy VISA disable pins (visa_all_disable and customer_disable) although they are being redefined for this version of the spec. These disable signals are re-mapped to provide four levels of security access, namely, VISA black, green, orange and red.

Figure 7-6. VISA ULM with security color code overlay

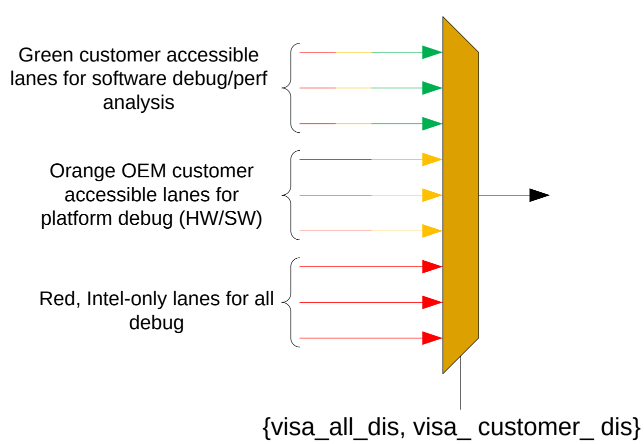
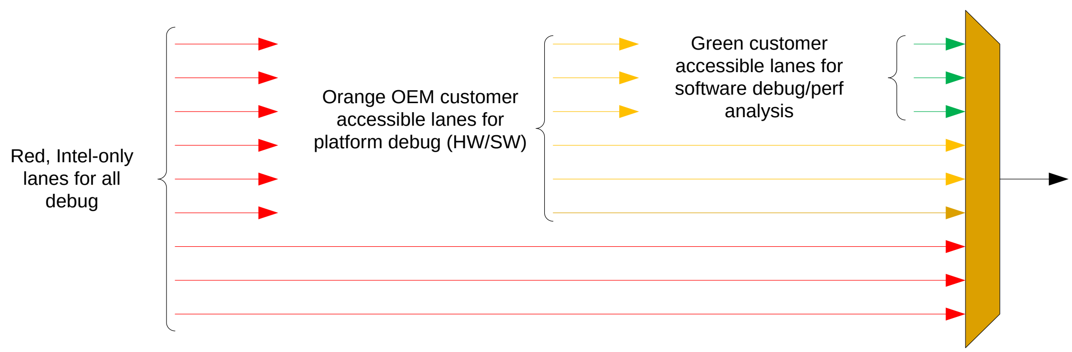


Figure 7-7 is another view of applying the color code description to the VISA ULM. It shows an increasing level of access to debug signals as we change from green to orange to red. It is not shown here but the VISA ULM mux can be completely shut off. This would translated into VISA black which is necessary when the SoC component is disabled or its personality has been programmed yet into the fuses.

Figure 7-7. Another view of VISA security color codes on a ULM



There are two ways to set the parameters that defined the VISA color coded lanes. One is set the CUST_VIS_TOP_LANE and the OEM_VIS_TOP_LANE. The defaults are shown but it is up to the user to set them properly.

1. parameter CUST_VIS_TOP_LANE = {SEL_WIDTH{1'b1}}, // default is all visible
2. parameter OEM_VIS_TOP_LANE = CUST_VIS_TOP_LANE, // default is same as cust vis top lane

However, the VISA architect states that those parameters are generally not used and most people use the $cust_top_vis_lane to set the top of the lane for the green debug signals. The user doesn't have to count the number of lanes to ensure they have the proper values set. If they user the $sign parameter value it sets the top lane value for that level of access. This follows with the new $oem_top_vis_lane for the orange debug signals. Figure 7-8 shows an example of using the parameters. Assume there are 20 input lanes for this VISA ULM. This IP-block requires 4 green lanes, 8 orange lanes and the rest red. If we set $cust_top_vis_lane to 3 then input lanes [3:0] will be green. Similarly, if $oem_top_vis_lane is set to 11 then input lanes [11:8] will be orange. The remaining lanes are red.

Figure 7-8. Applying security parameters to a VISA ULM

{visa_all_dis, visa_ customer_ dis}		ULM lane selection results
0	0	Red bucket
0	1	Green bucket
1	0	Orange bucket
1	1	Black, all disabled, clock gated

VISA parameters:
$cust_top_vis_lane = G
$oem_top_vis_lane = O
Example:
G= 3, O=11, number of input Visa lanes = 20
ulm_in[3:0]: green
ulm_in[11:4]: orange
ulm_in[19:12]: red

7.1.4 VISA reset control

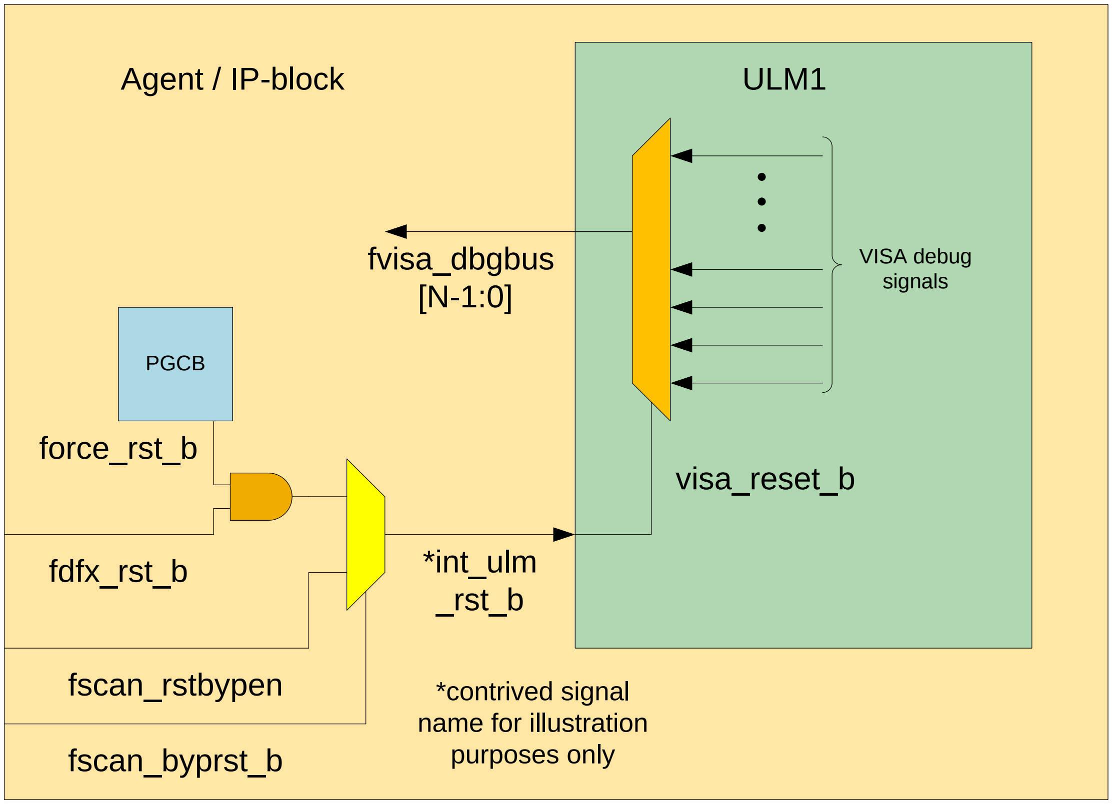
During the IOSF rev1.2 HAS publication a security hole was discovered with respect to using VISA during early boot debug. SoC silicon validation teams must be able to debug during the early boot process. For example, the Power Management Controller (PMC) is a critical IP-block to observe VISA signals to determine bring up issues. The DFx Security Plug-in (DSP) has the ability to allow full access to all debug signals and DFx features during this early boot window. Not all IP-blocks require this capability but for those that do a strap value indicates the level of access. For example, a typical setting would be to enable all DFx features and enable VISA red. The problem occurs when transitioning from red to green or orange to green and the IP-block was accessed during debug. The VISA register values do not change when the security level changes. This leaves the IP-block vulnerable when the security policy is established.

To resolve the issue, it was decided that the DFx aggregator would drive a reset signal to all ULMs. The IOSF designated name is fdfx_rst_b and it is connected to the visa_reset_b signal (as shown in Figure 7-9). The reason for the generic name is because the fdfx_rst_b was defined as a general purpose warm DFx reset in a previous IOSF DFx spec revision. The signal was not used and it was re-purposes for the ULM reset. The reader should refer to the Chassis DFx Aggregator HAS for more information on how to control this reset signal.

Soft IP-blocks with autonomous power gating must logically OR the PGCB force reset signal with the ULM reset (fdfx_rst_b). Since both signals are active low assertion this physically means the two signals are logically combined with an AND gate. This clears the ULM after each power gating and prevents spurious settings in the ULM that could expose red debug signals.

The fdfx_rst_b signal must be bypassed for scan testing. For this publication (IOSF DFx HAS rev1.2.2) it is required that the IP-block provide the scan reset bypass. A future version may move the mux into the VISA xLM IP-block.

Figure 7-9. VISA reset connected to Chassis DFx reset



7.1.5 VISA start and end ID description

It is expected that the integration teams manage the startID values for the IPs that they integrate. Also, it is suggested that they use the auto-ID feature tool in the VISA toolkit to assign the IDs as needed. Use of the endID is only allowed by the HIPs. Soft-IPs cannot use this interface because it is not needed and may actually be a detriment to the integration team's ability to manage the ULM IDs.

Refer to the following link for more information:

https://dtspedia.intel.com/Visa_IT/Signal_file_format#Automatically_generating_VISA_unit_IDs

7.1.6 VISA rules

IP developers should refer to the VISA web site to understand the rules regulating VISA implementation within an IP:

http://goto/visa

IPDS rules:

http://goto/ipds

7.2 Trigger sources and events

For the purpose of this specification the Cluster Trigger Block (CTB) is the trigger processing IP-block that is in the list of defined DFD IP-blocks for the Chassis DFx Gen2 generation. The CTB DFV IP-block that can generate trigger events based on a Boolean combination of trigger event inputs that can be counted or delayed in time. There are a variety of similar trigger block such as an Atom core micro breakpoint controller or the Common Trigger Sequencer (CTS). These trigger blocks act as a conduit to transmit the debug event (you may think of it as a single wire message) which may be modified by the trigger to manipulate that message to add context for debug use models. From an IOSF DFx interface perspective the outputs from an IP-block are named asta_trigsrc[N] and inputs to the IP-block are labeld as fsta_trigev[N]. IP-blocks that require cross-triggering would require both sets of signals. A protocol recognizer and trace injector like the PSF Fabric Trace Hook may have both types of trigger source/event debug signals. This IP acts has two functions. First, it acts as a mirroring function to replicate existing functional transactions for storage in a debug region of memory and second it has match/mask logic to identify when a particular or group of transactions have been passed through the PSF IP. The purpose for the input trigger event is to define a start and stop trace window in which the tracing logic operates so that only the address range of interested is traced and not all packets are replicated. The trigger source output detects when a specific transaction or a range transaction types have occurred.

The trigger source/trigger event signal group are defined at the IOSF DFx interface for IP-blocks but they are also the basis to form a network within the DFx fabric of the SoC. The Cluster Trigger Blocks in the regional DFx unit act as the trigger fabric to coalesce trigger events to reduce global wiring. The number of trigger source outputs from the CTB is equal to the number of resources; in general, this is usually only four wires. This means that only four unique triggers can exist within the SoC at one time. The CTBs have the benefit to logically combine regional trigger together to develop a composite trigger to work within this restriction.

If an agent generates trigger events for use by the SoC for on-die tracing or other debug activation features (such as array/freeze/dump) then the agent/IP-block must implement the asta_trigsrc signal on the interface. Similarly, if the agent/IP-block has a hardware function to activate, for example, error injection then the agent/IP-block must implement the fsta_trigev signal. Both of these signals are conditional based on need and they have parameterized widths.

7.2.1 DFx actions

A DFx action is a passive DFV feature that is different than the trigger IP-blocks. They are features that alter the behavior of an IP due to an assertion from the trigger IP-block within a region. Actions do not source trigger information. An example of a DFx action is a force clock stop (or clock freeze) to extract the current state of the flops in a particular cluster or region to analyze their contents for debug. The IOSF DFx signal is {f,a}sta_dfxact[U-1:0]. An earlier version of the IOSF DFx HAS (specifically rev1.2) already defines the array/freeze/dump (AFD) as a unique action since it is embedded in the DFx memory wrapper. We are essentially expanding this to a general interface signal that remains within the control of the region DFx unit that controls this IP-block.

In addition to debug and validation, the trigger IP-block and DFx actions can be used for survivability to work around a particular class of bugs that can be relieved through back-pressuring queues or causing a recoverable error. It should be obvious that the user must be careful with patching mechanisms so as not to cause adverse side effects on the performance of the platform. However, if corner case bugs rarely happen, then it can be argued that the fix is a nuisance but is less costly than a full stepping.

7.3 Access to Control and Status Registers for DFV

The post-silicon debug and validation teams require an out of band access to all functional registers within the SoC using the JTAG TAP controller. This allows a post-silicon tool to read any register if the primary fabric is hung. This feature implies two more requirements to accomplish this goal. 1. The SoC provides a TAP to Sideband logic block to translate a JTAG serial protocol to the Sideband protocol for register reads and writes. 2. All registers are accessible by the Sideband interface. This access requirement should be viewed from first silicon debug and validation point of view and all other security restrictions that limit access are still expected to be layered on top of this mechanism.

7.4 DFx Secure Policy Interface

Applying security to DFx is critical in protecting internally stored keys and other intellectual property of our SoC designs. The openness demanded by debugging and the closeness required by security are diametrically opposing requirements that cannot be completely resolved by a single point solution. The overall application of security on DFx features is a layered approach including firmware, a DFX aggregator IP, the DFx secure policy interface and a DFx secure plugin IP. This is a broadcast architecture where the DFx aggregator IP sets a policy which is sent on the DFx secure bus to all IPs in the SoC. The DFx secure plugin interprets the policy and translates it into enabling or disabling a set of DFx features. It should be noted here that the overall DFx security has changed since its initial development in Chassis 2.0 and reader should be familiar with the Chassis 2.1 DFx Security HAS. However, a few key aspects remain. One is that the VISA and TAP IPs rely upon a progression of openness for different security policies. Meaning, that a Secure Locked policy is most restrictive and an OEM Unlocked policy is more open with Security Unlock being the most open. However, if a customer part is returned to Intel for debug the policy available will be Intel Unlocked which is not as open as Security Unlocked so that the customer’s keys can be protected in the fuse block. In the past, this specification has used colors to represent security levels while some people do like this method of indicating an unlocked level it has proved to be easy to communicate intent. The obvious problem is how many colors are acceptable to describe the policy table. In general, we used green for Security Locked, orange for OEM Unlock and the user reserved policies and red for Security and Intel Unlocked. In most of the DFx IPs we can collapse the policy 2 and policy 4 together. In other DFT IP specs this may be referred to as red2 and red4 respectively to shorten the name for quick communication among colleagues. This is done for the TAPs and VISA ULMs. Using this as a reference we can then describe the openness of TAPs and VISA this way; as we move from green to orange to red more opcodes are available for access. This means that if we are in an orange policy we (including authenticated customers) then we have access to orange opcodes and green opcodes. If it is a red policy then all the opcodes are available. The same is applied to VISA signal groups where a set of signals in the green group are available in the green (Security Locked) and orange (OEM Unlocked) security state. VISA is different because it is implemented with a feature to completely turn it off with no signals available this would be defined as VISA black. The TAPs do not have this feature because if the TAP is addressed (TAP link architecture) or accessible as a child of a parent TAP (SoC TAP hierarchical architecture) then it must support some minimum level of access like the Bypass opcode otherwise, security would break the spec defined functionality. The overall security solution involving customers to obtain keys that is authenticated by the security engine is beyond the scope of this specification. The DFx secure policy plug-in is required by all IP-blocks. It is available in the Intel Reuse Repository (IRR) and compliant to this specification.

This DFx secure policy interface is a group of signals composed of one bus and two control signals that distributes a uniform policy value to all IP-blocks in the SoC. The signals are defined as fdfx_earlyboot_exit, fdfx_secure_policy[3:0] and fdfx_policy_update . IP-blocks instantiate a security plug-in IP that translates the current policy and enables or disables access to DFx features within the agent or IP-block. The DFx secure policy signal group is composed of a binary encoded policy bus, a latch enable and an early boot debug exit signals. This signal group is distributed throughout the SoC from DFx aggregator. The aggregator encodes the policy based on the Debug Life Cycle State (DLCS) from the fuse block and who is authenticated to access what level of DFx features. In other words, it is a combination of DLCS and the type of key a user had to gain access. No other source can drive this signal group other than the aggregator. Polices are defined in detail in the Chassis DFx security framework specification (reference listed in section 7.15) and it not re-iterated here. This section will define the DFx security plug-in IP and describe its functionality with examples.

To help describe the plug-in IP we need to review the policy table listed in Table 7-3. Policy values are defined by the Chassis 2.1 DFx Security HAS rev1.0 (refer docs listed in section 17.2). All policy values must by fully decoded and unused “user” policy values are decoded as OEM unlocked. The DFx secure policy bus is defined as a parameter but it is fixed with a width of four (4) bits for this version of the spec which is passed down from Chassis DFx spec as a requirement. It is mandatory that all IPs be the same width to eliminate security holes due to aliasing of the most significant bits. In the description field there are some indications of how a VISA unit level mux (ULM) or a slave TAP is expected to function within the scope of that policy. These are only illustrations and should be used as such. The use of color is an attempt to identify the security classification to help the reader interpret what level of access is being interpreted by the policy. A security described as green means it is public and anyone has access. From a VISA perspective the debug signals are benign. An orange level of access opens up more VISA signals and TAP opcodes. It was originally intended for the OEMs to use in-house for debugging their platforms in a pre-production scenario. A DFx security level that is labeled as red is only meant for Intel. Therefore, a color is meant to convey a how open the DFx features are from the customer’s point of view. Red means is the most “wide-open” level of access and should be viewed similar to a red colored document (labeled as Intel Top Secret). However, there are limitations to the use of colors as a label. The use of security colors as applied to VISA signals or TAP opcodes are useful for illustration purposes and employed throughout this document.

Table 7-3. DFx secure policy bus encoding

Policy Name	DFx Secure Policy Bus Encode	Generalized descriptions with TAP and VISA examples
Security Locked	0000	Locked, VISA signals that provide general control information (VISA green). TAP opcodes that provide status (TAP green).
Functionality Locked	0001	Locked, no VISA signals (VISA Black), TAP green
Security Unlocked	0010	Completely unlocked, all VISA signals (VISA red), all TAPs, all TAP opcodes (TAP red) are available.
Delayed Authentication Locked	0011	This is same as Security Locked but the SoC is allowed to go an unlock state in the future. VISA green, TAP green

Policy Name	DFx Secure Policy Bus Encode	Generalized descriptions with TAP and VISA examples
Intel Unlocked	0100	Most IPs are the same as Security Unlocked but there are exceptions like the fuse controller.
OEM Unlocked	0101	Loosely defined as partial unlock of DFx features. It is a group of features that are available in this policy in addition to the security locked group. VISA green and orange signals. TAP green and orange opcodes.
enDebug Unlocked	0110	Depends on the IP, in general VISA is green, the slave TAP (sTAP) is red (unlocked) the Chip Level TAP (CLTAP) is green (locked).
Infrared Unlocked	0111	This policy is specific for the Graphics IP but treated the same as policy 4 for enabling DFx features for all other IPs. Graphics IP uses it for PAVP debug enable.
DRAM debug Unlocked	1000	This is an example of a specific policy assignment for one IP. For all other IPs it is treated the same as policy 5 (partial unlock).
"User 3" Unlocked	1001	Reserved for future use and treated the same as policy 5. (OEM Unlocked)
"User 4" Unlocked	1010	Reserved for future use and treated the same as policy 5. (OEM Unlocked)
"User 5" Unlocked	1011	Reserved for future use and treated the same as policy 5. (OEM Unlocked)
"User 6" Unlocked	1100	Reserved for future use and treated the same as policy 5. (OEM Unlocked)
"User 7" Unlocked	1101	Reserved for future use and treated the same as policy 5. (OEM Unlocked)
"User 8" Unlocked	1110	Reserved for future use and treated the same as policy 5. (OEM Unlocked)
Part Disabled	1111	Locked, no VISA signals (VISA Black), TAP green

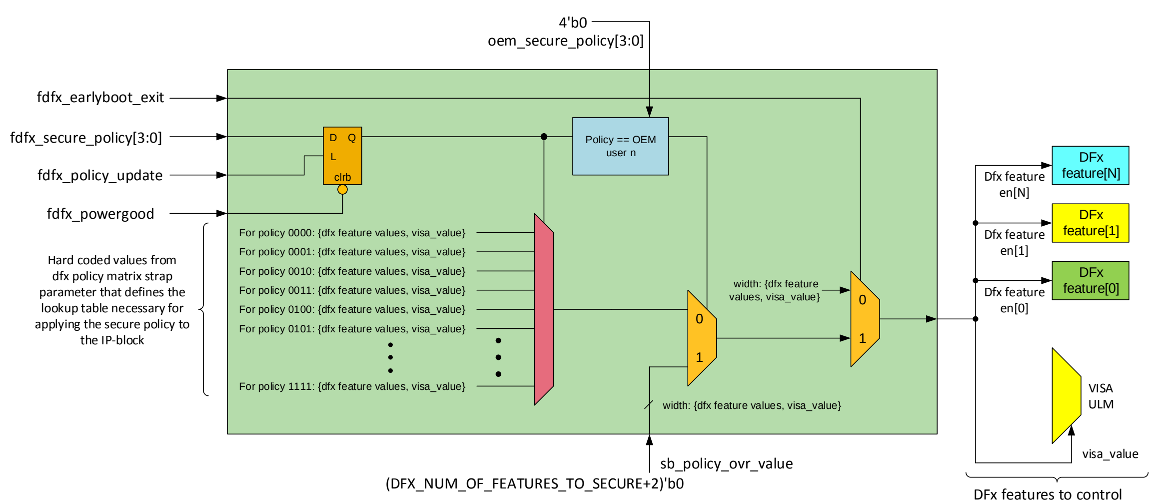
7.4.1 DFx secure plug-in IP-block

The DFx security plug-in IP-block is shown in Figure 7-10. DFx security plug-in block diagram. There are four sets IOSF DFx fabric facing interface signals, two supporting IP-facing signals buses and two output signal groups. Also, several parameters provide flexibility to the IP developer in how security enabling is applied to their IP-block. The left side of the block shows the IOSF DFx defined signals are listed in section 7.8. They are labeled as as fdfx_earlyboot_exit, fdfx_secure_policy[3:0] and fdfx_policy_update in the diagram. The policy is broadcasted to all IP-blocks on the secure policy bus. The IP-facing input control signals labeled oem_secure_policy[3:0] and sb_policy_ovr_value are no longer used and must be tied to logic 0. There is one set of output security control signal that connect directly to the VISA Unit Level Mux (ULM).

The policy value is latched if the policy_update signal is asserted followed by a lookup table that is defined by the policy matrix parameter. A latch is used rather than a flop to prevent clock glitch attacks. The update signal is controlled by the DFx aggregator.

There are two key parameters that are available to the IP developer to enforce security on their particular feature. One is the policy matrix parameter that is a two dimensional value with 16 rows and N number of column bits. The second is the number of DFx features to enable. This represents the variable N in the previous sentence. Each row represents the policy values and the columns are a group of bits that define the DFx feature enables and the VISA enable. Refer to example in section 7.1.3 for the VISA information. The output directly controls the assigned DFx features within the IP. For a small set of IP-blocks, it may be necessary to perform debug on the IP-block early in the boot flow. Not all IP-blocks need this feature but for those that do, there is a parameterized value that enables a set of DFx features and the VISA level of access necessary to debug and validate within this window of operation. Most IP-blocks will set this hardcoded value equivalent to a red level of access with all DFx features disabled and VISA red color code. During the boot process after the security policy has been determined, this window will be closed and the early boot exit signal will de-assert. Since VISA is used in nearly all IPs it was not included as part of the DFx features to enable output bus. Another reason is legacy, VISA already had two defined signals to enable a group of signals. These two signals are combined and overloaded with a new definition to form four groups of signals with different level of access: black (off), green, orange and red. A Chip-Level TAP (CLTAP) and slave TAP (sTAP) have their own DFx secure plugin and traditionally had a fixed lookup table. Even though this appears to be a duplication of logic from an IP point of view it help propagate consistency among the TAPs and prove successful in getting the security correct for one of the most important debug features because the TAP allows users access when portions of the SoC are not functioning correctly at power on.

Figure 7-10. DFx security plug-in block diagram



7.4.2 DFx secure plugin signal list

The DFx secure policy plug-in signals listed in Table 7-4.

Table 7-4. DFx secure policy plug-in IP signal list

Signal	I/O	Description
fdfx_secure_policy [N-1:0]	I	**Fabric DFx secure policy[N:0]:** Refer to the DFV signal section 7.7. Where N = DFXSECURE_WIDTH = 4 for this revision of the IOSF DFx HAS. Note: The latch is cleared to policy 0000 with the de-assertion of the fdfx_powergood.
fdfx_policy_update	I	**Fabric DFx policy update.** Refer to the DFV signal section 7.7.
fdfx_earlyboot_exit	I	**Fabric DFx early boot exit**. Refer to the DFV signal section 7.7.
dfxsecure_feature_en [M-1:0]	O	**DFx secure feature enable.** This bus enables each assigned DFx feature based on the policy assignment.
visa_all_dis	O	**VISA all disable.** This signal name is being re-used but its functionality is re-defined for this revision of the spec.
visa_customer_dis	O	**VISA customer disable.** This signal name is being re-used but its functionality is re-defined for this revision of the spec.
sb_policy_ovr_value [M+1:0]	I	**Chassis 2.1:** This feature is obsolete. The implementation team must connect sb_policy_ovr_value = 0. Refer to section 7.17.1 for historical information.
oem_secure_policy [N-1:0]	I	**Chassis 2.1:** This feature is obsolete. The implementation team must connect oem_secure_policy = 0. Refer to section 7.17.1 for historical information.
fdfx_powergood	I	**fdfx_powergood.** This is the power good reset pin. It is connected to the policy latch to clear it. Note: This signal is equivalent to **dfx_powergood_rst_b**.

7.4.3 DFx secure plug-in parameters

Table 7-5 lists the plug-in IP-block parameters for agents and IP-blocks to configure secure policy decoder for their use model. The hard IP-blocks will be fixed for this process generation.

Table 7-5. DFx secure plug-in IP parameters

Parameter name	Parameter value(s)	Comments
DFX_SECURE_ WIDTH	4	This parameter is fixed at 4 for this revision of the spec. (Defined in the Chassis mod-dfx Gen3 HAS)
DFX_NUM_OF_ FEATURES_TO_ SECURE	Set to the number of features to secure (minimum = 1)	Although, theoretically it can be any number of features but it is likely to be in the single digits. Any value set by the user does not include VISA as a feature to secure. If there are no DFx features to enable then the value is set to one (1) and the dfxsecure_feature_en[0:0] signal is not connected.

DFX_SECURE_POLICY_MATRIX	[DFX_SECURE_WIDTH-1:0][DFX_NUM_OF_FEATURES_TO _SECURE+1:0]	This parameter is a fixed number of values based on this version of spec. It determines the lookup table necessary to assign the appropriate policy with the DFx feature(s) including VISA access. Since the policy bus is fixed at 4 for this version of the spec it is a 16 row by [N+1:0] number of bits.
DFX_EARLYBOOT_FEATURE_ENABLE	[DFX_NUM_OF_FEATURES_TO _SECURE+1:0]	This parameter sets the hard coded value for the early debug window for this agent/IP-block. For most IP-blocks this will be VISA green only. Most IPs: DFX_EARLYBOOT_FEATURE_ENABLE[1:0] = VISA_GREEN DFX_EARLYBOOT_FEATURE_ENABLE[1:0] = {[DFX_NUM_OF_FEATURES_TO _SECURE:2]}{1'b0}}
The following list of parameters are for reference only. The user must not change these values		
USE_SB_OVR	0	Chassis 2.1: Set to 0. This feature is obsolete. The reader can review section 7.17.1 for historical information.
VISA_BLACK	11	This is used to define the VISA access value in an enumerated parameter format. This value will clock gate the output flops and prevent bypass from functioning.
VISA_GREEN	01	This is used to define the VISA access value in an enumerated parameter format. A VISA green level of access provides debug signals to customers without a key.
VISA_ORANGE	10	This is used to define the VISA access value in an enumerated parameter format. A VISA orange level of access provides debug signals to customers with a key.
VISA_RED	00	This is used to define the VISA access value in an enumerated parameter format. A VISA red level of access provides debug signals to Intel's use models.

7.4.4 DFx secure policy example

Figure 7-11 shows a fabricated example where two agents that have a different set of DFx assets to protect. This example is only for illustration purposes only and may or may not be an actual implementation of any particular SoC product. It is the IP-block's responsibility to provide security on their DFx features and the SoC integration team's responsibility to assign the various assets to the security requirements of the project. Agent 1 has a three DFx features with VISA. We do not include VISA as part of the feature count because we are assuming that nearly all agents and IP-blocks have VISA. If an IP-block doesn't have VISA then the outputs are simply not used. Agent 2 has no other DFx features besides VISA so the parameter for the number of DFx features to enable is set to the minimum value which is one. Due to the implementation of parameters we must have a minimum setting of one. The output is ignored and not used but the VISA control outputs are connected to the VISA ULM and is used.

Agent 1's two of the three DFx features are picked from actual DFx features in an existing SoC but they are not in any one particular IP-block. The IOSF ON-Die Logic Analyzer Trigger (ODLAT) example was implemented in the A-unit of the first Pondicherry SoC System Agent (SSA). An ODLAT performs a comparison of an expected command, address, and data slice from registers against transaction on the IOSF interface. If a match occurs then a trigger is asserted. This trigger output may be useful in trace operations to a central trace controller hub (such as North Peak) or as global trigger event to cause an array/freeze/dump. Match and mask values allow programmability of how the comparison will result in an output trigger based on specific transaction matches or don't care conditions with the mask registers. This feature has two levels of capabilities with level 1 to output a trigger assertion based on a matching command and address only. Level 2 will output a trigger assertion based on a matching command, address and one Dword of data. Even though this is one DFx feature it is beneficial to assign more than one enable to provide more levels of access. This allows for the possibility of giving the customer partial access to a DFx feature. It is not important what the third DFx feature actually is so we label it as simply "featureX". We want to show the ability of the security plug-in to use the spare user defined policies and a Sideband override capability and we need several DFx features to explain this. Most agents only contain VISA as its primary debug feature for customer use. The DFx aggregator in the bottom left of the diagram translates the Debug Life Cycle States, the appropriate key unlock and broadcasts the policy state on the DFx secure policy bus throughout the SoC.

Figure 7-11. DFx secure policy use model example 1

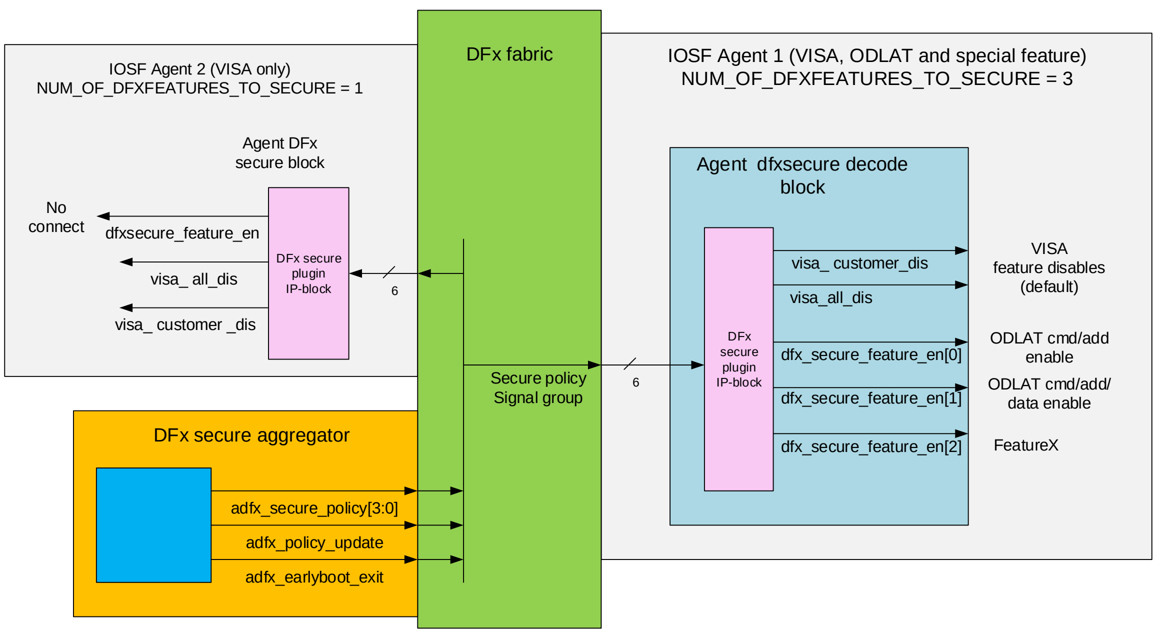


Figure 7-12 is the truth table that would be implemented by the secure policy plug-in IP for Agent 1. The left hand side of the table is the security policy briefly explained earlier in this section. Across the top of the right hand side of the table are the outputs from the DFx secure decoder IP. The outputs are dfx_secure_feature_en[N:0] and the VISA security enabling signals (visa_all_dis and visa_customer_dis). The bit values on the right hand side are the desired security settings for each of the features. All policies are decoded any unused "user" defined policy states must be decoded to be the same value as defined by policy 0x5. A red box highlights one row of the table and can be seen highlighted in the parameter definition at the bottom of the table. Every entry of the table is described by the policy matrix parameter.

Figure 7-12. Computing a policy matrix value for agent 1

Where M= [NUM_OF_DFXFEATUERS_TO_SECURE+1:0]
(The number of DFx features are 3 in this example + 2 bits for VISA)

	fdfx_secure_policy[3:0]				dfxsecurfeature[2]	dfxsecurfeature[1]	dfxsecurfeature[0]	Refer to VISA table {visa_all_dis, visa_ customer_ dis}		
Security Locked, Policy 0:	0	0	0	0	0	0	0	0	1	// VISA green lanes, no DFx features enabled
Functionally Locked, Policy 1:	0	0	0	1	0	0	0	1	1	// VISA disabled, no DFx features enabled
Security Unlocked, Policy 2:	0	0	1	0	1	1	1	0	0	// Red access to all features enabled
Delayed Auth. Locked , Policy 3:	0	0	1	1	0	0	0	0	1	// VISA green, DFx features disabled
Intel Unlocked, Policy 4:	0	1	0	0	1	1	1	0	0	// Red access to all features enabled
OEM unlocked : Policy 5:	0	1	0	1	0	0	1	1	0	// VISA orange lanes + ODLAT cmd/addr
enDebug Unlocked: Policy 6:	0	1	1	0	1	1	1	0	1	// VISA green, all DFx features, see note1
Infrared Unlocked, Policy 7:	0	1	1	1	1	1	1	0	0	// Special for GT, same as policy 4 for others
DRAM debug Unlocked, Policy 8:	1	0	0	0	0	0	1	1	0	// Special for DRAM, same as policy 5 for others
User 3 unlocked,Policy 9:	1	0	0	1	0	0	1	1	0	// same as OEM unlocked (policy 5)
User 4 unlocked, Policy A:	1	0	1	0	0	0	1	1	0	// same as OEM unlocked (policy 5)
User 5 unlocked, Policy B:	1	0	1	1	0	0	1	1	0	// same as OEM unlocked (policy 5)
User 6 unlocked, Policy C:	1	1	0	0	0	0	1	1	0	// same as OEM unlocked (policy 5)
User 7 unlocked, Policy D:	1	1	0	1	0	0	1	1	0	// same as OEM unlocked (policy 5)
User 8 unlocked, Policy E:	1	1	1	0	0	0	1	1	0	// same as OEM unlocked (policy 5)
Part Disabled, Policy F:	1	1	1	1	0	0	0	1	1	// VISA disabled, DFx feature disabled

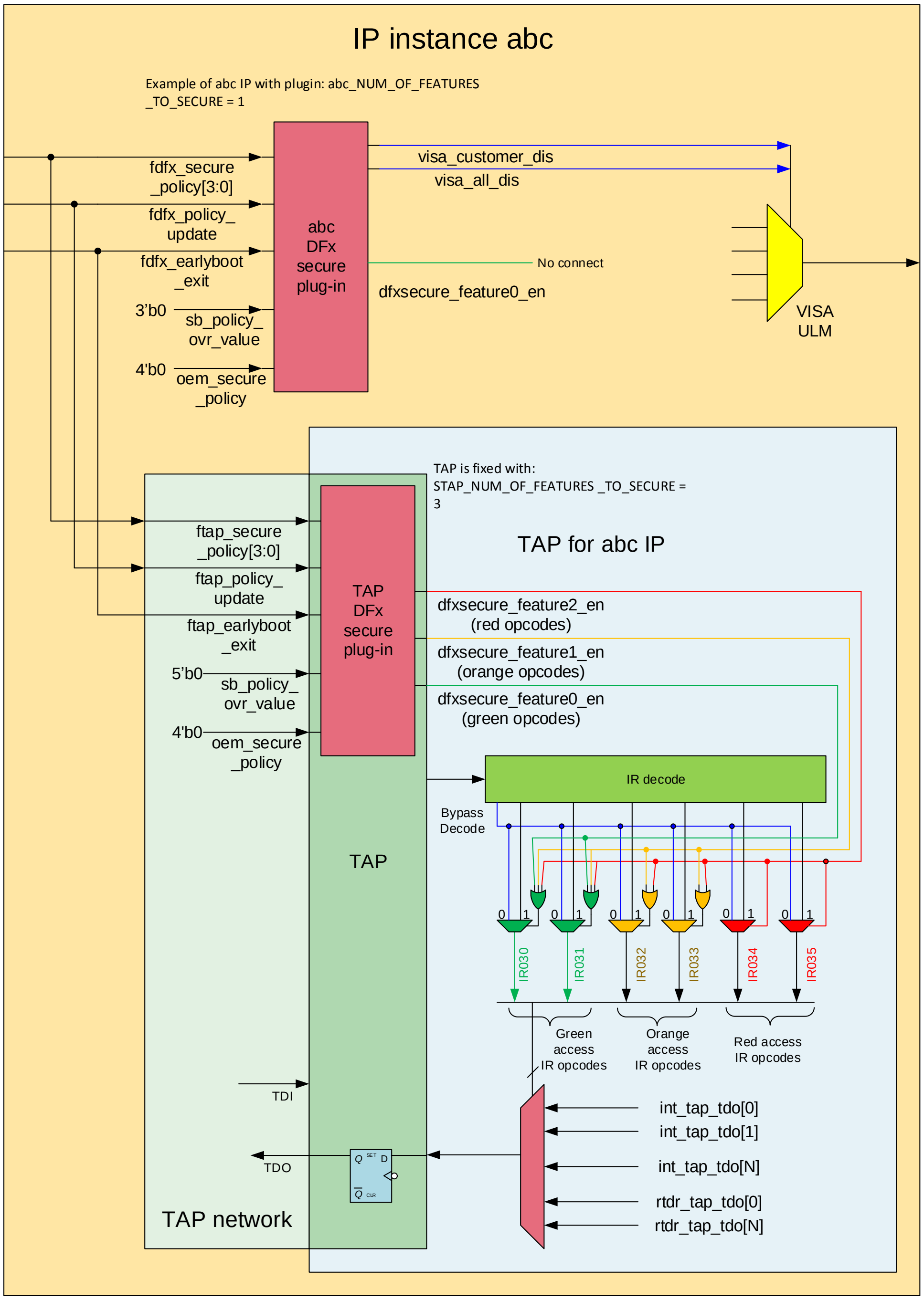
POLICY_MATRIX [7:0][M] = {11100, 11101, 00110, 11100, 00001, 11100, 00011, 00001}

POLICY_MATRIX [14:8][M] = {00110} POLICY_MATRIX [15][M] = {00011}

7.4.5 Agent/IP-block example with IP and TAP plug-in

Shown in Figure 7-13 is an example of agent (IP-block) with a DFx security plug-in and a TAP. Since the TAP already comes with the plug-in the IP-block developer must connect the DFx security plug-in signals together and route them up to the agent (IP) top level.

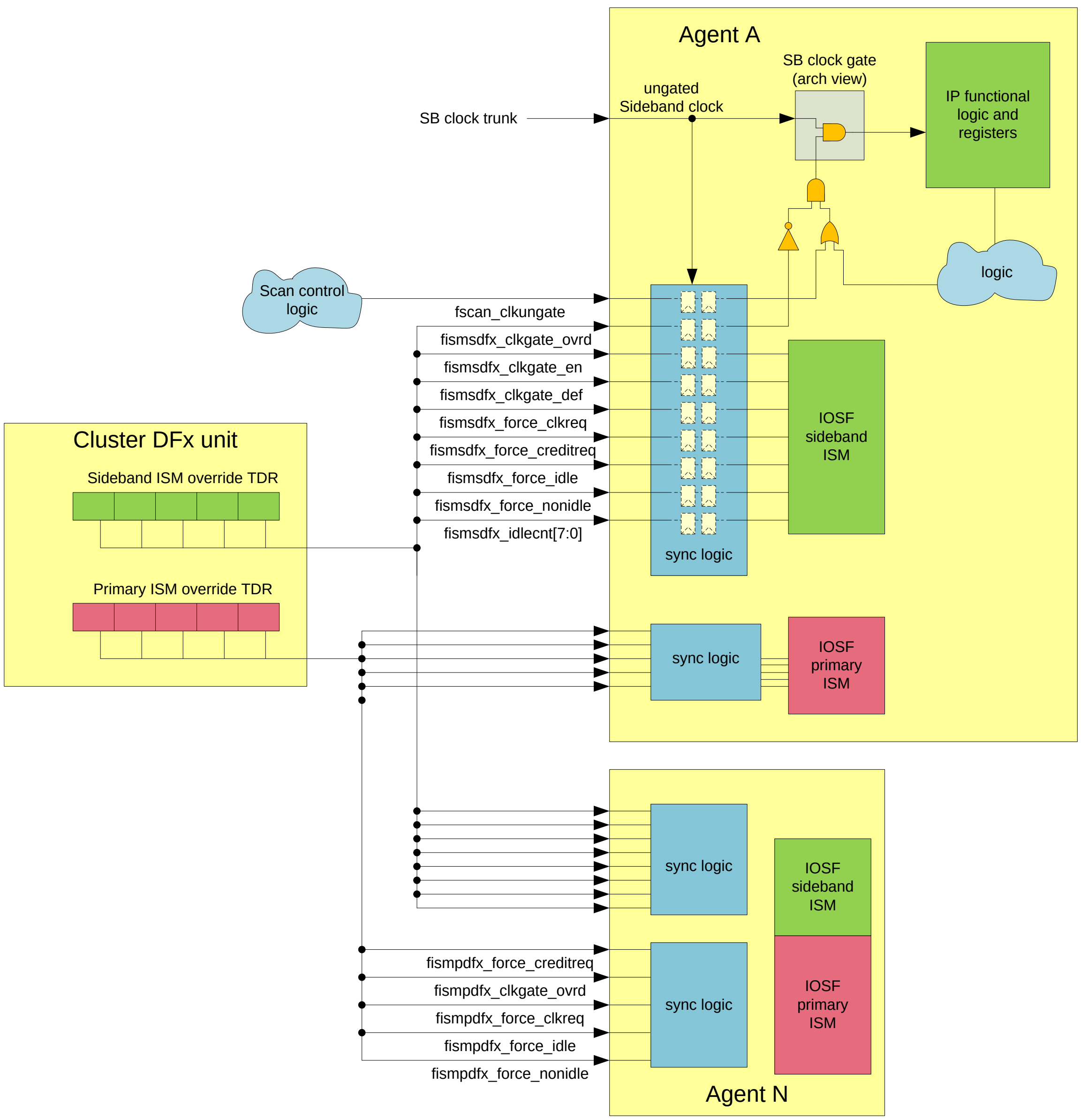
Figure 7-13. Agent/IP-block with security plug-in and TAP



7.5 ISM override for debug

The Idle State Machine (ISM) in the agent's IOSF primary and sideband interfaces must have override signals for debug. There are five (5) signals for both Sideband and IOSF Primary endpoints and three (3) additional signal groups for just Sideband. Details for these signals are defined in this section (Section 7.5) and in the Table 7-7. They are driven from a TAP test data register (TDR) in the Cluster DFx Unit (CDU). The SoC integration team has the choice to fan out one set of TAP test data registers for each group of primary and sideband ISM overrides or a unique TDR for each IP (shown in Figure 7-14). Since the impact to gate count may be high it may more advantageous to group together for each sideband and primary interface. It is implementation dependent on whether to provide a per IP, per sideband, per primary TAP bit for each set of control signals..

Figure 7-14. TAP TDR connectivity example for IOSF ISM override signals



7.5.1 IOSF Primary override signal description

This section describes the IOSF Primary ISM override signals and the use model requirements for the applying them to the endpoint.

1) fismpdfx_force_idle:
    a) Signal override ISM use model description:
        i) If Agent ISM is in IDLE state;
            (1) It remains IDLE state until:
                (a) If Fabric ISM is not in ACTIVE_REQ or CREDIT_REQ state
                (b) And credit initialization is completed
        ii) If Agent ISM is in active state
            (1) If a transaction is progress, it completes the current transaction and waits for 16-32 clock cycles and then moves to IDLE state. This feature assumes the clock is running, meaning, the clock isn't forced to be gated.
    b) Design considerations:
        i) This signal must be synchronized to the Primary clock domain (ungated clock trunk). The signal source is from a TCK clock domain test data register. It must be on the output of the trunk clock gate but not the local clock gate.
    c) Use mode considerations:
        i) This signal is meant to be used exclusively from the other overrides. It is a user error if they assert all of the overrides or any random configuration. There is no expectation of the ISM to handle this condition. Use of this signal assumes the trunk clock gate is ungated by the user and the clock is running.
2) fismpdfx_force_notidle:
    a) Signal override ISM use model description:
        i) Agent ISM eventually moves ACTIVE state
            (1) If Fabric ISM is in ACTIVE_REQ state
            (2) Or clock is running and credit initialization is done
    b) Design considerations:
        i) This signal must be synchronized to the Primary clock domain (ungated clock trunk). The signal source is from a TCK clock domain test data register. It must be on the output of the trunk clock gate but not the local clock gate.
    c) Use mode considerations:
        i) This signal is meant to be used exclusively from the other overrides. It is a user error if they assert all of the overrides or any random configuration. There is no expectation of the ISM to handle this condition. Use of this signal assumes the trunk clock gate is ungated by the user and the clock is running.
3) fismpdfx_force_creditreq:
    a) Signal override ISM use model description:
        i) Agent ISM moves CREDIT_REQ state
            (1) If Agent is active (clock is running) or Fabric ISM is in CREDIT_REQ state

b) Design considerations:

i) This signal must be synchronized to the Primary clock domain (ungated clock trunk). The signal source is from a TCK clock domain test data register. It must be on the output of the trunk clock gate but not the local clock gate.

c) Use mode considerations:

i) This signal is meant to be used exclusively from the other overrides. It is a user error if they assert all of the overrides or any random configuration. There is no expectation of the ISM to handle this condition. Use of this signal assumes the trunk clock gate is ungated by the user and the clock is running.

4) fismpdfx_force_clkreq:

a) Signal override ISM use model description:

i) Agent requests for clock

(1) If current clock is not running and Fabric Clock ACK is de-asserted

b) Design considerations:

i) This signal is expected to be asynchronously and logically combined with other control signals within the prim/side interface design. It is design dependent if the output cone of logic is synchronized to the trunk clock as necessary for proper functionality. For some IP-blocks, the clkreq signal may include other inputs such as a pin assertion that wakes the IP-block from a low power state.

ii) When asserted (*_force_clkreq = 1) and the prim_clkack = 0 then this will asynchronously assert the clock req for this interface.

iii) When deasserted (*_force_clkreq =0) this signal will not cause any action within the IP-block. If the signal transitions from 1 to 0, internally the clkreq it will continue to assert until prim_clkack is logic 1 so the handshake protocol is maintained and the IOSF specification is preserved.

c) Use mode considerations:

i) This signal is meant to be used exclusively from the other overrides. The user of these signals must ensure the clocks are ungated at the trunk or internally to the agent. For the sideband, the *_clkgate_ovrd is assert first (if necessary) so that register bits (registers in the SB clock domain) containing the other clock gate overrides are accessible.

5) fismpdfx_clkgate_ovrd

a) Signal override ISM use model description:

i) This signal does not connect to the ISM. It applies only to the local clock gate.

b) Design considerations:

i) This signal must be synchronized to the Primary clock domain. The signal source is from a TCK clock domain test data register. It must be on the output of the trunk clock gate but not the local clock gate.

ii) This signal is logically OR'd with the scan control signal for controlling the clock ungate (fscan_clkungate).

c) Use model considerations:

i) When asserted, the clock gate is overridden. This will force clock gate to be disabled and enable the clock to be running.

ii) This signal is optional for the primary. This is due to the changes in the chassis clock unit and how the clock gates are controlled and distributed. The IP-block may be controlling its own clock gates.

7.5.2 IOSF Sideband override signals

This section describes the Sideband ISM override signals and the use model requirements for the applying them to the endpoint.

1) fismsdfx_force_creditreq:
   a) Signal override ISM use model description:
      i) Agent ISM moves CREDIT_REQ state
         (1) If Agent is active (clock is running) or Fabric ISM is in CREDIT_REQ state
   b) Design considerations:
      i) This signal must be synchronized to the Sideband clock domain (ungated clock trunk). The signal source is from a TCK clock domain test data register. It must be on the output of the trunk clock gate but not the local clock gate.
   c) Use mode considerations:
      i) This signal is meant to be used exclusively from the other overrides. It is a user error if they assert all of the overrides or any random configuration. There is no expectation of the ISM to handle this condition. Use of this signal assumes the trunk clock gate is ungated by the user and the clock is running.
2) fismsdfx_clkgate_ovrd
   a) Use model description:
      i) This purpose of this signal is to “force” the clock gate to shut off the clock for power measurement analysis. This signal is may be not useful for debug.
   b) Design considerations:
      i) This signal must be synchronized to the Sideband (Primary) clock domain. The signal source is from a TCK clock domain test data register. It must be on the output of the trunk clock gate and not the local clock gate.
      ii) This signal is required for the Sideband but it is optional for the primary. This is due to the changes in the chassis clock unit and how the clock gates are controlled and distributed.
3) fismsdfx_force_clkreq:
   a) Signal override ISM use model description:
      i) Agent requests for clock
         (1) If current clock is not running and Fabric Clock ACK is de-asserted
   b) Design considerations:
      i) This signal is expected to be asynchronously and logically combined with other control signals within the prim/side interface design. It is design dependent if the output cone of logic is synchronized to the trunk clock as necessary for proper functionality. For some IP-blocks, the clkreq signal may include other inputs such as a pin assertion that wakes the IP-block from a low power state.

ii) When asserted (*_force_clkreq = 1) and the side_clkack = 0 then this will asynchronously assert the clock req for this interface.

iii) When deasserted (*_force_clkreq =0) this signal will not cause any action within the IP-block. If the signal transitions from 1 to 0, internally the clkreq it will continue to assert until side_clkack is logic 1 so the handshake protocol is maintained and the IOSF specification is preserved.

c) Use mode considerations:

i) This signal is meant to be used exclusively from the other overrides. The user of these signals must ensure the clocks are ungated at the trunk or internally to the agent. For the sideband, the *_clkgate_ovrd is assert first (if necessary) so that register bits (registers in the SB clock domain) containing the other clock gate overrides are accessible.

4) fismsdfx_force_idle:

a) Signal override ISM use model description:

i) If Agent ISM is in IDLE state;

(1) It remains IDLE state until:

(a) If Fabric ISM is not in ACTIVE_REQ or CREDIT_REQ state

(b) And credit initialization is completed

ii) If Agent ISM is in active state

(1) If some transaction is progress, then it completes the current transaction then waits for 16-32 clock cycles and then moves to IDLE state. This assumes the clock is running.

b) Design considerations:

i) This signal must be synchronized to the Sideband clock domain (ungated clock trunk). The signal source is from a TCK clock domain test data register. It must be on the output of the trunk clock gate but not the local clock gate.

c) Use mode considerations:

i) This signal is meant to be used exclusively from the other overrides. It is a user error if they assert all of the overrides or any random configuration. There is no expectation of the ISM to handle this condition. Use of this signal assumes the trunk clock gate is ungated by the user and the clock is running.

5) fismsdfx_force_notidle:

a) Signal override ISM use model description:

i) Agent ISM eventually moves ACTIVE state

(1) If Fabric ISM is in ACTIVE_REQ state

(2) Or clock is running and credit initialization is done

b) Design considerations:

i) This signal must be synchronized to the Sideband clock domain (ungated clock trunk). The signal source is from a TCK clock domain test data register. It must be on the output of the trunk clock gate but not the local clock gate.

c) Use mode considerations:

i) This signal is meant to be used exclusively from the other overrides. It is a user error if they assert all of the overrides or any random configuration. There is no expectation of the ISM to handle this condition. Use of this signal assumes the trunk clock gate is ungated by the user and the clock is running.

6) fismsdfx_idlecnt[7:0]
   a) Use model description:
      i) Idle count limit for ISM which is determined when the endpoint should transition to IDLE_REQ.
   b) Design considerations:
      i) Recommended default value = 8’h10 (16 decimal)
      ii) A value of zero breaks the implementation of the Sideband endpoint resulting in an unresponsive endpoint. It will be considered as user error.
      iii) Connected to Sideband endpoint signal labeled as: cgctrl_idlecnt[7:0]
      iv) Synchronization to side_clk is required since this signal is assumed to be driven from a TAP register operating in the TCK clock domain.
7) fismsdfx_clkgate_en
   a) Use model description: Clock gate enable
      i) When set to logic 1 it enables the ISM to leave ACTIVE and allows gating the side_clk if in the IDLE state (normal operation).
      ii) When set to logic 0, the ISM never leaves ACTIVE once it gets into that state and the clock is never gated.
   b) Design considerations:
      i) Recommended default value = 1’b1.
      ii) Connected to Sideband endpoint signal labeled as: cgctrl_clkgaten
      iii) Synchronization to side_clk is required since this signal is assumed to be driven from a TAP register operating in the TCK clock domain.
8) fismsdfx_clkgate_def
   a) Use model description: Clock gate defeature
      i) When set to logic 1, it disables side_clk gating within the endpoint when the ISM is in the IDLE state.
      ii) When set to logic 0, the endpoint is in normal operation.
   b) Design considerations:
      i) Recommended default value = 1’b0.
      ii) Connected to Sideband endpoint signal labeled as: cgctrl_clkgatedef
      iii) Synchronization to side_clk is required since this signal is assumed to be driven from a TAP register operating in the TCK clock domain.

Table 7-6. Use model examples

Expected Use model	ISM DFx override signals fism{p,s}dfx_*					Other signals	Comments
The use models listed are hypothetical to provide a sense of the how the signals might be used. It doesn't mean that they will resolve a specific bug but they can be used with other investigations or techniques.	clkgate_ovrd	clkreq	creditreq	idle	notidle		
Sideband use model 1 (SB1-step1): Write local Cluster DFx Unit's TAP to enable the bit.	1	0	0	0	0	Disable clock gate	Purpose is to conduct power measurements by enabling clock gates per IP or group of IPs.
SB1-step2	1	0	0	0	0	N/A	Take measurements
Use model 2 (UM2-step1): IP-block is misbehaving with suspicious credit accounting, enable VISA path to the Intel ® Trace Hub to trace debug signals and force credit initialization.	0	0	0	0	0	enable IP VISA	The user must ensure the clock trunk is ungated.
UM2-step 2: Write local Cluster DFx Unit's TAP to enable the bit.	0	0	1	0	0	observe signals	
Use model 3 (UM3-step1): IP-block is misbehaving with idle conditions, enable VISA path to Intel ® Trace Hub to trace debug signals and force ISM to idle	0	0	0	0	0	enable IP VISA	If necessary, execute use model #1 to force an active clock. The user must ensure the clock trunk is ungated.
UM3 – step 2: Write local Cluster DFx Unit's TAP to enable the bit.	0	0	0	1	0	observe signals	
Use model 4 (UM4-step1): Same as UM3 but we want to force the IP's ISM to not return to the Idle state.	0	0	0	0	0	enable IP VISA	The user must ensure the clock trunk is ungated.
UM4 – step 2: Write local Cluster DFx Unit's TAP to enable the bit.	0	0	0	0	1	observe signals	
Use mode 5 (UM5-step1):	0	0	0	0	0	enable ITH	

Issues with the IP-block requesting the clock (wake event problems). Enable Intel ® Trace Hub and observe debug signals while the clkreq is forced							
UM5-step2: Write local Cluster DFx Unit's TAP to enable the bit.	0	1	0	0	0	observe signals	

7.6 Microprocessor debug control

IOSF-based IP-blocks have been employing processors core such as the Minute-IA since the IOSF DFx HAS rev1.2 release however, those early core did not contain run control. Now that they have the capability the IOSF DFx interface is updated to include it. The preq / prdy set of pins are driven from the Run Control Module (RCM) in the Master DFx Unit. Processor-request (preq) signal informs the processor to take action to break from its normal execution. Once the processor has entered into this special debug subroutine it asserts Processor-ready back to the requesting source that the processor is now ready for debug commands. The RCM essentially manages the signaling as a centralized distribution hub to broadcast preq from the host controller to any processor or forward the prdy from one processor to another.

The signals are defined as active high for Chassis DFx Gen2 rev1.0. When IOSF DFx HAS rev1.2.2 was published it is known that Lakemont 2.1 Minute-IA processor is active low. It is expected that later version of the processor will be compliant with the active high polarity requirements. The RCM IP is parameterized to handle active low signaling. The signals are defined in Table 7-7.

7.7 Sideband parity error defeature

The Sideband router and endpoint supports parity checking for SoCs that require protection against events that can change the logic value of a flop or gate. This may only be required for some SoCs due to the large size of the die, the small geometries of the transistors and high reliability requirements of the product. The defeature signal allows the integration team to control the signal as necessary for survivability. It is expected that the power management controller will broadcast the parity defeature value throughout the DFx fabric to enable or disable parity as necessary for proper functionality. This implies that during a fuse pull event, the PMC can defeature the capability to prevent a hang condition with a non-responsive endpoint due to an errant parity check.

When the parity defeature is asserted the Sideband endpoint treats all parity as good parity. Incoming payload from the Sideband router will be propagated to the agent as if it was good parity regardless of the parity logic result. Also, the outgoing transactions will have the parity signal (sbe_sbi_parity_err_out) deasserted even if the parity generation logic created a bad result. The signal is listed in Table 7-11 and is required on the IP's top level interface.

7.8 DFD Signal Interface Description

A graphical view of the DFV signals are shown in Figure 7-15 and Figure 7-16 with Table 7-7 describing the signals in detail. The integration team has the freedom to assign any reasonable number of characters as a prefix or suffix as applied to the root name to identify the signal in a full chip SoC.

Figure 7-15. Graphical diagram of DFV signal interface

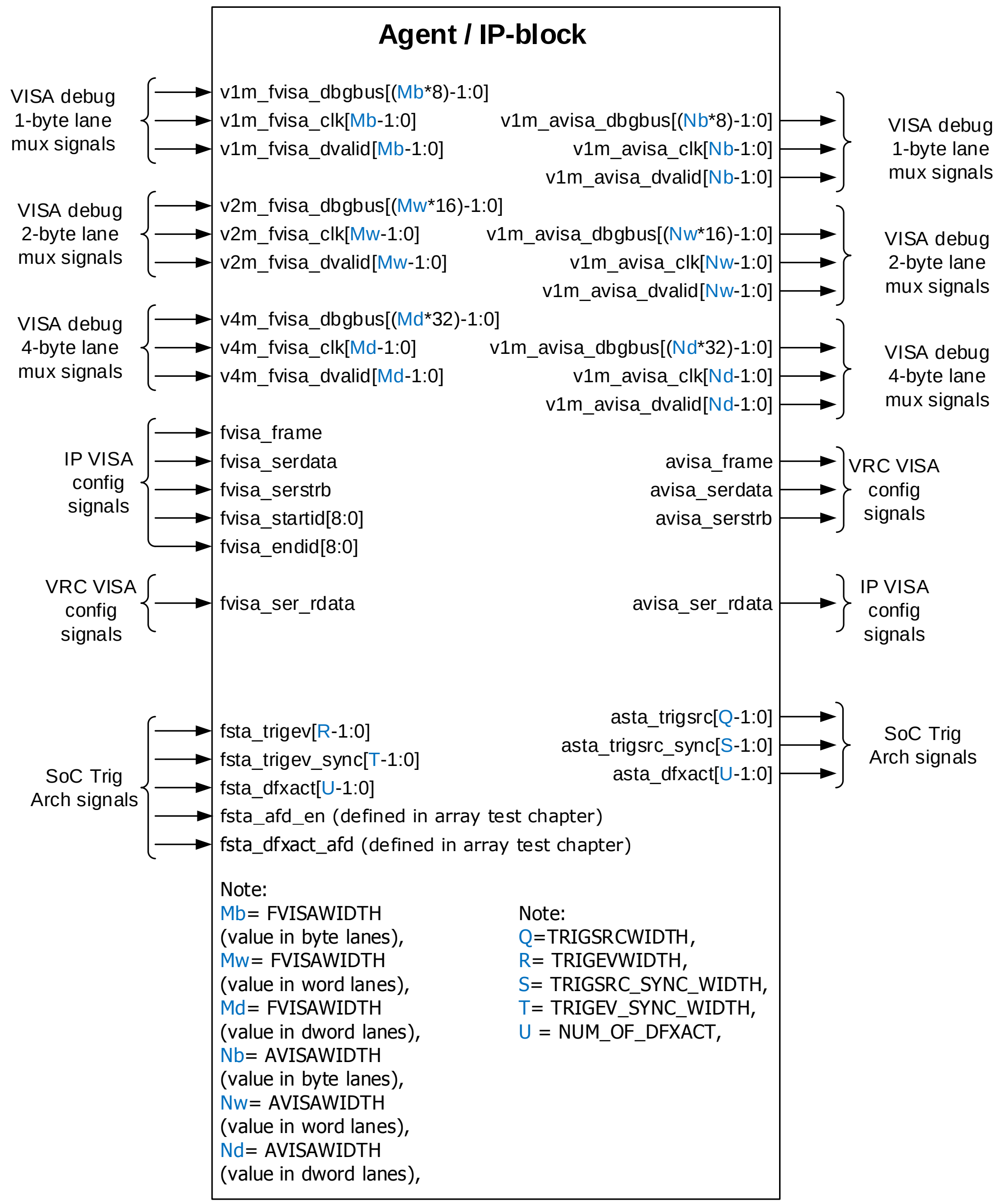


Figure 7-16. Graphical view of DFV signals continued

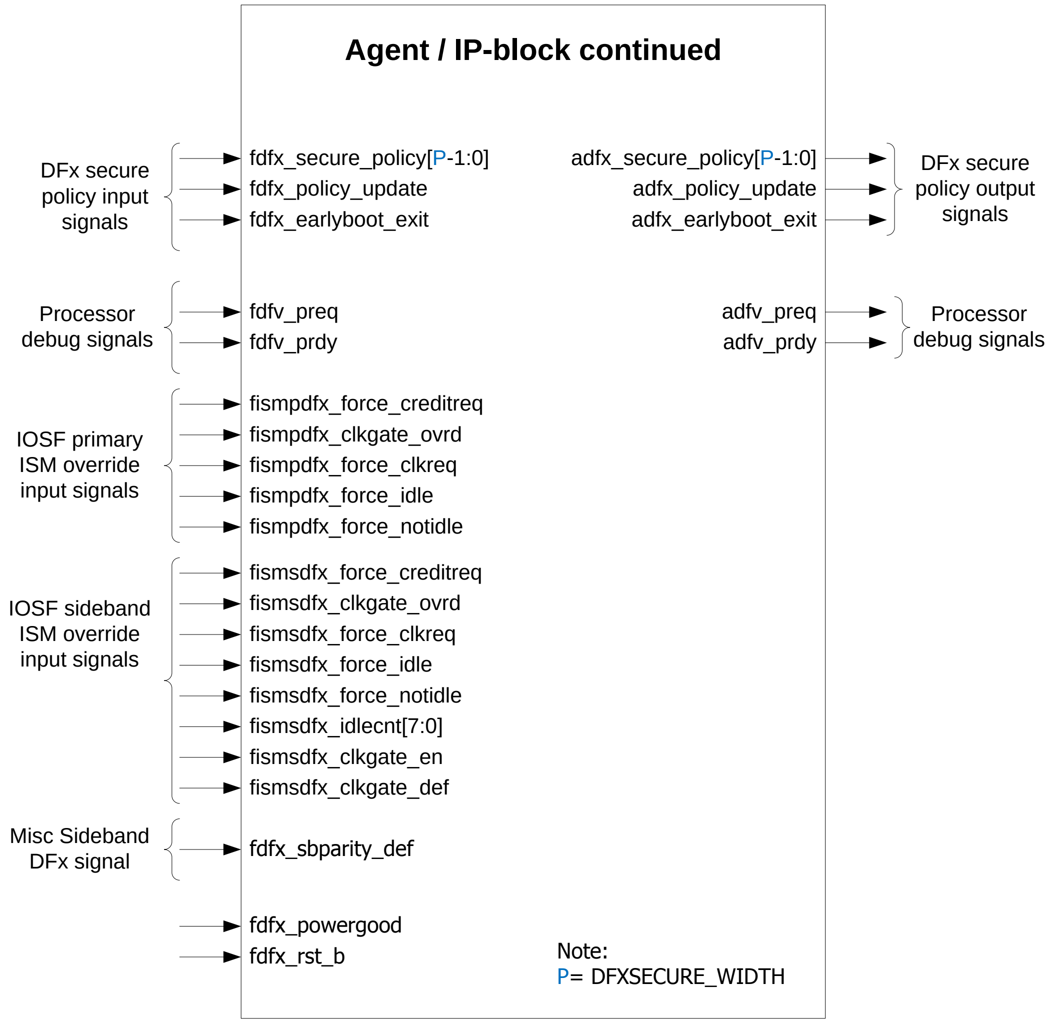


Table 7-7. VISA DFV signal descriptions

Signal	I/O	R/O/C[1]	Description
Fabric sourced: VISA debug bus			
v1m_fvisa_dbgbus [(FVISAWIDTH*8)-1:0]	O	C	**Fabric VISA Debug Bus:** This is the legacy VISA definition for IP debug observability. The debug bus is based on a byte lane mux width. The minimum grouping is one byte (8 bits) and each increment is one byte. If FVISAWIDTH = 2 then 2 * 8 = 16 bit width (*_dbgbus[15:0).
v1m_fvisa_clk [FVISAWIDTH -1:0]	O	C	**Fabric VISA Debug Bus Clock:** This signal is the clock reference for each byte lane of the debug bus. There is one clock per byte lane. This is the legacy VISA clock definition.

Signal	I/O	R/O/C[1]	Description
v1m_fvisa_dvalid [FVISAWIDTH-1:0]	O	C	**Fabric Data Valid:** This signal indicates when the VISA debug data from avisa_dbgbus is valid when this agent implements the optional VISA sync gasket. Each data valid corresponds to a byte lane. This is the legacy VISA data valid definition.
v2m_fvisa_dbgbus [(FVISA2MWIDTH*16)-1:0]	O	C	**Fabric VISA 2-byte Mux (v2m) Debug Bus:** VISA observability debug bus for this fabric IP based on a 16-bit word lane mux width. The minimum grouping is one word and each increment is a word lane. If FVISA2MWIDTH = 2 then 2 * 16 = 32 bit width (*_dbgbus[31:0).
v2m_fvisa_clk [FVISA2MWIDTH -1:0]	O	C	**Fabric VISA V2M Clock:** This signal is the clock reference for each word lane of the debug bus. There is one clock per word lane.
v2m_fvisa_dvalid [FVISA2MWIDTH-1:0]	O	C	**Fabric VISA V2M Data Valid :** This signal indicates when the VISA debug data from *_dbgbus is valid when this agent implements the optional VISA sync gasket. Each data valid corresponds to a word lane.
v4m_fvisa_dbgbus [(FVISA4MWIDTH*32)-1:0]	O	C	**Fabric VISA 4-byte Mux (v4m) Debug Bus:** VISA observability debug bus for this fabric IP based on a 32-bit double word lane mux width. The minimum grouping is one Dword and each increment is a word lane. If FVISA4MWIDTH = 2, then 2 * 32 = 64 bit width (*_dbgbus[63:0).
v4m_fvisa_clk [FVISA4MWIDTH -1:0]	O	C	**Fabric VISA V4M Clock:** This signal is the clock reference for each Dword lane of the debug bus. There is one clock per word lane.
v4m_fvisa_dvalid [FVISA4MWIDTH-1:0]	O	C	**Fabric VISA V4M Data Valid:** This signal indicates when the VISA debug data from *_dbgbus is valid when this agent implements the optional VISA sync gasket. Each data valid corresponds to a Dword lane.
v8m_fvisa_dbgbus [(FVISA8MWIDTH*64)-1:0]	O	C	**Fabric VISA 8-byte Mux (v8m) Debug Bus:** VISA observability debug bus for this fabric IP based on a 64-bit quad-word lane mux width. The minimum grouping is one Qword and each increment is a word lane. If FVISA8MWIDTH = 2, then v8m_avisa_dbgbus = 2 * 64 = 128 bit width.
v8m_fvisa_clk [FVISA8MWIDTH -1:0]	O	C	**Fabric VISA V8M Clock:** This signal is the clock reference for each Qword lane of the debug bus. There is one clock per word lane.
v8m_fvisa_dvalid [FVISA8MWIDTH-1:0]	O	C	**Fabric VISA V8M Data Valid:** This signal indicates when the VISA debug data from *_dbgbus is valid when this agent implements the optional VISA sync gasket. Each data valid corresponds to a quad word lane.
Agent sourced: VISA debug bus			
v1m_avisa_dbgbus [(AVISAWIDTH*8)-1:0]	O	O	**Agent VISA 1-byte Mux (v1m) Debug Bus:** This is the legacy VISA definition for IP debug observability. The debug bus is based on a byte lane mux width. The minimum grouping is one byte (8 bits) and each increment is one byte. If AVISAWIDTH = 2 then 2 * 8 = 16 bit width (*_dbgbus[15:0) .

Signal	I/O	R/O/C[1]	Description
v1m_avisa_clk [AVISAWIDTH -1:0]	O	O	**Agent VISA V1M Clock:** This signal is the clock reference for each byte lane of the debug bus. There is one clock per byte lane. This is the legacy VISA clock definition.
v1m_avisa_dvalid [AVISAWIDTH-1:0]	O	O	**Agent Data V1M Valid:** This signal indicates when the VISA debug data from *_dbgbus is valid when this agent implements the optional VISA sync gasket. Each data valid corresponds to a byte lane. This is the legacy VISA data valid definition.
v2m_avisa_dbgbus [(AVISA2MWIDTH*16)-1:0]	O	O	**Agent VISA 2-byte Mux (v2m) Debug Bus:** VISA observability debug bus for this agent based on a 16-bit word lane mux width. The minimum grouping is one word and each increment is a word lane. If AVISA2MWIDTH = 2 then 2 * 16 = 32 bit width (*_dbgbus[31:0).
v2m_avisa_clk [AVISA2MWIDTH -1:0]	O	O	**Agent VISA V2M Clock:** This signal is the clock reference for each word lane of the debug bus. There is one clock per word lane.
v2m_avisa_dvalid [AVISA2MWIDTH-1:0]	O	O	**Agent VISA V2M Data Valid:** This signal indicates when the VISA debug data from *_dbgbus is valid when this agent implements the optional VISA sync gasket. Each data valid corresponds to a word lane.
v4m_avisa_dbgbus [(AVISA4MWIDTH*32)-1:0]	O	O	**Agent VISA 4-byte Mux (v4m) Debug Bus:** VISA observability debug bus for this agent based on a 32-bit double word lane mux width. The minimum grouping is one Dword and each increment is a word lane. If AVISA4MWIDTH = 2, then 2 * 32 = 64 bit width (*_dbgbus[63:0).
v4m_avisa_clk [AVISA4MWIDTH -1:0]	O	O	**Agent VISA V4M Clock:** This signal is the clock reference for each Dword lane of the debug bus. There is one clock per word lane.
v4m_avisa_dvalid [AVISA4MWIDTH-1:0]	O	O	**Agent VISA V4M Data Valid:** This signal indicates when the VISA debug data from *_dbgbus is valid when this agent implements the optional VISA sync gasket. Each data valid corresponds to a Dword lane.
v8m_avisa_dbgbus [(AVISA8MWIDTH*64)-1:0]	O	O	**Agent VISA 8-byte Mux (v8m) Debug Bus:** VISA observability debug bus for this agent based on a 64-bit quad-word lane mux width. The minimum grouping is one Qword and each increment is a word lane. If AVISA8MWIDTH = 2, then v8m_avisa_dbgbus = 2 * 64 = 128 bit width.
v8m_avisa_clk [AVISA8MWIDTH -1:0]	O	O	**Agent VISA V8M Clock:** This signal is the clock reference for each Qword lane of the debug bus. There is one clock per word lane.
v8m_avisa_dvalid [AVISA8MWIDTH-1:0]	O	O	**Agent VISA V8M Data Valid:** This signal indicates when the VISA debug data from *_dbgbus is valid when this agent implements the optional VISA sync gasket. Each data valid corresponds to a quad word lane.

Signal	I/O	R/O/C[1]	Description
VISA serial configuration bus to IPs (SIPs/HIPs)			
fvisa_serstrb	I	R	**Fabric Serial Strobe:** This strobe indicates when the data bit on fvisa_serdata is valid. Note: This signal is connected to serial_cfg_in[0] when using the VISA insertion tool.
fvisa_frame	I	R	**Fabric VISA Frame:** This signal frames the serial register access packet that is sent from the VISA central controller to the PLMs and then on to the ULMs. Note: This signal is connected to serial_cfg_in[1] when using the VISA insertion tool.
fvisa_serdata	I	R	**Fabric Serial Data:** This is the data stream sent from the VISA central controller to the PLMs then on to the ULMs. The data stream contains control, address and data. The data is the local VISA mux control register information. Note: This signal is connected to serial_cfg_in[2] when using the VISA insertion tool.
avisa_ser_rdata	O	C	**Agent Serial Read Data:** Agent read data return signal back to the VISA register controller. Conditional: This signal is required when VISA 4.x is used and the read capability is enabled.
IP that contains the VISA Register Controller (e.g. Intel Trace Hub)			
avisa_serstrb	O	O	**Agent Serial Strobe:** This strobe indicates when the data bit on avisa_serdata is valid. Note: This signal is connected to serial_cfg_out[0] when using the VISA insertion tool.
avisa_frame	O	O	**Agent VISA Frame:** This signal frames the serial register access packet that is sent from the VISA central controller to the PLMs and then on to the ULMs. Note: This signal is connected to serial_cfg_out[1] when using the VISA insertion tool.
avisa_serdata	O	O	**Agent Serial Data:** This is the data stream sent from the VISA central controller to the PLMs then on to the ULMs. The data stream contains control, address and data. The data is the local VISA mux control register information. Note: This signal is connected to serial_cfg_out[2] when using the VISA insertion tool.
fvisa_ser_rdata	I	C	**Fabric Serial Read Data:** Fabric serial read data input from VISA network of PLMs. This signal is the read return data from IP ULMs. Conditional: This signal is required when VISA 4.x is used and the read capability is enabled.
VISA strap pin interface signals			

Signal	I/O	R/O/C[1]	Description
fvisa_startid[8:0]	I	O	**Fabric VISA starting ID[8:0]:** This signal bus identifies the starting ULM/PLM unit ID for the muxes within the IP-block. It is implementation dependent for the IP-block developer to manage the number of unit IDs for the overall IP using the startid and endid signal groups. Note: This pin interface is expected to be a strap value for the ULMs (or PLMs) and not a fabric bus.
fvisa_endid[8:0]	I	O	**Fabric VISA ending ID[8:0]:** This signal bus identifies the ending ULM/PLM unit ID for the muxes with the HIP. It is implementation dependent for the HIP developer to manage the number of unit IDs for the overall IP using the startid and endid signal groups. The endid is required if the register IDs are exhausted for one startid value. There are 32 IDs available and a typical ULM with 4 lanes consumes 5 IDs. Note: This pin interface is expected to be a strap value for the ULMs (or PLMs) and not a fabric bus. Note: This signal interface is intended only for HIPs. Any SIP should not use this interface because the SoC integration team will use the auto-ID feature in the VISA toolkit.

NOTE:
1. R = required, O = optional, C = conditional. Conditional means that if the feature is used then the signals labeled as conditional become required.

Table 7-8. Trigger DFV signals

Signal	I/O	R/O/C[1]	Description
asta_trigsrc [TRIGSRCWIDTH-1:0]	O	C	**Agent SoC Trigger Architecture (STA) Trigger Source [N-1:0]:** This is an asynchronous output signal from this IP-block's internal logic that is generating triggers for use by the SoC. There may be TRIGSRCWIDTH number of trigger outputs from this agent. Note: This is signal is expected to be a multi-cycle path for the backend timing analysis tools.
fsta_trigev [TRIGEVWIDTH-1:0]	I	C	**Fabric SoC Trigger Architecture (STA) Trigger Event [N-1:0]:** This is a debug trigger event input from either the regional or master DFx unit to this agent. This event is used for asserting response functions within the agent for debug, validation, and survivability features where an event driven assertion alters the behavior of the agent. One example is an error injection logic based on an event generated from a trigger source. There may be TRIGEVWIDTH number of trigger event inputs to this agent. Note: This is signal is expected to be a multi-cycle path for the backend timing analysis tools.

asta_trigsrc_sync [TRIGSRC_SYNC_WIDTH-1:0]	O	C	**Agent SoC Trigger Architecture (STA) Trigger Source Sync[N-1:0]:** This is a synchronous output signal from an internal logic block that is generating triggers for use by the regional and master DFx units in the fabric. There may be TRIGSRC_SYNC_WIDTH number of trigger outputs from this agent. Note: This is signal is a synchronous path that requires a single cycle or flop stage to reach the destination. This signal cannot be a multi-cycle path.
fsta_trigev_sync [TRIGEV_SYNC_WIDTH-1:0]	I	C	**Fabric SoC Trigger Architecture (STA) Trigger Event Sync [N-1:0]:** This is a debug trigger event input from either the regional or master DFx unit to this agent. This event is used for asserting response functions within the agent for debug, validation, and survivability features where an event driven assertion alters the behavior of the agent. One example is an error injection logic based on an event generated from a trigger source. There may be TRIGEV_SYNC_WIDTH number of trigger event inputs to this agent. Note: This is signal is a synchronous path that requires a single cycle or flop stage to reach the destination. This signal cannot be a multi-cycle path.
asta_dfxact [NUM_OF_DFXACT-1:0]	O	C	**Agent SoC Trigger Architecutre (STA) DFx Action[P-1:0].** This signal is intended to be a DFx action output from a trigger control block. For Chassis DFx Gen2 this would apply to the Cluster Trigger Block (CTB). DFx actions are driving passive DFx IPs that alter the behavior of the SoC. For example, an array/freeze/dump is a passive consumer of a DFx action. A trigger block such as a micro breakpoint controller would use the fsta_trigev signal. **Note:** this signal is for new IP and SoC developments.
fsta_dfxact [NUM_OF_DFXACT-1:0]	I	C	**Fabric SoC Trigger Architecutre (STA) DFx Action[P-1:0].** This signal enables a DFx action to occur such as a clock stop (clock freeze), array/freeze/dump (AFD), launch a TAP2SB transaction, etc. It is a passive DFx feature that a trigger event is expected to enable. It is not intended for active trigger IPs such as micro breakpoint controllers or transaction match/mask IPs (similar to PSF Fabric Trace Hook). **Note:** this signal is for new IP and SoC developments.
fsta_dfxact_afd	I	C	**Fabric SoC trigger arch DFx action for array/freeze/dump:** Defined in Section 5.5. Reprinted here to show all of the {f,a}sta_* signals.
fsta_afd_en	I	C	**Fabric SoC trigger arch DFx action enable:** Defined in Section 5.5. Reprinted here to show all of the {f,a}sta_* signals.

> **NOTE:**
> [1]R = required, O = optional, C = conditional. Conditional means that if the feature is used then the signals labeled as conditional become required.

Table 7-9. DFV security signals

Signal	I/O	R/O/C[1]	Description
fdfx_secure_policy [DFXSECURE_WIDTH-1:0]	I	R	**Fabric DFx security policy.** This bus is a binary encoded value of the security policy that is the current state of the SoC. The parameter value is constant for a given SoC generation and aligned with a process node. All IP-blocks must have the same width. For this revision of the IOSF DFx HAS: DFXSECURE_WIDTH = 4 Note: An agent/IP-block may have two DFx secure policy plug-in blocks one for sTAP and one for the agent/IP-block.
fdfx_policy_update	I	R	**Fabric DFx policy update.** This is the latch enable signal to capture the policy value to prevent glitches. 0: Latch values 1: Update to new policy value Note: An agent/IP-block may have two DFx secure policy plug-in blocks one for sTAP and one for the agent/IP-block.
fdfx_earlyboot_exit	I	R	**Fabric DFx early boot exit.** This signal indicates when the early boot debug window is closed. 0: Debug capabilities are available during this phase of the boot flow 1: DFx security policy must be used Note: An agent/IP-block may have two DFx secure policy plug-in blocks one for sTAP and one for the agent/IP-block.
adfx_secure_policy [DFXSECURE_WIDTH-1:0]	O	O	**Agent DFx security policy.** This bus is a binary encoded value of the security policy that is the current state of the SoC. The parameter value is constant for a given SoC generation and aligned with a process node. All IP-blocks must have the same width. The security agent will output this bus as the source of the security policy information but this may be used as a pass-through from the agent to another IP-block within an agent. For 14nm chassis: DFXSECURE_WIDTH = 4
adfx_policy_update	O	O	**Agent DFx policy update.** This signal is the latch enables to capture the policy value to prevent glitches. 0: Latch values 1: Update to new policy value

adfx_earlyboot_exit	O	O	**Fabric DFx early boot exit.** This signal indicates when the early boot debug window is closed. 0: Debug capabilities are available during this phase of the boot flow 1: DFx security policy must be used
NOTE: [1]R = required, O = optional, C = conditional. Conditional means that if the feature is used then the signals labeled as conditional become required.			

Table 7-10. ISM override DFV signals

Signal	I/O	R/O/C[1]	Description
Primary ISM debug override signals			
fismpdfx_force_creditreq	I	R	**Fabric Primary ISM DFx force credit request:** This input forces the primary Idle State Machine to request its credits. 0: normal operation 1: force IP to request its credits Requirements Note: If an agent/IP-block provides a TAP TDR to drive the ISM override signals within the agent (IP-block) then this signal can be waived from being required on the IOSF DFx interface. This input signal may be distributed to N number of primary interfaces within this IP. There is no requirement to independently control each ISM override.
fismpdfx_clkgate_ovrd	I	R	**Fabric Primary ISM DFx clock gate override:** This input overrides the local clock gating mux to the Idle State Machine, meaning, that it enables the clock locally. To use this feature for power measurements the SoC must drive this signal individually meaning each Sideband ISM and each primary ISM from any local TAP for this to be effective. 0: normal operation 1: override clock gate by forcing the clock to turn on. Requirements Note: If an agent/IP-block provides a TAP TDR to drive the ISM override signals within the agent (IP-block) then this signal can be waived from being required on the IOSF DFx interface. This is signal is optional because it can be controlled internally to the IP-block with registers bits on the Sideband interface. This assumes the register override bits are on the Sideband clock domain and the fismsdfx_clkgate_ovrd is implemented to override the sideband clock gate (required). This input signal may be distributed to N number of primary interfaces within this IP. There is no requirement to independently control each ISM override.

fismpdfx_force_clkreq	I	R	**Fabric Primary ISM DFx force clock request:** This input forces the agent to request a clock active condition in the Idle State Machine. This use model assumes to be in clock idle and we force a clock active to the fabric. 0: normal operation 1: force clock request Requirements Note: If an agent/IP-block provides a TAP TDR to drive the ISM override signals within the agent (IP-block) then this signal can be waived from being required on the IOSF DFx interface. This input signal may be distributed to N number of primary interfaces within this IP. There is no requirement to independently control each ISM override.
fismpdfx_force_idle	I	R	**Fabric Primary ISM DFx force idle:** This input forces the agent to the idle state of the Idle State Machine. 0: normal operation 1: force idle Requirements Note: If an agent/IP-block provides a TAP TDR to drive the ISM override signals within the agent (IP-block) then this signal can be waived from being required on the IOSF DFx interface. This input signal may be distributed to N number of primary interfaces within this IP. There is no requirement to independently control each ISM override.
fismpdfx_force_notidle	I	R	**Fabric Primary ISM DFx force not idle:** This input forces the agent's Idle State Machine not to go to idle. 0: normal operation 1: force not idle Requirements Note: If an agent/IP-block provides a TAP TDR to drive the ISM override signals within the agent (IP-block) then this signal can be waived from being required on the IOSF DFx interface. This input signal may be distributed to N number of primary interfaces within this IP. There is no requirement to independently control each ISM override.
Sideband ISM debug override signals			

fismsdfx_force_creditreq	I	R	**Fabric Sideband ISM DFx force credit request:** This input forces the primary Idle State Machine to request its credits. 0: normal operation 1: force IP to request its credits Requirements Note: If an agent/IP-block provides a TAP TDR to drive the ISM override signals within the agent (IP-block) then this signal can be waived from being required on the IOSF DFx interface. This input signal may be distributed to N number of Sideband interfaces within this IP. There is no requirement to independently control each ISM override.
fismsdfx_clkgate_ovrd	I	R	**Fabric Sideband ISM DFx clock gate override:** This input overrides the local clock gating mux to the Idle State Machine, meaning, that it enables the clock locally. 0: normal operation 1: override clock gate by forcing the clock to turn on Note1: This signal is connected to the ISM signal named jta_clkgate_ovrd. Use of this signal as a bus is optional. Requirements Note: If an agent/IP-block provides a TAP TDR to drive the ISM override signals within the agent (IP-block) then this signal can be waived from being required on the IOSF DFx interface. This input signal may be distributed to N number of Sideband interfaces within this IP. There is no requirement to independently control each ISM override.
fismsdfx_force_clkreq	I	R	**Fabric Sideband ISM DFx force clock request:** This input forces the agent to request a clock active condition in the Idle State Machine. This use model assumes to be in clock idle and we force a clock active to the fabric. 0: normal operation 1: force clock request Note1: This signal is connected to the ISM signal named jta_force_clkreq. Use of this signal as a bus is optional. Requirements Note: If an agent/IP-block provides a TAP TDR to drive the ISM override signals within the agent (IP-block) then this signal can be waived from being required on the IOSF DFx interface. This input signal may be distributed to N number of Sideband interfaces within this IP. There is no requirement to independently control each ISM override.

fismsdfx_force_idle	I	R	**Fabric Sideband ISM DFx force idle:** This input forces the agent to the idle state of the Idle State Machine. 0: normal operation 1: force idle Note1: This signal is connected to the ISM signal named jta_force_idle. Use of this signal as a bus is optional. Requirements Note: If an agent/IP-block provides a TAP TDR to drive the ISM override signals within the agent (IP-block) then this signal can be waived from being required on the IOSF DFx interface. This input signal may be distributed to N number of Sideband interfaces within this IP. There is no requirement to independently control each ISM override.
fismsdfx_force_notidle	I	R	**Fabric Sideband ISM DFx force not idle:** This input forces the agent's Idle State Machine not to go to idle. 0: normal operation 1: force not idle Note1: This signal is connected to the ISM signal named jta_force_notidle. Use of this signal as a bus is optional. Requirements Note: If an agent/IP-block provides a TAP TDR to drive the ISM override signals within the agent (IP-block) then this signal can be waived from being required on the IOSF DFx interface. This input signal may be distributed to N number of Sideband interfaces within this IP. There is no requirement to independently control each ISM override.
fismsdfx_idlecnt[7:0]	I	R	**Fabric Sideband ISM DFx idle count:** This bus sets the idle counter default value that determines when the endpoint should transition to IDLE_REQ. Default value is 8'h10 (16 dec). This input signal may be distributed to N number of Sideband interfaces within this IP. There is no requirement to independently control each ISM override.
fismsdfx_clkgate_en	I	R	**Fabric Sideband ISM DFx clock gate enable:** This input forces the agent's Idle State Machine not to go to idle. 0: ISM never leaves the ACTIVE state which forces the clock to remain on. 1: Normal operation. Allows ISM to leave ACTIVE. Clocking gating occurs normally once in IDLE. Default value = 1'b1 This input signal may be distributed to N number of Sideband interfaces within this IP. There is no requirement to independently control each ISM override.

fismsdfx_clkgate_def	I	R	**Fabric Sideband ISM DFx clock gate defeature:** This signal will defeature the clock gating enable signal (fismsdfx_clkgate_def). 0: normal operation 1: Disable clock gating feature Default value = 1'b0 This input signal may be distributed to N number of Sideband interfaces within this IP. There is no requirement to independently control each ISM override.
NOTE: [1]R = required, O = optional, C = conditional. Conditional means that if the feature is used then the signals labeled as conditional become required.			

Table 7-11. General DFV signals

Signal	I/O	R/O/C[1]	Description
General debug signals			
fdfx_sbparity_def	I	R	**Fabric DFx Sideband parity defeature.** This signal disables parity checking within the Sideband endpoint. When asserted all parity must be treated as good parity. Incoming payload from the Sideband router will be propagated to the agent as if it was good parity regardless of the parity logic result. Also, the outgoing transactions will have the parity signal (sbe_sbi_parity_err_out) deasserted even if the parity generation logic created a bad result. 0: Enable parity checking 1: Disable parity checking
fdfx_pgcb_bypass [NUM_OF_PGCBS-1:0]	I	C	**Fabric DFx PGCB bypass.** This signal controls the Power Gate Common Block's bypass muxes to enable a DFx override signal to control the enabling of this IP-block's PGCB instantiation. 0: Normal PGCB operation 1: PGCB is bypassed and forces the override value **Note:** This signal is part of the Misc DFT signal group but remain here for legacy reasons. Refer to section 6.2 for more information.

fdfx_pgcb_ovr [NUM_OF_PGCBS-1:0]	I	C	**Fabric DFx PGCB override (value).** This signal controls the DFx sequencer inside of the PGCB block. The DFx sequencer automates the activation/deactivation of the power management control signals to power up or down the domain that this PGCB controls. **0:** Power gate device (PGD) is force on, meaning, the IP-block will be powered up **1:** Power gate device (PGD) is force off, meaning, the IP-block will be powered down **Note:** This signal is part of the Misc DFT signal group but remain here for legacy reasons. Refer to section 6.2 for more information.
fdfx_preq	I	C	**Fabric DFV preq.** This signal forces the processor to break execution and halt for debug operations. If an IP-block contains a Minute-IA processor (Lakemont rev2.1 or later) or processor that supports entering probe mode with a hardware assertion then this signal is required. **0:** Normal processor operation **1:** preq asserted. **Note:** This signal is defined active high. Previous IP-block releases with active low signaling ("_b" in the signal name) will be waived.
adfx_prdy	O	C	**Agent DFV prdy.** This signal indicates that a processor has entered into probe mode due to a previous preq assertion or due to an internal event. If an IP-block contains a Minute-IA processor (Lakemont rev2.1 or later) or processor that supports entering probe mode with a hardware assertion then this signal is required. **0:** Normal processor operation **1:** prdy is asserted **Note:** This signal is defined active high. Previous IP-block releases with active low signaling ("_b" in the signal name) will be waived.
fdfx_powergood	I	R	Refer to section 6.2. Note: This signal is equivalent to **dfx_powergood_rst_b**.
fdfx_rst_b	I	C	**Fabric DFx reset bar:** This signal is for resetting the VISA ULM only. It is the responsibility of the IP-block developer to internally logically combine this signal with force_rst_b if a PGCB block is instantiated in the IP. **Note 1:** Soft-IP blocks are required to include fdfx_rst_b on the IP interface if this IP has VISA. **Note 2:** If a hard-IP block is designated as a secure IO IP then it must implement fdfx_rst_b connecting to all VISA ULMs/PLMs within the IP. Otherwise, for all other hard IP-blocks it is optional.

NOTE:
[1]R = required, O = optional, C = conditional. Conditional means that if the feature is used then the signals labeled as conditional become required.

7.9 Transaction Cycle/Data Flows

Not applicable

7.10 Ordering/Coherency Rules

Not applicable

7.11 Performance/Bandwidth Analysis

Not applicable

7.12 Exception List Requirements

Not applicable

7.13 Programming Model

Not applicable

7.14 Power Management Capabilities

Not applicable

7.15 Security Feature Requirements

IOSF DFx is compliant to the Chassis DFx Security Framework specification.

Link:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/chassisWG/SitePages/Home.aspx

Directory:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/chassisWG/Security/HAS/DFx Security Framework

VISA security

7.16 DFx Requirements

Not applicable

7.16.1 DFV/DFD Requirements

Not applicable

7.16.2 DFT Requirements

VISA rev2.12 (or later) supports scan.

7.16.2.1 Burn-in Specific Requirements

Not applicable

7.16.3 DFM Requirements

Not applicable

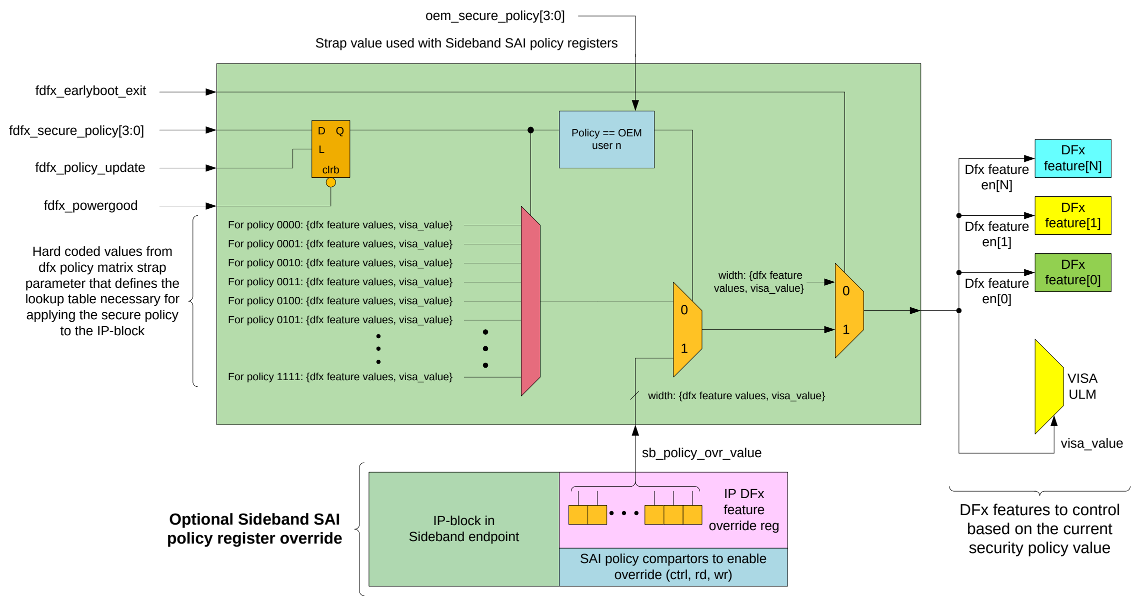
7.17 Legacy or obsolete use models

7.17.1 DFx secure plugin

The following information is for reference only. The OEM secure policy value detection logic and Sideband override feature is obsolete for Chassis DFx 2.1.

A Sideband endpoint may implement a secure DFx override capability (shown in) where a set of SAI policy registers (control, read, write) allow access to a register that contains an override value equal to the number of DFx features to enable plus the two VISA values. It is IP-block specific on how the Sideband override is implemented but it must be consistent with the use of SAI policy registers in the main IOSF HAS and the Chassis DFx security framework HAS.

Figure 7-17. DFx security plug-in with optional Sideband override



To use the Sideband endpoint override option, there are two sets of signal buses that must be used and one parameter to enable the feature. When the parameter USE_SB_OVR is set to logic 1 it will enable the policy override. The user sets the oem_secure_policy[3:0] to one of the user defined (unused) policy states. These are listed in Table 7-4 as user 1 through 8 unlocked (hex values 0x7 through 0xE). Then the IP-block developer provides the SAI policy registers to enable the override according to the security framework HAS document. Once enabled with an authenticated SAI source, a register in the IP-block contains the bits that are the same width as one row in the policy matrix (namely, [NUM_OF_FEATURES_TO _SECURE+1:0]). The output of the register is connected to sb_policy_ovr_value. describes the how the policy compare block determines if the Sideband is used or not.

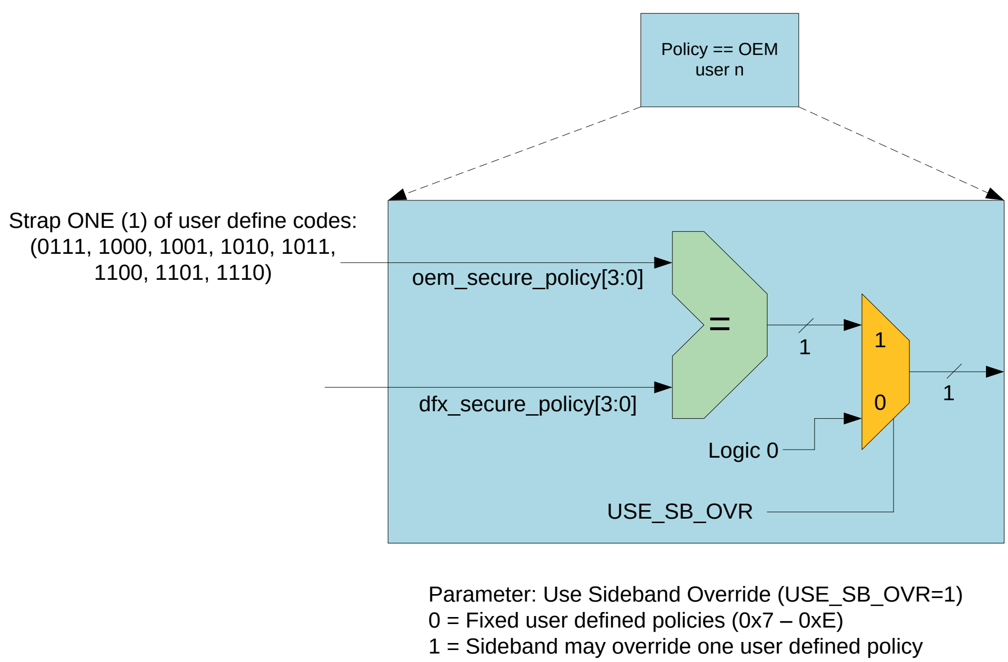
Table 7-12. DFx secure plugin obsolete signal list

Signal	I/O	Description
sb_policy_ovr_value [M+1:0]	I	**Sideband policy override value.** This signal bus is the same width as the policy matrix parameter. It is the number of features to secure plus the two bits from the concatenation of the VISA signals. **Chassis 2.1:** This feature is obsolete. The implementation team must connect sb_policy_ovr_value = 0.
oem_secure_policy [N-1:0]	I	**OEM specific secure policy value.** This is a strap on the IP-block that allows a user-defined value specifically between 0x7 and 0xE to use the sideband policy override value (sb_policy_ovr_value). If the Sideband is not used then this bus input should be set to the OEM unlock policy value of 0x0. **Chassis 2.1:** Set to 0x0.

Table 7-13. DFx secure plugin obsolete parameter list

Parameter name	Parameter value(s)	Comments
USE_SB_OVR	{0, 1}	Chassis 2.1: Set to 0. This feature is obsolete. If this parameter is set then the plug-in will use the sb_policy_ovr_value input. 0: Disregard sb_policy_ovr_value and oem_secure_policy. Internally, the Sideband override (OEM control override) mux is forced to logic 0. 1: The sb_policy_ovr_value and oem_secure_policy are active and the IP-block owner must set the value according to the DFx use model.

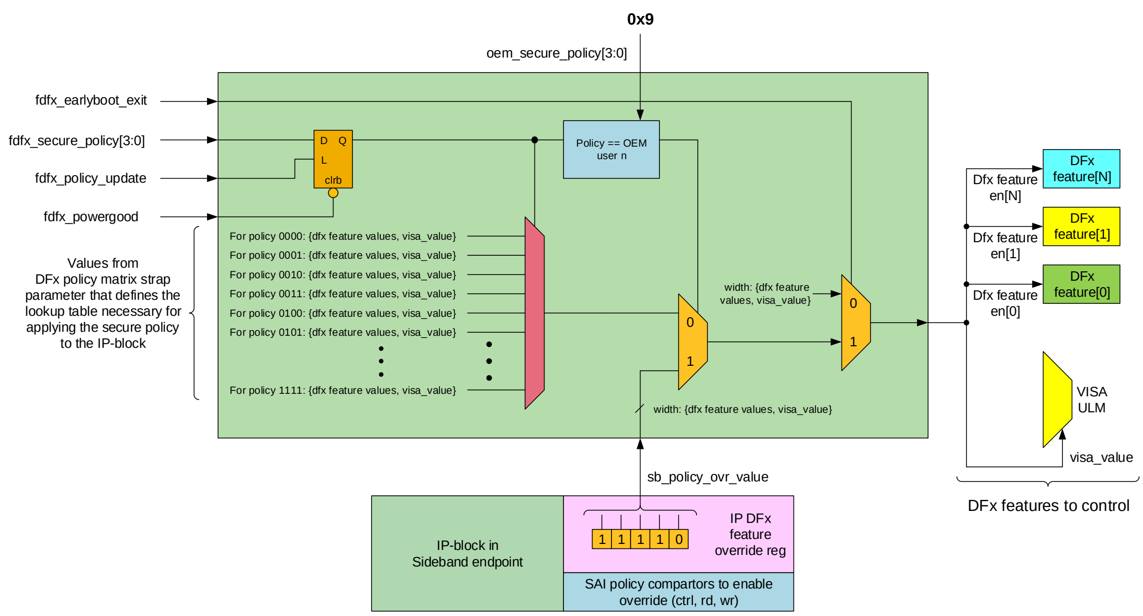
Figure 7-18. Sideband override for targeted OEM enabling



7.17.1.1 DFx secure plugin with Sideband override example

This use model reuses the example of an IP-block defined in section 7.4.4 but in this example, (shown in Figure 7-19) it is desired to use one of the user defined policy values (0x9 through 0xE) to provide a programmable override using the IP-block's Sideband endpoint. This endpoint must implement the SAI policy registers so that the override only occurs via an authenticated agent most likely it is a trusted firmware agent. In this example, 0x9 is available for overriding the DFx features. The Sideband endpoint implements the SAI policy registers to enable access the DFx feature override register. It is implementation specific how this occurs and the IP-block developer must comply with security requirements in the Chassis security framework specification. The width of the register is the same as the value in the policy matrix, namely, [NUM_OF_FEATURES_TO _SECURE+1:0]. This register output is then connected to the sb_policy_ovr_value. When the fdfx_secure_policy bus is equal to 0x9 and then the value in the local Sideband endpoint register is used as the new set of DFx feature enables. In this example, all of the DFx features are enabled (dfx_feature_en[2:0] = 3’b111) with VISA output set to 2’b10 which keeps the IP’s signal list at the orange group level. The user has full access to the IOSF ODLAT transaction matching on all commands, address and data with FeatureX debug feature.

Figure 7-19. DFx secure plug-in example with Sideband override



§

8 Register Description

The base registers are listed in the DFx IP high level architectural specifications for each feature in this document. Specific register definitions based each SoC’s specific parameters maybe found in the SoC DFx HAS documents.

§

9 Implementation Details

The reader should refer to specific SoC MAS documents or individual DFx feature High Architectural Specifications.

§

10 Signal Interface

The signal interfaces are described in each chapter to make it easier for the reader to refer to signal information within the chapter. However, this spec is compliant to the IDGa/CSA template HAS (as of this writing) so the information is duplicated here.

10.1 TAP Signal Interface Description

A graphical view of the TAP interface signals are shown in Figure 10-1 and Figure 10-2. This diagram summarizes the available TAP signals that support the usage models described previously. Table 10-1 presents a complete list of signal names. The integration team has the freedom to assign any reasonable number of characters as a prefix or suffix as applied to the root name to identify the signal in a full chip SoC.

Figure 10-1. TAP signal interface graphical view 1

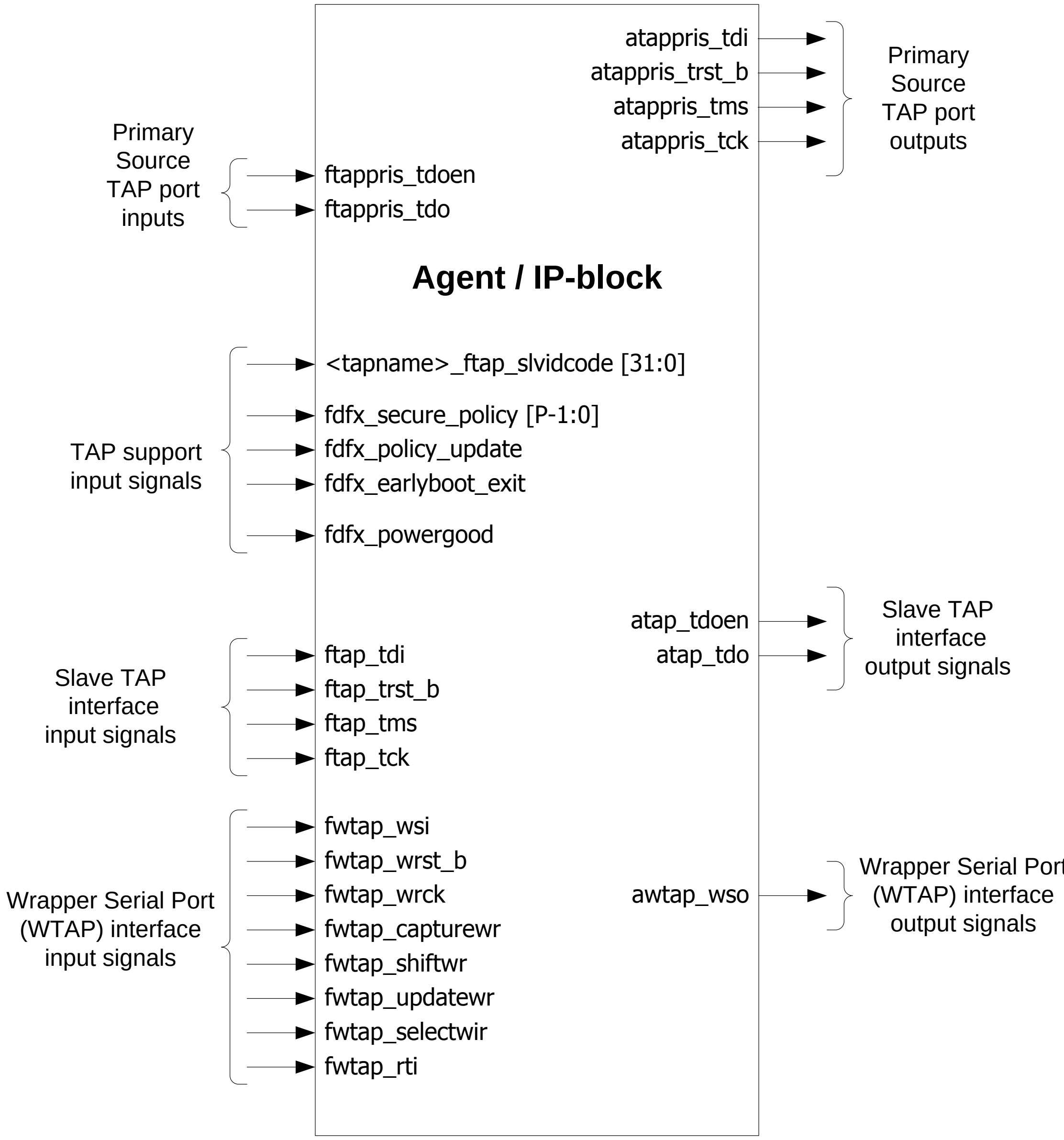


Figure 10-2. TAP signal interface graphical view 2

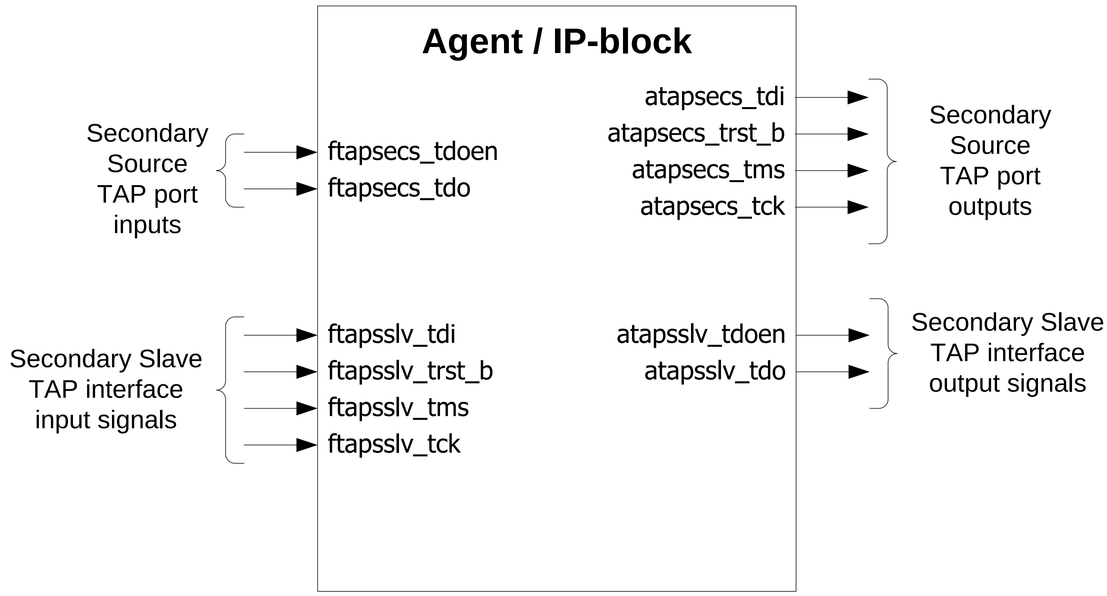


Table 10-1. TAP interface signal description

Signal	I/O	R/O/C[1]	Description
Slave TAP signals			
ftap_tck[2]	I	C	**Fabric TAP clock:** This is the TAP clock from the fabric that originates from package pins connected to the TAP network.
ftap_tms	I	C	**Fabric TAP test mode:** This is the TAP finite state machine test mode control signal from the fabric that originates from package pins connected to the TAP network.
ftap_trst_b	I	C	**Fabric TAP reset bar:** This is the TAP reset from the fabric that may originate from a package pin connecting to the master TAP controller and the TAP network.
ftap_tdi	I	C	**Fabric Test Data In:** This is a test data in (TDI) from the fabric that originates from the master TAP controller through the TAP network.
atap_tdo	O	C	**Agent Test Data Out:** This is the test data out (TDO) from this agent.
atap_tdoen	O	C	**Agent Test Data Out Enable:** This is the test data out enable to drive the tristate enable on the TDO pad control. atap_tdoen = Shift-DR OR Shift_IR

Signal	I/O	R/O/C[1]	Description
Agent TAP support signals			
<tapname>_ftap_slvidcode[31:0]	I	R	**Fabric Slave ID Code:** This is a signal bus port provides the slave ID code for the sTAP. Note1: This signal bus is required if a TAP interface is required. Note2: Use of the tapname is optional. It is useful for agents/IP-blocks with more than one TAP within the module to uniquely identify the codes. The SIP TAP RTL IP-block will not provide a prefix.
fdfx_powergood	I	R	Refer to section 6.2. Note: This signal is equivalent to **dfx_powergood_rst_b**.
fdfx_secure_policy [DFXSECURE_WIDTH-1:0]	I	R	**Fabric DFx security policy.** This bus is a binary encoded value of the security policy that is the current state of the SoC. The parameter value is constant for a given SoC generation and aligned with a process node. All IP-blocks must have the same width. For this revision of the IOSF DFx HAS: DFXSECURE_WIDTH = 4 Note: An agent/IP-block may have two DFx secure policy plug-in blocks one for sTAP and one for the agent/IP-block. It is the IP-block's responsibility to connect both together. Note2:
fdfx_policy_update	I	R	**Fabric DFx policy update.** This signal is the latch enable to capture the policy value to prevent glitches. 0: Latch values 1: Update to new policy value Note: An agent/IP-block may have two DFx secure policy plug-in blocks one for sTAP and one for the agent/IP-block. It is the IP-block's responsibility to connect both together.
fdfx_earlyboot_exit	I	R	**Fabric DFx early boot exit.** This signal indicates when the early boot debug window is closed. 0: Debug capabilities are available during this phase of the boot flow 1: DFx security policy must be used Note: An agent/IP-block may have two DFx secure policy plug-in blocks one for sTAP and one for the agent/IP-block. It is the IP-block's responsibility to connect both together.
WTAP signals			
awtap_wso	O	O	**Agent WTAP Serial port Output:** This is the Wrapper Serial Port (WTAP) data output signal.
fwtap_wsi	I	O	**Fabric WTAP Serial Port Input:** This is the Wrapper Serial Port (WTAP) data input signal.
fwtap_wrst_b	I	O	**Fabric WTAP Reset Bar:** This is the Wrapper Serial Port (WTAP) reset input signal. This signal is active low.
fwtap_wrck[2]	I	O	**Fabric WSP Clock:** This is the Wrapper Serial Port (WTAP) clock input signal.

Signal	I/O	R/O/C[1]	Description
fwtap_capturewr	I	O	**Fabric WTAP Capture Wrapper Register control:** This is the Wrapper Serial Port (WTAP) capture control signal to enable capturing input data from WIR or test data registers.
fwtap_shiftwr	I	O	**Fabric WTAP Shift Wrapper Register control:** This is the Wrapper Serial Port (WTAP) shift control signal to enable shifting of the WIR and test data registers.
fwtap_updatewr	I	O	**Fabric WTAP Update Wrapper Register control:** This is the Wrapper Serial Port (WTAP) update control signal to enable updating a shadow WIR or test register from the contents of the shift register.
fwtap_selectwir	I	O	**Fabric WTAP Select Wrapper Instruction Register:** This is the Wrapper Serial Port (WTAP) select control signal to select either the IR register or a test data register the actions by the capture, shift, and update control signals. 0: Any decoded data register will be active in the DR-Scan branch from the controlling sTAP/mTAP FSM that is driving this WTAP's control signals. 1: The WTAP's instruction register will be active in the IR-Scan branch from the controlling sTAP/mTAP FSM that is driving this WTAP's control signals.
fwtap_rti	I	O	**Fabric WTAP Run-Test/Idle:** This is the Wrapper Serial Port (WTAP) control signal to indicate that the driving TAP state machine is in the Run-Test/Idle state. One possible use model is to execute an operation after the test data register was updated, for example, start a BIST engine after all the registers are updated.
Primary source for TAP signals			
atappris_tck[2]	O	O	**Agent Primary Source TAP TCK:** This is the signal source of Test Clock (TCK) from this agent to the CLTAP. This signal is available for those situations where the CLTAP package pins are located in this agent to control the TAP network.
atappris_tms	O	O	**Agent Primary Source TAP TMS:** This is the signal source of Test Mode Select from this agent to the CLTAP. Refer to atappris_tck for more description of the primary source TAP interface.
atappris_trst_b	O	O	**Agent Primary Source TAP TRST_b:** This is the signal source of Test Reset bar from this agent to the CLTAP. Refer to atappris_tck for more description of the primary source TAP interface.
atappris_tdi	O	O	**Agent Primary Source TAP TDI:** This is the signal source of Test Data In from this agent to the CLTAP. Refer to atappris_tck for more description of the primary source TAP interface.
ftappris_tdo	I	O	**Fabric Primary Source TAP TDO:** This is the primary signal sink of Test Data Out from the CLTAP to this agent where the master TAP package pins are located. Refer to atappris_tck for more description of the primary source TAP interface.

Signal	I/O	R/O/C[1]	Description
ftappris_tdoen	I	O	**Fabric Primary Source TAP TDOen:** This is the primary signal sink of TDO enable from the CLTAP to this agent to control the tri-state enable of the physical layer that drive the master TAP TDO. Refer to atappris_tck for more description of the primary source TAP interface.
Secondary source for TAP signals			
atapsecs_tck[2]	O	O	**Agent Secondary Source TAP TCK:** This is the secondary signal source of Test Clock from this agent to the TAP network. This agent is supplying an additional TAP for reusing test vectors to the core or for increased HVM throughput from the tester.
atapsecs_tms	O	O	**Agent Secondary Source TAP TMS:** This is the secondary signal source of Test Mode Select from this agent to the TAP network. Refer to atapsecs_tck for more description of the primary source TAP interface.
atapsecs_trst_b	O	O	**Agent Secondary Source TAP TRST_b:** This is the secondary signal source of Test Reset bar from this agent to the TAP network. Refer to atapsecs_tck for more description of the primary source TAP interface.
atapsecs_tdi	O	O	**Agent Secondary Source TAP TDI:** This is the secondary signal source of Test Data In from this agent to the TAP network. Refer to atapsecs_tck for more description of the primary source TAP interface.
ftapsecs_tdo	I	O	**Fabric Secondary Source TAP TDO:** This is the secondary signal sink of Test Data Out from the fabric to this agent where the secondary TAP package pins are located.
ftapsecs_tdoen	I	O	**Fabric Secondary Source TAP TDOen:** This is the secondary signal sink of TDO Enable from the fabric to this agent to control the tri-state enable of the physical layer that drives the secondary TAP TDO.
Secondary slave TAP signals			
ftapsslv_tck[2]	I	C	**Fabric Secondary Slave TAP clock:** This is the TAP clock from the fabric that originates from package pins connected to the master TAP controller elsewhere in the component. Note: The secondary slave TAP port is required for the slave TAP version that is available in the Intel Reuse Repository. It is optional for a hard-IP TAP. Use of the secondary slave TAP port is SoC dependent. One use model supports a tertiary TAP port to an assigned set of GPIO pins.
ftapsslv_tms	I	C	**Fabric Secondary Slave TAP test mode:** This is the TAP finite state machine test mode control signal from the fabric that originates from package pins connected to the master TAP controller elsewhere in the component. Note: Refer to the explanation note in the ftapslv_tck signal description.

Signal	I/O	R/O/C[1]	Description
ftapsslv_trst_b	I	C	**Fabric Secondary Slave TAP reset bar:** This is the TAP reset from the fabric that may originate from a package pin connecting to the master TAP controller and the TAP network. Note: Refer to the explanation note in the ftapslv_tck signal description.
ftapsslv_tdi	I	C	**Fabric Secondary Slave Test Data In (other TDI):** This is a test data in (TDI) from the master TAP controller elsewhere in the component. Note: Refer to the explanation note in the ftapslv_tck signal description.
atapsslv_tdo	O	C	**Agent Secondary Slave Test Data Out:** This is the test data out (TDO) from this agent. Note: Refer to the explanation note in the ftapslv_tck signal description.
atapsslv_tdoen	O	C	**Agent Test Data Out Enable:** This is the test data out enable to drive the tristate enable on the TDO pad control. Note: Refer to the explanation note in the ftapslv_tck signal description. Also, this signal is defined as: atapslv_tdoen = Shift-DR OR Shift-IR.

NOTE:

[1]Note1: R = required, O = optional, C = conditional. Conditional means that if the feature is used then the signals labeled as conditional become required.

[2]Note 2: The TCK clock signals are expected to be all synchronous and in phase with respect to the driving TCK source. The source may be either the primary or secondary port.

Note 3: The symbol M is the number of agent ports on the package when the extended platform TAP port is supported (M = MTAP_EXI_NUM_OF_TAP_ AGENTS_ON_PLATFORM)

Note3: The slave TAP IP-block from IRR is required to support the secondary slave interface and it is optional for a hard IP-block. It is SoC dependent on how this interface will be used. A secondary slave TAP supports a hierarchical-hybrid topology or a local tertiary TAP port.

10.2 Scan Signal Interface Description

A graphical view of the scan interface signal set is shown in Figure 10-3. The signal details are listed in . The integration team has the freedom to assign any reasonable number of characters as a prefix or suffix as applied to the root name to identify the signal in a full chip SoC. The parameter summary is list in .

Figure 10-3. Agent specific scan signal interface block diagram

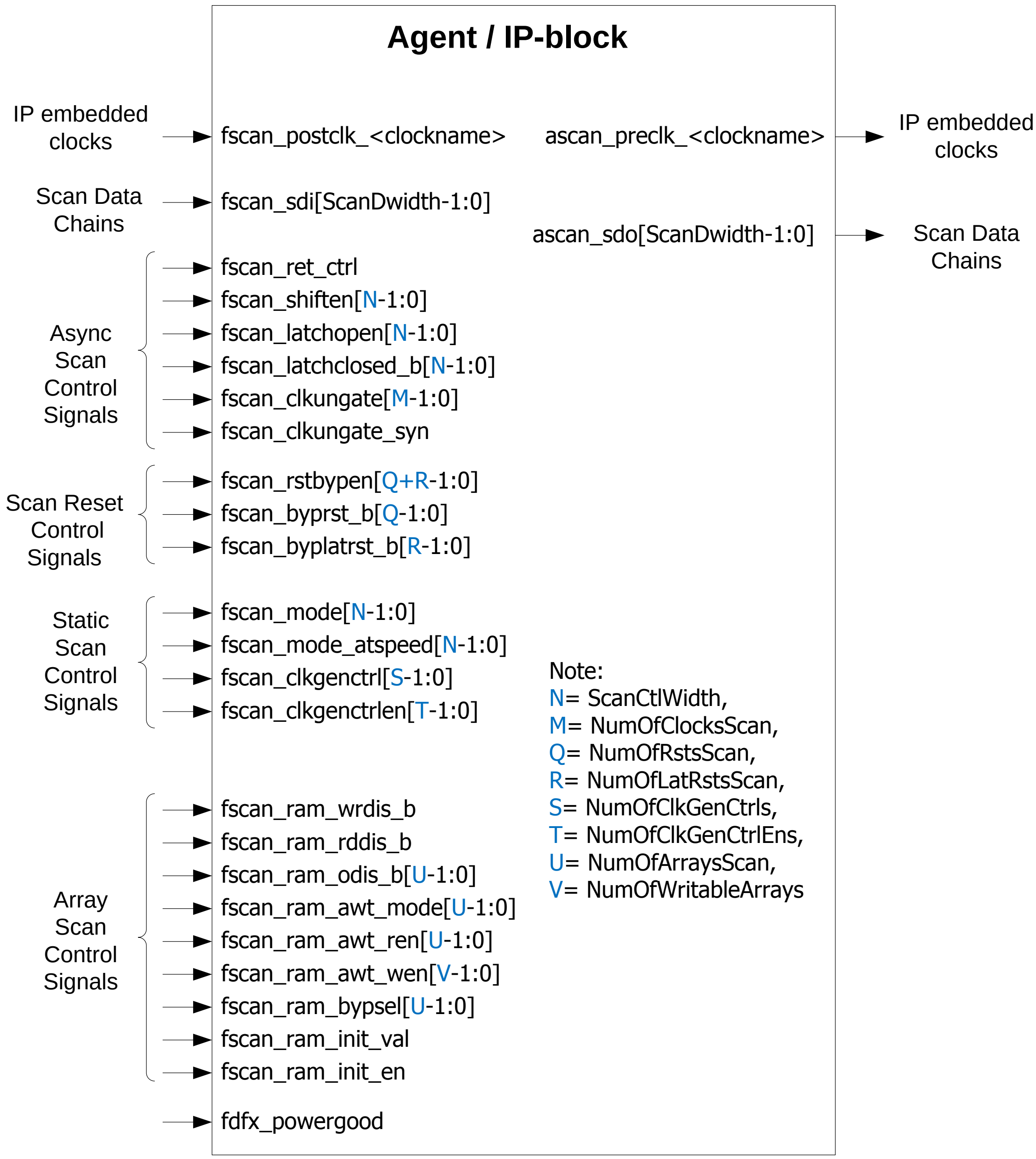


Table 10-2. Scan interface signal descriptions

Signal	I/O	R/O/C[1]	Description
Embedded clock control			
ascan_preclk_ <clockname>	O	C	**Fabric Pre-clock <source clock name>:** This signal, and its accompanying postclk signal, is optionally available for those logic blocks that produce internally generated clocks for their logic. This port allows IP-blocks to export these derived clocks for control by scan clock control logic outside the IP-block (wrapper, partition, or full chip). These internally generated clocks should be directly sent out, prior to functional use i.e. "pre" scan control. Note: The pre and post must exist as pairs. If there is need for ascan_preclk_<clockname> then interface must also have fscan_postclk_<clockname>
fscan_postclk_ <clockname>	I	C	**Agent Post-clock <source clock name>:** This input signal is associated with accompanying preclk output signal. It allows IP-blocks to receive the "post" scan clock control version of internally derived clocks. This version of the clock connects to all modules within this agent/IP-block that were originally connected to the internally-generated derived functional clock. Note: The pre and post must exist as pairs.
Scan data chain signals			
fscan_sdi [ScanDwidth-1:0]	I	O	**Fabric Scan Data In:** This signal bus is the scan data inputs for all of the serially-stitched scan flops/latches within this IP-agent.
ascan_sdo [ScanDwidth-1:0]	O	O	**Agent Scan Data Out:** This signal bus is the scan data outputs for all of the serially-stitched scan flops/latches within this agent.
Asynchronous scan control signals			
fscan_ret_ctrl	I	C	**Fabric scan retention control:** This signal determines the state of the retention cell within a retention flop for scan operations. A mux in the Power Gate Common Block (PGCB) is controlled by fscan_mode. When enabled, the signal is controlled from an asynchronous scan controller (SASC) in the DFx fabric. **Note:** This is signal is required if the IP-block supports retention cells.
fscan_shiften [ScanCtlWidth-1:0]	I	R	**Fabric Scan Shift Enable:** This signal determines whether the data chains are enabled for shifting this does not apply to the control chains. This signal is bused to support hard IP-block modular physical layer that requires scan control per lane. Use of a bit vector for this signal name is optional. The value of ScanCtlWidth is implementation-dependent and may vary per-agent or per-lane for a hard-IP IO agent.

Signal	I/O	R/O/C[1]	Description
fscan_latchopen [ScanCtlWidth-1:0]	I	C	**Fabric Scan Latch Open Enable:** This signal controls the latch open during scan operations. This signal is bused to support hard-IP block modular physical layer that requires scan control per lane. Use of a bit vector for this signal name is optional. The value of ScanCtlWidth is implementation-dependent and may vary per-agent or per-lane for a hard-IP IO agent. Note: If this IP-block contains latches then this signal must be used to control them.
fscan_latchclosed_b [ScanCtlWidth-1:0]	I	C	**Fabric Scan Latch Closed bar:** This signal controls the latch closed during scan operations. This signal is bused to support hard-IP block modular physical layer that requires scan control per lane. Use of a bit vector for this signal name is optional. The value of ScanCtlWidth is implementation-dependent and may vary per-agent or per-lane for a hard-IP IO agent. Note: If this IP-block contains latches then this signal must be used to control them.
fscan_clkungate	I	R	**Fabric Scan Clock Ungate:** This signal controls the clock gating logic during scan operations. This signal is bused to support hard-IP block modular physical layer that requires scan control per lane.
fscan_clkungate_syn	I	R	**Fabric Scan Clock Ungate for Synthesis Inserted Clock Gates:** This signal controls the clock gating logic inserted during synthesis. This signal cannot be used interchangeably with the fscan_clkungate signal that is used exclusively to control clock gating logic that exists in the pre-synthesis design. This signal controls the clock gating logic that is added after synthesis for scan operations.
Scan reset control signals			
fscan_rstbypen [(NumOfRstsScan+ NumOfLatRstsScan) -1:0]	I	C	**Fabric Scan Reset Bypass Enable:** This signal will enable the ability for the bypass reset signals to be active. The reset override signal group must be implemented for IP-blocks with embedded or derived internal reset signals. • 0: Reset bypass and Latch reset bypass are ignored. • 1: Reset bypass and Latch reset bypass are active. Use of a bit vector for this signal name is optional. The value of NumOfRstsScan is implementation-dependent and may vary per-agent or per-lane. This signal is bused to support hard-IP block modular physical layer that requires scan control per lane. Note: It is expected that a soft-IP agent will only use a single control wire for enabling the reset bypasses. A hard-IP agent that implements scan on a per lane basis will require a vector set of enable signals.

Signal	I/O	R/O/C[1]	Description
fscan_byprst_b [NumOfRstsScan-1:0]	I	C	**Fabric Scan Bypass Reset bar:** This signal is a reset input for scan operations that bypasses the internal agent reset logic and applies a reset directly to the agent. The reset override signal group must be implemented for IP-blocks with embedded or derived internal reset signals. Note1: Use of a bit vector for this signal name is optional. The value of NumOfRstsScan is implementation-dependent and may vary per-agent or per-lane. Note2: This signal is enabled with fscan_rstbypen.
fscan_byplatrst_b [NumOfLatRstsScan-1:0]	I	C	**Fabric Scan Bypass Latch Reset bar:** This signal is a reset input for scan operations that bypasses the internal agent reset logic and applies a reset directly to the latches within the agent. This signal is bused to support hard-IP block modular physical layer that requires scan control per lane. Note1: Use of a bit vector for this signal name is optional. The value of NumOfLatRstsScan is implementation-dependent and may vary per-agent or per-lane. Note2: This signal is enabled with fscan_rstbypen.
Static scan control signals			
fscan_mode [ScanCtlWidth-1:0]	I	R	**Fabric Scan Mode:** This signal enables modes within this agent for scan operations. This signal is bused to support a hard IP-block's physical layer that requires scan control per lane. Use of a bit vector for this signal is optional. The value ScanCtlWidth is implementation dependent and may vary per-agent or per-lane. Soft-IP use model: This signal may or may not be used depending on the attributes within the IP-block that need to be made scan friendly. However, it is still required on the interface. Hard-IP use model: Its primary use is to enable the SCC/SCRC controller for this hard-IP partition. Other scan enabling features should be controlled by the asynchronous control signal group. If a scan attribute is unique to this partition and a corresponding control signal is not available than a local TAP test/debug register will enable its actions.
fscan_mode_atspeed [ScanCtlWidth-1:0]	I	C	**Fabric Scan At-speed Mode:** This signal enables the at-speed mode for this agent. This signal is bused to support hard-IP block modular physical layer that requires scan control per lane. Use of a bit vector for this signal name is optional. The value of ScanCtlWidth is implementation-dependent and may vary per-agent or per-lane.

Signal	I/O	R/O/C[1]	Description
fscan_clkgenctrl [NumOfClkGenCtrls-1:0]	I	C	**Fabric Scan Clock Generator Control:** This signal bus overrides clock control values within the agent. The override value is enabled with fscan_clkgenstrlen. This bus may be composed of clock select override and other miscellaneous control signals used for conditioning the clock selects. For agents with a TAP, these signals would be connected to the output of an assigned test data register. For agents without a TAP, this signal bus that is connected to a scan control logic block (SCC/SCRC) within the DFx fabric. Note: This signal may be a single bit.
fscan_clkgenctrlen [NumOfClkGenCtrlEns-1:0]	I	C	**Fabric Scan Clock Generator Control Enable:** This signal (or signal group) is the enable for the fscan_clkgenctrl override control bus. A mux override control can manipulate the signal only during scan operations. For agents with a TAP, these signals would be connected to the output of an assigned test data register. For agents without a TAP, this signal group is a bus that is connected to and by controlled by the scan control logic block (SCC). If this signal is a bus then bit[0] of the interface is assigned to the SCC the fscan_clkgenctrlen[0]: This bit may be assigned to select between internal functional clocks and external SCC clocks. It is implementation dependent to bus this signal or use it as a single bit.
Array shadow logic for scan control signals			
fscan_ram_wrdis_b	I	C	**Fabric Scan RAM Write Disable bar:** This signal controls the write enable on the agent's array during scan operations. Note: This is signal is required for any IP-block's memory wrapper that contains an array, RAM, FIFO, etc.
fscan_ram_rddis_b	I	C	**Fabric Scan RAM Read Disable bar:** This signal controls the read enable on the agent's array during scan operations. Note: This is signal is required for any IP-block's memory wrapper that contains an array, RAM, FIFO, etc.
fscan_ram_odis_b [NumOfArraysScan-1:0]	I	C	**Fabric Scan RAM Output Disable bar:** This signal controls masking output of the agent's array during scan operations. Note: This is signal is required for any IP-block's memory wrapper that contains an array, RAM, FIFO, etc.
fscan_ram_awt_mode [NumOfArraysScan-1:0]	I	C	**Fabric Scan RAM AWT Mode:** This signal enables the mode to conduct scan operations on an Array Write Through (AWT) testing model for arrays. Note: This is signal is required for any IP-block's memory wrapper that contains an array, RAM, FIFO, etc.

Signal	I/O	R/O/C[1]	Description
fscan_ram_awt_ren [NumOfArraysScan -1:0]	I	C	**Fabric Scan RAM AWT Read Enable:** This signal is the read enable for scan operations using an Array Write Through (AWT) testing model for arrays. Note: This is signal is required for any IP-block's memory wrapper that contains an array, RAM, FIFO, etc.
fscan_ram_awt_wen [NumOfWritableArrays -1:0]	I	C	**Fabric Scan RAM AWT Write Enable:** This signal is the write enable for scan operations using an Array Write Through (AWT) testing model for arrays. Note: This is signal is required for any IP-block's memory wrapper that contains an array, RAM, FIFO, etc.
fscan_ram_bypsel [NumOfArraysScan -1:0]	I	C	**Fabric Scan RAM Bypass Select:** This signal selects the bypass path around the array to conduct scan operations on this type of array test configuration. Note: This is signal is required for any IP-block's memory wrapper that contains an array, RAM, FIFO, etc.
fscan_ram_init_val	I	C	**Proposed signal: Fabric Scan RAM initialization value.** This signal is the RAM initialization value to control the DFx muxes with in the array wrapper. 0: Mux points to functional array controls 1: Mux points to DFT array controls (from BIST)
fscan_ram_init_en	I	C	**Proposed signal: Fabric Scan RAM initialization enable.** This signal controls the array initialization for scan operations. 0: normal operation 1: enable initialization
Reset input signals			
fdfx_powergood	I	R	Refer to section 6.2. Note: This signal is equivalent to **dfx_powergood_rst_b**.

NOTES:
[1]Note1: R = required, O = optional, C = conditional. If the IP-block contains logic that requires specific controls to manage that logic during test operations then the signal is required. For example, if an IP-block has an array then the scan test controls for arrays are required.

10.2.1 Scan signal parameter summary

Table 10-3. Scan parameter summary table

Parameter Name	Letter designation from diagram	Description
ScanCtlWidth	N	**Scan Control Width:** This strap is primarily used for hard-IP agents where the scan control segregated per lane. However, soft-IP agents can certainly take advantages of this feature.
ScanDwidth	-	**Scan Data chain width:** This strap value determines the number of scan data chains.

Parameter Name	Letter designation from diagram	Description
NumOfClocksScan	M	**Number of Clocks for Scan:** This value determines the number of clocks that are supported by this agent that require scan override control. Soft-IP agents: This value is used for the number of clocks that require bypassing. Hard-IP agents: This value may be set to the same value as ScanCtlWidth to control the scan logic on a per lane basis.
NumOfRstsScan	Q	**Number of Resets for Scan:** This value determines the number of resets that are supported by this agent that require scan override control. Soft-IP agents: This value is used for the number of resets that require bypassing. Hard-IP agents: This value may be set to the same value as ScanCtlWidth to control the scan logic on a per lane basis.
NumOfLatRstsScan	R	**Number of Resets with Latch-based design for Scan:** This value determines the number of resets associated with latches that are supported by this agent that require scan override control. Soft-IP agents: This value is used for the number of latched based resets that require bypassing. Hard-IP agents: This value may be set to the same value as ScanCtlWidth to control the scan logic on a per lane basis.
NumOfClkGenCtrls	S	**Number of Clock Generate Control Overrides:** This value determines the number of clock control signal overrides for the fscan_clkgenctrl signal group.
NumOfClkGenCtrlEns	T	**Number of Clock Generate Control Enables:** This value determines the number of clock control signal overrides for the fscan_clkgenctrl signal group.
NumOfArraysScan	U	**Number of Arrays for Scan:** This value determines the number of arrays that require scan control overrides. NumOfArraysScan = NumberOf_RFarrays + NumOf_SRAMarrays + NumOf_ROMarrays
NumOfWritableArrays	V	**Number of Arrays for RF and SRAM:** This value is the number of arrays that do not include ROMs. NumOfArraysScan = NumberOf_RFarrays + NumOf_SRAMarrays
NumOf_RFarrays	-	**Number of Arrays for RF array type**
NumOf_SRAMarrays	-	**Number of Arrays for SRAM array type**
NumOf_ROMarrays	-	**Number of Arrays for ROM array type**

10.3 Array Signal Interface Description

A graphical view of the array test interface is divided among the three memory types and shown in Figure 10-4 through Figure 10-7. A more detailed signal description is listed in

Table 10-4 through Table 10-7. The integration team has the freedom to assign any reasonable number of characters as a prefix or suffix as applied to the root name to identify the signal in a full chip SoC.

Figure 10-4. Register file array test support signals

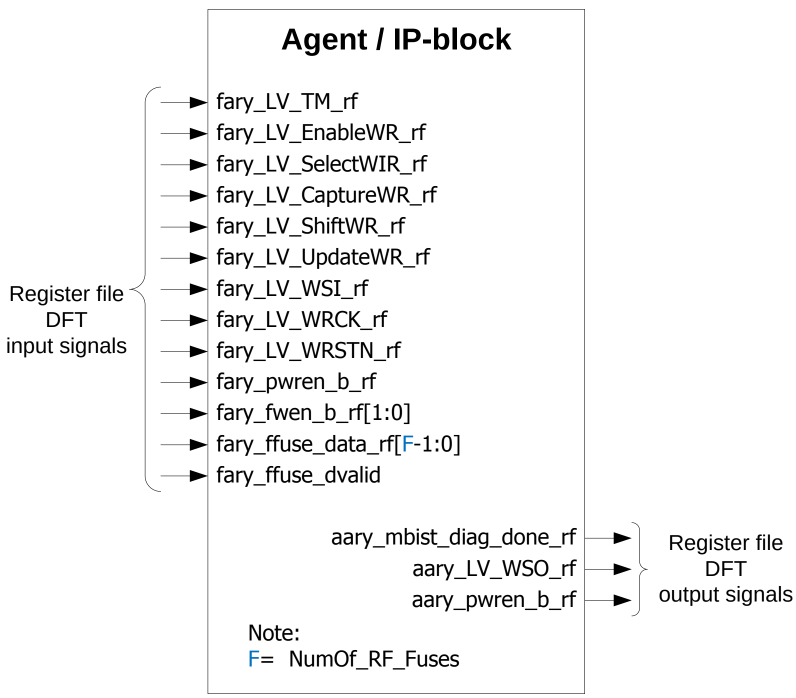


Figure 10-5. SRAM array support signals continued

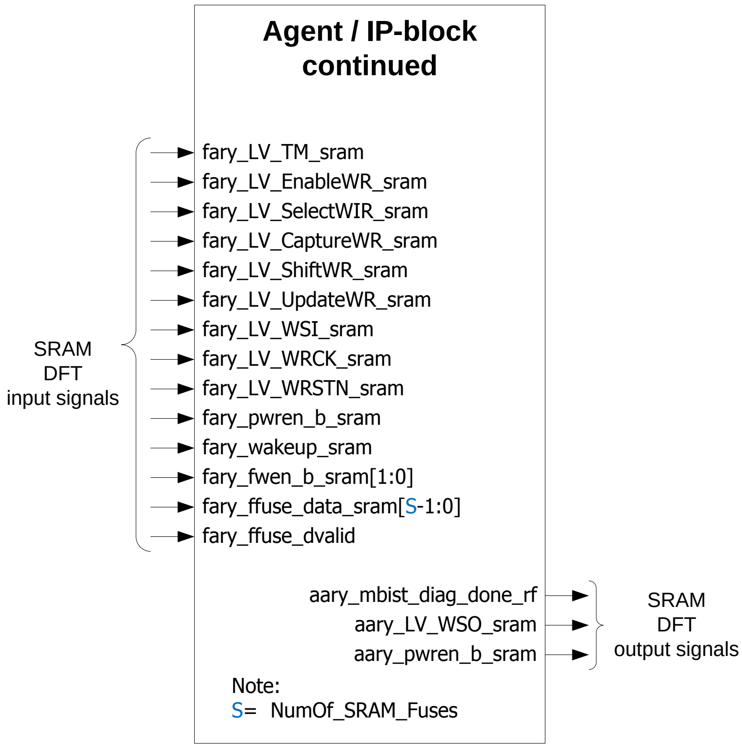


Figure 10-6. Read-only memory (ROM) array support signals continued

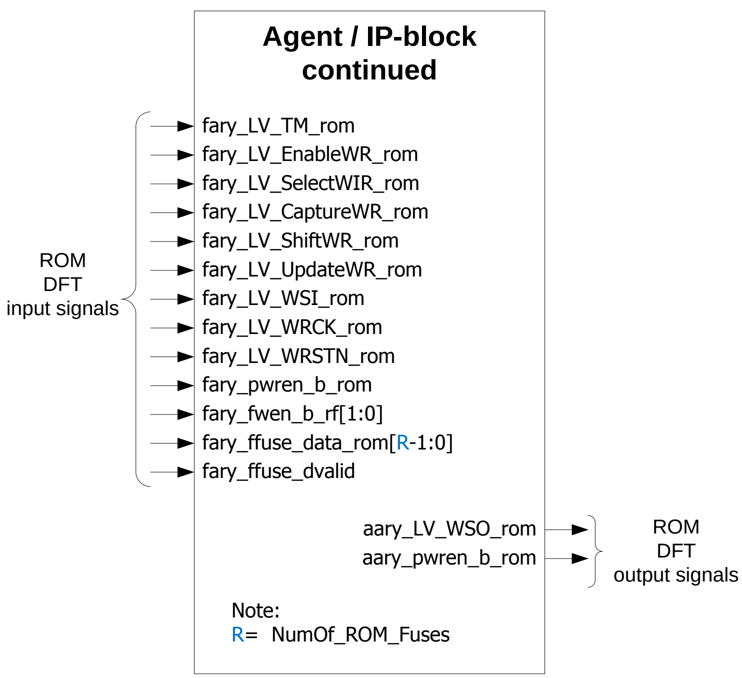


Figure 10-7. LYA and miscellaneous array test signal support

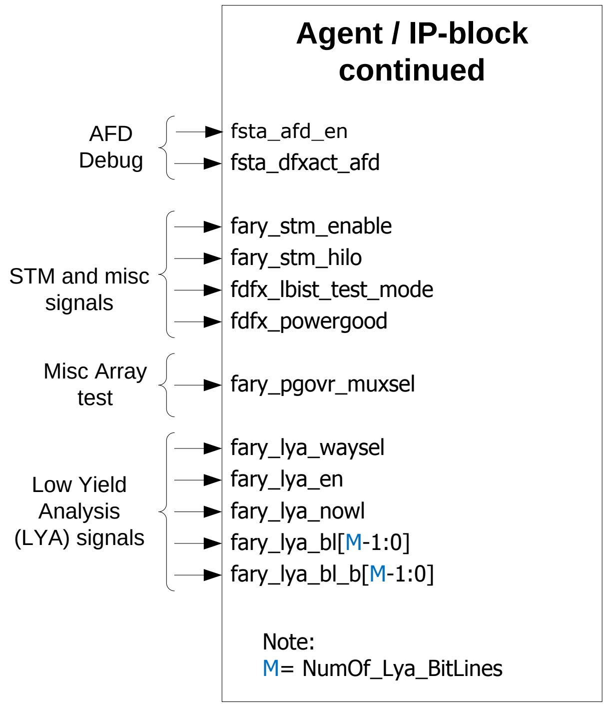


Table 10-4. Register file array test signal table

Signal	I/O	R/C[1]	Description
MBIST test signals for register files			
fary_LV_TM_rf	I	C	**Fabric array Logic Vision (MBIST) Test Mode for RFs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal informs the BIST collar that is in test mode for register file associated at the top level of the agent (IP-block). Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_LV_EnableWR_rf	I	C	**Fabric array Logic Vision (MBIST) Enable WTAP Register for RFs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal enables the embedded WTAP register in the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_LV_SelectWIR_rf	I	C	**Fabric array Logic Vision (MBIST) Select WTAP IR for RFs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.

Signal	I/O	R/C[1]	Description
fary_LV_CaptureWR_rf	I	C	**Fabric array Logic Vision (MBIST) Capture WTAP register for RFs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_LV_ShiftWR_rf	I	C	**Fabric array Logic Vision (MBIST) Shift WTAP register for RFs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_LV_UpdateWR_rf	I	C	**Fabric array Logic Vision (MBIST) Update WTAP register for RFs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_LV_WSI_rf	I	C	**Fabric array Logic Vision (MBIST) WTAP serial input for RFs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_LV_WRCK_rf	I	C	**Fabric array Logic Vision (MBIST) WTAP clock for RFs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_LV_WRSTN_rf	I	C	**Fabric array Logic Vision (MBIST) WTAP reset (bar) for RFs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
aary_LV_WSO_rf	O	C	**Fabric array Logic Vision (MBIST) WTAP serial output for RFs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_pwren_b_rf	I	C	**Fabric array power enable bar for RFs.** This signal enables the power for array test operations.

Signal	I/O	R/C[1]	Description
aary_pwren_b_rf	O	C	**Agent array power enable bar for RFs.** This signal enables the power for array test operations. This signal is available as an output so the integration team can serially stitch the control signal for several similar arrays.
fary_fwen_b_rf [1:0]	I	C	**Fabric array firewall enable bar for RFs.** This signal is active low and it forces the firewall to be enabled. [0]: Enables the firewall all the inputs to EBB which includes data in, address, enables, STM i/ps, clock, lya* signals,wakeup_ms01h,clock etc. [1]: Enables firewall the data out from EBB
aary_mbist_diag_done_rf	O	C	**Agent array MBIST diagnostic done signal for RF type.** This signal indicates that either the MBIST controller is done or an error occurred such that the controller stopped.
fary_ffuse_data_rf [NumOf_RF_Fuses - 1:0]	I	C	**Fabric array for RFs using IOSF fuse data bus.** This signal bus is re-used from IOSF fuse data bus defined in the miscellaneous DFx signal interface. The fary_ffuse_data_rf signal bus is generic and assigned to the FUSE_MISC_RF_IN signal bus on the memory. If an agent/IP-block has no RF memories then the parameter is set to NumOf_RF_Fuses =1.
fary_ffuse_dvalid	I	C	**Fabric array for RFs using IOSF fuse data valid.** This signal indicates that the fuse data is valid. There is only one fuse data valid per agent/IP-block.
[1]Note1: R = required, C = conditional. If an IP-block has arrays, FIFOs, ROMs or SRAM memories then these signals are required.			

Table 10-5. SRAM Array test signal table

Signal	I/O	R/C[1]	Description
MBIST test signals for SRAMs			
fary_LV_TM_sram	I	C	**Fabric array Logic Vision (MBIST) Test Mode for SRAMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal informs the BIST collar that is in test mode for register file associated at the top level of the agent (IP-block). Note: If an agent/IP-block requires an SRAM then this signal is required.
fary_LV_EnableWR_sram	I	C	**Fabric array Logic Vision (MBIST) Enable WTAP Register for SRAMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal enables the embedded WTAP register in the BIST collar. Note: If an agent/IP-block requires an SRAM then this signal is required.

Signal	I/O	R/C[1]	Description
fary_LV_SelectWIR_ sram	I	C	**Fabric array Logic Vision (MBIST) Select WTAP IR for SRAMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires an SRAM then this signal is required.
fary_LV_CaptureWR_ sram	I	C	**Fabric array Logic Vision (MBIST) Capture WTAP register for SRAMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires an SRAM then this signal is required.
fary_LV_ShiftWR_ sram	I	C	**Fabric array Logic Vision (MBIST) Shift WTAP register for SRAMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. **Note:** If an agent/IP-block requires an SRAM then this signal is required.
fary_LV_UpdateWR_ sram	I	C	**Fabric array Logic Vision (MBIST) Update WTAP register for SRAMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires an SRAM then this signal is required.
fary_LV_WSI_ sram	I	C	**Fabric array Logic Vision (MBIST) WTAP serial input for SRAMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires an SRAM then this signal is required.
fary_LV_WRCK_ sram	I	C	**Fabric array Logic Vision (MBIST) WTAP clock for SRAMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires an SRAM then this signal is required.
fary_LV_WRSTN_ sram	I	C	**Fabric array Logic Vision (MBIST) WTAP reset (bar) for SRAMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires an SRAM then this signal is required.

Signal	I/O	R/C[1]	Description
aary_LV_WSO_ sram	O	C	**Fabric array Logic Vision (MBIST) WTAP serial output for SRAMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires an SRAM then this signal is required.
fary_pwren_b_sram	I	C	**Fabric array power enable bar for SRAMs.** This signal enables the power for array test operations.
aary_pwren_b_sram	O	C	**Agent array power enable bar for SRAMs.** This signal enables the power for array test operations. This signal is available as an output so the integration team can serially stitch the control signal for several similar arrays.
fary_wakeup_sram	I	C	**Fabric wakeup for SRAMs.** This signal forces the memory to wake up from a sleep mode. 0: Memory goes into sleep mode with many power saving features enabled. 1: Assertion of on this signal causes the SRAM to come out of sleep and get ready for access.
fary_fwen_b_sram [1:0]	I	C	**Fabric array firewall enable bar for SRAMs.** This signal is active low and it forces the firewall to be enabled. [0]: Enables the firewall all the inputs to EBB which includes data in, address, enables, STM i/ps, clock, lya* signals,wakeup_ms01h,clock etc. [1]: Enables firewall the data out from EBB
fary_ffuse_data_sram [NumOf_SRAM_Fuses - 1:0]	I	C	**Fabric array for SRAMs using IOSF fuse data bus.** This signal bus is re-used from IOSF fuse data bus defined in the miscellaneous DFx signal interface. The fary_ffuse_data_sram signal bus is generic and assigned to the FUSE_MISC_SSA_IN signal bus on the memory. If an agent/IP-block has no RF memories then the parameter is set to NumOf_SRAM_Fuses =1.
fary_ffuse_dvalid	I	C	**Fabric array for RFs using IOSF fuse data valid.** This signal indicates that the fuse data is valid. There is only one fuse data valid per agent/IP-block.

[1]Note1: R = required, C = conditional. If an IP-block has arrays, FIFOs, ROMs or SRAM memories then these signals are required.

1

Table 10-6. Read-only memory (ROM) array test signal table

Signal	I/O	R/C[1]	Description
MBIST test signals for ROMs			
fary_LV_TM_rom	I	C	**Fabric array Logic Vision (MBIST) Test Mode for ROMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal informs the BIST collar that is in test mode for register file associated at the top level of the agent (IP-block). Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_LV_EnableWR_rom	I	C	**Fabric array Logic Vision (MBIST) Enable WTAP Register for ROMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal enables the embedded WTAP register in the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_LV_SelectWIR_rom	I	C	**Fabric array Logic Vision (MBIST) Select WTAP IR for ROMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_LV_CaptureWR_rom	I	C	**Fabric array Logic Vision (MBIST) Capture WTAP register for ROMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_LV_ShiftWR_rom	I	C	**Fabric array Logic Vision (MBIST) Shift WTAP register for ROMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_LV_UpdateWR_rom	I	C	**Fabric array Logic Vision (MBIST) Update WTAP register for ROMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_LV_WSI_rom	I	C	**Fabric array Logic Vision (MBIST) WTAP serial input for ROMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.

Signal	I/O	R/C[1]	Description
fary_LV_WRCK_rom	I	C	**Fabric array Logic Vision (MBIST) WTAP clock for ROMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_LV_WRSTN_rom	I	C	**Fabric array Logic Vision (MBIST) WTAP reset (bar) for ROMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
aary_LV_WSO_rom	O	C	**Fabric array Logic Vision (MBIST) WTAP serial output for ROMs.** This signal originates from a Logic Vision embedded MBIST TAP controller in the regional DFx units. This signal is part of the MBIST WTAP interface that communicates the test algorithms to/from the BIST collar. Note: If an agent/IP-block requires a register file/array then this signal is required.
fary_pwren_b_rom	I	C	**Fabric array power enable bar for ROMs.** This signal enables the power for array test operations.
aary_pwren_b_rom	O	C	**Agent array power enable bar for ROMs.** This signal enables the power for array test operations. This signal is available as an output so the integration team can serially stitch the control signal for several similar arrays.
fary_fwen_b_rom [1:0]	I	C	**Fabric array firewall enable bar for ROMs.** This signal is active low and it forces the firewall to be enabled. [0]: Enables the firewall all the inputs to EBB which includes data in, address, enables, STM i/ps, clock, lya* signals,wakeup_ms01h,clock etc. [1]: Enables firewall the data out from EBB
aary_mbist_diag_ done_sram	O	C	**Agent array MBIST diagnostic done signal for RF type.** This signal indicates that either the MBIST controller is done or an error occurred such that the controller stopped
fary_ffuse_data_rom [NumOf_RF_Fuses - 1:0]	I	C	**Fabric array using IOSF fuse data bus.** This signal bus is re-used from IOSF fuse data bus defined in the miscellaneous DFx signal interface. The fary_ffuse_data_rom signal bus is generic and assigned to the FUSE_MISC_ROM_IN signal bus on the memory. If an agent/IP-block has no RF memories then the parameter is set to NumOf_ROM_Fuses =1.
fary_ffuse_dvalid	I	C	**Fabric array for RFs using IOSF fuse data valid.** This signal indicates that the fuse data is valid. There is only one fuse data valid per agent/IP-block.

[1]Note1: R = required, C = conditional. If an IP-block has arrays, FIFOs, ROMs or SRAM memories then these signals are required.

1

Table 10-7. Miscellaneous array test signal table

Signal	I/O	R/O/C[1]	Description
Low Yield Analysis (LYA) DFT signals			
fary_lya_waysel	I	C	**Fabric array, LYA signal group, slice select.** This signal select a LYA data slice. fary_lya_waysel connect to: lyawaysel_ms00h internally.
fary_lya_dataen	I	C	**Fabric array, LYA signal group, data enable.** This signal is the LYA data enable. fary_lya_dataen connects to: lyaen_ms00h internally.
fary_lya_nowl	I	C	**Fabric array, LYA signal group, no write line.** This signal indicates a read or write to a line of memory cells. 0: write 1: read fary_lya_nowl connects to: lyanowl_ms00h internally.
fary_lya_nowrysel	I	C	**Fabric array, LYA signal group, no write ysel.** This signal indicates a no write with 'y' select. fary_lay_nowrysel connects to lyanowrysel_ms00h internally.
fary_lya_bl[M-1:0]	I	C	**Fabric array, LYA signal group bit line input.** This signal is used for low yield analysis of the arrays. It is parameterized for M number of inputs. The minimum and default is set to 2. fary_lya_bl[1:0] connects to: lyabl_ms00l [1:0]
fary_lya_bl_b[M-1:0]	I	C	**Fabric low yield analysis bit line bar input.** This signal is used for low yield analysis of the arrays. It is parameterized for M number of inputs. It is parameterized for M number of inputs. The minimum and default is set to 2. fary_lya_bl_b[1:0] connects to: lyabl_ms00_bl [1:0]
Miscellaneous array DFT signals			
fary_pgovr_muxsel	I	C	**Fabric array power gate override mux select.** This signal overrides the internal mux to selects between the internal power gate controls to the arrays/RF/ROM and the TAP test data register. **0:** Normal operation, internal power gate controls **1:** Force EBBs to use TAP TDR bits to drive values
fary_stm_enable	I	C	**Fabric STM enable.** This signal enables the Stability Test Mode (STM) for arrays. 0: STM disabled 1: STM enabled and block the normal write driver signal.
fary_stm_hilo	I	C	**Fabric STB high/low value.** This signal determines if the bitline or bit line bar is floating to the SRAM. 0: Bitline_b is floating 1: Bitline is floating
fdfx_lbist_test_mode	I	C	**Fabric DFx LBIST test mode:** This signal supports LBIST operations in the presence of arrays. This signal is conditional based on the SoC decision to use the LBIST flow.

fdfx_powergood	I	C	Refer to section 6.2. Note: This signal is equivalent to **dfx_powergood_rst_b**.
Array debug signals			
fsta_dfxact_afd	I	C	**Fabric SoC trigger arch DFx action for array/freeze/dump:** This signal forces the array's (or register file's) DFx mux inputs to select the BIST engine to extract it contents. 0: Normal array (RF) operation. The functional path to the array is active. 1: Force the DFx mux to select the MBIST. Note: This signal is sourced from the Cluster Trigger Block in the regional DFx unit.
fsta_afd_en	I	C	**Fabric SoC trigger arch DFx action enable:** This signal allows the silicon validation team to select which arrays can be enabled for AFD. This signal is connected to a cluster DFx unit TAP bit.
NOTE: [1]R = required, O = optional, C = conditional. Conditional means that if the feature is used then the signals labeled as conditional become required.			

10.4 Miscellaneous DFT Signal Interface Description

Figure 10-8 and Figure 10-9 shows a graphical view of the miscellaneous test and boundary scan control signals. **Error! Reference source not found.** provides a more detailed description of the function, signal direction, signal name, and whether it's required or not. The integration team has the freedom to assign any reasonable number of characters as a prefix or suffix as applied to the root name to identify the signal in a full chip SoC.

Figure 10-8. Miscellaneous test signal diagram

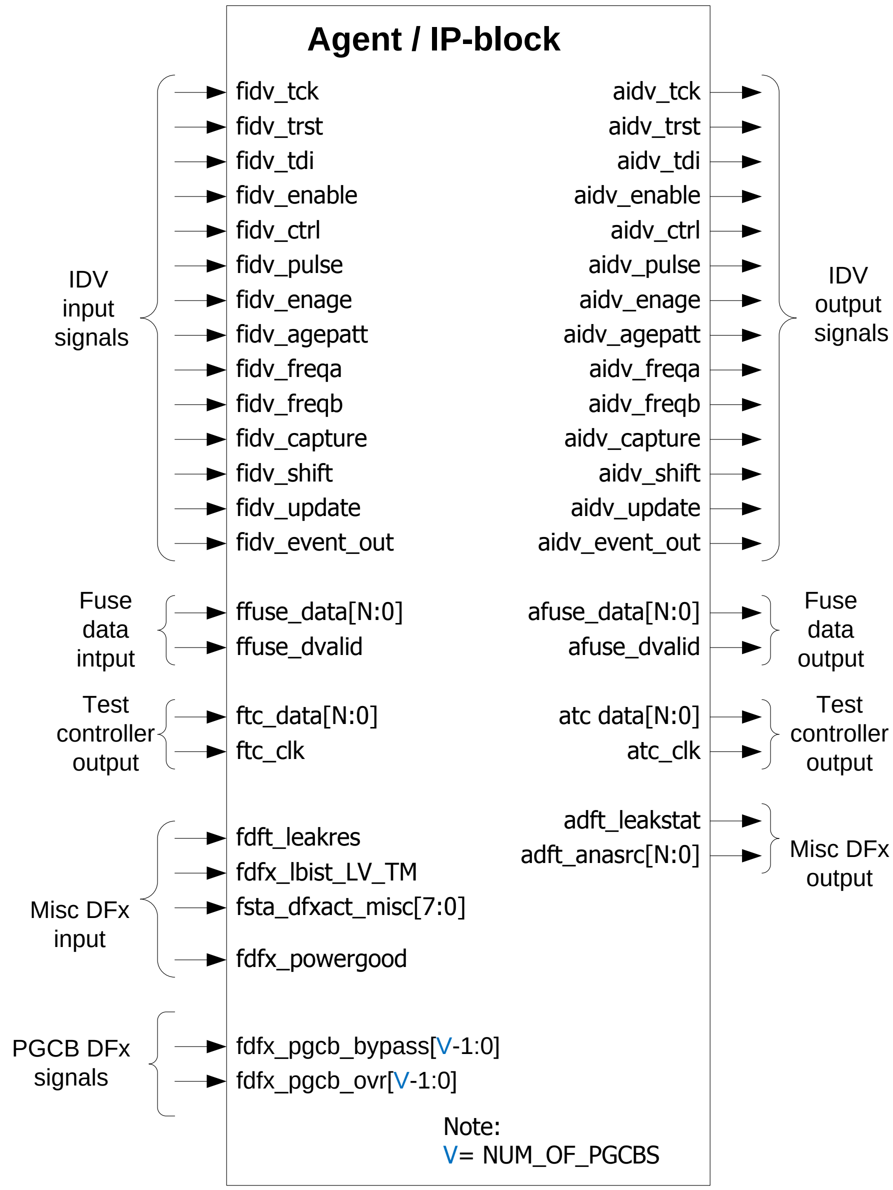
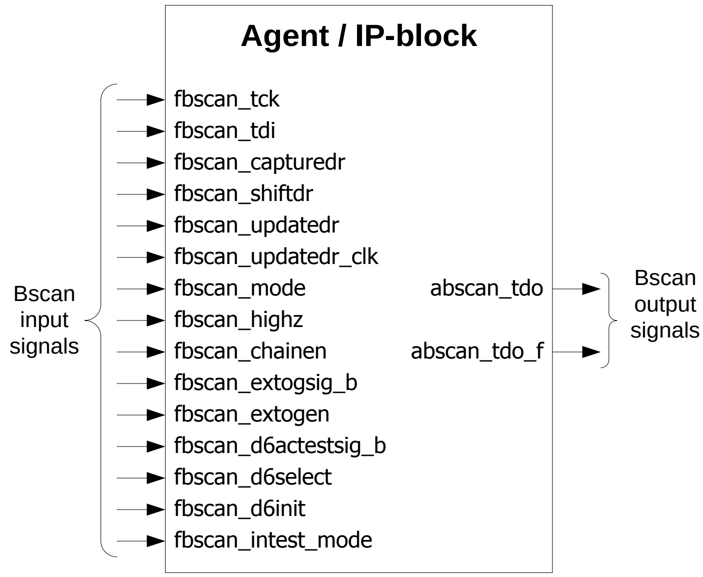


Figure 10-9. Boundary Scan control signals



Boundary scan support signals

Signal	I/O	R/O/C[1]	Description
fbscan_tck[2]	I	C	**Fabric Boundary Scan Test Clock Input.** This signal is the test clock input for *this* agent's section of the boundary scan chain stitched through the physical layer pins in the hard-IP macro. Generally, this version of the TCK signal is routed backwards with respect to the data (fbscan_tdi) signal. This fbscan_tck signal and the ftap_tck are eventually connected to the same package pin source. Its use is implementation-dependent.
fbscan_tdi	I	C	**Fabric Boundary Scan Test Data Input.** This signal is the test data input for *this* agent's section of the boundary scan chain stitched through the physical layer pins in the hard-IP macro.
fbscan_capturedr	I	C	**Fabric Boundary Scan Capture-DR Input.** This signal is the boundary scan decoded Capture-DR control signal for *this* agent's physical layer pins in the hard-IP macro.
fbscan_shiftdr	I	C	**Fabric Boundary Scan Shift-DR Input.** This signal is the boundary scan decoded Shift-DR control signal for *this* agent's physical layer pins in the hard-IP macro.
fbscan_updatedr	I	C	**Fabric Boundary Scan Update-DR Input.** This signal is the boundary scan decoded Update-DR control signal for *this* agent's physical layer pins in the hard-IP macro. There are two potential bscan cell designs; this signal supports the cell with a negative edge (TCK) clock enable flop. This signal is connected to the bscan cell's enable input.

Signal	I/O	R/O/C[1]	Description
fbscan_updatedr_clk	I	C	**Fabric Boundary Scan Update-DR Clock.** This signal is the boundary scan decoded Update-DR control signal for *this* agent's physical layer pins in the hard-IP macro. There are two potential bscan cell designs; this signal supports the cell with the updatedr directly connected to the bscan cell's clock input.
fbscan_mode	I	C	**Fabric Boundary Scan Mode Input.** This signal is the boundary scan decoded Mode control signal for *this* agent's physical layer pins in the hard-IP macro.
fbscan_highz	I	C	**Fabric Boundary Scan High-Z control.** This signal is the test data output for *this* agent's section of the boundary scan chain stitched through the physical layer pins in the hard-IP macro.
fbscan_chainen	I	C	**Fabric Boundary Scan Chain Enable.** This signal enables circuits in the physical layer to allow for boundary scan sample/preload operations to occur. It may be used to enable the proper pull ups/downs or to turn on sense amps. This signal is active when any boundary scan instruction is decoded in the master TAP.
fbscan_extogen	I	C	**Fabric Boundary Scan Extest Toggle Enable.** This signal enables the EXTEST_TOGGLE function when the instruction is valid. EXTEST_TOGGLE is a modified EXTEST boundary scan function that toggles bscan cell as an output at a period equal to the TCK frequency. This may be used for any IO type. For high speed serial IOs, this signal enables both the transmit (Tx) and receive (Rx) paths as an output for toggling logic values at the IO voltage level from the component to a test card.
fbscan_extogsig_b	I	C	**Fabric Boundary Scan Extest Toggle Signal bar.** This signal provides the toggling signal source when the EXTEST_TOGGLE instruction is enabled.
fbscan_d6init	I	C	**Fabric Boundary Scan "dot6" Initialization Signal.** This signal will initialize or clear the hysteresis memory (depending on the implement choice) in the IO circuit that captures detected values on the pins.
fbscan_d6select	I	C	**Fabric Boundary Scan "dot 6" Select:** This signal enables the test circuitry for dot6 test mode and indicates when either of the two train or pulse 1149.6 modes are active.
fbscan_d6actestsig_b	I	C	**Fabric Boundary Scan "dot 6" AC Test Signal bar:** This signal enables the 1149.6 test mode for use with IR instructions EXTEST_TRAIN and EXTEST_PULSE to perform the boundary scan test function for AC-coupled differential IOs.
fbscan_intest_mode	I	C	**Fabric Boundary Scan Intest Mode:** This signal enables an INTEST boundary-scan test mode. This is rarely used but available to SoCs.
abscan_tdo	O	C	**Agent Boundary Scan Test Data Output.** This signal is the test data output for *this* agent's section of the boundary scan chain stitched through the physical layer pins in the hard-IP macro.

Signal	I/O	R/O/C[1]	Description
abscan_tdo_f	O	C	**Agent Boundary Scan Test Data Output on Falling edge.** This signal is the test data output for *this* agent's section of the boundary scan chain. The output is asserted on the falling edge of TCK. If boundary scan is supported by this TAP then this signal is required.
fdfx_rst_b	I	C	**Fabric DFx reset bar:** This signal is sinked by the IP-block for resetting designated DFx logic. The signal is controlled within the DFx fabric, the Chassis reset unit and/or the Power Management Controller (PMC). It is the responsibility of the integration team to properly connect this signal and assert it for each IP. **Note:** Soft-IP blocks are required to include this signal on the IP interface and connect it to the VISA reset. If a hard-IP block is designated as a secure IO IP then it must implement this reset connecting to all VISA ULMs/PLMs within the IP. Otherwise, for all other hard IP-blocks it is optional.

NOTE:

[1]Note1: R = required, O = optional, C = Conditional. Conditional means that if the feature is used then the signals labeled as conditional become required. It may be obvious but if the IP-block supports boundary scan the boundary signals are required.

[2]Note2: The tck clock signals are expected to be synchronous and in phase with respect to the driving tck source. Usually, this is the primary TAP tck for most use models but may include the secondary TAP port tck signal. For example, a mux cannot be used to select between a TAP tck signal and the boundary scan tck signal within an agent.

Table 10-8. IDV control signals

Signal	I/O	R/O/C[1]	Description
fidv_tck[2]	I	C	**Fabric IDV TCK:** TAP clock from the previous IDV that originates from the master TAP or fabric TAP controller.
fidv_trst	I	C	**Fabric IDV TRST:** This is signal will reset the IDV fublet. 0: IDV normal operation 1: IDV will be in reset until the signal is logic 0.
fidv_tdi	I	C	**Fabric IDV TDI:** TAP data input.
fidv_enable	I	C	**Fabric IDV Enable:** This signal will enable the IDV fublet. The actual signal on the fublet is named as idvdisable_b but the functionality is the same. IOSF DFx spec strives to name signals with positive true logic except for active low resets.
fidv_ctrl	I	C	**Fabric IDV Control:** This signal enables a 1-bit flop datapath through the IDV.

fidv_pulse	I	C	**Fabric IDV Pulse:** This signal to gate scan in/out feature of IDV, thus disabling IDV and forcing the address to 0 and disabling all oscillators. 0: normal IDV operation. 1: When it is held high it will gate any scan in/out function and force the idvtdo output to 0. It will also force idvoscaddr[0] and idvoscaddr_b[0] low at the same time. This triggers the address decoder block to disable all the oscillators, regardless of the values of idvoscaddr[4:1] and idvoscaddr_b[4:1].
fidv_freqa	I	C	**Fabric IDV Previous Frequency Oscillator A:** Output from previous IDV oscillator is an input to this IDV. If this is the first IDV, then this input is grounded.
fidv_freqb	I	C	**Fabric IDV Previous Frequency Oscillator B:** Output from previous IDV oscillator is an input to this IDV. If this is the first IDV, then this input is grounded.
fidv_enage	I	O	**Fabric IDV Enable Aging:** This control signal will enable the aging oscillator feature of this IDV.
fidv_agepatt	I	O	**Fabric IDV Age Pattern:** This control signal will enable the age pattern logic in the IDV.
fidv_capture	I	O	**Fabric IDV Capture:** This control signal will capture status or shadow register data during the Capture-DR state. IDVs are serially linked together and this input signal is connected from the previous IDV in the chain. The purpose of this signal is for future extensions of the IDV controller where an internal TAP data register provides more sophisticated capabilities within the fublets.
fidv_shift	I	O	**Fabric IDV Shift:** This control signal will shift the TAP data register during the Shift-DR state. IDVs are serially linked together and this input signal is connected from the previous IDV in the chain. This signal is for future extensions of the IDV controller where an internal TAP data register provides more sophisticated capabilities within the fublets.
fidv_update	I	O	**Fabric IDV Update:** This control signal will update the shadow register with data from the shift register during the Update-DR state. IDVs are serially linked together and this input signal is connected from the previous IDV in the chain. This signal is for future extensions of the IDV controller where an internal TAP data register provides more sophisticated capabilities within the fublets.
fidv_event_out	I	O	**Fabric IDV Event Output:** An event output indicates expected results from a specific feature within the fublet. IDVs are serially linked together and this input signal is connected from the previous IDV in the chain. This signal is for future extensions of the IDV controller where an internal TAP data register provides more sophisticated capabilities within the fublets.
aidv_tck[2]	O	C	**Agent IDV TCK:** This control signal passes through to the next IDV or to the IDV controller. See fidv_* for the definition.
aidv_trst	O	C	**Agent IDV TRST:** IDV that originates from the IDV controller.

aidv_tdo	O	C	**Agent IDV TDO:** This control signal passes through to the next IDV or to the IDV controller. See fidv_* for the definition.
aidv_enable	O	C	**Agent IDV Enable:** This control signal passes through to the next IDV or to the IDV controller. See fidv_* for the definition.
aidv_ctrl	O	C	**Agent IDV Control:** This control signal passes through to the next IDV or to the IDV controller. See fidv_* for the definition
aidv_pulse	O	C	**Agent IDV pulse:** This control signal passes through to the next IDV or to the IDV controller. See fidv_* for the definition.
aidv_freqa	O	C	**Agent IDV Previous Frequency Oscillator A:** This control signal passes through to the next IDV or to the IDV controller. See fidv_* for the definition.
aidv_freqb	O	C	**Agent IDV Previous Frequency Oscillator B:** This control signal passes through to the next IDV or to the IDV controller. See fidv_* for the definition.
aidv_enage	O	O	**Agent IDV Enable Aging:** This control signal passes through to the next IDV or to the IDV controller. See fidv_* for the definition.
aidv_agepatt	O	O	**Agent IDV Age Pattern:** This control signal passes through to the next IDV or to the IDV controller. See fidv_* for the definition.
aidv_capture	O	O	**Agent IDV Capture:** This control signal passes through to the next IDV or to the IDV controller. See fidv_* for the definition.
aidv_shift	O	O	**Agent IDV Shift:** This control signal passes through to the next IDV or to the IDV controller. See fidv_* for the definition.
aidv_update	O	O	**Agent IDV Update:** This control signal passes through to the next IDV or to the IDV controller. See fidv_* for the definition.
aidv_event_out	O	O	**Agent IDV Event Output:** This control signal passes through to the next IDV or to the IDV controller. See fidv_* for the definition.

NOTE:

[1]Note1: R = required, O = optional, C = Conditional. Conditional means that if the feature is used then the signals labeled as conditional become required. It may be obvious but if the IP-block supports boundary scan the boundary signals are required.

1

Table 10-9. Parallel fuse data interface

Signal	I/O	R/O/C[1]	Description
ffuse_data [FFDWIDTH-1:0]	I	O	**Fabric Fuse Data[N:0]:** This bus delivers the fuse data from a centralized fuse controller through the fabric to an IOSF agent. This data is then used for specialized configuration control of features for circuits. For example, a PLL may have divider ratio settings that are fixed for a particular market segment. These values must be delivered before the deassertion of reset. Use of this bus implies a direct connection of the fuse values from the fuse block to this agent.
ffuse_dvalid	I	O	**Fabric Fuse Data Valid:** This signal indicates when the fuse values from the fabric to the IOSF agent are valid for reading.
afuse_data [AFDWIDTH-1:0]	O	O	**Agent Fuse Data[N:0]:** This bus delivers the fuse data from an IOSF agent to the fabric that is then distributed to other agents as needed. This agent may be a security block that contains the fuses rather than a DFx control block in the fabric.
afuse_dvalid	O	O	**Agent Fuse Data Valid:** This signal indicates when the fuse values from the agent to the fabric are valid for reading.

NOTE:
[1]Note1: R = required, O = optional, C = Conditional. Conditional means that if the feature is used then the signals labeled as conditional become required. It may be obvious but if the IP-block supports boundary scan the boundary signals are required.

Table 10-10. SoC level IO test control signals

Signal	I/O	R/O/C[1]	Description
adft_leakstat	O	O	**Agent DFT Leakage Status:** This is a source signal from this hard-IP agent that generated an output response from the leakage testing digital logic in the physical layer.
fdft_leakres	I	O	**Fabric DFT Leak Results:** This is the sink signal for the results of the leakage testing from one or more hard-IP agents on the fabric. This signal is intended for hard-IP agents to output the results from internal digital logic to the tester reusing general purpose IOs.
adft_anasrc [ANASRCWIDTH-1:0]	O	O	**Fabric DFT Analog Source[N:0]:** This is the source signal for analog from Small Signal Array (SSA) Low Yield Analysis testing. The fabric collects the signals from all of the agents and delivers them to a hard-IP agent with an analog pad. An analog pass-gate mux will select which of the agent's output to select.

NOTE:
[1]Note1: R = required, O = optional, C = Conditional. Conditional means that if the feature is used then the signals labeled as conditional become required. It may be obvious but if the IP-block supports boundary scan the boundary signals are required.

Table 10-11. Test controller signals for content transport to/from IP-blocks

Signal	I/O	R/O/C[1]	Description
atc_clk	O	O	**Agent Test Controller Clock:** This is a reference clock that may be used optionally by the IOSF TAM Test Controller to deliver data sent by the TAM.
atc_data [TestCtrlOut_WIDTH-1:0]	O	O	**Agent Test Controller Data [N:0]:** This signal bus provides the test controller with data streaming from the tester. The fabric will gather all of the available atc_data[N:0] signals and deliver them as a single bus to the IOSF TAM test controller. Any agent (presumed to be a hard-IP agent) that can provide a bypass capability on its IOs is a candidate to participate in this bus.
ftc_clk	I	O	**Fabric Test Controller Clock:** This is a reference clock that may be used optionally by the IOSF TAM Test Controller to deliver data sent by the TAM.
ftc_data [TestCtrlIn_WIDTH-1:0]	I	O	**Fabric Test Controller Data [N:0]:** This signal bus provides the tester with data streaming from the IOSF TAM test controller (TAM-TC). The data content from test controller is usually in the form of compare results destined to the tester for pass/fail determination. Any agent (presumed to be a hard-IP agent) that can provide a bypass capability on its IOs is a candidate to participate in this bus.

NOTE:
[1]Note1: R = required, O = optional, C = Conditional. Conditional means that if the feature is used then the signals labeled as conditional become required. It may be obvious but if the IP-block supports boundary scan the boundary signals are required.

Table 10-12. Miscellaneous DFx support signals

Signal	I/O	R/O/C[1]	Description
fdfx_lbist_LV_TM	I	O	**Fabric DFx LBIST Logic Vision Test Mode:** This is the LV specific test mode signal from the LV TAP for use the LBIST controller logic.

fsta_dfxact_misc[7:0]	I	O	**Fabric SoC Trigger Arch DFx action for miscellaneous actions:** This signal group is available for the SoC integration team to implement an IP-block specific action. 0: Normal operation. 1: Force the assigned miscellaneous DFx action.
fdfx_pgcb_bypass [NUM_OF_PGCBS-1:0]	I	C	**Fabric DFx PGCB bypass.** This signal controls the Power Gate Common Block's bypass muxes to enable a DFx override signal to control the enabling of this IP-block's PGCB instantiation. **0:** Normal PGCB operation **1:** PGCB is bypassed and forces the override value **Note:** Refer to section 6.2 for more information.
fdfx_pgcb_ovr [NUM_OF_PGCBS-1:0]	I	C	**Fabric DFx PGCB override (value).** This signal controls the DFx sequencer inside of the PGCB block. The DFx sequencer automates the activation/deactivation of the power management control signals to power up or down the domain that this PGCB controls. **0:** Power gate device (PGD) is force on, meaning, the IP-block will be powered up **1:** Power gate device (PGD) is force off, meaning, the IP-block will be powered down **Note:** Refer to section 6.2 for more information.
fdfx_powergood	I	R	**Fabric DFx power good:** The DFx power good signal is required by all agents/IP-blocks that have DFx associated with them. Note: This signal is equivalent to **dfx_powergood_rst_b.**
NOTE: [1]Note1: R = required, O = optional, C = Conditional. Conditional means that if the feature is used then the signals labeled as conditional become required. It may be obvious but if the IP-block supports boundary scan the boundary signals are required.			

10.5 DFV Signal Interface Description

A graphical view of the DFV signals is shown in Figure 10-1 and Figure 10-2 with a detailed listing describing the signals in **Error! Reference source not found.**. The integration team has the freedom to assign any reasonable number of characters as a prefix or suffix as applied to the root name to identify the signal in a full chip SoC.

Figure 10-10. Graphical diagram of DFV signal interface

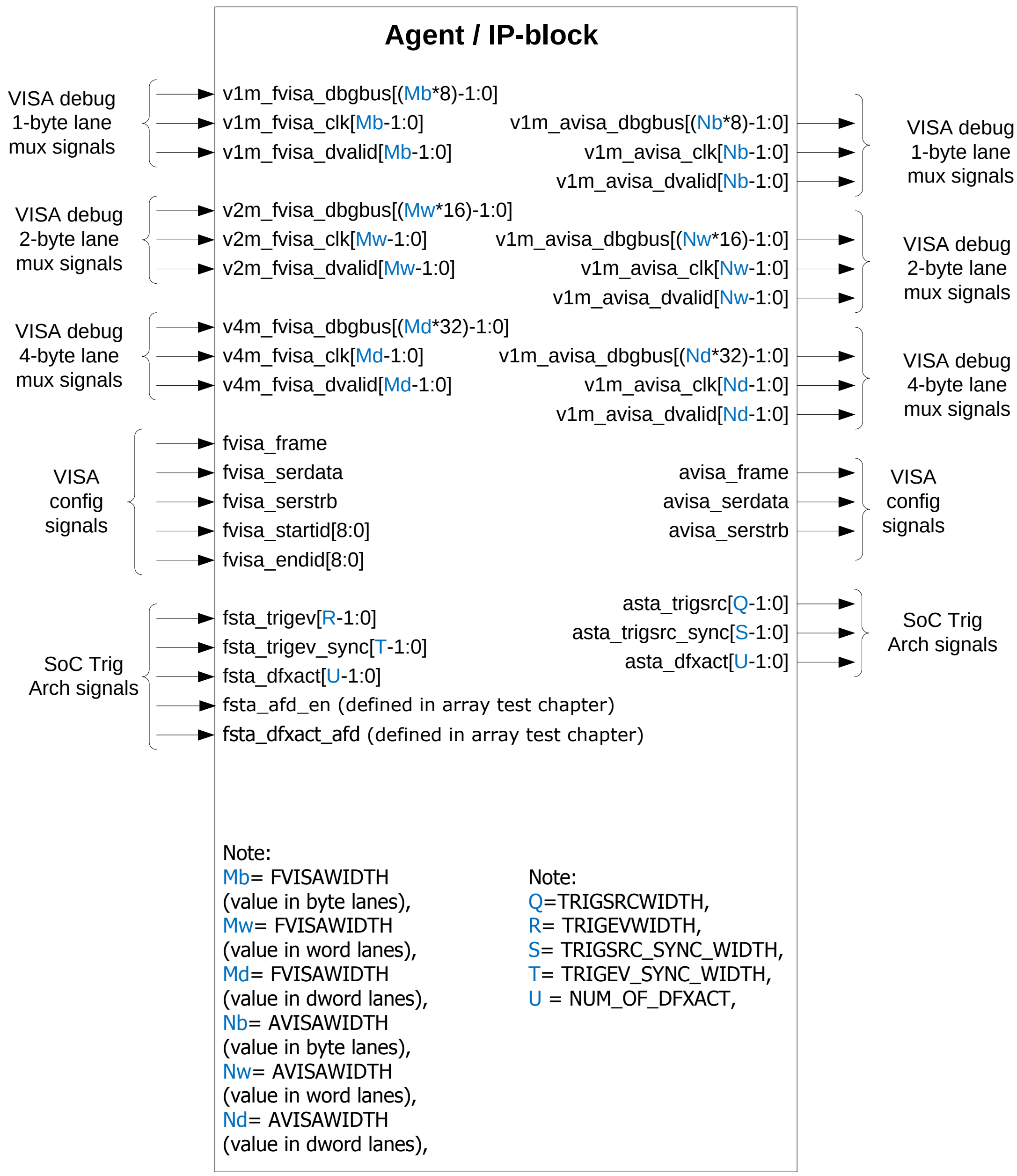


Figure 10-11. Graphical view of DFV signals continued

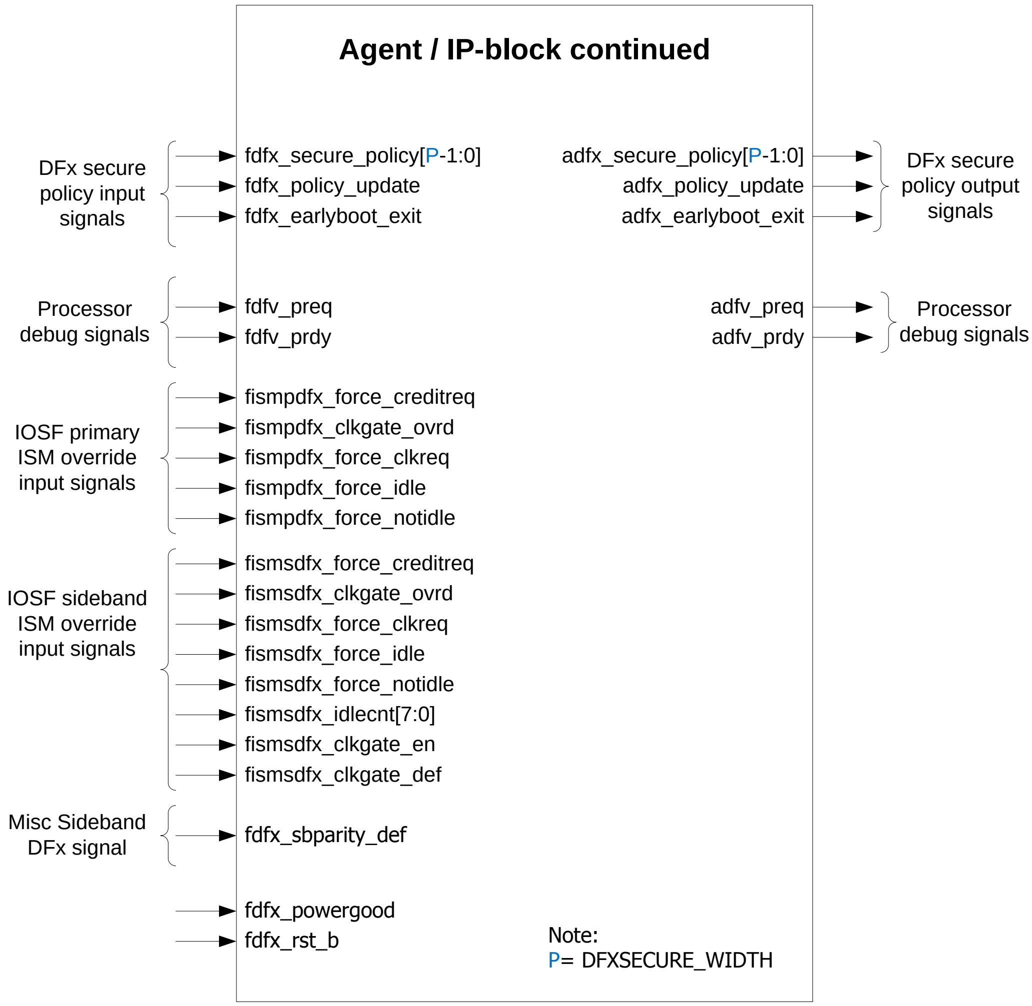


Table 10-13. VISA DFV signal descriptions

Signal	I/O	R/O/C[1]	Description
Fabric sourced: VISA debug bus			
v1m_fvisa_dbgbus [(FVISAWIDTH*8)-1:0]	O	C	**Fabric VISA Debug Bus:** This is the legacy VISA definition for IP debug observability. The debug bus is based on a byte lane mux width. The minimum grouping is one byte (8 bits) and each increment is one byte. If FVISAWIDTH = 2 then 2 * 8 = 16 bit width (*_dbgbus[15:0).
v1m_fvisa_clk [FVISAWIDTH -1:0]	O	C	**Fabric VISA Debug Bus Clock:** This signal is the clock reference for each byte lane of the debug bus. There is one clock per byte lane. This is the legacy VISA clock definition.

Signal	I/O	R/O/C[1]	Description
v1m_fvisa_dvalid [FVISAWIDTH-1:0]	O	C	**Fabric Data Valid:** This signal indicates when the VISA debug data from avisa_dbgbus is valid when this agent implements the optional VISA sync gasket. Each data valid corresponds to a byte lane. This is the legacy VISA data valid definition.
v2m_fvisa_dbgbus [(FVISA2MWIDTH*16)-1:0]	O	C	**Fabric VISA 2-byte Mux (v2m) Debug Bus:** VISA observability debug bus for this fabric IP based on a 16-bit word lane mux width. The minimum grouping is one word and each increment is a word lane. If FVISA2MWIDTH = 2 then 2 * 16 = 32 bit width (*_dbgbus[31:0).
v2m_fvisa_clk [FVISA2MWIDTH -1:0]	O	C	**Fabric VISA V2M Clock:** This signal is the clock reference for each word lane of the debug bus. There is one clock per word lane.
v2m_fvisa_dvalid [FVISA2MWIDTH-1:0]	O	C	**Fabric VISA V2M Data Valid :** This signal indicates when the VISA debug data from *_dbgbus is valid when this agent implements the optional VISA sync gasket. Each data valid corresponds to a word lane.
v4m_fvisa_dbgbus [(FVISA4MWIDTH*32)-1:0]	O	C	**Fabric VISA 4-byte Mux (v4m) Debug Bus:** VISA observability debug bus for this fabric IP based on a 32-bit double word lane mux width. The minimum grouping is one Dword and each increment is a word lane. If FVISA4MWIDTH = 2, then 2 * 32 = 64 bit width (*_dbgbus[63:0).
v4m_fvisa_clk [FVISA4MWIDTH -1:0]	O	C	**Fabric VISA V4M Clock:** This signal is the clock reference for each Dword lane of the debug bus. There is one clock per word lane.
v4m_fvisa_dvalid [FVISA4MWIDTH-1:0]	O	C	**Fabric VISA V4M Data Valid:** This signal indicates when the VISA debug data from *_dbgbus is valid when this agent implements the optional VISA sync gasket. Each data valid corresponds to a Dword lane.
v8m_fvisa_dbgbus [(FVISA8MWIDTH*64)-1:0]	O	C	**Fabric VISA 8-byte Mux (v8m) Debug Bus:** VISA observability debug bus for this fabric IP based on a 64-bit quad-word lane mux width. The minimum grouping is one Qword and each increment is a word lane. If FVISA8MWIDTH = 2, then v8m_avisa_dbgbus = 2 * 64 = 128 bit width.
v8m_fvisa_clk [FVISA8MWIDTH -1:0]	O	C	**Fabric VISA V8M Clock:** This signal is the clock reference for each Qword lane of the debug bus. There is one clock per word lane.
v8m_fvisa_dvalid [FVISA8MWIDTH-1:0]	O	C	**Fabric VISA V8M Data Valid:** This signal indicates when the VISA debug data from *_dbgbus is valid when this agent implements the optional VISA sync gasket. Each data valid corresponds to a quad word lane.
Agent sourced: VISA debug bus			
v1m_avisa_dbgbus [(AVISAWIDTH*8)-1:0]	O	O	**Agent VISA 1-byte Mux (v1m) Debug Bus:** This is the legacy VISA definition for IP debug observability. The debug bus is based on a byte lane mux width. The minimum grouping is one byte (8 bits) and each increment is one byte. If AVISAWIDTH = 2 then 2 * 8 = 16 bit width (*_dbgbus[15:0) .

Signal	I/O	R/O/C[1]	Description
v1m_avisa_clk [AVISAWIDTH -1:0]	O	O	**Agent VISA V1M Clock:** This signal is the clock reference for each byte lane of the debug bus. There is one clock per byte lane. This is the legacy VISA clock definition.
v1m_avisa_dvalid [AVISAWIDTH-1:0]	O	O	**Agent Data V1M Valid:** This signal indicates when the VISA debug data from *_dbgbus is valid when this agent implements the optional VISA sync gasket. Each data valid corresponds to a byte lane. This is the legacy VISA data valid definition.
v2m_avisa_dbgbus [(AVISA2MWIDTH*16)-1:0]	O	O	**Agent VISA 2-byte Mux (v2m) Debug Bus:** VISA observability debug bus for this agent based on a 16-bit word lane mux width. The minimum grouping is one word and each increment is a word lane. If AVISA2MWIDTH = 2 then 2 * 16 = 32 bit width (*_dbgbus[31:0).
v2m_avisa_clk [AVISA2MWIDTH -1:0]	O	O	**Agent VISA V2M Clock:** This signal is the clock reference for each word lane of the debug bus. There is one clock per word lane.
v2m_avisa_dvalid [AVISA2MWIDTH-1:0]	O	O	**Agent VISA V2M Data Valid:** This signal indicates when the VISA debug data from *_dbgbus is valid when this agent implements the optional VISA sync gasket. Each data valid corresponds to a word lane.
v4m_avisa_dbgbus [(AVISA4MWIDTH*32)-1:0]	O	O	**Agent VISA 4-byte Mux (v4m) Debug Bus:** VISA observability debug bus for this agent based on a 32-bit double word lane mux width. The minimum grouping is one Dword and each increment is a word lane. If AVISA4MWIDTH = 2, then 2 * 32 = 64 bit width (*_dbgbus[63:0).
v4m_avisa_clk [AVISA4MWIDTH -1:0]	O	O	**Agent VISA V4M Clock:** This signal is the clock reference for each Dword lane of the debug bus. There is one clock per word lane.
v4m_avisa_dvalid [AVISA4MWIDTH-1:0]	O	O	**Agent VISA V4M Data Valid:** This signal indicates when the VISA debug data from *_dbgbus is valid when this agent implements the optional VISA sync gasket. Each data valid corresponds to a Dword lane.
v8m_avisa_dbgbus [(AVISA8MWIDTH*64)-1:0]	O	O	**Agent VISA 8-byte Mux (v8m) Debug Bus:** VISA observability debug bus for this agent based on a 64-bit quad-word lane mux width. The minimum grouping is one Qword and each increment is a word lane. If AVISA8MWIDTH = 2, then v8m_avisa_dbgbus = 2 * 64 = 128 bit width.
v8m_avisa_clk [AVISA8MWIDTH -1:0]	O	O	**Agent VISA V8M Clock:** This signal is the clock reference for each Qword lane of the debug bus. There is one clock per word lane.
v8m_avisa_dvalid [AVISA8MWIDTH-1:0]	O	O	**Agent VISA V8M Data Valid:** This signal indicates when the VISA debug data from *_dbgbus is valid when this agent implements the optional VISA sync gasket. Each data valid corresponds to a quad word lane.

Signal	I/O	R/O/C[1]	Description
Fabric sourced: VISA serial configuration bus			
fvisa_serstrb	I	R	**Fabric Serial Strobe:** This strobe indicates when the data bit on fvisa_serdata is valid. Note: This signal is connected to serial_cfg_in[0] when using the VISA insertion tool.
fvisa_frame	I	R	**Fabric VISA Frame:** This signal frames the serial register access packet that is sent from the VISA central controller to the PLMs and then on to the ULMs. Note: This signal is connected to serial_cfg_in[1] when using the VISA insertion tool.
fvisa_serdata	I	R	**Fabric Serial Data:** This is the data stream sent from the VISA central controller to the PLMs then on to the ULMs. The data stream contains control, address and data. The data is the local VISA mux control register information. Note: This signal is connected to serial_cfg_in[2] when using the VISA insertion tool.
VISA agent sourced: VISA serial configuration bus			
avisa_serstrb	O	O	**Agent Serial Strobe:** This strobe indicates when the data bit on avisa_serdata is valid. Note: This signal is connected to serial_cfg_out[0] when using the VISA insertion tool.
avisa_frame	O	O	**Agent VISA Frame:** This signal frames the serial register access packet that is sent from the VISA central controller to the PLMs and then on to the ULMs. Note: This signal is connected to serial_cfg_out[1] when using the VISA insertion tool.
avisa_serdata	O	O	**Agent Serial Data:** This is the data stream sent from the VISA central controller to the PLMs then on to the ULMs. The data stream contains control, address and data. The data is the local VISA mux control register information. Note: This signal is connected to serial_cfg_out[2] when using the VISA insertion tool.
VISA strap pin interface signals			
fvisa_startid[8:0]	I	O	**Fabric VISA starting ID[8:0]:** This signal bus identifies the starting ULM/PLM unit ID for the muxes within the IP-block. It is implementation dependent for the IP-block developer to manage the number of unit IDs for the overall IP using the startid and endid signal groups. Note: This pin interface is expected to be a strap value for the ULMs (or PLMs) and not a fabric bus.

Signal	I/O	R/O/C[1]	Description
fvisa_endid[8:0]	I	O	**Fabric VISA ending ID[8:0]:** This signal bus identifies the ending ULM/PLM unit ID for the muxes with the HIP. It is implementation dependent for the HIP developer to manage the number of unit IDs for the overall IP using the startid and endid signal groups. The endid is required if the register IDs are exhausted for one startid value. There are 32 IDs available and a typical ULM with 4 lanes consumes 5 IDs. Note: This pin interface is expected to be a strap value for the ULMs (or PLMs) and not a fabric bus. Note: This signal interface is intended only for HIPs. Any SIP should not use this interface because the SoC integration team will use the auto-ID feature in the VISA toolkit.

NOTE:
1. R = required, O = optional, C = conditional. Conditional means that if the feature is used then the signals labeled as conditional become required.

Table 10-14. Trigger DFV signals

Signal	I/O	R/O/C[1]	Description
asta_trigsrc [TRIGSRCWIDTH-1:0]	O	C	**Agent SoC Trigger Architecture (STA) Trigger Source [N-1:0]:** This is an asynchronous output signal from this IP-block's internal logic that is generating triggers for use by the SoC. There may be TRIGSRCWIDTH number of trigger outputs from this agent. Note: This is signal is expected to be a multi-cycle path for the backend timing analysis tools.
fsta_trigev [TRIGEVWIDTH-1:0]	I	C	**Fabric SoC Trigger Architecture (STA) Trigger Event [N-1:0]:** This is a debug trigger event input from either the regional or master DFx unit to this agent. This event is used for asserting response functions within the agent for debug, validation, and survivability features where an event driven assertion alters the behavior of the agent. One example is an error injection logic based on an event generated from a trigger source. There may be TRIGEVWIDTH number of trigger event inputs to this agent. Note: This is signal is expected to be a multi-cycle path for the backend timing analysis tools.
asta_trigsrc_sync [TRIGSRC_SYNC_WIDTH-1:0]	O	C	**Agent SoC Trigger Architecture (STA) Trigger Source Sync[N-1:0]:** This is a synchronous output signal from an internal logic block that is generating triggers for use by the regional and master DFx units in the fabric. There may be TRIGSRC_SYNC_WIDTH number of trigger outputs from this agent. Note: This is signal is a synchronous path that requires a single cycle or flop stage to reach the destination. This signal cannot be a multi-cycle path.

fsta_trigev_sync [TRIGEV_SYNC_WIDTH-1:0]	I	C	**Fabric SoC Trigger Architecture (STA) Trigger Event Sync [N-1:0]:** This is a debug trigger event input from either the regional or master DFx unit to this agent. This event is used for asserting response functions within the agent for debug, validation, and survivability features where an event driven assertion alters the behavior of the agent. One example is an error injection logic based on an event generated from a trigger source. There may be TRIGEV_SYNC_WIDTH number of trigger event inputs to this agent. Note: This is signal is a synchronous path that requires a single cycle or flop stage to reach the destination. This signal cannot be a multi-cycle path.
asta_dfxact [NUM_OF_DFXACT-1:0]	O	C	**Agent SoC Trigger Architecutre (STA) DFx Action[P-1:0].** This signal is intended to be a DFx action output from a trigger control block. For Chassis DFx Gen2 this would apply to the Cluster Trigger Block (CTB). DFx actions are driving passive DFx IPs that alter the behavior of the SoC. For example, an array/freeze/dump is a passive consumer of a DFx action. A trigger block such as a micro breakpoint controller would use the fsta_trigev signal. **Note:** this signal is for new IP and SoC developments.
fsta_dfxact [NUM_OF_DFXACT-1:0]	I	C	**Fabric SoC Trigger Architecutre (STA) DFx Action[P-1:0].** This signal enables a DFx action to occur such as a clock stop (clock freeze), array/freeze/dump (AFD), launch a TAP2SB transaction, etc. It is a passive DFx feature that a trigger event is expected to enable. It is not intended for active trigger IPs such as micro breakpoint controllers or transaction match/mask IPs (similar to PSF Fabric Trace Hook). **Note:** this signal is for new IP and SoC developments.
fsta_dfxact_afd	I	C	**Fabric SoC trigger arch DFx action for array/freeze/dump:** Defined in Section 5.5. Reprinted here to show all of the {f,a}sta_* signals.
fsta_afd_en	I	C	**Fabric SoC trigger arch DFx action enable:** Defined in Section 5.5. Reprinted here to show all of the {f,a}sta_* signals.

NOTE:
[1]R = required, O = optional, C = conditional. Conditional means that if the feature is used then the signals labeled as conditional become required.

1

1 ## Table 10-15. DFV security signals

Signal	I/O	R/O/C[1]	Description
fdfx_secure_policy [DFXSECURE_WIDTH-1:0]	I	R	**Fabric DFx security policy.** This bus is a binary encoded value of the security policy that is the current state of the SoC. The parameter value is constant for a given SoC generation and aligned with a process node. All IP-blocks must have the same width. For this revision of the IOSF DFx HAS: DFXSECURE_WIDTH = 4 Note: An agent/IP-block may have two DFx secure policy plug-in blocks one for sTAP and one for the agent/IP-block.
fdfx_policy_update	I	R	**Fabric DFx policy update.** This is the latch enable signal to capture the policy value to prevent glitches. 0: Latch values 1: Update to new policy value Note: An agent/IP-block may have two DFx secure policy plug-in blocks one for sTAP and one for the agent/IP-block.
fdfx_earlyboot_exit	I	R	**Fabric DFx early boot exit.** This signal indicates when the early boot debug window is closed. 0: Debug capabilities are available during this phase of the boot flow 1: DFx security policy must be used Note: An agent/IP-block may have two DFx secure policy plug-in blocks one for sTAP and one for the agent/IP-block.
adfx_secure_policy [DFXSECURE_WIDTH-1:0]	O	O	**Agent DFx security policy.** This bus is a binary encoded value of the security policy that is the current state of the SoC. The parameter value is constant for a given SoC generation and aligned with a process node. All IP-blocks must have the same width. The security agent will output this bus as the source of the security policy information but this may be used as a pass-through from the agent to another IP-block within an agent. For 14nm chassis: DFXSECURE_WIDTH = 4
adfx_policy_update	O	O	**Agent DFx policy update.** This signal is the latch enables to capture the policy value to prevent glitches. 0: Latch values 1: Update to new policy value
adfx_earlyboot_exit	O	O	**Fabric DFx early boot exit.** This signal indicates when the early boot debug window is closed. 0: Debug capabilities are available during this phase of the boot flow 1: DFx security policy must be used

> **NOTE:**
> [1]R = required, O = optional, C = conditional. Conditional means that if the feature is used then the signals labeled as conditional become required.

Table 10-16. ISM override DFV signals

Signal	I/O	R/O/C[1]	Description
Primary ISM debug override signals			
fismpdfx_force_creditreq	I	R	**Fabric Primary ISM DFx force credit request:** This input forces the primary Idle State Machine to request its credits. 0: normal operation 1: force IP to request its credits Requirements Note: If an agent/IP-block provides a TAP TDR to drive the ISM override signals within the agent (IP-block) then this signal can be waived from being required on the IOSF DFx interface. This input signal may be distributed to N number of primary interfaces within this IP. There is no requirement to independently control each ISM override.
fismpdfx_clkgate_ovrd	I	R	**Fabric Primary ISM DFx clock gate override:** This input overrides the local clock gating mux to the Idle State Machine, meaning, that it enables the clock locally. To use this feature for power measurements the SoC must drive this signal individually meaning each Sideband ISM and each primary ISM from any local TAP for this to be effective. 0: normal operation 1: override clock gate by forcing the clock to turn on. Requirements Note: If an agent/IP-block provides a TAP TDR to drive the ISM override signals within the agent (IP-block) then this signal can be waived from being required on the IOSF DFx interface. This is signal is optional because it can be controlled internally to the IP-block with registers bits on the Sideband interface. This assumes the register override bits are on the Sideband clock domain and the fismsdfx_clkgate_ovrd is implemented to override the sideband clock gate (required). This input signal may be distributed to N number of primary interfaces within this IP. There is no requirement to independently control each ISM override.

fismpdfx_force_clkreq	I	R	**Fabric Primary ISM DFx force clock request:** This input forces the agent to request a clock active condition in the Idle State Machine. This use model assumes to be in clock idle and we force a clock active to the fabric. 0: normal operation 1: force clock request Requirements Note: If an agent/IP-block provides a TAP TDR to drive the ISM override signals within the agent (IP-block) then this signal can be waived from being required on the IOSF DFx interface. This input signal may be distributed to N number of primary interfaces within this IP. There is no requirement to independently control each ISM override.
fismpdfx_force_idle	I	R	**Fabric Primary ISM DFx force idle:** This input forces the agent to the idle state of the Idle State Machine. 0: normal operation 1: force idle Requirements Note: If an agent/IP-block provides a TAP TDR to drive the ISM override signals within the agent (IP-block) then this signal can be waived from being required on the IOSF DFx interface. This input signal may be distributed to N number of primary interfaces within this IP. There is no requirement to independently control each ISM override.
fismpdfx_force_notidle	I	R	**Fabric Primary ISM DFx force not idle:** This input forces the agent's Idle State Machine not to go to idle. 0: normal operation 1: force not idle Requirements Note: If an agent/IP-block provides a TAP TDR to drive the ISM override signals within the agent (IP-block) then this signal can be waived from being required on the IOSF DFx interface. This input signal may be distributed to N number of primary interfaces within this IP. There is no requirement to independently control each ISM override.
Sideband ISM debug override signals			

fismsdfx_force_creditreq	I	R	**Fabric Sideband ISM DFx force credit request:** This input forces the primary Idle State Machine to request its credits. 0: normal operation 1: force IP to request its credits Requirements Note: If an agent/IP-block provides a TAP TDR to drive the ISM override signals within the agent (IP-block) then this signal can be waived from being required on the IOSF DFx interface. This input signal may be distributed to N number of Sideband interfaces within this IP. There is no requirement to independently control each ISM override.
fismsdfx_clkgate_ovrd	I	R	**Fabric Sideband ISM DFx clock gate override:** This input overrides the local clock gating mux to the Idle State Machine, meaning, that it enables the clock locally. 0: normal operation 1: override clock gate by forcing the clock to turn on Note1: This signal is connected to the ISM signal named jta_clkgate_ovrd. Use of this signal as a bus is optional. Requirements Note: If an agent/IP-block provides a TAP TDR to drive the ISM override signals within the agent (IP-block) then this signal can be waived from being required on the IOSF DFx interface. This input signal may be distributed to N number of Sideband interfaces within this IP. There is no requirement to independently control each ISM override.
fismsdfx_force_clkreq	I	R	**Fabric Sideband ISM DFx force clock request:** This input forces the agent to request a clock active condition in the Idle State Machine. This use model assumes to be in clock idle and we force a clock active to the fabric. 0: normal operation 1: force clock request Note1: This signal is connected to the ISM signal named jta_force_clkreq. Use of this signal as a bus is optional. Requirements Note: If an agent/IP-block provides a TAP TDR to drive the ISM override signals within the agent (IP-block) then this signal can be waived from being required on the IOSF DFx interface. This input signal may be distributed to N number of Sideband interfaces within this IP. There is no requirement to independently control each ISM override.

fismsdfx_force_idle	I	R	**Fabric Sideband ISM DFx force idle:** This input forces the agent to the idle state of the Idle State Machine. 0: normal operation 1: force idle Note1: This signal is connected to the ISM signal named jta_force_idle. Use of this signal as a bus is optional. Requirements Note: If an agent/IP-block provides a TAP TDR to drive the ISM override signals within the agent (IP-block) then this signal can be waived from being required on the IOSF DFx interface. This input signal may be distributed to N number of Sideband interfaces within this IP. There is no requirement to independently control each ISM override.
fismsdfx_force_notidle	I	R	**Fabric Sideband ISM DFx force not idle:** This input forces the agent's Idle State Machine not to go to idle. 0: normal operation 1: force not idle Note1: This signal is connected to the ISM signal named jta_force_notidle. Use of this signal as a bus is optional. Requirements Note: If an agent/IP-block provides a TAP TDR to drive the ISM override signals within the agent (IP-block) then this signal can be waived from being required on the IOSF DFx interface. This input signal may be distributed to N number of Sideband interfaces within this IP. There is no requirement to independently control each ISM override.
fismsdfx_idlecnt[7:0]	I	R	**Fabric Sideband ISM DFx idle count:** This bus sets the idle counter default value that determines when the endpoint should transition to IDLE_REQ. Default value is 8'h10 (16 dec). This input signal may be distributed to N number of Sideband interfaces within this IP. There is no requirement to independently control each ISM override.
fismsdfx_clkgate_en	I	R	**Fabric Sideband ISM DFx clock gate enable:** This input forces the agent's Idle State Machine not to go to idle. 0: ISM never leaves the ACTIVE state which forces the clock to remain on. 1: Normal operation. Allows ISM to leave ACTIVE. Clocking gating occurs normally once in IDLE. Default value = 1'b1 This input signal may be distributed to N number of Sideband interfaces within this IP. There is no requirement to independently control each ISM override.

fismsdfx_clkgate_def	I	R	**Fabric Sideband ISM DFx clock gate defeature:** This signal will defeature the clock gating enable signal (fismsdfx_clkgate_def). 0: normal operation 1: Disable clock gating feature Default value = 1’b0 This input signal may be distributed to N number of Sideband interfaces within this IP. There is no requirement to independently control each ISM override.
NOTE: [1]R = required, O = optional, C = conditional. Conditional means that if the feature is used then the signals labeled as conditional become required.			

Table 10-17. General DFV signals

Signal	I/O	R/O/C[1]	Description
General debug signals			
fdfx_sbparity_def	I	R	**Fabric DFx Sideband parity defeature.** This signal disables parity checking within the Sideband endpoint. When asserted all parity must be treated as good parity. Incoming payload from the Sideband router will be propagated to the agent as if it was good parity regardless of the parity logic result. Also, the outgoing transactions will have the parity signal (sbe_sbi_parity_err_out) deasserted even if the parity generation logic created a bad result. 0: Enable parity checking 1: Disable parity checking
fdfx_pgcb_bypass [NUM_OF_PGCBS-1:0]	I	C	**Fabric DFx PGCB bypass.** This signal controls the Power Gate Common Block’s bypass muxes to enable a DFx override signal to control the enabling of this IP-block’s PGCB instantiation. 0: Normal PGCB operation 1: PGCB is bypassed and forces the override value **Note:** This signal is part of the Misc DFT signal group but remain here for legacy reasons. Refer to section 6.2 for more information.

fdfx_pgcb_ovr [NUM_OF_PGCBS-1:0]	I	C	**Fabric DFx PGCB override (value).** This signal controls the DFx sequencer inside of the PGCB block. The DFx sequencer automates the activation/deactivation of the power management control signals to power up or down the domain that this PGCB controls. **0:** Power gate device (PGD) is force on, meaning, the IP-block will be powered up **1:** Power gate device (PGD) is force off, meaning, the IP-block will be powered down **Note:** This signal is part of the Misc DFT signal group but remain here for legacy reasons. Refer to section 6.2 for more information.
fdfx_preq	I	C	**Fabric DFV preq.** This signal forces the processor to break execution and halt for debug operations. If an IP-block contains a Minute-IA processor (Lakemont rev2.1 or later) or processor that supports entering probe mode with a hardware assertion then this signal is required. **0:** Normal processor operation **1:** preq asserted. **Note:** This signal is defined active high. Previous IP-block releases with active low signaling ("_b" in the signal name) will be waived.
adfx_prdy	O	C	**Agent DFV prdy.** This signal indicates that a processor has entered into probe mode due to a previous preq assertion or due to an internal event. If an IP-block contains a Minute-IA processor (Lakemont rev2.1 or later) or processor that supports entering probe mode with a hardware assertion then this signal is required. **0:** Normal processor operation **1:** prdy is asserted **Note:** This signal is defined active high. Previous IP-block releases with active low signaling ("_b" in the signal name) will be waived.
fdfx_powergood	I	R	Refer to section 6.2. Note: This signal is equivalent to **dfx_powergood_rst_b**.
fdfx_rst_b	I	C	**Fabric DFx reset bar:** This signal is for resetting the VISA ULM only. It is the responsibility of the IP-block developer to internally logically combine this signal with force_rst_b if a PGCB block is instantiated in the IP. **Note 1:** Soft-IP blocks are required to include fdfx_rst_b on the IP interface if this IP has VISA. **Note 2:** If a hard-IP block is designated as a secure IO IP then it must implement fdfx_rst_b connecting to all VISA ULMs/PLMs within the IP. Otherwise, for all other hard IP-blocks it is optional.

NOTE:
[1]R = required, O = optional, C = conditional. Conditional means that if the feature is used then the signals labeled as conditional become required.

§

11 Feature Use Models

Use models are presented throughout the chapters when needed to explain how a particular feature works.

§

12 Re-Use Collateral & Information

This chapter is not relevant for this type of HAS document. This is an interface specification and not a specific DFx IP.

§

13 Validation Strategy

This chapter is not relevant for this type of HAS document. This is an interface specification and not a specific DFx IP.

§

14 Prior Work

Refer to the IOSF DFx rev1.1 for prior work on IOSF DFx.

§

15 Open Issues

15.1 List of Open Issues

1. No open issues as of this revision.

§

16 Feature Wish List

1) A potential future feature to support would be IEEE1687.
2) The IEEE1149.1-2012 spec was approved during the final draft publication. Although we have the opcodes reserved for the features, the IOSF may need to include more details in the boundary scan section of this document.
3) The IEEE 1149.8.1 is still in draft form as of this publication. Although we have the opcodes reserved for the features, the IOSF may need to include more details in the boundary scan section of this document.
4) It was a mistake to label the power good reset signal as fdfx_powergood. It should be fdfx_powergood_rst_b. In a future revision of this spec, it may change. There is no difference in these two signals.
   a) fdfx_powergood = 0…………..: the power is not good so reset the flop
   b) fdfx_powergood_rst_b = 0…: reset the flop (active low reset)
   c) fdfx_powergood =1…………….: the power is good so do not reset.
   d) fdfx_powergood_rst_b = 1…: no reset
5) The VISA section should be updated in the next revision. Althought the information at a high level is still valid it continues to evolve and the reader should refer to the VISA specification for more information.

§

17 Glossary and Reference

List documents on which this HAS is based or that could provide further understanding into the unit's functionality.

17.1 Terminology

Table 17-1. Terminology

Term	Description
TAP	Test Access Port: This is the serial port access mechanism from the IEEE1149.1 specification. While the 1149.1 spec describes an entire boundary scan based test system, the TAP is referring to the serial communication port portion of the spec. This is what we would use to access Intel-only test data registers to perform specific test and debug operations.
CLTAP	Chip-Level TAP. There is only one CLTAP in an SoC.
sTAP	Slave TAP. This is the Intel defined SoC TAP HAS compliant TAP used for Intel IP-blocks and DFx fabric TAPs.
TAP hierarchical topology	The CLTAP controls a set of slave TAPs on it level of hierarchy. A select register determines which TAP is Normal with CLTAP. A slave TAP can be the master of a sub-network. This multi-dimensional array of TAPs is the critical component in a modular-DFx fabric that allows rapid changes without impacting the integration.
TAP hierarchical-hybrid	The TAPs at the CLTAP TAP network (hierarchy level0) can be programmatically set to the primary or secondary TAP network. The hierarchical-hybrid network was an attempt at making the TAPs at hierarchical level 1 accessible from the secondary TAP port. This is useful for third party TAP host controllers that do not understand the 1149.7 network. They require their own TAP to be the sole TAP between TDI and TDO. For example, as of this writing the Logic Vision MBIST TAP must be the only TAP between TDI and TDO when using their software. A pure hierarchical topology will address this with the use of the tertiary TAP port.
SCC	Scan Clock Controller: This is a scan IP-block that controls the local functional clock for scan test operations.
SRC	Scan Reset Controller: This is a scan IP-block that controls the functional reset for scan test operations. It is essentially a striped down version of the SCC.
SRC	Scan Asynchronous Scan Controller: This is a scan IP-block that controls the scan signals such as the fscan_latchopen or the fscan_clkungate for scan test operations. It is essentially a striped down version of the SCC.
IDV	Intra-Die Variation monitor. This is a DFx IP-block that is placed at regular intervals across the die to monitor variations in process parasitics.

Term	Description
VISA	Visualization of Internal Signals Architecutre (VISA). This is a methodology and tool flow to instrument IP-blocks and fabrics with signals expected to assist in debugging the IP. It is composed of three muxes at different levels of hierarchy. A Unit Level Mux (ULM) is the lowest level mux and usually found in IP-blocks. A Partition Level Mux (PLM) collect outputs from ULMs and is located in the DFx fabric within the Region or Cluster DFx unit. The Central Level Mux (CLM) collects the outputs from all of the PLMs. CLM contains a crossbar output and N number of output byte lanes. The cross bar output is used for triggering and counting of events with the SoCHAP counters inside of Lakemore. The byte lane outputs connect to the On-Die Logic Analyzer (ODLA) IP-block in Lakemore to compress the debug data and output to one of several destinations.
ISM	Idle State Machine. This is the IOSF primary and sideband state machine that initializes credits and manages clock gating for power management.

17.2 Reference Documents

The following specifications provide additional details about the various DFx features described within this specification.

Table 17-2. Reference Documents

Document	Document No./Location
SoC (JTAG) TAP HAS rev 1.0 or later	
SoC Scan Requirements Handbook rev 0.70 or later	
Visualization of Internal Signals Architecture (VISA) rev 0.80 or later	
Memory BIST Flow and Debug Handbook rev2.0	
North Peak collection of HAS documents (rev0.70).	
Chassis 2.1 DFx Security HAS rev1.0 (or later)	https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/chassisWG/Security/HAS

17.3 Author / Acknowledgement List

Table 17-3. Author - Acknowledgement List

Name	Group/Division (eg. IDGa/CSA)	Description of contribution
Mike Wiznerowicz	IDGa/CSA	Primary author, owner of this specification.
		Many people contributed to this spec’s development without them this document would not be possible: For TAP, scan, misc: Bowden, Scott J; Easter, Jonathan P; Hussey, Jenifer Lee, Andrew Yoon Fah; Liew, Vui Yong; Sachan, Sunjiv; Bulusu, Shivaprashant; Seshadri, Sandhya; Tice, Christopher; Pappu, Lakshminarayana; others… For the DFV: Baartmans, Sean; Ruybalid, Victor; Sandri, Jason G; For the DFx security sections: Carrieri, Enrico D; Neve De Mevergnies, Michael Sastry, Manoj R;

17.4 Web Links

17.4.1 Chassis/SoC DFx HAS repository:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC%20DFx/Forms/AllItems.aspx

17.4.2 IOSF DFx:

Text:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC DFx/Interface DFx (IOSF, etc.)

Link:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC%20DFx/Interface%20DFx%20(IOSF,%20etc.)/IOSF%20DFx

17.4.3 SoC TAP:

Text:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC DFx/DFx Fabrics (TAP, etc.)/TAP

Link:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC%20DFx/DFx%20Fabrics%20(TAP,%20etc.)/TAP

17.4.4 Scan components:

Text:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC DFx/DFx IPs (Lakemore, SCC, etc.)/DFT HAS Docs (scan, TAM, etc.)

Link:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC%20DFx/DFx%20IPs%20(Lakemore,%20SCC,%20etc.)/DFT%20HAS%20Docs%20(scan,%20TAM,%20etc.)/SCC%20components

17.4.5 SoC DFT Handbook:

Text:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC DFx/DFx Overview Docs/SoC DFT Handbook

Link:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC%20DFx/DFx%20Overview%20Docs/SoC%20DFT%20Handbook

17.4.6 SoC Scan Requirements:

Text:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC DFx/DFx IPs (Lakemore, SCC, etc.)/DFT HAS Docs (scan, TAM, etc.)/SoC Scan Requirements Handbook

Link:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC%20DFx/DFx%20IPs%20(Lakemore,%20SCC,%20etc.)/DFT%20HAS%20Docs%20(scan,%20TAM,%20etc.)/SoC%20Scan%20Requirements%20Handbook

17.4.7 MBIST:

Text:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC DFx/DFx Methodologies (MBIST, LBIST, VISA, etc.)/Memory BIST

Link:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC%20DFx/DFx%20Methodologies%20(MBIST,%20PSMI,%20VISA,%20etc.)/Memory%20BIST

17.4.8 VISA:

Text:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC DFx/DFx Methodologies (MBIST, LBIST, VISA, etc.)/VISA

Link:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC%20DFx/DFx%20Methodologies%20(MBIST,%20PSMI,%20VISA,%20etc.)/VISA

17.4.9 DFx Security Framework:

Link:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/chassisWG/Security/HAS

17.4.10 Soft IP-block DFx:

Text:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC DFx/DFx for SIP and HIP/SIP DFx

Link:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC%20DFx/DFx%20for%20SIP%20and%20HIP/SIP%20DFx

17.4.11 Hard IP-block DFx:

Text:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC DFx/DFx for SIP and HIP/HIP DFx/14nm HIP

Link:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC%20DFx/DFx%20for%20SIP%20and%20HIP/HIP%20DFx

17.4.12 Chassis Test Controller and functional test:

Text:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC DFx/DFx IPs (Lakemore, SCC, etc.)/DFT HAS Docs (scan, TAM, etc.)/Chassis Test Controller/10_Spec releases

Link:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC%20DFx/DFx%20IPs%20(Lakemore,%20SCC,%20etc.)/DFT%20HAS%20Docs%20(scan,%20TAM,%20etc.)/Chassis%20Test%20Controller

17.4.13 North Peak:

Text:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC DFx/DFx IPs (Lakemore, SCC, etc.)/Debug_Val HAS Docs (Lkmr, NrthPk, etc)/20_North Peak

Link:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC%20DFx/DFx%20IPs%20(Lakemore,%20SCC,%20etc.)/Debug_Val%20HAS%20Docs%20(Lkmr,%20NrthPk,%20etc)/20_North%20Peak

17.4.14 Cluster Trigger Block Architecture:

Text:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC DFx/DFxIPs(Lakemore, SCC, etc.)/Debug_Val HAS Docs (Lkmr, NrthPk, etc)/SoC Trigger Architecture

Link:

https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/DFxChassisWG/SoC%20DFx/DFx%20IPs%20(Lakemore,%20SCC,%20etc.)/Debug_Val%20HAS%20Docs%20(Lkmr,%20NrthPk,%20etc)/SoC%20Trigger%20Architecture

§