Slave TAP (sTAP)

INTEGRATION GUIDE

IP Rev. 2020WW05-PIC6-V1

January 2020

Intel Top Secret

Contents

About This Template

How to Use This Template

Do not remove any headings from this document. If you do not need the headings to describe your IP, enter “Not applicable” under the heading. This lets the reader know that you did not overlook this topic.

In the main document that follows, add new headings that you need to fully describe the integration of this IP. Add them in the appropriate chapters.

Most red text in this document contains instructions for filling out the section where it appears. The tag for most of this red text is called “Gaps.” You should replace this text with the content appropriate for that section, ensuring that the text is tagged appropriately (for example, with the BodyText or List Bullet style). If a section is not relevant, do not remove it; instead just replace the “Gap” text with “Not applicable” and apply the BodyText style.

Goal of This Document

This document should contain all information an integration team would need to accomplish the task without needing to seek help from another source. Try not to refer to other documents for required information; do so only if you include specific instructions for obtaining those documents, and only if you are sure your audience has access to them. Verify all links. This should be a self-contained guide for integration.

1 Introduction

1.1 Audience

The information in this document is intended for an integration team that is integrating this IP/IPSS into an SoC.

1.2 References

If you need more information on this IP/IPSS, you may find these documents helpful.

Document Title	Location
Slave TAP (sTAP) External Architecture Description	Included in the 'doc' directory for this release
Slave TAP (sTAP) Product Brief	Included in the 'doc' directory for this release

1.3 Contact Information

If you need additional help, use the contact information below.

Function	Name	Email
IP Verification	Bulusu, Shivaprashant Bandana, V, Sudheer Chelli, Vijaya	shivaprashant.bulusu@intel.com sudheer.v.bandana@intel.com vijaya.chelli@intel.com
Design	Kandula,Rakesh Adithya, B S	rakesh.kandula@intel.com b.s.adithya@intel.com
Doc Template Owner	Flowers, Susann	susann.flowers@intel.com

1.4 Terminology

The table below defines uncommon terms used in this document.

Term	Definition
sTAP	Slave TAP
Cltapc	Cluster TAP or Master Tap

1.5 Document Revision History

Fill in the revision dates below for each revision level. Add a revision level, if needed.

Each SS/IP drop will have a complimentary Integration Guide revision updated with changes in the <SS/IP root path>/doc directory. If the guide is published online, then a link in the 'doc' directory will point to that drop's Integration Guide release.

Revision Number	Description of Change	Date	Revised By
0.1	Draft	1 Sept 2011	Herb
0.2	Initial formatting	8 Sept 2011	Sandy

Revision Number	Description of Change	Date	Revised By
0.21	Draft edit	12 Sept 2011	Herb
0.22	Added comments from Sandy	13 Sept 2011	Herb
0.23	Formatted and fixed headings and tables.	13 Sept 2011	Sandy
025	Edit 2	3 Oct 2011	Herb
0.41	Updated copyright page, repositioned figure captions.	13 Oct 2011	Sandy
0.45	Major edit	28 Nov 2011	Sandy
1.0v4	HDK update	08 Apr 2015	Sudheer
PIC3_R1P0_V1	PCR 1604200691 is updated	08 Jul 2016	Sudheer
PIC4_RTL1P0_V1	TSA/MAT setup. Global macros updated in RTL.	15 June 2017	Adithya
PIC4_RTL1P0_V2	Bug fixes in SWCOMP	12 November 2017	Adithya
PIC5_RTL1P0_V1	TSA updates. Enabling of Spyglass CDC, Lint and VCLP	22 March 2018	Adithya
PIC5_RTL1P0_V2	TSA Prime updates. New TDR (0x22) added. Caliber enabled.	27 July 2018	Adithya
PIC5_RTL1P0_V3	Hot fix	1 Nov 2018	Adithya
PIC5_RTL1P0_V4	Tool updates. New customer ADPS added	18 Jan 2018	Adithya
PIC5_RTL1P0_V5	Tool updates. HSD’s addressed	21 Aug 2019	Adithya
PIC6_RTL1P0_V1	Tool updates. HSD’s addressed	27 Jan 2020	Adithya

2 Quick Start

Make sure everyone can run listed scripts. Specify if any special access is needed.

2.1 Downloading Sub IP

Not applicable to this IP/IPSS; all sub-IPs are uniquified.

Downloading the VCs Required for Compilation:-

1. Go to this link to download the version of latest JTAG BFM 2.4 version
2. Transfer the .gz file to a UNIX server and place it in a shared area of your site.
3. Enter the following command:

```
setenv JTAG_BFM_VER <shared JTAG BFM dir> in
cfg/ace/temmplates/stap.env.template
```

Note: Follow these steps whenever you want the latest version of the JTAG BFM.

2.2 Firmware Version

Not applicable to this IP/IPSS

2.3 Design Libraries

Also see section 4.4, HIP Libraries Included in Release, and section 8.4, Verification Libraries.

Library	Version	Special Usage
Ctech	v15ww50e	For all Ctech cells in ICL configuration
Ctech	v15ww50e	For all Ctech cells in ICPLP configuration

2.4 Defines and Variables

See also section 5.5, RTL Configuration Parameters and Overrides.

2.5 IP/IPSS Straps

See also section 5.9, Fuses, for fuse-related straps.

There is only one strap used in this IP. It is the input port `ftap_slvidcode`. Every instance of this IP in an SOC should have a unique 32 bit value strapped on this input.

Strap	Purpose
`ftap_slvidcode`	

2.6 Integrity Checks

Following are steps for running standalone integrity checks of this IP/IPSS. It is assumed that the environment variable `IP_ROOT` is set to the path of the IP collateral.

For detailed information about the bring-up flows, see the Flows section in the Verification Plan Reference (located in the 'doc' directory for this release).

The commands shown below are examples. Enter appropriate ones for your IP/IPSS.

1. Run the environment script:

   ```
   setenv IP_ROOT <release_path>
   cd $IP_ROOT/scripts
   run_env
   ```

2. Run Lintra:

   ```
   cd $IP_ROOT/scripts
   run_lintra
   ```

3. Compile the model:

   ```
   cd $IP_ROOT/tools/ace
   ace -cc
   ```

4. Run a simple test.

   To run basic test:

   ```
   cd $IP_ROOT/tools/ace
   ace -x
   ```

   To run the sample IP-level test "test1" (included in the release):

   ```
   ace -x -t test1
   ```

   To run the test interactively:

   ```
   ace -x -t by1_test -sd gui
   ```

5. Run Synthesis:

   ```
   cd $IP_ROOT/scripts
   run_syn
   ```

6. Run CDC:

   ```
   cd $IP_ROOT/scripts
   run_cdc
   ```

7. Run LEC:

   ```
   cd $IP_ROOT/scripts
   run_lec
   ```

8. Run Scan:

   ```
   cd $IP_ROOT/scripts
   run_scan
   ```

To run standalone integrity checks of this IP, follow the below steps:

1. Navigate to the directory where you downloaded the IP package.

2. Wash the unwanted groups and keep only the needed groups as below as HDK env sourcing will not happen if the shell has more than 16 groups.

   ```
   make set_wash
   ```

3. To source the SIP HDK env:

   ```
   source /p/hdk/rtl/hdk.rc -cfg sip
   ```

   This is the command if want to resource the env on the same shell.

   ```
       source /p/hdk/rtl/hdk.rc -cfg sip -reentrant
   ```

4. To run DOA/ To run basic test case the model with VCS:

   ```
   make run_vcs_test CUST=<CUST_NAME>
   Eg: make run_vcs_test CUST=ADL
   ```

5. To run simple tests.
   - To run a basic test:

     ```
     make run_vcs_test CUST=<CUST_NAME> TESTCASENAME=<testcasename>
     ```

   - To run the test interactively:

     ```
     make run_vcs_test_GUI CUST=<CUST_NAME> TESTCASENAME=<testcasename>
     ```

6. To run Lintra:

   This is HDK based approach. HDK commands with customer specific switch is used to run lintra. To run lintra, follow the below readme present.

   Follow the commands present in `README.txt`.

   ```
       #To run lintra compile
       make run_lint_comp CUST=<CUST_NAME>


       #To run lintra elab
       make run_lint_elab CUST=<CUST_NAME>

       ## To run Spyglass lintra  compile
       ## make run_sglint_comp CUST=ADL
       ## make run_sglint_comp CUST=ADP

       ## To run Spyglass lintra elab
       ## make run_sglint_elab CUST=ADL
       ## make run_sglint_elab CUST=ADP
   ```

7. To run DC and FEV after generation of RTL file list

   ```
   make run_lint_dc_fv CUST=<CUST_NAME>
   ```

8. To run CDC:

   ```
   #To run CDC for 1273
   make run_cdc_lint CUST=ADP
   #To run CDC for 1274
   make run_cdc_lint CUST=ADL

   ## To run Spyglass CDC  compile
   ## make run_sgcdc_comp CUST=ADL
   ## make run_sgcdc_comp CUST=ADP

   ## To run Spyglass cdc test
   ## make run_sgcdc_test CUST=ADL
   ```

```
## make run_sgcdc_test CUST=ADP
```

The following command is a workaround to set the CDC license pointer.

```
    source /nfs/site/eda/group/cse/setups/mentor/mentor.dft
```

9. To run Spyglass:

   Not applicable to this IP/IPSS

10. To run Scan:

    Not applicable to this IP/IPSS

11. To run EMULATION:

    ```
    make run_emulation CUST=<CUST_NAME>
    ```

12. To run collage:

    ```
    make run_collage CUST=<CUST_NAME>
    ```

13. To run CDC-LINT and open the GUI after running CDC:

    ```
    make run_cdc_lint CUST=<CUST_NAME>
    ```

14. To open GUI for CDC LINT:

    ```
    make run_cdc_lint_gui CUST=<CUST_NAME>
    ```

15. To run ZIRCON:

    ```
    source tools/zirconqa/run_zircon
    source tools/zirconqa/run_zircon_upload_ipdashboard
    source tools/zirconqa/run_zircon_upload_socdashboard
    ```

2.6.1 Use the following steps to run CHEETAH flow

1. Wash the unwanted groups and keep the groups as below

   ```
   Wash -n intelall soc siphdk cdftsip cdft n7 n7blr n7fe mp_tech_n7

   hdkenv
   ```

2. To compile the model in your area, do the following

   ```
   simbuild -dut stap -ace_args ace -ccud -elab_opts "-Xctdiag=cfgverbose" -
   vlog_opts "+define+INTEL_SVA_OFF" -ace_args- -1c -CUST <CUST_NAME> -1c-

   Eg: simbuild -dut stap -ace_args ace -ccud -elab_opts "-Xctdiag=cfgverbose"
   -vlog_opts "+define+INTEL_SVA_OFF" -ace_args- -1c -CUST MTPLP -1c-
   ```

3. To generate 2 stage rtl file list, do the following

   ```
   febe -s all +s v2k_prep +s flg_v2k -1c -CUST <CUST_NAME> -1c-

   Eg: febe -s all +s v2k_prep +s flg_v2k -1c -CUST MTPLP -1c-
   ```

4. To generate gen_collaterals

   ```
   febe -s all +s gen_collateral -1c -CUST <CUST_NAME> -1c-

   Eg: febe -s all +s gen_collateral -1c -CUST MTPLP -1c-
   ```

5. To source into CHEETAH environment

```
/p/hdk/bin/cth_eps -groups n7fe,cdft,cdftsip,siphdk,soc,n7,n7blr,mp_tech_n7
-ward_root
$MODEL_ROOT/target/<BLOCK_NAME>/<CUST_NAME>/*/aceroot/results/DC/<USER>.febe
/ -type private -ward_id febe -scope
sipcth,sipn6,a0,p00,2019.09,n7_tsmc_snps_h240_M13,scs

Eg: (It's a single command)
/p/hdk/bin/cth_eps -groups n7fe,cdft,cdftsip,siphdk,soc,n7,n7blr,mp_tech_n7
-ward_root $MODEL_ROOT/target/stap/
MTPLP/*/aceroot/results/DC/badithya.febe/ -type private -ward_id febe -scope
sipcth,sipn6,a0,p00,2019.09,n7_tsmc_snps_h240_M13,scs

pte_setup -wfs r2g_tools -stage all

cth_r2g populate -id <BLOCK_ID> -block <BLOCK_NAME> -init cig-block -
febe_output_path
$MODEL_ROOT/target/<BLOCK_NAME>/<CUST_NAME>/*/aceroot/results/DC/<USER>.febe
/fe  _collaterals/

Eg: cth_r2g populate -id febe -block stap -init cig-block - febe_output_path
$MODEL_ROOT/target/stap/MTPLP/*/aceroot/results/DC/badithya.febe/fe
_collaterals/
```

6. For running DC using command line

```
cth_r2g autorun -id <BLOCK_ID> -block <BLOCK_NAME> -run_flow dc_baseline

Eg: cth_r2g autorun -id febe -block stap -run_flow dc_baseline
```

7. For running FV using command line

```
cth_r2g autorun -id <BLOCK_ID> -block <BLOCK_NAME> -run_flow fev_r2syn

Eg: cth_r2g autorun -id febe -block stap -run_flow fev_r2syn
```

8. To run FISHTAIL

```
source tools/fishtail/<CUST_NAME>/ft_cth.csh

Eg: source tools/fishtail/MTPLP/ft_cth.csh
```

9. To run SAGE

```
cth_r2g autorun -id <BLOCK_ID> -block <BLOCK_NAME> -run_flow atpg_sage_syn

Eg: cth_r2g autorun -id febe -block stap -run_flow atpg_sage_syn
```

10. To run CALIBER

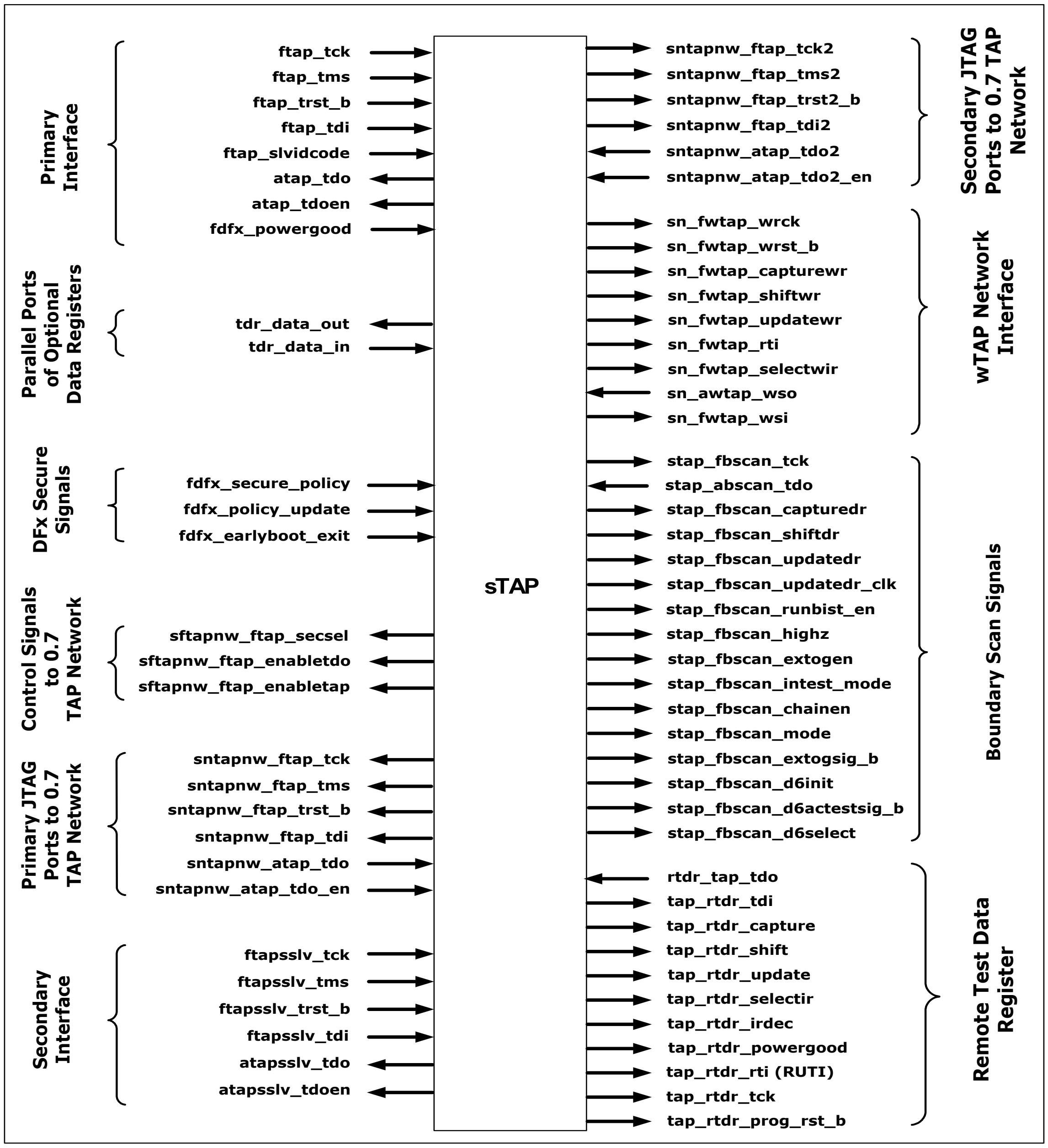
```
cth_r2g autorun -block <BLOCK_NAME> -id <BLCOK_ID> -run_flow
sta_syn_dmsa,syn_caliber

Eg: cth_r2g autorun -block stap -id febe -run_flow sta_syn_dmsa,syn_caliber
```

3 Overview

3.1 sTAP Block Diagram

Figure 1. sTAP Block Diagram



3.2 Top-level Signals / Functional Ports

Table 1. Top-Level Primary Interface Signals

Signal Name	Direction	Source/ Destination	Width
ftap_tck	Input	TAP Network	1
ftap_tms	Input	TAP Network	1
ftap_trst_b	Input	TAP Network	1
ftap_tdi	Input	TAP Network	1
ftap_slvidcode	Input	TAP Network	STAP_WIDTH_OF_SLVIDCODE
atap_tdo	Output	TAP Network	1
atap_tdo_en	Output	TAP Network	1
fdfx_powergood	Input	Clock and Reset	1

Table 2. Parallel Ports of Optional Data Registers

Signal Name	Direction	Source/ Destination	Width
tdr_data_out	Output	TAP Application	STAP_TOTAL_WIDTH_OF_TEST_DATA_REGISTERS
tdr_data_in	Input	TAP Application	STAP_TOTAL_WIDTH_OF_TEST_DATA_REGISTERS

Table 3. DFx Secure Signals

Signal Name	Direction	Source/ Destination	Width
fdfx_secure_policy	Input	Security Engine Fuse Bus	STAP_DFX_SECURE_WIDTH
fdfx_policy_update	Input	Security Engine	1
fdfx_earlyboot_exit	Input	Security Engine	1

Table 4. Control Signals to 0.7 TAP Network

Signal Name	Direction	Source/ Destination	Width
sftapnw_ftap_secsel	Output	0.7 TAP Network	STAP_NUMBER_OF_TAPS
sftapnw_ftap_enabletdo	Output	0.7 TAP Network	STAP_NUMBER_OF_TAPS
sftapnw_ftap_enabletap	Output	0.7 TAP Network	STAP_NUMBER_OF_TAPS

Table 5. Primary JTAG Ports to 0.7 TAP Network Signals

Signal Name	Direction	Source/ Destination	Width
sntapnw_ftap_tck	Output	0.7 TAP Network	1
sntapnw_ftap_tms	Output	0.7 TAP Network	1
sntapnw_ftap_trst_b	Output	0.7 TAP Network	1
sntapnw_ftap_tdi	Output	0.7 TAP Network	1
sntapnw_atap_tdo	Input	0.7 TAP Network	1
sntapnw_atap_tdo_en	Input	0.7 TAP Network	STAP_NUMBER_OF_TAPS

Table 6. Secondary Interface Signals

Signal Name	Direction	Source/ Destination	Width
ftapsslv_tck	Input	TAP Network	1
ftapsslv_tms	Input	TAP Network	1
ftapsslv_trst_b	Input	TAP Network	1
ftapsslv_tdi	Input	TAP Network	1
atapsslv_tdo	Output	TAP Network	1
atapsslv_tdoen	Output	TAP Network	1

Table 7. Secondary JTAG Ports to 0.7 TAP Network Signals

Signal Name	Direction	Source/ Destination	Width
sntapnw_ftap_tck2	Output	0.7 TAP Network	1
sntapnw_ftap_tms2	Output	0.7 TAP Network	1
sntapnw_ftap_trst2_b	Output	0.7 TAP Network	1
sntapnw_ftap_tdi2	Output	0.7 TAP Network	1
sntapnw_atap_tdo2	Input	0.7 TAP Network	1
sntapnw_atap_tdo2_en	Input	0.7 TAP Network	STAP_NUMBER_OF_TAPS

Table 8. wTAP Network Interface Signals

Signal Name	Direction	Source/ Destination	Width
sn_fwtap_wrck	Output	wTAP Network	1
sn_fwtap_wrst_b	Output	wTAP Network	1
sn_fwtap_capturewr	Output	wTAP Network	1
sn_fwtap_shiftwr	Output	wTAP Network	1
sn_fwtap_updatewr	Output	wTAP Network	1
sn_fwtap_rti	Output	wTAP Network	1
sn_fwtap_selectwir	Output	wTAP Network	1
sn_awtap_wso	Input	wTAP Network	STAP_NUMBER_OF_WTAPS
sn_fwtap_wsi	Output	wTAP Network	STAP_NUMBER_OF_WTAPS

Table 9. Boundary Scan Signals

Signal Name	Direction	Source/ Destination	Width
stap_fbscan_tck	Output	BScan Cell	1
stap_abscan_tdo	Input	BScan Cell	1
stap_fbscan_capturedr	Output	BScan Cell	1
stap_fbscan_shiftdr	Output	BScan Cell	1
stap_fbscan_updatedr	Output	BScan Cell	1
stap_fbscan_updatedr_clk	Output	BScan Cell	1

Signal Name	Direction	Source/ Destination	Width
stap_fbscan_runbist_en	Output	BScan Cell	1
stap_fbscan_highz	Output	BScan Cell	1
stap_fbscan_extogen	Output	BScan Cell	1
stap_fbscan_intest_mode	Output	BScan Cell	1
stap_fbscan_chainen	Output	BScan Cell	1
stap_fbscan_mode	Output	BScan Cell	1
stap_fbscan_extogsig_b	Output	BScan Cell	1
stap_fbscan_d6init	Output	BScan Cell	1
stap_fbscan_d6actestsig_b	Output	BScan Cell	1
stap_fbscan_d6select	Output	BScan Cell	1

Table 10. Remote Test Data Register Signals

Signal Name	Direction	Source/ Destination	Width
rtdr_tap_tdo	Input	Remote TDR	1 (when STAP_NUMBER_OF_REMOTE_TEST_DATA_REGISTERS = 0). STAP_NUMBER_OF_REMOTE_TEST_DATA_REGISTERS (when STAP_RTRD_IS_BUSSED = 1).
tap_rtdr_tdi	Output	Remote TDR	1 (when STAP_RTRD_IS_BUSSED = 0). STAP_NUMBER_OF_REMOTE_TEST_DATA_REGISTERS (when STAP_RTRD_IS_BUSSED = 1).
tap_rtdr_capture	Output	Remote TDR	1 (when STAP_RTRD_IS_BUSSED = 0). STAP_NUMBER_OF_REMOTE_TEST_DATA_REGISTERS (when STAP_RTRD_IS_BUSSED = 1).
tap_rtdr_shift	Output	Remote TDR	1 (when STAP_RTRD_IS_BUSSED = 0). STAP_NUMBER_OF_REMOTE_TEST_DATA_REGISTERS (when STAP_RTRD_IS_BUSSED = 1).
tap_rtdr_update	Output	Remote TDR	1 (when STAP_RTRD_IS_BUSSED = 0). STAP_NUMBER_OF_REMOTE_TEST_DATA_REGISTERS (when STAP_RTRD_IS_BUSSED = 1).
tap_rtdr_selectir	Output	Remote TDR	1
tap_rtdr_irdec	Output	Remote TDR	STAP_NUMBER_OF_REMOTE_TEST_DATA_REGISTERS
tap_rtdr_powergood	Output	Remote TDR	1
tap_rtdr_rti	Output	Remote TDR	1
tap_rtdr_tck	Output	Remote TDR	1
tap_rtdr_prog_rst_b	Output	Remote TDR	STAP_NUMBER_OF_REMOTE_TEST_DATA_REGISTERS

Table 11. 3.2.11 FSM Signals

Signal Name	Direction	Source/ Destination	Width
stap_fsm_tlrs	Output	FSM Signal	1

Table 12. 3.2.12 Isolation Enable Signal

Signal Name	Direction	Source/ Destination	Width
stap_isol_en_b	Input	From the SoC/PHY	1

Table 13. 3.2.13 Powerdomain Reset Signal

Signal Name	Direction	Source/ Destination	Width
ftap_pwrdomain_rst_b	Input	From the SoC/PHY	1

3.2.1 Signal List File

See the signal list file in `tools/collage/10nm/reports/stap.build.summary`

3.2.2 Tie-offs, Opens, and Gates

Not applicable tot his IP/IPSS; there are no tie-offs.

3.2.3 "Ad Hoc" Pins, Non-standard, or Non-compliant

Refer to section x.x, Signal List.

3.2.4 Control Signals

Refer to section x.x, Signal List.

3.2.5 Reset Ports

Refer to section x.x, Signal List.

3.2.6 Naming Scheme

Refer to section x.x, Signal List.

3.2.7 End Points

Refer to section section x.x, Signal List.

3.3 Forces

Not applicable to this IP/IPSS; there are no forces in this IP.

4 Tools and Methodology for Integration

4.1 Configuration Updates

Not applicable to this IP/IPSS; there are no Tool overrides in this IP.

4.2 Supported Tools

The following tools are used in the integration of this IP/IPSS. For versions supported by each release, see Release Notes in the "doc" directory of the release package.

- VCSMX - J-2014.12-SP1
- OVM
- ACE
- Saola
- Lintra
- Design Compiler
- Conformal
- Designware
- Debussy/Verdi
- Spyglass
- VGGNU
- Nebulon
- CDC - QuestaCDC

4.3 Environment Variables

Set the following environment variables as listed.

To set the environment variables, run the following commands from your release directory:

```
cd <release_directory>
setenv JTAG_BFM_VER <shared JTAG BFM dir>/CHASSIS_JTAGBFM_2015WW12_R2.3

setenv MODEL_ROOT $cwd
```

To compile the model in your area:

```
make run_vcs CUST=<CUST_NAME>
```

To run the default bypass test:

```
make run_vcs_test CUST=<CUST_NAME>
```

Table 14. Environment Variables

Variable	Description

4.4 HIP Libraries Included in Release

4.4.1 HIP IPToolData.pm Change

Not appicable to this IP/IPSS

4.4.2 Register Files or SRAM

Not appicable to this IP/IPSS

4.4.3 M-PHY and Related Libraries

Not appicable to this IP/IPSS

4.5 Directory Structure for Tools

Following is a high-level view of the directory structure.

```
|-- ace
|    |-- emulation
|    |-- lib
|-- doc
|-- pwa
|    `-- results
|-- scripts
|-- source
|    `-- rtl
|-- subIP
|    |-- SIP_DFx_DFxSecurePlugin_2015WW24_R1.0_v4
|    `-- DfxSecurePlugin -> SIP_DFx_DFxSecurePlugin_2015WW24_R1.0_v4
|-- tools
|    |-- cdc
|    |-- collage
|    |-- emulation
|    |-- emulint
|    |-- fpv
|    |-- lint
|    |-- ptpx
|    |-- report
|    |-- reports
|    |-- spyglass
|    |-- syn
|    |-- xprop
|    `-- zirconqa
|-- verif
|    |-- lib
|    |-- tb
|    `-- tests
`---- wvrin
 |-- log
 |-- viol
  `-- wvr
```

4.6 Ace

Paths to acerc: `$IP_ROOT/ace/stap.acerc`

Location of udf file: `$IP_ROOT/ace/stap_hdl.udf`

Model to use when importing libraries: stap

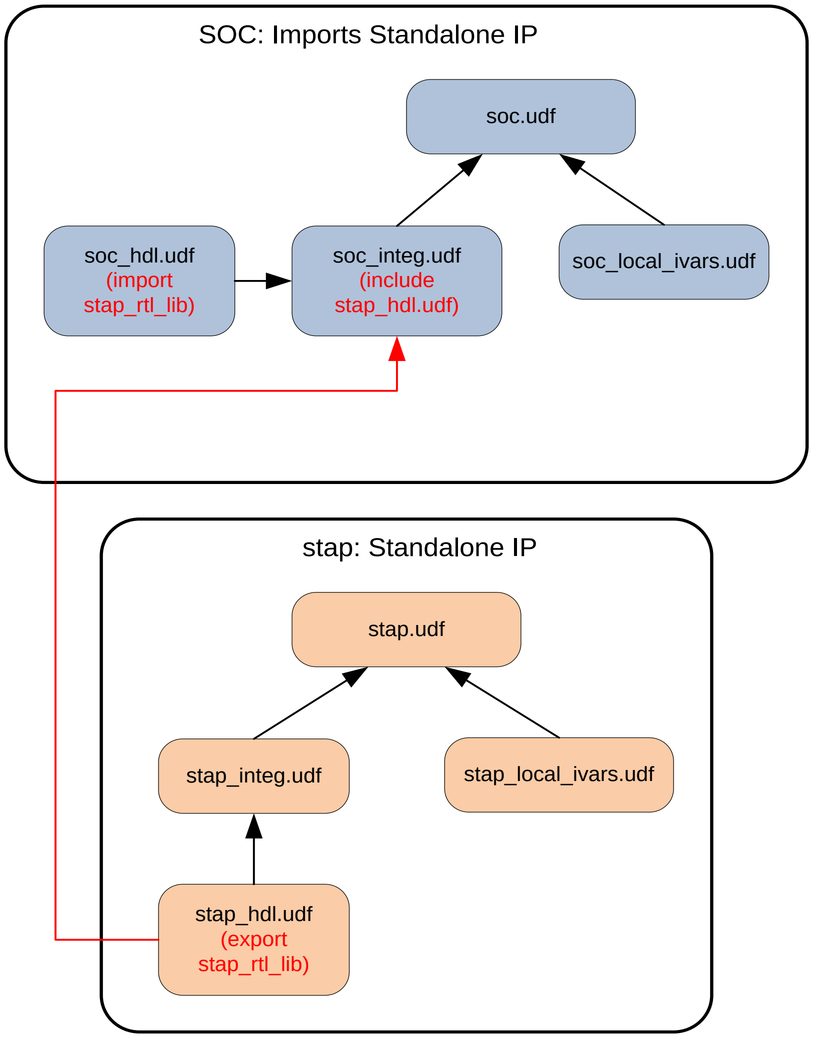
Elaboration options not exported: N/A

Required content in sip_shared_libs:

```
$hdl_spec = {
    '-vlog_files' => [
                   'verif/tb/emulate_stap_tdo.sv',
                   'verif/tb/stap_rtdr_ref.sv',
                   'verif/tb/STapTop.sv',
    ],
    '-vlog_lib_dirs' => [
        $ENV{'UVM_HOME'}.'/src/',$ENV{'OVM_HOME'}.'/src/',$ENV{'XVM_HOME'}.'/src/',
    ],
    '-vlog_lib_files' => [
    ],
    '-vlog_incdirs' => [
        'verif',
        'verif/tb',
        'verif/tests',
        'verif/tb/include',
        'source/rtl/include',
        'verif/tb',
        $ENV{'ACE_HOME'}.'/lib/Verilog',
        $ENV{'OVM_HOME'}.'/src/',
    ],
    '-vlog_opts' => [
        '+define+VCS ',
        '+libext+.v+.sv -sverilog',
        '-ntb_opts pcs -timescale=1ps/1ps'
    ],
    '-vhdl_files' => [
    ],
    '-vcom_opts' => '-xlrm ',
    '-hdl_spec' => [
    ],
};
```

Figure 2 illustrates the ACE integration.

Figure 2. Ace Integration in an SoC



In order to run ACE, the SoC team must import the `stap.udf` in their ACE setup. The steps are as follows:

1. Place the current IP that you downloaded from IRR in the `subIP` directory of the project in which it will be used.
2. In the `soc.acerc` file of the SoC, define stap as a subscope.
3. In the `soc.acerc` file of the SoC, add the directory where you place the IP to the search path.
4. In the `soc_integ.udf` file, add `stap_hdl.udf` as an included `udf`.
5. In the `soc_hdl.udf` file, import the `stap_rtl_lib`. Declare these as relevant dependent libs in the higher-level RTL libraries where this module will be instantiated.
6. Run the `ace -show_udfs` command to see if the `stap_hdl.udf` shows up.

7. Run the `ace -show_models` command to see if the model of the SoC is clearly shown as the current scope.

4.7 Lintra

Lintra Version: 19.1p6_shOpt64

Lintra location: `$IP_ROOT/tools/lint/`

Location of waiver files: `$IP_ROOT/tools/lint/ADL/sipstap_waiver_w.xml`

Location of Lintra patches & configuration: `verif/tests/lintra/stap/stap.opt`

Location of Lintra report file for warnings and errors: `tools/lint/ADL/stap_violations.xml`

This is HDK based approach. HDK commands with customer specific switch is used to run lintra. To run lintra, follow the below readme present.

- To run lintra compile:

```
make run_lint_comp CUST=<CUST_NAME>
```

4.8 To run lintra elab: Synthesis

To run synthesis and LEC. In HDK flow, FEBE environment is used to run syn and LEC.

```
make run_lint_dc_fv CUST=<CUST_NAME>
```

4.8.1 Clocks

Table 15. Primary Clocks

Sl. No.	Clock Name	Clock Period	Clock Waveform	Clock Source
1	ftap_tck	10000	{0 5000}	{ftap_tck}

Table 16. Generated Clocks

Sl. No.	Clock	Master Clock	Master Clock Source	Edges	Source
1	sntapnw_ftap_tck	ftap_tck	ftap_tck	Positive Edge	Positive Edge
2	sntapnw_ftap_tck2	ftap_tck	ftap_tck	Positive Edge	Positive Edge

4.8.2 Clock Diagram

Not applicable to this IP/IPSS; this is a single clock domain IP.

4.8.3 Constraint Files

Files are present in the `tools/syn/<project_name>/stap/inputs` folder.

4.8.4 Synthesis Best Known Methods (BKMs)

Not applicable to this IP/IPSS

4.8.5 Scan Insertion

Not applicable to this IP/IPSS

4.8.6 Latches

There are total four latches. IP has latches from Dfx SecurePlugin.

4.9 Formal Verification

The verification of assertions is covered through directed and random tests.

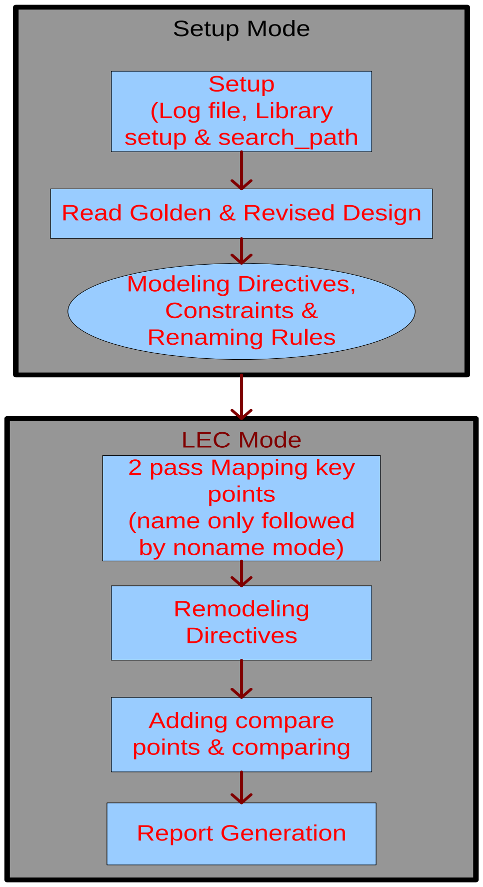
To verify the assertions, you can use the following command to add various ifdefs and run the FPV (Formal Property Verification) tool:

```
source $IP_ROOT/tools/fpv/run_inspect
```

Formal verification is done using Conformal-LEC from Cadence. A high-level conformal flow is illustrated in Figure 3.

The reports are present at the `tools/fev` directory depending on which customer is addressed.

Figure 3. LEC Flow Diagram


Setup Mode
Setup (Log file, Library setup & search_path
Read Golden & Revised Design
Modeling Directives, Constraints & Renaming Rules
LEC Mode
2 pass Mapping key points (name only followed by noname mode)
Remodeling Directives
Adding compare points & comparing
Report Generation

4.10 CDC

```
   make run_cdc_ADL
   make run_cdc_ADP

  ## To run Spyglass CDC  compile
  ## make run_sgcdc_comp CUST=ADL
  ## make run_sgcdc_comp CUST=ADP

  ## To run Spyglass cdc test
  ## make run_sgcdc_test CUST=ADL
  ## make run_sgcdc_test CUST=ADP
```

The following command is a workaround to set the CDC license pointer:

```
source /nfs/site/eda/group/cse/setups/mentor/mentor.dft
```

4.11 Scan

Not applicable to this IP/IPSS

4.12 DFD Triggers and Actions

Not applicable to this IP/IPSS

4.13 Safety Verification

Not applicable to this IP/IPSS

5 Design Information for SoC Integration

This chapter is primarily targeted to the integration team responsible for integrating this IP/IPSS into an SoC. For subsystems, it describes the main SS and co-IPs.

5.1 RTL Directory Structure

```
source
`-- rtl
    |-- ctech_lib
    |-- include
    |-- stap
    |-- stap_rtl.hdl
    |-- stap_rtl.hdl_gls
    |-- stap_rtl.hdl_gls_1274
    `-- stap_rtl.hdl_rtl
subIP
|-- DFx_Secure_Plugin_PIC3_2016WW27_RTL1P0_V1
|   |-- ace
|   |-- doc
|   |-- source
|   `-- verif
`-- DfxSecurePlugin -> DFx_Secure_Plugin_PIC3_2016WW27_RTL1P0_V1/
```

5.2 Building Blocks for SoC Integration

Not applicable to this IP/IPSS

5.3 Embedded Building Blocks/Custom Logic

Not applicable to this IP/IPSS

5.4 Macros used by IP/IPSS

Not applicable to this IP/IPSS

5.5 RTL Configuration Parameters and Overrides

In order to use the sTAP IP, you must generate the RTL and Testbench configuration parameters files.Script-based Parameter Generation

To generate the default configuration parameters for use with the RTL and Testbench designs, use the provided Perl script. To run the script:

```
cd $IP_ROOT/scripts
perl parameters.pl
```

The script creates four parameter files with a user-specified name pre-appended to the file name. For example, if you specify "Pentium" then the following files are generated.

- `stap_params_include.inc`
- `tb_param.inc`
- `Pentium_sTAP_soc_param_values.inc`
- `Pentium_sTAP_soc_param_overide.inc`

- `Pentium_stap_instance.sv`

The first two files are for the running the IP in standalone mode with the Env that is supplied with the release. The third and fourth files are the RTL parameter files to be used by the SoC or IP team members instantiating the sTAP in their designs.

The fifth file, `Pentium_stap_instance.sv`, includes the third and fourth files (`Pentium_sTAP_soc_param_values.inc` and `Pentium_sTAP_soc_param_overide.inc`), and generates the sTAP module interface while instantiating the sTAP. You can use this file “as is” to instantiate the sTAP module in your SoC.

Before running in standalone mode, manually copy the generated RTL and Testbench parameter files to the appropriate directories:

- Copy the RTL parameter file as follows:

  ```
  cp $IP_ROOT/scripts/stap_params_include.inc \
  $IP_ROOT/source/rtl/include/stap_params_include.inc
  ```

- Copy the Testbench parameter file as follows:

  ```
  cp $IP_ROOT/scripts/tb_param.inc \
  $IP_ROOT/verif/tb/include/tb_param.inc
  ```

- Copy the appropriate `*_soc_*.inc` files to your project-specific `include` directory.

Next, edit the user-definable parameters in the RTL parameter file:

Open the RTL parameter file (in this example, `stap_params_include.inc`) in a text editor and change the user-defined parameters (as indicated by bold font in the parameter tables in section 5.5.2) as necessary. Do not change the derived parameters (parameter values not in bold).Tool-based Parameter Generation

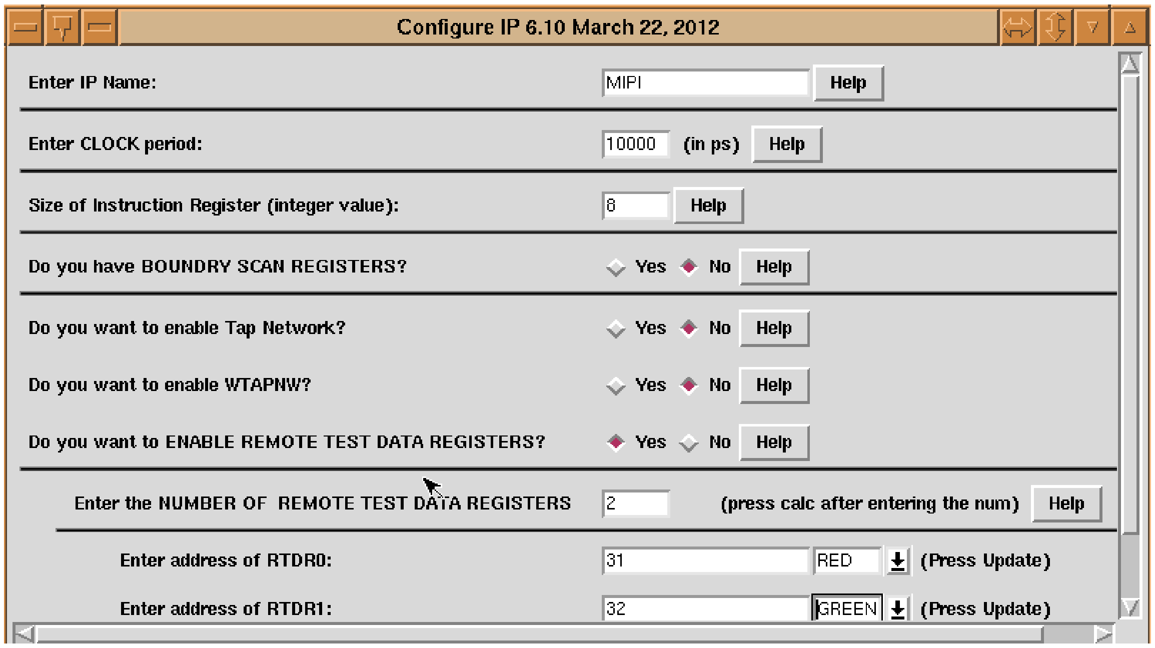
Instead of manually editing the RTL parameter file, you can use the Parameter Generation Tool to change the user-defined RTL parameter (see the parameter tables in section 5.5.2). When you start the Parameter Generation tool, it generates a SystemVerilog parameter override file for instantiating the IP.

Note: The Synopsys simulator (VCS) requires that if any configuration parameter is to be overridden, then all configuration parameters must be overridden.

To start the Parameter Generation tool (Figure 4) and generate the RTL parameters files, enter the following commands:

```
cd $IP_ROOT/scripts/dfx_parameter_gui/
/p/com/eda/intel/siputils/prod/bin/configIP  stap.config  -odf  input_file.inc
```

Figure 4. Parameter Generation Tool



Configure IP 6.10 March 22, 2012

Enter IP Name: MIPI

Enter CLOCK period: 10000 (in ps)

Size of Instruction Register (integer value): 8

Do you have BOUNDRY SCAN REGISTERS? Yes [✓ No]

Do you want to enable Tap Network? Yes [✓ No]

Do you want to enable WTAPNW? Yes [✓ No]

Do you want to ENABLE REMOTE TEST DATA REGISTERS? [✓ Yes] No

Enter the NUMBER OF REMOTE TEST DATA REGISTERS 2 (press calc after entering the num)

Enter address of RTDR0: 31 RED (Press Update)

Enter address of RTDR1: 32 GREEN (Press Update)

The -odf stands for output data file. The argument next to -odf is the name of a file where you want to save the generated configuration. You can then use this file with an argument -if (input file) to reload the same configuration in the tool. The Parameter Generation Tool generates three parameter files with a user-supplied name prepended to the file names. For example, if you specify "MIPI", the following files are generated after you click the Save button:

- stap_params_include.inc
- MIPI_sTAP_soc_param_values.inc
- MIPI_sTAP_soc_param_overide.inc

The first file is for running the sTAP in standalone mode with the supplied IP Env. The next two are the parameter files to be used by the SoC or IP team members to instantiate the sTAP.

If you enter an inappropriate value in a field (for example, a value that is out of range or a special character) the Parameter Generation tool will display a message in the console log window that provides guidance on the valid values by giving warning or error messages. Once you have entered all the parameters, click the Calc button (in the row of buttons at the bottom of the window). The Parameter Generation tool performs a complete validation check on all parameter values. It displays the calculated values in the console log window. If you want to update the window and check for warnings, click the Update button.

After you have finished entering or updating all of the values, you can click the Calc button again to allow the tool to process the new configuration (this is optional). If no warning or error messages appear in the console log window (the white text box just above the line of buttons at the bottom of the window), click the Save button.

After saving the file, if you want to edit some of the parameters, you can upload it again. You must upload the `input_file.inc` file, not the other three. In the example case, the file you would want to upload is `MIPI_sTAP_rtl.inc`. To upload the file, use the following command:

```
/p/com/eda/intel/siputils/prod/bin/configIP stap.config \
   -if input_file.inc
```

Note: It is advisable to prefix the `input_file.inc` with the IP name. For example, if you are generating this for MIPI, you can call the input file `MIPI_input_file.inc`. If you then want to generate parameters for another IP, say USB2, in the same directory, you can call the other file `USB2_input_file.inc`. In this way, you will not accidentally overwrite one of your `input_file.inc`

Once your file has loaded, you can edit the parameter values. When you have finished editing the values, you can click the Calc button to allow the Parameter Generation Tool to perform a validation check. Once you are satisfied with your changes click the Save button. The tool will automatically generate and save all three parameter files. If you do not want to save your changes, click the Exit button to exit the tool.

In order to retain saved changes and load from an existing file, type the following command.

```
/p/com/eda/intel/siputils/prod/bin/configIP stap.config \
  -odf MIPI_input_file.inc -if MIPI_input_file.inc
```

Note: If you do not click the Save button before exiting the tool, none of the three parameter files will be generated or saved.

To display a PDF file with detailed information on how to use the Parameter Generation Tool, click the Help button.

Parameter Name	**Range**	**Default**	**Description (Including Interdependencies)**
STAP_NUMBER_OF_BITS_FOR_SLICE	16	16	Used for internal computation (as 16 bit field representation is used in compound parameters). Do not change this parameter.
STAP_SIZE_OF_EACH_INSTRUCTION	8 – 32	16	Specifies the size of the instruction register.

5.5.1 Mandatory Parameters

Not applicable to this IP/IPSS

5.5.2 Test Data Register Parameters

Parameter	**Range**	**Default**	**Description (Including Interdependencies)**
STAP_NUMBER_OF_TEST_DATA _REGISTERS	0 – N	2	Specifies the number of user-defined optional TDRs that are present in the sTAP.

Parameter	Range	Default	Description (Including Interdependencies)
STAP_ENABLE_ITDR_PROG_RST	0 - 1	0	Specifies the programmable reset option for ITDRs. If the value of this parameter is set to 1, the TAPC_ITDRRSTSEL and TAPC_TDRRSTEN registers comes into existence. You must specify which register has the programmable reset option by writing 1 to the corresponding bit position for each of the ITDRs in the TAPC_ITDRRSTSEL register. The applicable programmable reset is determined by the value programmed in TAPC_TDRRSTEN register.
STAP_TOTAL_WIDTH_OF_TEST _DATA_REGISTERS	0 – N	64	Specifies the combined total widths of all the user-defined optional TDRs that are present in the sTAP.
STAP_NUMBER_OF_TOTAL _REGISTERS	2 – N (N is a positiv e integer)	18	Specifies the maximum number of registers possible for the sTAP.
STAP_INSTRUCTION_FOR_DATA _REGISTERS	0 - N	User-defined opcodes: { {8'h0C, STAP_SECURE_GRE EN,}{8'hFF, STAP_SECURE_GRE EN} }	Holds an array containing user-defined opcodes with associated security level for each opcodes.
STAP_SIZE_OF_EACH_TEST _DATA_REGISTER	0 - N	{ 16'h0 }	Holds an array containing the user defined width for each TDR. It will contain as many entries as the value of STAP_NUMBER_OF_ TEST_DATA_REGISTERS.
STAP_MSB_VALUES_OF_TEST _DATA_REGISTERS	0 - N	{ 16'h0 }	Holds an array containing the MSB value of each TDR. It will contain as many entries as the value of STAP_NUMBER_OF _TEST_DATA_REGISTERS. If this parameter is 0 then the port width will be 1 bit wide.
STAP_LSB_VALUES_OF_TEST _DATA_REGISTERS	0 - N	{ 16'h0 }	Holds an array containing the LSB value of each TDR. It will contain as many entries as the value of STAP_NUMBER _OF_TEST_DATA_REGISTERS. If this parameter is 0 then the port width will be 1 bit wide.

Parameter	Range	Default	Description (Including Interdependencies)
STAP_RESET_VALUES_OF_TEST_DATA_REGISTERS	0 - N	{ 1’b0 }	Holds an array containing the reset values of each TDR. This parameter is a per-bit value as wide as STAP_TOTAL_WIDTH_OF_TEST_DATA_REGISTERS. If this parameter is 0 then the port width will be 1 bit wide. This parameter is used to specify the reset state of every bit of each TDR.
STAP_SWCOMP_ACTIVE	0-1	0	0 : SWCOMP Module is absent 1: SWCOMP Module is present
STAP_SWCOMP_NUM_OF_COMPARE_BITS	8-32	10	This parameter specifies the number of compare bits present in the SWCOMP compare window
STAP_WTAPCTRL_RESET_VALUE	0-1	0	This parameter sets the default value for the STAP_WTAP_CTL register. It is based on the number of WTAPs. This value is only pratical for one WTAP controlled by a sTAP.
STAP_SUPPRESS_UPDATE_CAPTURE	0-1	0	Parameter enables SUPPRESS_UPDATE_CAPTURE register.

5.6 Integration Dependencies

Not applicable to this IP/IPSS

5.6.1 “Ad Hoc” Pins

Name	I/O	Functionality	Deassert
fdfx_powergood	in	See HAS document for description	See HAS document for description
Ftap_slvidcode	in	See HAS document for description	See HAS document for description
Stap_abscan_tdo	in	See HAS document for description	See HAS document for description
tdr_data_in	in	See HAS document for description	See HAS document for description
tdr_data_out	out	See HAS document for description	See HAS document for description
sntapnw_atap_tdo_en	in	See HAS document for description	See HAS document for description
ftap_pwrgood_rst_b	in	See HAS document for description	See HAS document for description
stap_isol_en_b	in	See HAS document for description	See HAS document for description

5.6.2 Validation Requirements

Not applicable to this IP/IPSS

5.6.3 Interface Signals Implemented for Security

Not applicable to this IP/IPSS

5.7 RTL Uniquification

DFx Secure Plugin is uniqified at the IP-level.

5.8 Registers

No RDL is provided for the IP. Customers need to generate it using CTT

Relative Path	Fuse RDL Filename	Description
`/target/<ss>/<ss_nebulon_lib>/nebulon/output/`	`<ss-regs>.svh`	SS RDL file for <x>

5.9 Fuses

Not applicable to this IP/IPSS

5.9.1 Fuse RDL Files

Table 17. RDL list:

Relative Path	Fuse RDL Filename	Description
`/target/mspmas_fuseagg_lib/socfusegen/Pullers/rdl/`	`fusegen_mspmas0_scf_<ss>.rdl`	SS aggregated IP RDL file
`/tools/srd/fuse_rdl/`	`m2mem_fuses.rdl`	M2M fuses from SS
`/tools/srd/fuse_rdl/`	`ss_mem_ddr_mee_fuse.rdl`	Agent logic fuses from SS

5.10 Clock, Power, and Reset Domains

- Slave TAP operates the state machine on a single clock (`ftap_tck`). The frequency can vary from 1 to 100 MHz.

The Secondary clock (`ftapsslv_tck`) is a pass through.

Following are the three resets in the design.

- `fdfx_powergood`
- `ftap_trst_`b
- `ftapsslv_trst_b`
- `ftap_pwrdomain_rst_b`

  There is no minimum assertion time for these resets. All private data registers are cleared by fdfx_powergood. Asserting TMfsffS high for five cycles resets the TAP state machine as described in the JTAG protocol.

5.10.1 Clock Domain Diagram

Not applicable to this IP/IPSS

5.10.2 Clock Requirements (HIP only)

HIP clock requirements are described in the file `clock_req.xlsx` that is published in the `doc` directory of the release.

The sTAP was tested at 100 MHz. The sTAP has the following two clock domains:

- `ftap_tck`
- `ftapsslv_tck`

The primary and secondary clocks are two asynchronous clock domains. The logic that runs on each domain is independent from the logic that runs on the other domain. There is no clock crossing between the two domains.

All remote TDRs work on the `ftap_tck` domain.

The associated ports that run on TCK2 (`ftapsslv_tck`) are:

- `ftapsslv_tck`
- `ftapsslv_tms`
- `ftapsslv_trst_b`
- `ftapsslv_tdi`
- `atapsslv_tdo`
- `atapsslv_tdoen`
- `sntapnw_ftap_tck2`
- `sntapnw_ftap_tms2`
- `sntapnw_ftap_trst2_b`
- `sntapnw_ftap_tdi2`
- `sntapnw_atap_tdo2`

`sntapnw_atap_tdo2_en`

5.10.3 Power Domains

Not applicable to this IP/IPSS

5.11 Power Information

5.11.1 Power Supply

Not applicable to this IP/IPSS

5.11.2 Static Clock Gating

Not applicable to this IP/IPSS

5.11.3 Power Gating

Not applicable to this IP/IPSS

5.11.4 Bumps and their Power Domains

Not applicable to this IP/IPSS

5.11.5 Voltage Rail Requirements (HIP only)

Voltage rail requirements are described in the file `voltage_rail_req.xlsx` that is published in the `doc` directory of the release.

5.12 System Startup

5.12.1 Power-up Requirements

Not applicable to this IP/IPSS

5.12.2 Power-up Sequence

The BFM provides tasks that let you create a custom power sequence by controlling the drive on `fdfx_powergood` and `ftap_trst_b`

5.12.3 Initialization Sequence

Not applicable to this IP/IPSS

5.12.4 Device Configuration

Not applicable to this IP/IPSS

5.12.5 Header for Windows Boot

Not applicable to this IP/IPSS

5.13 DFx Considerations

Not applicable to this IP/IPSS

5.13.1 DFx Top-Level Signals

Not applicable to this IP/IPSS

5.13.2 DFx Clock Definition

Not applicable to this IP/IPSS

5.13.3 Clock Crossings

Not applicable to this IP/IPSS

5.13.4 VISA

Not applicable to this IP/IPSS

5.13.5 DFD Triggers and Actions

Not applicable to this IP/IPSS

5.13.6 Debug Registers

Not applicable to this IP/IPSS

5.13.7 Scan – Clock Gating in RTL

Not applicable to this IP/IPSS

5.13.8 Scan – Reset Override

Not applicable to this IP/IPSS

5.13.9 Scan – Constraints and Coverage

Not applicable to this IP/IPSS

5.13.10 TAP and Associated Registers

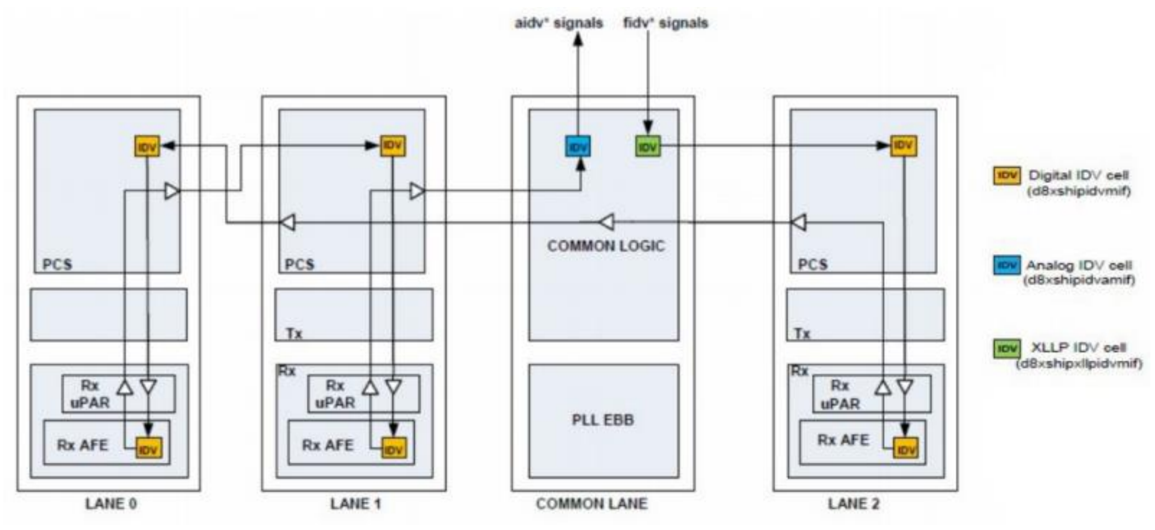
Not applicable to this IP/IPSS

5.13.11 Boundary Scan Parameters

Not applicable to this IP/IPSS

5.13.12 Intra-Die Variation (IDV) - for HIP only

Figure 5. IDV Circulation Example


aidv* signals
fidv* signals
IDV
PCS
Tx
Rx
Rx uPAR
Rx AFE
COMMON LOGIC
PLL EBB
LANE 0
LANE 1
COMMON LANE
LANE 2
Digital IDV cell (d8xshipidvmif)
Analog IDV cell (d8xshipidvamif)
XLLP IDV cell (d8xshipxllpidvmif)

List the lay coordinates of the cells.

Table 18. IDV Coordinates (example)

Order	IDV Type	X (um)	Y (um)	Lane Instance	Related Supply
0	d8xshipidvxlllpmif	632.520	214.662	com0	derived supply: vcca_1p05
1	d8xshipidvmif	664.440	69.426	lane0	derived supply: vcca_1p05
2	d8xshipidvmif	0.840	69.426	lane0	derived supply: vcca_1p05
3	d8xshipidvmif	664.440	176.358	lane1	derived supply: vcca_1p05
4	d8xshipidvmif	0.840	176.358	lane1	derived supply: vcca_1p05
5	d8xshipidvamif	639.240	249.774	com0	derived supply: vcca_1p05

Show the rough location of the IDV cells.

5.14 Security

See the Security Development Lifecycle site for information on security-related design practices:

https://sp2010.amr.ith.intel.com/sites/sdl/SitePages/Home.aspx

5.14.1 SAI Width

Not applicable to this IP/IPSS

5.14.2 Validation Requirements

Not applicable to this IP/IPSS

5.14.3 Interface Signals Implemented for Security

Not applicable to this IP/IPSS

5.15 Safety Support

See the Safety Development Lifecycle site for information on IP HW/SW design practices:

http://goto.intel.com/FuSaLifecycle

or

https://sharepoint.amr.ith.intel.com/sites/FUSA_GlobalDomain/FuSaPublic/Published%20FuSa%20Lifecycle/Forms/AllItems.aspx

5.15.1 IP Safety HW Compliance

Not applicable to this IP/IPSS

5.15.2 IP Safety SW Compliance

Not applicable to this IP/IPSS

5.15.3 Safety HW Validation Requirement

Not applicable to this IP/IPSS

5.15.4 Safety SW Validation Requirement

Not applicable to this IP/IPSS

5.16 Emulation Support

Emulation support is added and available at tools/emulation.

Emulation is enabled based on ace flow.

```
Make run_emulation CUST=<CUST_NAME>
```

5.17 Other Design Considerations

5.17.1 rTDR Connections

Figure 6 and Figure 7 illustrate the connections between an sTAP and rTDR module.

The STAP_RTDR_IS_BUSSED parameter specifies whether or not the control and data signals to the user-defined Remote Test Data Register-related pins are bussed.

Figure 6 illustrates a non-bussed connection (when STAP_RTDR_IS_BUSSED is set to 0). In this example, the control and data signals are 1 bit wide. RTDR0 is an instance of `stap_rtdr_ref.sv` and RTDR1 is a user-created rTDR module. In RTDR1, you should AND the control signals (`tap_rtdr_shift[0]`, `tap_rtdr_capture[0]`, and `tap_rtdr_update[0]`) with `tap_rtdr_irdec[N]` to pass it over to Shift and Shadow Registers. The widths and names of the signals in Figure 6 are:

- `rtdr_tap_tdo[N]`
- `tap_rtdr_tdi[0]`
- `tap_rtdr_capture[0]`
- `tap_rtdr_shift[0]`
- `tap_rtdr_update[0]`
- `tap_rtdr_irdec[N]`
- `tap_rtdr_powergood`
- `tap_rtdr_selectir`
- `tap_rtdr_tck`
- `tap_rtdr_rti (RUTI)`
- `tap_rtdr_prog_rst_b[0]`

Figure 7 illustrates a bussed connection (when STAP_RTDR_IS_BUSSED is 1). Here the control and data signals are N bits wide. The widths and names of the signals in Figure 7 are:

- `rtdr_tap_tdo[N]`
- `tap_rtdr_tdi[N]`
- `tap_rtdr_capture[N]`
- `tap_rtdr_shift[N]`
- `tap_rtdr_update[N]`
- `tap_rtdr_irdec[N]`
- `tap_rtdr_powergood`
- `tap_rtdr_selectir`
- `tap_rtdr_tck`
- `tap_rtdr_rti (RUTI)`
- `tap_rtdr_prog_rst_b[N]`

Figure 6. Non-Bussed Connections Between sTAP and RTDR Modules

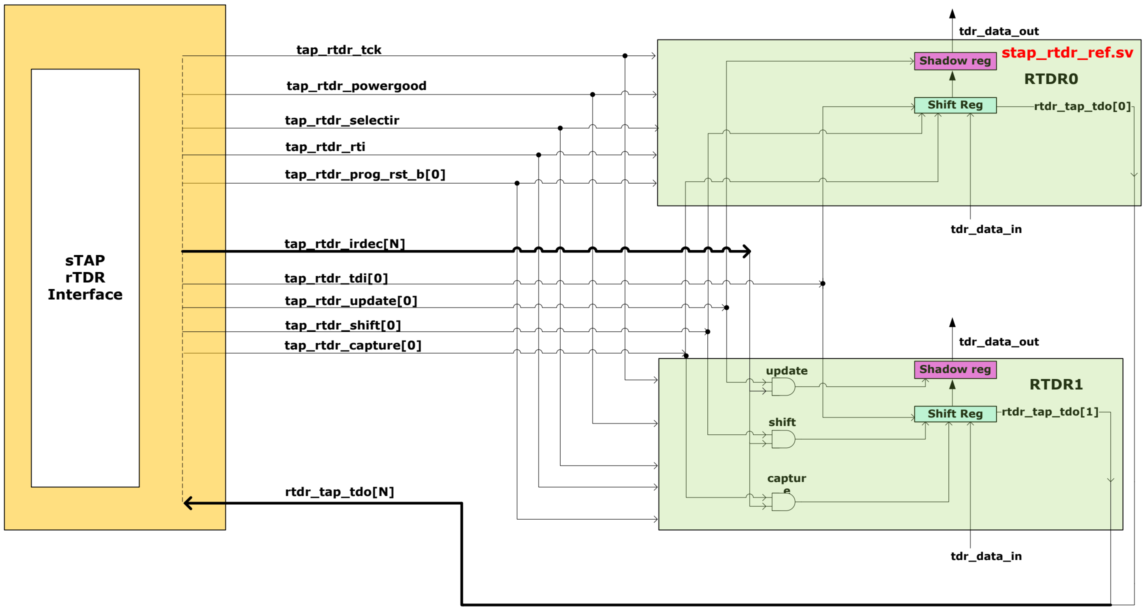
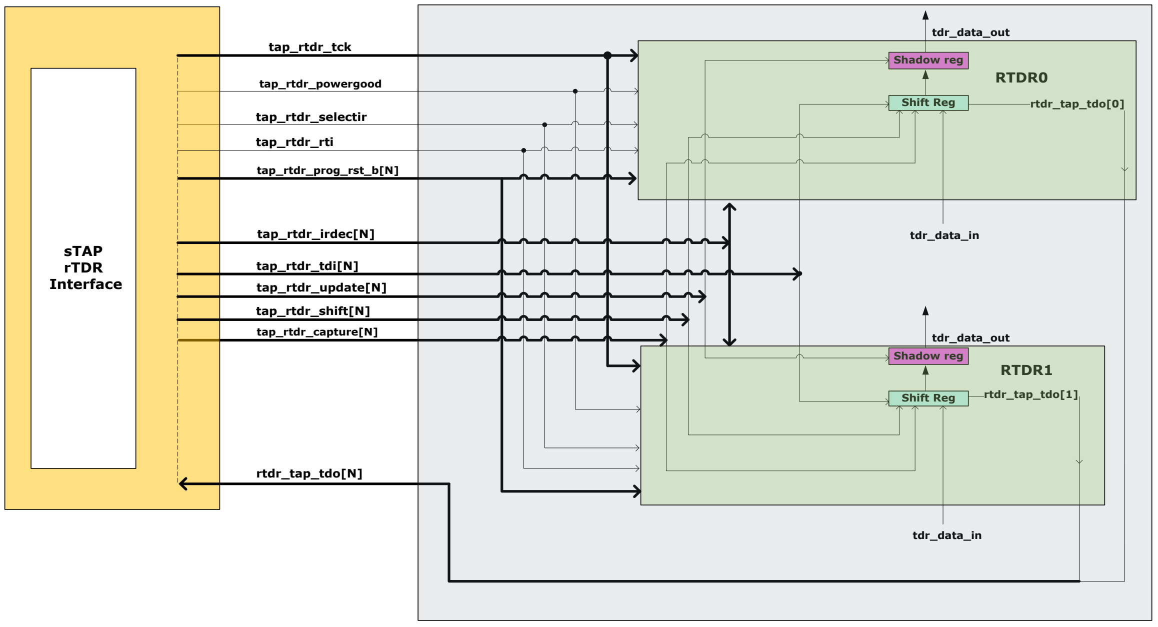


Figure 7. Bussed Connections Between sTAP and RTDR Modules



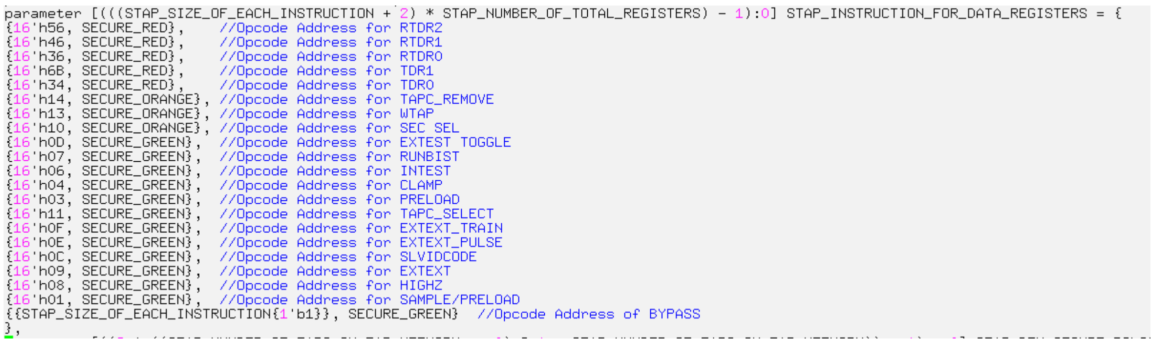
5.17.1.1 IR Decoder to RTDRs Connection Order

Figure 8 illustrates the order in which IR decoder connections are made to rTDRs for the max configuration (a configuration with all of the sTAP features enabled). The max config file is available at:

```
$IP_ROOT/source/rtl/include/configs/stap_params_include_max.inc
```

The `STAP_INSTRUCTION_FOR_DATA_REGISTERS` parameter holds the IR Table, as shown in Figure 8.

Figure 8. Snapshot of STAP_INSTRUCTION_FOR_DATA_REGISTERS



```
parameter [(((STAP_SIZE_OF_EACH_INSTRUCTION + 2) * STAP_NUMBER_OF_TOTAL_REGISTERS) - 1):0] STAP_INSTRUCTION_FOR_DATA_REGISTERS = {
{16'h56, SECURE_RED},     //Opcode Address for RTDR2
{16'h46, SECURE_RED},     //Opcode Address for RTDR1
{16'h36, SECURE_RED},     //Opcode Address for RTDR0
{16'h6B, SECURE_RED},     //Opcode Address for TDR1
{16'h34, SECURE_RED},     //Opcode Address for TDR0
{16'h14, SECURE_ORANGE}, //Opcode Address for TAPC_REMOVE
{16'h13, SECURE_ORANGE}, //Opcode Address for WTAP
{16'h10, SECURE_ORANGE}, //Opcode Address for SEC SEL
{16'h0D, SECURE_GREEN},  //Opcode Address for EXTEST TOGGLE
{16'h07, SECURE_GREEN},  //Opcode Address for RUNBIST
{16'h06, SECURE_GREEN},  //Opcode Address for INTEST
{16'h04, SECURE_GREEN},  //Opcode Address for CLAMP
{16'h03, SECURE_GREEN},  //Opcode Address for PRELOAD
{16'h11, SECURE_GREEN},  //Opcode Address for TAPC_SELECT
{16'h0F, SECURE_GREEN},  //Opcode Address for EXTEXT_TRAIN
{16'h0E, SECURE_GREEN},  //Opcode Address for EXTEXT_PULSE
{16'h0C, SECURE_GREEN},  //Opcode Address for SLVIDCODE
{16'h09, SECURE_GREEN},  //Opcode Address for EXTEXT
{16'h08, SECURE_GREEN},  //Opcode Address for HIGHZ
{16'h01, SECURE_GREEN},  //Opcode Address for SAMPLE/PRELOAD
{{STAP_SIZE_OF_EACH_INSTRUCTION{1'b1}}, SECURE_GREEN}  //Opcode Address of BYPASS
};
```

When entering the values for the parameter STAP_INSTRUCTION_FOR_DATA_REGISTERS array, you must enter them from the top down in the following order:

1. The RTDR opcodes
2. The Internal TDRs
3. The various opcodes resulting from the enabling of each feature

Table 19. IR Opcodes for Max Config (Supplied with the IP)

Opcode (Hex)	Register	IR_Decoder_Bit
56	RTDR2	stap_irdecoder_drselect(20) (tap_rtdr_irdec(2))
46	RTDR1	stap_irdecoder_drselect(19) (tap_rtdr_irdec(1))
36	RTDR0	stap_irdecoder_drselect(18) (tap_rtdr_irdec(0))
6B	Internal TDR1	stap_irdecoder_drselect(17)
34	Internal TDR0	stap_irdecoder_drselect(16)
14	TAPC_REMOVE	stap_irdecoder_drselect(15)
13	TAPC_WTAP_SEL	stap_irdecoder_drselect(14)
10	TAPC_SEC_SEL	stap_irdecoder_drselect(13)
0D	EXTEST_TOGGLE	stap_irdecoder_drselect(12)
07	RUNBIST	stap_irdecoder_drselect(11)
06	INTEST	stap_irdecoder_drselect(10)
04	CLAMP	stap_irdecoder_drselect(09)
03	PRELOAD	stap_irdecoder_drselect(08)
11	TAPC_SELECT	stap_irdecoder_drselect(07)
0F	EXTEST_TRAIN	stap_irdecoder_drselect(06)
0E	EXTEST_PULSE	stap_irdecoder_drselect(05)
0C	SLVIDCODE	stap_irdecoder_drselect(04)
09	EXTEST	stap_irdecoder_drselect(03)

Opcode (Hex)	Register	IR_Decoder_Bit
08	HIGHZ	stap_irdecoder_drselect(02)
01	SAMPLE/PRELOAD	stap_irdecoder_drselect(01)
FF	BYPASS	stap_irdecoder_drselect(00)

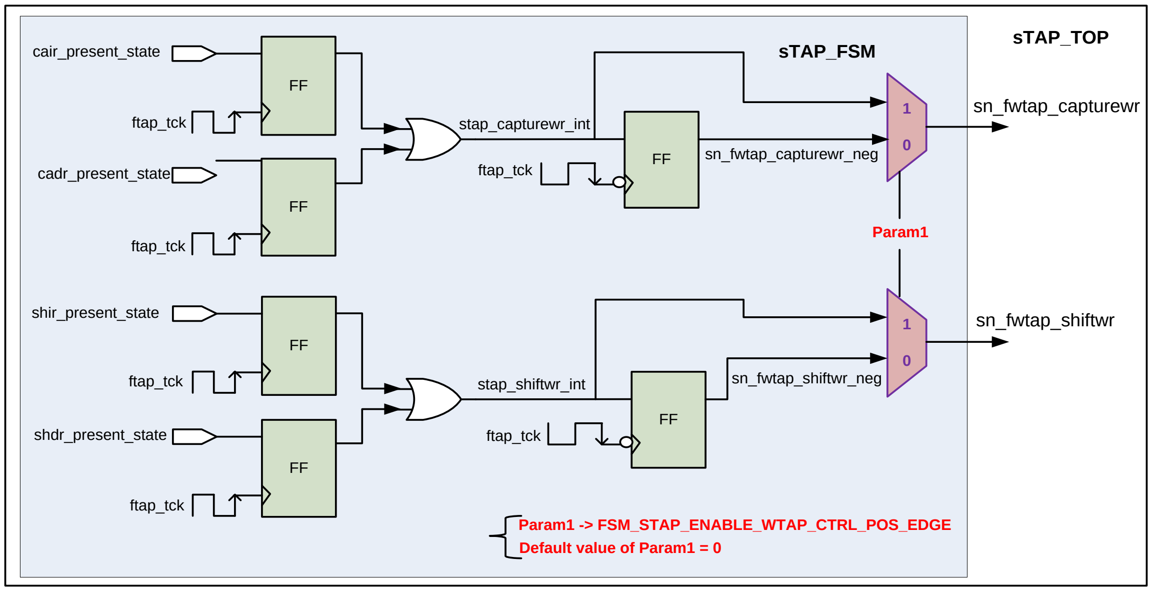
5.17.2 wTAP Control Signals

The internal control signals `stap_capturewr_int` and `stap_shiftwr_int` are captured with respect to the rising edge of the clock `ftap_tck` when the state machine is in CAIR/CADR and SHIR/SHDR respectively. These signals are flopped on negedge and become `sn_fwtap_capturewr_neg` and `sn_fwtap_shiftwr_neg`. The specific polarity of these signals is brought out through a mux, the final outputs of which are `sn_fwtap_capturewr` and `sn_fwtap_shiftwr`. This mux is controlled by the parameter `STAP_ENABLE_WTAP_CTRL_POS_EDGE`

The default value of parameter STAP_ENABLE_WTAP_CTRL_POS_EDGE is 0, which selects the falling edge clock capture control signals. To select the rising edge clock captured control when the wTAP is enabled, set the STAP_ENABLE_WTAP_CTRL_POS_EDGE value to 1.

Figure 9 illustrates the logic for the wTAP control signals in detail.

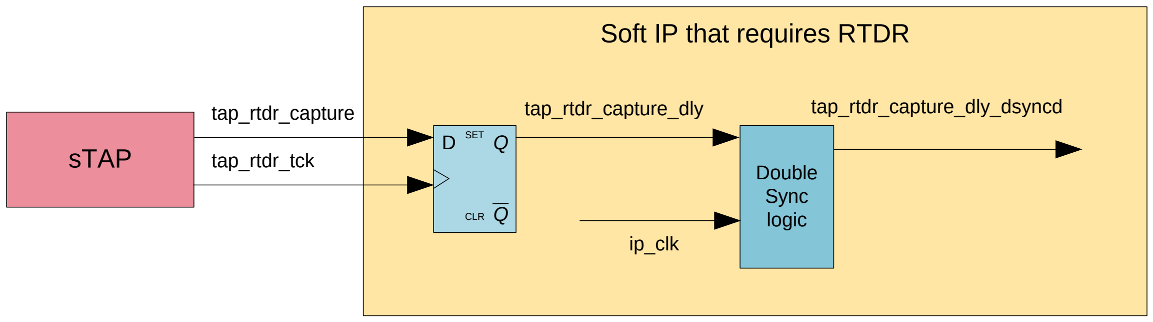
Figure 9. wTAP Control Signal Logic



5.17.3 RTDR Control Signals for use in Core Logic

If a SoftIP block implements a Remote TDR and has a requirement to operate it in a clock domain other than TAP, then the TAP control signals must be synchronized into the IP clock domain. For this, the rTDR control signals from sTAP have to first be flopped with `tap_rtdr_tck` and then double synchronized. This prevents any CDC errors pertaining to placement of combinatorial logic in front of a double sync flop. Figure 10 illustrates the logic you will need to use.

Figure 10. Synchronization Logic for tap_rtdr_capture



This logic can also be used when you need to have a double synchronized version of `tap_rtdr_update` and `tap_rtdr_irdec`.

6 Physical Integration

6.1 Memory Requirements

Not applicable to this IP/IPSS

Enter the following information for each array.

Array type and number of instances	Not Applicable
Functional usage (how many bits are used)	Not Applicable
Highest functional clock frequency	Not Applicable
Floorplan details	Not Applicable
Security requirements	Not Applicable
IP/IPSS power draw limitations for array testing	Not Applicable

6.2 Subsystem Requirements

6.2.1 Hierarchy Details

Not applicable to this IP/IPSS

6.2.2 Area and Floorplan Details

Not applicable to this IP/IPSS

6.2.2.1 Boundary

Not applicable to this IP/IPSS

6.2.2.2 Area Allocations and Confidence

Not applicable to this IP/IPSS

6.2.2.3 Interface Requirements and Confidence

Not applicable to this IP/IPSS

6.2.3 Fabric Convergence Requirements and Confidence

Not applicable to this IP/IPSS

6.2.4 OTH Straps (Logical)

Not applicable to this IP/IPSS

6.2.5 Clocking Domains and Entry/Exit Points

Not applicable to this IP/IPSS

6.2.6 Global Elements

Not applicable to this IP/IPSS

6.2.7 Power Domains and Power Grid Interface Requirements

Not applicable to this IP/IPSS

6.2.8 Timing Considerations

Not applicable to this IP/IPSS

6.2.9 Power

Not applicable to this IP/IPSS

6.2.10 QRE-related Information

Not applicable to this IP/IPSS

6.3 Open Issues

Not applicable to this IP/IPSS

7 Connectivity Chains and IP/IPSS-specific Features (HIP only)

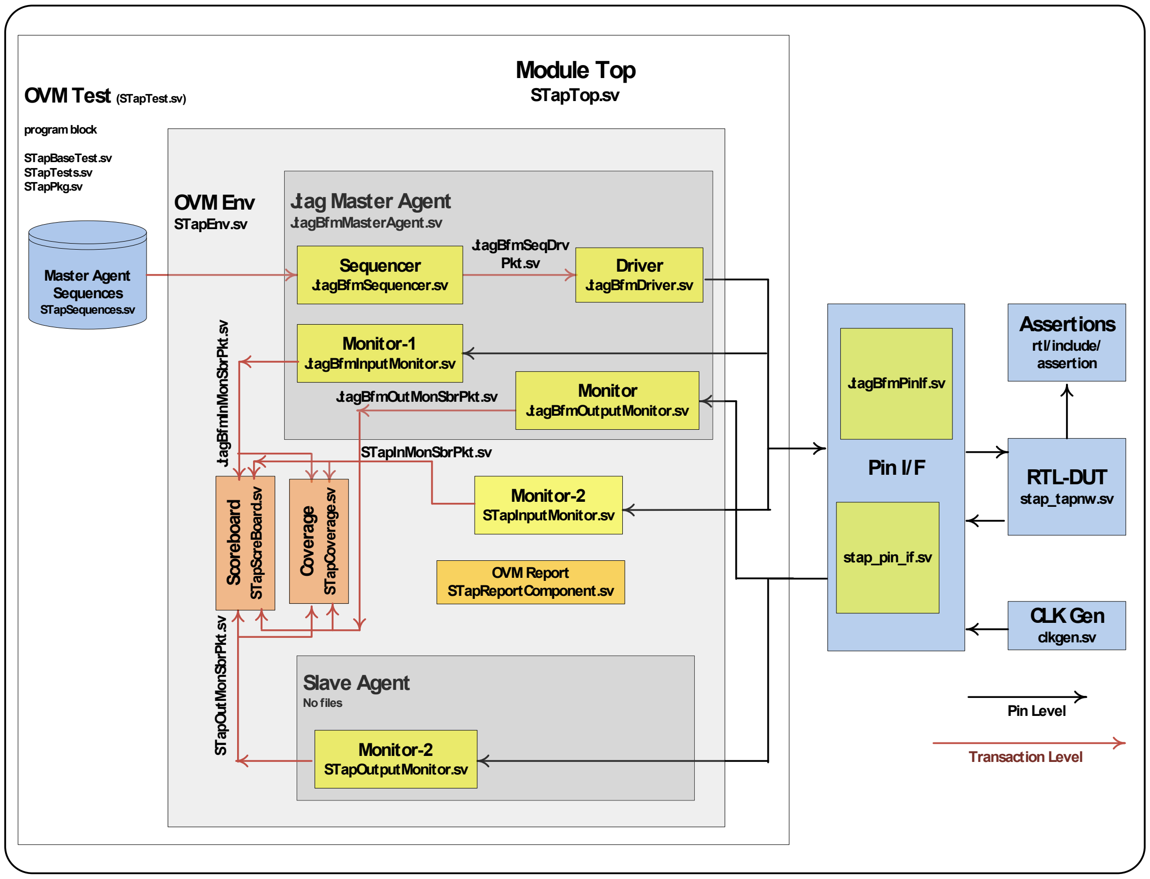
Not applicable to this IP/IPSS

8 Verification Environment for Integration

8.1 Testbench Overview

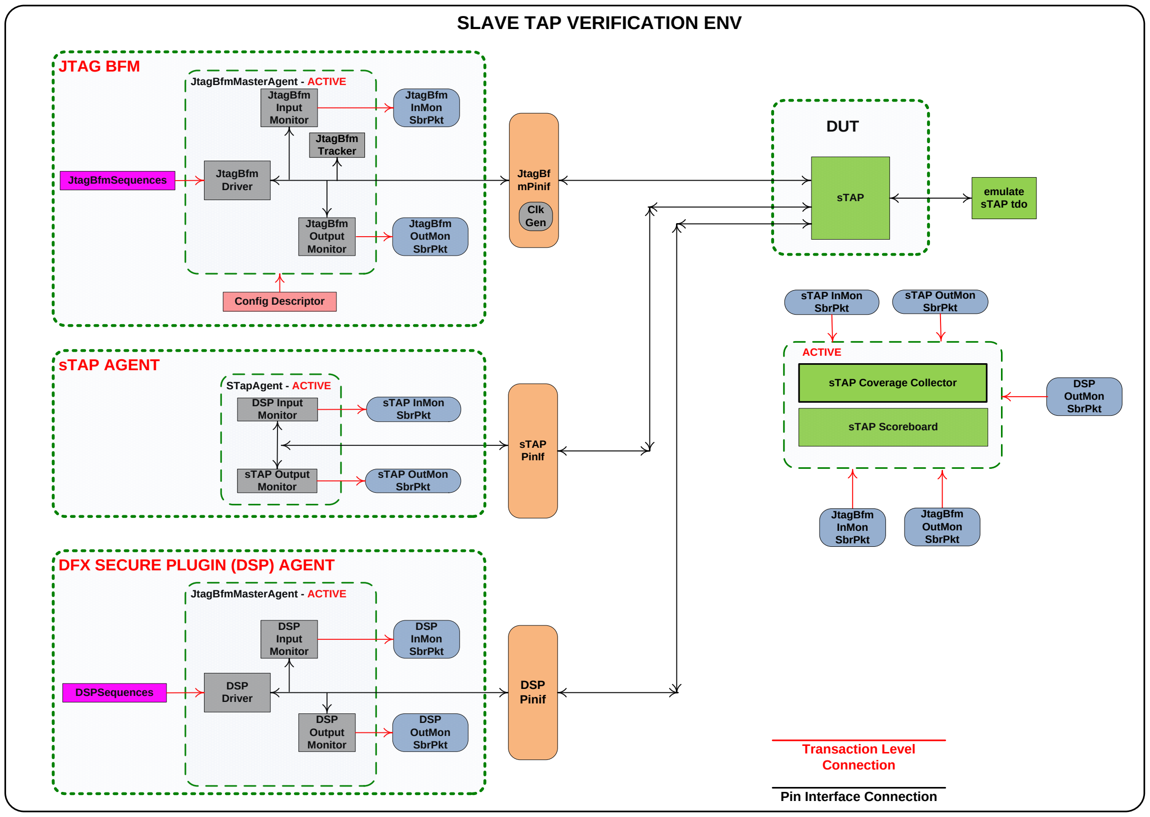
The Verification ENV is in OVM methodology. The Testbench architecture is illustrated in Figure 11. The exact files for each block are provided in the diagram. The Env class is extended from the `ovm_env`. It instantiates all the OVM components: master agent, slave agent, Scoreboard, Coverage, and OVM report.

Figure 11. Unit-Level Testbench Block Diagram without the DFx Secure Plugin



The verification environment is redrawn after integrating the DFx Secure Plugin verification.

Figure 12. Unit-Level Testbench Block Diagram with the DFx Secure Plugin



8.2 SoC-specific Validation

By integrating in Sandbox, we verified this IP for an SoC-Specific Validation.

8.3 Validation Parameters

8.3.1 #defines

```
Macro +define+JTAG_BFM_DEBUG_MSG during compile time is optional.
setenv GENERATED_FILES_FOR_JTAGBFM <user_preferred_area>
Example:
setenv GENERATED_FILES_FOR_JTAGBFM ${IP_ROOT}/verif/ctt_files/taplink
+define+INTEL_SVA_OFF+STAP_SVA_OFF+EMULATION"
+define+CTECH_LIB_META_ON+CTECH_LIB_PLUS1_ONLY
```

8.3.2 +define Command Arguments

```
+plusarg_save +CTECH_LIB_PLUS1_ONLY=1
+plusarg_save +CTECH_LIB_META_ON=1
```

8.3.3 Parameters

In order to configure the sTAP, you need to generate the RTL and Testbench configuration parameters file as described in section section x.x, Generating Configuration Parameters). The below table describes the Testbench configuration parameters for this IP.

Table 20. TB_PARAMETERS

Parameter Name	Range	Default	Description (Including Interdependencies)
MULTIPLE_TAP	Fixed at 0	0	Specifies the presence of multiple TAPs in the Env. The value 0 means no multiple TAPs are present.
NO_OF_SLAVE_TAP	Fixed at 1	1	Specifies the number of Slave TAPs present in the Testbench Env.
IDCODE_WIDTH	Fixed at 32	32	Specifies the width of IDCODE register.
CLOCK_PERIOD	INTEGER	10000	Specifies the clock period.
LENGTH_OF_BSCAN_CHAIN	INTEGER	3	Specifies the width of the Boundary Scan register.
SIZE_OF_IR_REG_STAP	Minimum 8	16	Specifies the size of the sTAP instruction register.
STAP_EN_TAP_NETWORK	0/1	1	Specifies whether the TAP Network connected to sTAP is enabled. If it is enabled, sTAP generates the required signals to drive the TAPNW.
STAP_NUM_OF_TAPS_IN_TAP_NETWORK	0 to N	4	Specifies the number of TAPs that can be present on a TAPNW.
STAP_NO_OF_TAPS_IN_TAP_NETWORK	1 to N	4	Specifies the number of TAPs present in a TAPNW. It is also used for some signal declaration, and if the number of TAPs in TAPNW is 0 then its value will be set to 1 to avoid compile warnings.
STAP_WIDTH_OF_SEC_SEL_REG_IN_STAP	0 to N	4	Specifies the width of the SEC SEL register in sTAP.
STAP_EN_WTAP_NETWORK	0/1	1	Specifies whether the wTAP Network connected to sTAP is enabled. If it is enabled, sTAP generates the required signals to drive the WTAPNW.
STAP_NUMBER_OF_WTAPS_IN_WTAP_NETWORK	0 to N	3	Specifies the number of wTAPs that can be present on a WTAPNW.
STAP_WTAP_NETWORK_SERIES_OR_PARALLEL	0/1	0	Specifies whether the wTAPs present on a WTAPNW are connected in parallel or in series: 0: Parallel 1: Series

Parameter Name	Range	Default	Description (Including Interdependencies)
STAP_NO_OF_WTAPS_IN_WTAP_NETWORK	1 to N	3	Specifies the number of wTAPs present in a WTAPNW. It is also used for some signal declaration, and if the number of wTAPs in the WTAPNW is 0 then the parameter value will be set to 1 to avoid compile warnings. If the number of wTAPs in the WTAPNW is more than 1, and the wTAPs are connected in series, the parameter value will also be set to 1 to avoid compile warnings.
STAP_NUM_OF_WTAPS_IN_WTAP_NETWORK	0 to N	3	Specifies number of wTAPs present in a WTAPNW. It is also used for some internal computation, and if the number of wTAPs in the WTAPNW is more than 1 and the wTAPs are connected in series, then the parameter value will be set to 1 to avoid data comparison errors during simulation.
NO_OF_RW_REG	INTEGER	13	Specifies the number of registers in the sTAP.
TOTAL_DATA_REGISTER_WIDTH	1-65535	135	Provides the sum of the register width of all register.
BYPASS_TRANS_WIDTH	1-65535	3	Specifies the number shifts in Bypass state.
RO_REGISTER_WIDTH	1-65535	37	Provides the width of the registers that do not contain parallel data.
TB_SECURE_GREEN	0-3	2'b00	Specifies the security policy setting on the TAP opcodes. This parameter sets the security level to green (low). When it is applied on an opcode, Intel and all Intel customers will get access to this opcode.
TB_SECURE_ORANGE	0-3	2'b01	Specifies the security policy setting on the TAP opcodes. This parameter sets the security level to Orange (medium). When it is applied on an opcode, it serves a partial unlock such that only selected Intel customers get access to this opcode.
TB_SECURE_RED	0-3	2'b10	Specifies the security policy setting on the TAP opcodes. This parameter sets the security level to Red (high). When it is applied on opcode STAP_INSTRUCTION_FOR_DATA_REGISTERS, the opcode becomes private and is visible only to debug in Intel.

Parameter Name	Range	Default	Description (Including Interdependencies)
RW_REG_ADDR	Depends on IR Width	{ //16'h36, //RTDR 16'h6B, 16'h34, 16'h14, 16'h13, 16'h10, 16'h0D, 16'h06, 16'h03, 16'h11, 16'h0F, 16'h0E, 16'h09, 16'h01 };	Defines the addresses for all registers.
RW_REG_COLOR	(2*NO_OF_RW_REG)-1:0	TB_SECURE_RED, TB_SECURE_RED, TB_SECURE_ORANGE, TB_SECURE_ORANGE, TB_SECURE_ORANGE, TB_SECURE_GREEN, TB_SECURE_GREEN, TB_SECURE_GREEN, TB_SECURE_GREEN, TB_SECURE_GREEN, TB_SECURE_GREEN, TB_SECURE_GREEN, TB_SECURE_GREEN	Defines the security level for each opcode.
CON_NO_OF_DR_REG_STAP		16'd13	Provides the sum of the register width of all the registers.

Parameter Name	Range	Default	Description (Including Interdependencies)
RW_REG_WIDTH	1-65535	{ 16'd32, 16'd32, 16'd1, 16'd3, 16'd4,16'd3, 16'd3, 16'd3, 16'd8, 16'd3, 16'd3, 16'd3, 16'd3 };	Defines the widths of all of the registers.
TB_DATA_REGISTER_RESET_VALUES	0/1 per bit of the register	{ 32'h99AA_5566 , 32'hAA55_6699 , 1'd0, 3'd0, 4'd0, 3'd0, 3'd0, 3'd0, 8'd0, 3'd0, 3'd0, 3'd0, 3'd0 };	Defines the reset value of all registers.
TB_LOAD_PIN_OR_NOT_LOOPBACK	0/1 per bit of the register	{ 32'h0000_FFFF, 32'h0000_FFFF, 1'd0, 3'd0, 4'd0, 3'd0, 3'd0, 3'd0, 8'd0, 3'd0, 3'd0, 3'd0, 3'd0 };	This parameter is a per-bit value used to specify whether the each bit of the TDR captures a fixed value (from tdr_data_in bus) or the looped-back value coming from the corresponding bit of tdr_data_out bus. 0: Loops back the TDR value from tdr_data_out in capture state 1: Takes the tdr_data_in value from core logic in capture state

Parameter Name	Range	Default	Description (Including Interdependencies)
NO_PARALLEL_OUT_BIT_WIDTH	0/1 per bit of the register	37	Specifies the width of the TDR data_out. This is used for data comparison in Scoreboard.
NUM_OF_DFX_FEATURES_TO_SECURE	0 to N	3	Specifies number of features supported in the current TAP controller.
DFX_SECURE_WIDTH	0 to N	4	Specifies width of ftap_dfxsecure and atap_dfxsecure ports.
SIZE_OF_IR_REG	INTEGER	16	Specifies size of the instruction register.
TB_REMOTE_TDR_ENABLE	0/1	0	Enables the use of Remote TDRs. Disabling the use of RTDRs makes the IP backward compatible.
TB_NO_OF_REMOTE_TDR	0 to N	0	Specifies the number of remote TDRs.
TB_NO_OF_REMOTE_TDR_NZ	1 to N	1	Specifies the number of remote TDRs. It is also used for some signal declaration, and if the number of Remote TDRs is 0 then its value will be set to 1 to avoid compile warnings.
TB_RTDR_IS_BUSSED_NZ	1 to N	1	1: A unique set of Capture, Shift, Update, and TDI signals are provided to each RTDR. 0: Single set of Capture, Shift, Update, and TDI signals is fanned out to all RTDRs.
TB_SIZE_OF_REMOTE_TDR	32	0	Generates the exact RTDR size for emulating the behavior in Testbench.
TB_DISABLE_MISC_DRIVE	0/1	0	A few of the miscellaneous pins on the sTAP are driven from the Pin Interface. By setting this to 1 the drive from pin interface is disabled allowing integration teams to drive their specific value. This parameter is set to 0 in the standalone test Env for this IP.
TB_DFX_NUM_OF_FEATURES_TO_SECURE	Integer	3	Used to override IP dfxsecure_plugin Testbench parameters and specifies the number of features supported in the current TAP controller.
TB_DFX_SECURE_WIDTH	Integer	4	Used to Overide ip-dfxsecure_plugin TB parameters and it Specifies width of ftap_dfxsecure and atap_dfxsecure ports.

Parameter Name	Range	Default	Description (Including Interdependencies)
TB_DFX_USE_SB_OVR	0/1	0	Specifies whether to use the user defined DFx Secure Feature values or the sb_policy_ovr_value. If this parameter is set then the DFx Secure Plugin will use the sb_policy_ovr_value input.
TB_CLK_PERIOD	Not Applicable	10ns	Specifies the clock period used in test environment.
STAP_NUM_OF_ITDRS	0 - N	0	Specifies the width of opcode 0x16. It is equal to number of ITDRs.

Table 21. DFx Security Parameters

Parameter	**Range**	**Default**	**Description (Including Interdependencies)**

STAP_DFX_SECURE_POLICY_MATRIX	Not Applicable	{
		 // 3 dfx features + 2 visa controls
		 {3'b001,2'b11}, // => Policy_15 (Part Disabled)
		 {3'b010,2'b11}, // => Policy_14 (User8 Unlocked
		 {3'b010,2'b11}, // => Policy_13 (User7 Unlocked)
		 {3'b010,2'b11}, // => Policy_12 (User6 Unlocked)
		 {3'b010,2'b11}, // => Policy_11 (User5 Unlocked)
		 {3'b010,2'b11}, // => Policy_10 (User4 Unlocked)
		 {3'b010,2'b11}, // => Policy_9 (User3 Unlocked)
		 {3'b010,2'b11}, // => Policy_8 (DRAM Debug Unlocked)
		 {3'b010,2'b11}, // => Policy_7 (InfraRed Unlocked)
		 {3'b100,2'b11}, // => Policy_6 (enDebug Unlocked)
		 {3'b010,2'b11}, // => Policy_5 (OEM Unlocked)
		 {3'b100,2'b11}, // => Policy_4 (Intel Unlocked)
		 {3'b001,2'b11}, // => Policy_3 (Delayed Auth Locked)
		 {3'b100,2'b11}, // => Policy_2 (Security Unlocked)
		 {3'b001,2'b11}, // => Policy_1 (Functionality Locked)
		 {3'b001,2'b11} // => Policy_0 (Security Locked)
		 }
			This parameter determines the lookup table necessary to assign the appropriate policy with the DFx feature(s) including VISA access

Table 22. TAP 0.7 Network Configuration Parameters

Parameter	**Range**	**Default**	**Description (Including Interdependencies)**
STAP_NUMBER_OF_TAPS_IN _TAP_NETWORK	0 – N	4	Specifies the number of TAPs that can be present in a TAP Network
STAP_DFX_SECURE_POLICY _SELECTREG	0 – 1	0	Determines the security policy settings of the TAPs on the TAP network
STAP_ENABLE_TAPC_REMOVE	0 – 1	1	This is a 1-bit DR opcode that enables the TAP TDI input to "pass-through" this TAP to the TAP.7 network. Writing a 1 to this register gates the internal TDI and TMS signals to the FSM/logic block. It will not add any flop delays in its path to output.

Table 23. wTAP Network Configuration Parameters

Parameter	**Range**	**Default**	**Description (Including Interdependencies)**
STAP_NUMBER_OF_WTAPS_IN _NETWORK	0 – N	3	Specifies the number of wTAPs that can be present in a wTAP Network.
STAP_WTAP_NETWORK_ONE_FOR _SERIES_ZERO_FOR_PARALLEL	0 – 1	0	Determines whether the wTAPs on a wTAP Network are connected in series or in parallel: 0: Parallel 1: Serial
STAP_ENABLE_WTAP_CTRL _POS_EDGE	0 – 1	1	When wTAP is enabled, this parameter allows you to specify whether control signals like fsm_capture_dr and fsm_shift_dr are enabled with the positive or negative edge of the clock: 0: Negative edge 1: Positive edge

Table 24. Remote Test Data Register Configuration Parameters

Parameter	**Range**	**Default**	**Description (Including Interdependencies)**
STAP_NUMBER_OF_REMOTE _TEST_DATA_REGISTERS	0 – N	0	Specifies the number of remote TDRs that are present in the sTAP
STAP_ENABLE_RTDR_PROG_RST	0 - 1	0	Specifies the programmable reset option for RTDRs. If the value of this parameter is set to 1, then the TAPC_RTDRRSTSEL and TAPC_TDRRSTEN registers come into existence. You must specify which register has the programmable reset option by writing 1 to the corresponding bit position for each of the RTDRs in TAPC_RTDRRSTSEL register. The applicable programmable reset is determined by the value programmed in TAPC_TDRRSTEN register.

Parameter	Range	Default	Description (Including Interdependencies)
STAP_RTDR_IS_BUSSED	0 – 1	1	1: A unique set of Capture, Shift, Update, and TDI signals are provided to each RTDR. 0: A single set of Capture, Shift, Update, and TDI signals are fanned out to all RTDRs.

8.3.4 Variables

Not applicable to this IP/IPSS

8.3.5 Overrides

Not applicable to this IP/IPSS

8.3.6 Plusargs

Following are the plusargs used.

simv_args "+fsdb=all +SLA_MAX_RUN_CLOCK=4000000 +OVM_VERBOSITY=OVM_FULL +CTECH_LIB_META_DISPLAY"

8.3.7 Pre- and Post- TI Imports

jtagBfm and DfxSecurePlugin Agent TI's are imported.

8.4 Verification Libraries

Not applicable to this IP/IPSS

8.5 Testbench Components and Connectivity

8.5.1 Testbench Directory Structure

Table 25. Testbench Directory Description

Content	Directory Relative to IP_ROOT
Environment Files and	`verif/tb/STapEnv.sv` `verif/tb/STapEnv_Converged.sv`
scoreboard	`verif/tb/STapScoreBoard.sv`
Tracker	`verif/tb/STapOutputMonitor.sv` `verif/tb/STapInputMonitor.sv`
coverage	`verif/tb/STapCoverage.sv`
PInn interaface	`verif/tb/stap_pin_if.sv`
Test Lib	`verif/tests/STAPTestPkg.sv`

8.5.2 Test Island

Test Island is a module in the environment. The only reusable part of this IP in the Testbench is the instantiation of the pin interface through the virtual interface container. There is a string-based association of the interface for this IP. A `set_config_object` with a unique string is set in the Test Island. A `get_config_object` of the same string is fetched in the `TapEnv`. In this way the interface is passed.

A consumer of this IP needs to instantiate the Test Island as is done in `TapTop.sv`.

The program block still is used in the module `top` to connect the Env and the RTL. However, no interfaces are passed here as this is done in the Test Island. The code sample below (from file `TapTop.sv`) instantiates the Test Island in the `top` module:

```
// From file TapTop.sv
module top();

    reg soc_fdfx_powergood;
    reg soc_clock;

       JtagBfmIntf #(.CLOCK_PERIOD (10000), .PWRGOOD_SRC (0), .CLK_SRC(0));
       Primary_if(soc_fdfx_powergood, soc_clock);
       stap_pin_if pif(Primary_if.jtagbfm_clk);
        DfxSecurePlugin_pin_if #(`STAP_DSP_TB_PARAMS_INST
                        dfxsecure_pins(Primary_if.jtagbfm_clk);

    //------------------
    // Instatiate the Test Island
    //------------------
    TapTestIsland i_TapTestIsland(
                                  //Inputs
                                  .pif   (pif),
                                  .Primary_if(Primary_if),
                                  .pins  (dfxsecure_pins)
                                 );
Endmodule
```

8.5.3 Making Test Island Connections

Table 26 describes the signal that needs to be connected at the SoC level.

Table 26. Test Island Connections

Signal	Connect to	Description
Primary_if	i_TapTestIsland	Passes the primary instance of JTAG BFM Interface to sTAP Test Island.
dfx_secure_pins	i_TapTestIsland	DFx Secure Plugin interface.
pif	i_TapTestIsland	sTAP interface that forms the non-standard JTAG interface.

8.5.4 Test Island Interfaces

Use STAP_Integ_TestIsland.sv for connecting interfaces.

8.5.5 Checkers and Trackers

STAP_Scoreboard is used to check all types of transactions inside STAP.

8.5.6 Monitors

The Input Monitor class is extended from the ovm_component base class. This class captures the pin interface signals that are sent to the DUT. It internally runs an FSM driven by the TMS to determine the state of the DUT. The address is the accumulated TDI value when the FSM is in shift IR state; similarly, data is the accumulated TDI value when the data is in shift DR state. The data and address are cleared when the FSM is in Run Test Idle or Test Logic Reset state. The address and data—together with the present state of the FSM, the Vercode, the IDCODE, and the parallel data input—form the transaction packet `TapInMonSbrPkt`, which is passed to the Scoreboard and black-box coverage.

The Output Monitor class is extended from the ovm_component base class. This class captures the TDO pin value serially and converts it to parallel. This class also collects the driver signals (TCK, TMS, trst_b) to the DUT and internally runs the FSM. The TDO is accumulated when the present state of this FSM is either in Shift IR or Shift DR state. This parallel data is sent to the Scoreboard and black-box coverage using the `TapOutMonSbrPkt` packet. The transaction packet is sent whenever the state machine comes out of the Shift IR or Shift DR states. After the packet is sent, the data and the address are cleared for the next capture. The Monitor also captures the parallel data out from the DUT (if the selected DR register has parallel data out).

8.5.7 Scoreboards

STAP_Scoreboard is used to check all types of transactions inside STAP.

8.5.8 BFMs

JTAGBFM is a separate VIP that is used in the verification of sTAP. Refer to the JTAG BFM documentation for the various APIs/tasks/functions.

8.5.8.1 Initial Condition of TAP Signals from the BFM

The BFM is modified to drive undriven states (described in IEEE 1149.1 specification) on TAP signals. Accordingly, at time t=0ps, TMS and `trst_b` are driven with logic high, and TDI and TCK are driven with logic low. (The RTL can handle various possibilities of assertion and de-assertion of `fdfx_powergood` and `trst_b`.)

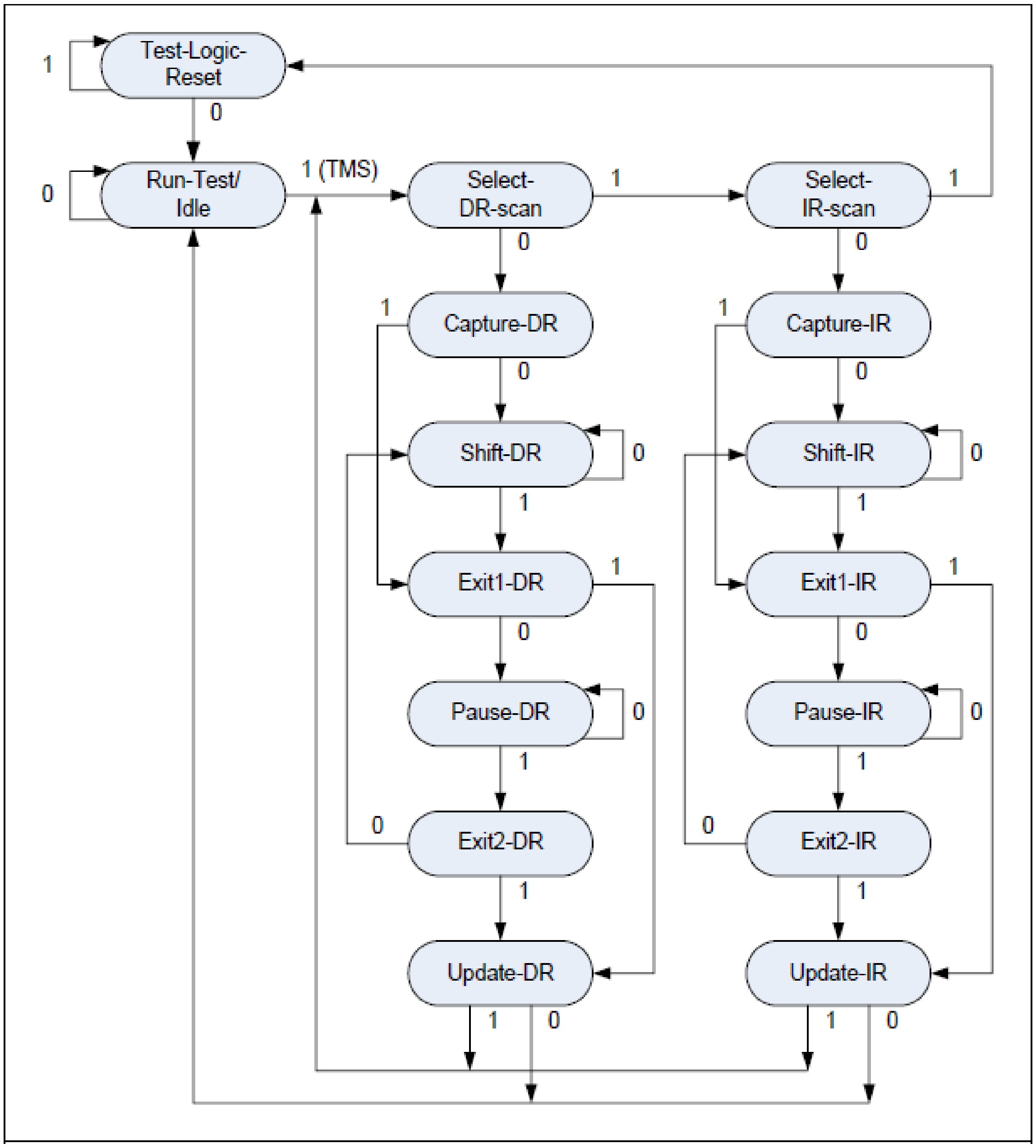
The `ForceReset` and `Reset` tasks provide various options to power-sequence and test the logic. In addition, by setting the `enable_clk_gating` variable in the test, you can have a free running clock or run the clock only when a transaction is performed. Further, the `park_clock_at` variable allows the clock to park TCK at the desired logic level when clock gating is enabled.

The existing Reset task just pulses the chosen reset. ForceReset followed by Idle holds the reset to the desired logic level. While the Reset task works only when TCK is present, ForceReset does not depend on TCK. Therefore, you can use ForceReset for power sequencing tests even when the clock is disabled.

8.5.8.2 TAP Finite State Machine

The TAP has a state machine that has 16 different states. Refer to IEEE1149.1 specification for the definition of each state. The state transition diagram is illustrated in Figure 13.

Figure 13. TAP FSM State Machine Flow Diagram



8.5.9 Other Structures

Not applicable to this IP/IPSS

8.6 Collage or Sandbox Files

There are no Sandbox files.

Collage is supported. The coreKit is available at the `tools/collage/coreKit` directory.

This IP contains a CoreKit for Collage to aid in an SoC RTL integration. There are several groups of interfaces as part of the CoreKit. Signals that cannot be grouped due to specific functionality should be connected at SoC level (CoreAssembler). Core Kit is available in `$IP_ROOT/tools/collage/builder/builder.stap.tcl`

Table 27. Collage Connections

Interface Groups	Description
jtag_pri	Primary JTAG ports.
jtag_sec	Secondary JTAG ports.
tapnw_pri	Network facing TAP ports from Primary.
tapnw_sec	Network facing TAP ports from Secondary.
tapnw_ctrl	Network control signals like enables and secondary selects.
rtdr	RTDR ports with STAP_RTDR_IS_BUSSED = 1
rtdr_bussed_0	RTDR ports with STAP_RTDR_IS_BUSSED = 0
iosf_dfx_dfxsecure_plugin	DFx Secure Plugin ports.
bscan_stap	Boundary Scan interface.
wtap	wTAP Network interface.

8.7 Safety Files

Not applicable to this IP/IPSS

9 Workarounds, Waivers, and Disabled Assertions

9.1 Disabled Assertions

Not applicable to this IP/IPSS

9.2 Waivers

Not applicable to this IP/IPSS

9.3 Other Workarounds

Not applicable to this IP/IPSS

10 Integration Checks and Tests

Once sTAP is integrated into a cluster, then one should run Slaveidcode and Bypass Tests.

Here is a code snippet of the actual test. First, in order to emulate the Debug Life Cycles States, the sequencer of DFx Secure Plugin Agent is used to drive the SECURITY_LOCKED policy on the policy bus. Next, the fdfx_powergood is asserted using the JTAG BFM sequencer to remove the TAP out of reset. Then the SECURITY_UNLOCKED sequence is run to set the correct set of features that are visible either to Intel, selective customer or to all. Then the actual test case sequence is executed.

```
class TapTestBypassAndSlvidCode extends STapBaseTest;

    `ovm_component_utils(TapTestBypassAndSlvidCode)
    SecurityLocked    SL;
    PowergoodReset    PG;
    SecurityUnlocked  SUL;
    TapSequenceBypass SP;
    OnlySLVIDCODE     SLVID;

    function new (string name = "TapTestBypassAndSlvidCode", ovm_component parent =
null);
        super.new(name,parent);
    endfunction : new

    virtual function void build();
        super.build();
    endfunction : build

    virtual task run();
        ovm_report_info("TapTestBypassAndSlvidCode","Test Starts!!!");
        SL  = new("Security Locked Mode");
        PG  = new("Powergood reset");
        SUL = new("Security Unlocked Mode");
        SP  = new("ALL Primary Mode");
        SLVID = new("SLVIDCODE");

        //PG.start(Env.stap_DfxSecurePlugin_Agent.i_DfxSecurePlugin_Seqr);
        SL.start(Env.stap_DfxSecurePlugin_Agent.i_DfxSecurePlugin_Seqr);
        PG.start(Env.stap_JtagMasterAgent.Sequencer);
        SUL.start(Env.stap_DfxSecurePlugin_Agent.i_DfxSecurePlugin_Seqr);
        SP.start(Env.stap_JtagMasterAgent.Sequencer);
        SLVID.start(Env.stap_JtagMasterAgent.Sequencer);
        ovm_report_info("TapTestBypass","Test Completed!!!");
        global_stop_request();
    endtask : run
endclass : TapTestBypassAndSlvidCode
```

By default, all assertions are turned on. You can turn off the assertions by providing the switch INTEL_SVA_OFF