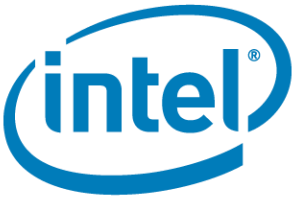


DFx Secure Plugin

INTEGRATION GUIDE

IP Rev. PIC5_RTL1P0_V3
Jan 2019

Intel Restricted Secret

This document contains information on products in the design phase of development.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your Intel account manager or distributor to obtain the latest specifications and before placing your product order.

Copies of documents that have an order number and are referenced in this document or in other Intel literature can be obtained from your Intel account manager or distributor.

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 to this IP" 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 into an SoC.

1.2 Supported Projects

This document supports the following projects at the listed RTL maturity level.

Project Name	IP Maturity Level
ALL/MULTI	PIC4

1.3 Terminology

The table below defines uncommon terms used in this document.

Term	Definition
DSP	Dfx Secure Plugin

1.4 Related Documents

Document Title	Location
IOSF DFx HAS Rev 1.3_rc3	Included in the 'doc' directory for this release
DFx Secure Plugin Product Brief	Included in the 'doc' directory for this release
DFx Secure Plugin Verification Plan Reference	Included in the 'doc' directory for this release
DFx Secure Plugin HAS	Included in the 'doc' directory for this release

1.5 Contact Information

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

Name	Function	Email
Bulusu, Shivaprashant Bandana V, Sudheer	Verification	shivaprashant.bulusu@intel.com sudheer.v.bandana@intel.com
Rakesh, Kandula Adithya, B, S	Design	rakesh.kandula@intel.com b.s.adithya@intel.com
Flowers, Susann	Doc template owner	susann.flowers@intel.com

1.6 Document Revision History

Revision Number	Description of Change	Date	Revised By
1.0v2	Updated document to current content template	21 Apr 2014	Kathryn Craven-Sarr

Revision Number	Description of Change	Date	Revised By
1.0v3	Updated new template	15 Dec 2014	Vikram Sharma
1.0v4	HDK update	7 Apr 2015	Vikram Sharma
1.0v5	Update for ICL	1 Jan 2016	Sudheer V Bandana
PIC3_R1P0_V1	PCR 1604197232 is updated	1 July 2016	Devendar Reddy D
PIC4_RTL1P0_V1	TSA/MAT enabled	15 June 2017	BS Adithya
PIC5_RTL1P0_V1	TFM Updates. Spyglass Lint and Spyglass CDC enabled	22 March 2018	BS Adithya
PIC5_RTL1P0_V2	TSA Prime Release	26 July 2018	BS Adithya
PIC5_RTL1P0_V3	ADP-S Samsung 14nm customer added	18 Jan 2019	BS Adithya

2 Quick Start

2.1 Downloading Sub IP

Not applicable to this IP

2.2 Firmware Version

Not applicable to this IP

2.3 Integrity Checks for Standalone IP

Following are the steps for running standalone integrity checks of this IP. Complete set of standard README_HDK is provided at every IP-level called README_HDK.txt, which has all the tool running commands. CUST_OPTION is to be provided for every run. Failing to provide the input, the default customer would be taken. Tool version will be changed.

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:

   ```
   make set_hdk
   ```

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

   ```
   make set_hdk_pre
   ```

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 Lintra:

   This is HDK based approach. HDK simbuild 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_compile CUST=<CUST_NAME>
   Eg: make run_lint_compile CUST=ADL


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

   #To run lintra compile for Spyglass
   make run_sglint_compile CUST=<CUST_NAME>
   Eg: make run_sglint_compile CUST=ADL


   #To run lintra elab
   make run_sglint_elab CUST=<CUST_NAME>
   ```

```
Rg: make run_sglint_elab CUST=ADL
```

6. To run synthesis and LEC after generating the file list. In HDK flow, FEBE environment is used to run syn and LEC.

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

7. To run EMULATION:

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

8. To run CDC:

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

For Spyglass
make run_sgcdc_comp  CUST=<CUST_NAME>
make run_sgcdc_test CUST=<CUST_NAME>
```

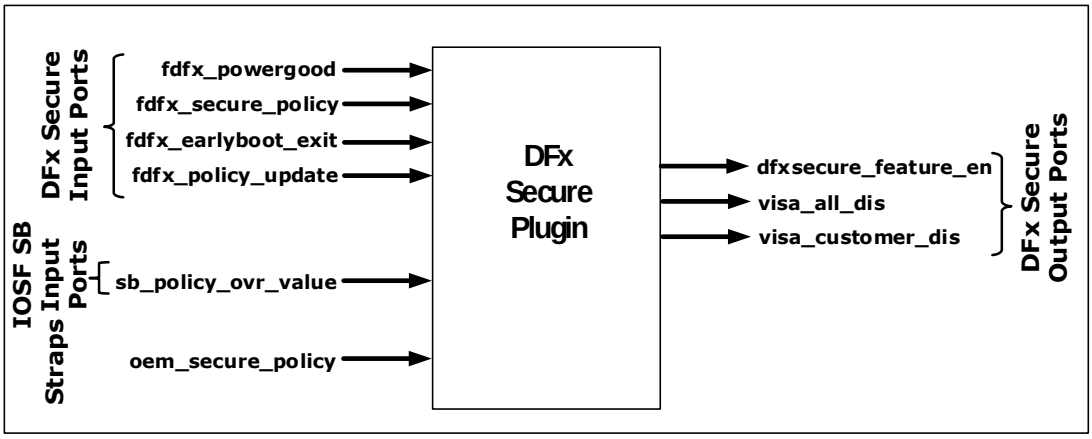
9. To run collage:

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

3 Overview

3.1 IP Block Diagram

Figure 1. DFx Secure Plugin Block Diagram



3.2 Functional Top-Level Signals

Table 1. Top-Level Signals

Signal Name	Direction	Source/ Destination	Width
fdfx_powergood	Input	SoC Chassis	1
fdfx_secure_policy	Input	Security Aggregator	DFX_SECURE_WIDTH
fdfx_earlyboot_exit	Input	Security Aggregator	1
fdfx_policy_update	Input	Security Aggregator	1
sb_policy_ovr_value	Input	Sideband	DFX_NUM_OF_FEATURES_TO_SECURE + 1
oem_secure_policy	Input	Strap	DFX_SECURE_WIDTH
dfxsecure_feature_en	Output	To be consumed by the IP	DFX_NUM_OF_FEATURES_TO_SECURE
visa_all_dis	Output	To be consumed by the IP	1
visa_customer_dis	Output	To be consumed by the IP	1

4 Design Information for Integration

This chapter is primarily targeted for the IP integration team responsible for integrating this IP into a SoC.

4.1 RTL Directory Structure

```
source/
`-- rtl
    |-- dfxsecure_plugin
    |   `-- dfxsecure_plugin.sv
    |-- dfxsecure_plugin_rtl.hdl
    |-- dfxsecure_plugin_rtl.hdl_gls
    |-- dfxsecure_plugin_rtl.hdl_rtl
    `-- include
        |-- assertions
        |   `-- dfxsecure_include.sv
        |-- dfxsecure_defines_include.inc
        `-- dfxsecure_params_include.vh
```

4.2 Clock, Power and Reset Domain

This IP consists only of combinatorial logic and does not reside on any clock or power domain.

4.2.1 Clock Domains

Not applicable to this IP

4.2.2 Power Domains

Not applicable to this IP

4.2.3 Resets

There is only one `fdfx_powergood` reset in the design and no minimum assertion time for this reset. The latch in the design is cleared by `fdfx_powergood`. This reset is similar to `powergood_rst_b` in the SoC which is an active-low signal.

4.3 Embedded Building Blocks/Custom Logic

Not applicable to this IP

4.4 RTL Configuration Parameters

In order to configure the DFx Secure Plugin, you need to generate the RTL and Testbench configuration parameters file as described in section 4.6, "Generating Configuration Parameters". Once created, you can adjust the user-defined RTL parameters either by manually editing the RTL parameters file or by using the Parameter Generation Tool.

4.4.1 Mandatory Parameters

Table 2. RTL Configuration Parameters

Parameter Name	Range	Default	Description (Including Interdependencies)
DFX_NUM_OF_FEATURES _TO_SECURE	1 – N	1	Specifies the number of features supported in the current DFx Secure Plugin Implementation. The minimum number of features to secure are one. Theoretically, DFx Secure Plugin can support any number of features, but in practice, the number of features is likely to be in the single digits.
DFX_SECURE_WIDTH	4	4	Specifies the width of the DFx Secure Policy bus. Its value has been fixed for the current DFx Secure Plugin implementation. This parameter is fixed at 4 for the 14 nm chassis generation (Chassis Gen3). **Note:** Should not change this parameter.

DFX_SECURE_POLICY _MATRIX	Not Applicable	User-defined DFx Secure Feature values:
		{
		{1'b0,DFX_VISA_BLACK},
		{1'b0,DFX_VISA_ORANGE},
		{1'b0,DFX_VISA_ORANGE},
		{1'b0,DFX_VISA_ORANGE},
		{1'b0,DFX_VISA_ORANGE},
		{1'b0,DFX_VISA_ORANGE},
		{1'b0,DFX_VISA_ORANGE},
		{1'b0,DFX_VISA_ORANGE},
		{1'b1,DFX_VISA_RED},
		{1'b0,DFX_VISA_BLACK},
		{1'b0,DFX_VISA_ORANGE}
		{1'b1,DFX_VISA_RED},
		{1'b0,DFX_VISA_GREEN},
		{1'b1,DFX_VISA_RED},
		{1'b0,DFX_VISA_BLACK},
		{1'b0,DFX_VISA_GREEN}
		}
			Holds an array containing user-defined DFx Secure Feature values.
			This parameter specifies the lookup table necessary to assign the appropriate policy to the DFx feature(s), including VISA access.

Parameter Name	Range	Default	Description (Including Interdependencies)
DFX_USE_SB_OVR	0 – 1	0	Specifies whether to use the sb_policy_ovr_value or the user-defined DFx Secure Feature values. If this parameter is set, then the DFx Secure Plugin uses the sb_policy_ovr_value input.
DFX_VISA_BLACK	0 - 3	2'b11	Specifies the VISA access value in an enumerated parameter format. This value clock gates the output flops and prevents bypass from functioning.
DFX_VISA_GREEN	0 - 3	2'b01	Specifies the VISA access level in an enumerated parameter format. The VISA Green access provides debug signals to customers without a key.
DFX_VISA_ORANGE	0 - 3	2'b10	Specifies the VISA access level in an enumerated parameter format. The VISA Orange access level provides debug signals to customers with a key.
DFX_VISA_RED	0 - 3	2'b00	Specifies the VISA access level in an enumerated parameter format. The VISA Red access level provides debug signals to Intel's use models.
DFX_EARLYBOOT_FEATURE_ENABLE	Not Applicable	3'b001	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. However, there are DFx features in suspend well (SUS) that require full access. Most IPs: DFX_EARLYBOOT_FEATURE_ENABLE[1:0] = VISA_BLACK DFX_EARLYBOOT_FEATURE_ENABLE[1:0] = {[DFX_NUM_OF_FEATURES_TO_SECURE:2]}{1'b0}}

4.4.2 Test Data Register Parameters

Not applicable to this IP

4.5 Testbench Configuration Parameters

In order to configure the DFx Secure Plugin, you need to generate the RTL and Testbench configuration parameters file as described in section 4.6, "Generating Configuration Parameters". Table 3 describes the Testbench configuration parameters for this IP.

Table 3. Testbench Configuration Parameters

Parameter Name	Range	Default	Description (Including Interdependencies)
TB_DFX_NUM_OF_FEATURES_TO_SECURE	1 – N	1	Specifies the number of features supported in the current DFx Secure Plugin implementation. The minimum number of features to secure is one. Although theoretically it can be any number of features, it is most likely to be in the single digits.
TB_DFX_SECURE_WIDTH	4	4	Specifies the width of DFx Secure Policy bus. Its value has been fixed for the current DFx Secure Plugin implementation. This parameter is fixed at 4 for the 14 nm chassis generation (Chassis Gen3). **Note:** Do not change this parameter.
TB_DFX_USE_SB_OVR	0 – 1	0	Specifies whether to use the sb_policy_ovr_value or the user defined DFx Secure Feature values. If this parameter is set, then the DFx Secure Plugin will use the sb_policy_ovr_value input.
TB_CLK_PERIOD	Not Applicable	No Clock	Specifies the clock period used in a test environment.

4.6 Generating Configuration Parameters

In order to use the DFx Secure Plugin IP, you must generate the RTL and Testbench configuration parameters files.

4.6.1 Tool-based Parameter Generation

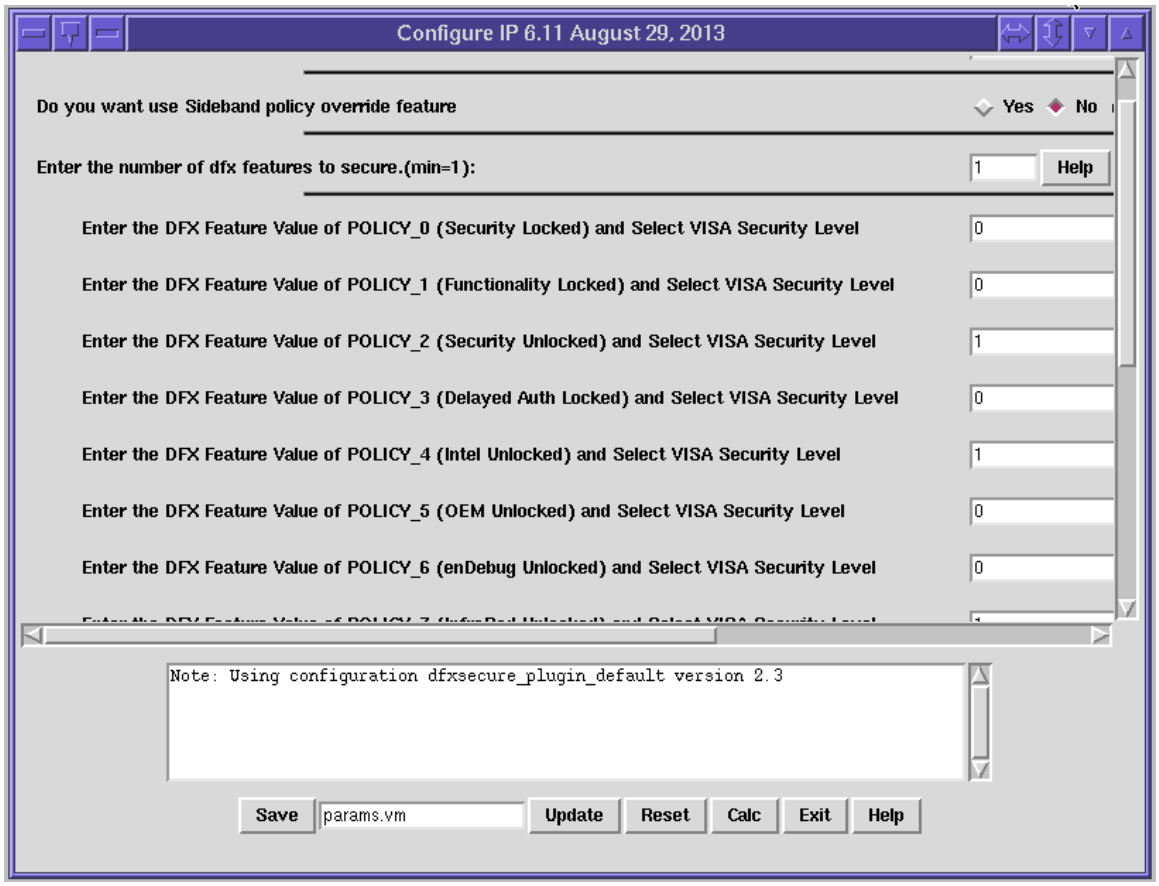
Instead of manually editing the RTL parameter file, you can use the Parameter Generation Tool to change the user-defined RTL parameter (see Table 2 for a list of parameters). When you start the Parameter Generation tool, it generates a System Verilog parameter override file for instantiating the IP.

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 as shown in Figure 2, enter the below commands:

```
cd $IP_ROOT/scripts/dfx_parameter_gui/
/p/com/eda/intel/siputils/prod/bin/configIP dfxsecure_plugin.config
-if <ip_name>_input_file.vh
```

Figure 2. Parameter Generation Tool



The Parameter Generation Tool generates seven parameter files, four with a user-supplied name prepended to the file name. For example, if you specify "MIPI", the files listed in the Table 4 are generated when you click the Save button.

Table 4. Parameter Generation Tool Generated Files

Generated Files	Description
MIPI_dfxsecureplugin_param_values.vh	This file contains the values of user-specific uniquified parameters. This file is used in (i) Include directory where the RTL can find the value of the parameters. (ii) This file is also used in SoC_TestIsland. It should be included above the module definition of SoC_TestIsland. To avoid any conflicts between RTL and verif, place this file with the same name in both directories.
MIPI_dfxsecureplugin_param_overide.vh	This file contains the SV parameter override definition. This file is used in (i) RTL instantiation (ii) This file is used in SoC_TestIsland. This file should be included in the parameter instantiation of the DfxSecurePlugin_TestIsland module in SoC_TestIsland.

Generated Files	Description
MIPI_DfxSecurePlugin_TbDefines.svh	This file contains the parameters required by the Verification Environment. This file should be included in the following files that contain: (i) Topmost testbench module. (ii) Cluster Package. (iii) BaseTest or Program Block. It is used in the parameter instantiation of TopTb, SoCTestIsland and DfxSecurePluginPinIf.
MIPI_input_file.vh	Currently, generated configuration is saved into this file for later retrieval. When the tool is invoked with this file, it will pre populate the ConfigIP GUI with the previously generated parameters.
dfxsecure_params_include.vh	Contains the parameters of the RTL module. IP consumers should ignore this file.
dsp_tb_params.vh	Contains the same content as dfxsecure_params_include.vh file, but is used for inclusion in TestIsland. This has been done to remove the dependency on RTL parameters file. The RTL parameters file gets uniquified when the DfxSecurePlugin gets instantiated in another IP, which causes the TestIsland to fail during compilation. To avoid the TestIsland failure to compile, this new file has been introduced.
params.vm	An intermediate file generated by the parameter generation tool. Ignore this file.

The Parameter Generation Tool also allows you to enter DFx Policy Matrix values for the features of each policy. You can select the VISA Security Levels for each policy from drop down menus. If you do not enter feature values and VISA Security Levels for Policies, the tool selects the default policy matrix values as shown in Table 5.

Table 5. DFx Secure Policies

Policy	Default
POLICY_0	1'b0,VISA_GREEN
POLICY_1	1'b0, VISA_BLACK
POLICY_2	1'b1, VISA_RED
POLICY_3	1'b0, VISA_GREEN
POLICY_4	1'b1, VISA_RED
POLICY_5	1'b0, VISA_ORANGE
POLICY_6	1'b0, VISA_BLACK
POLICY_7	1'b1, VISA_RED
POLICY_8	1'b0, VISA_ORANGE
POLICY_9	1'b0, VISA_ORANGE
POLICY_10	1'b0, VISA_ORANGE
POLICY_11	1'b0, VISA_ORANGE
POLICY_12	1'b0, VISA_ORANGE
POLICY_13	1'b0, VISA_ORANGE
POLICY_14	1'b0, VISA_ORANGE

Policy	Default
POLICY_15	1'b0, VISA_BLACK

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 displays 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" (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 warnings or errors 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 `<ip_name>_input_file.vh` file, not the other two. In the example case, the file you would want to upload is `MIPI_input_file.vh`. To upload the file, use the below commands:

```
/p/com/eda/intel/siputils/prod/bin/configIP dfxsecure_plugin.config
-odf <ip_name>_input_file.vh -if <ip_name>_input_file.vh
```

In the example above, the command would be:

```
/p/com/eda/intel/siputils/prod/bin/configIP dfxsecure_plugin.config
-if MIPI_input_file.vh
```

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.

Note: If you do not click the Save button before exiting the tool, none of the 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.

4.7 IP Straps

Two straps are used in Dfx Secure Plugin: `oem_secure_policy` and `fdfx_earlyboot_exit`.

When the parameter value of DFX_USE_SB_OVR is zero, oem_secure_policy is driven with a value of 0x0, otherwise, oem_secure_policy is driven with available policy values.

During debug this strap will provide the values for various features, until the earlyboot_exit goes high.

4.8 Fuses

There are no fuses for this IP.

4.9 Power Information

Not applicable to this IP

4.9.1 Power Supply

Not applicable to this IP

4.9.2 Static Clock Gating

Not applicable to this IP

4.9.3 Power Gating

Not applicable to this IP

4.9.4 Bumps and Their Power Domains

Not applicable to this IP

4.10 Power-up Requirements

Not applicable to this IP

4.11 Macros used by IP

Not applicable to this IP

4.12 Other Design Considerations

This IP does not run on any clocks.

4.13 DFx Considerations

Not applicable to this IP

4.13.1 DFx Top-Level Signals

Not applicable to this IP

4.13.2 DFx Clock Definition

Not applicable to this IP

4.13.3 Clock Crossings

Not applicable to this IP

4.13.4 VISA

Not applicable to this IP

4.13.5 Debug Registers

Not applicable to this IP

4.13.6 Scan – Clock Gating in RTL

Not applicable to this IP

4.13.7 Scan – Reset Override

Not applicable to this IP

4.13.8 Scan – Constraints and Coverage

Not applicable to this IP

4.13.9 TAP and Associated Registers

Not applicable to this IP

4.13.10 Boundary Scan Parameters

Not applicable to this IP

4.14 System Startup

4.14.1 Power-up Sequence

Features that are enabled by security can work only after `early_boot_exit` goes high.

4.14.2 Initialization Sequence

Drive SECURITY_UNLOCKED value on fdfx_secure_policy to start using all the features in pre-silicon.

4.14.3 Device Configuration

Not applicable to this IP

4.14.4 Header for Windows Boot

Not applicable to this IP

4.15 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

4.16 SAI Width

Not applicable to this IP

4.17 Integration Dependencies

Not applicable to this IP

4.17.1 "Ad Hoc" Pins

Name	I/O	Functionality
Dfxsecure_feature_en	Out	This is a legacy implementation where the decoded value of the fdfx_secure_policy enables each assigned DFx feature based on the policy assignment from the policy matrix.
Fdfx_powergood	In	Reset - powergood
Oem_secure_policy	In	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. Default: Set to 0x0 if not used.
Sb_policy_ovr_value	In	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 signal. If the sideband policy override feature is not used then connect sb_policy_ovr_value=policy_fivehex.
Visa_all_dis	Out	This output gets asserted when policy indicates that none of the features are enabled (VISA_BLACK Policy).
Visa_customer_dis	Out	This output gets asserted when fdfx_secure_policy is VISA_GREEN policy or VISA_BLACK policy.

4.17.2 Validation Requirements

Not applicable to this IP

4.17.3 Interface Signals Implemented for Security

Not applicable to this IP

4.18 RTL Design Libraries

Library	Version	Special Usage
CTECH	Ww45e	Not Applicable

4.19 RTL Uniquification

Not applicable to this IP

4.20 Emulation Support

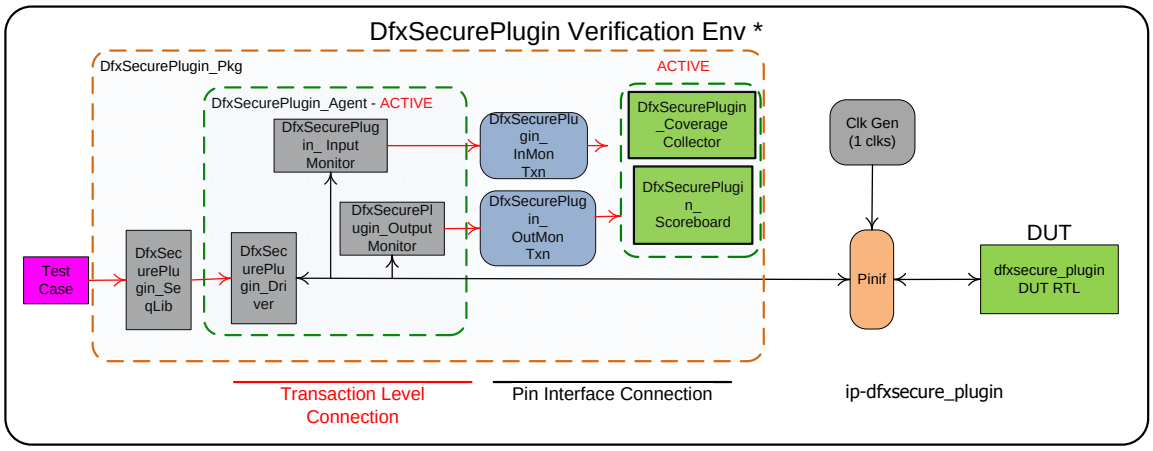
Emulation is located in `$IP_ROOT/tools/emulation/` folder.

5 Verification Information for Integration

5.1 IP Testbench Overview

The Verification ENV is in OVM methodology. The Testbench architecture is illustrated in Figure 3. 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 3. Unit-Level Testbench Block Diagram



5.2 Reusable IP Testbench Components

5.2.1 Test Island

The following describes the signal that needs to be connected at the SoC level.

Table 6. Test Island Connections

Signal	Connect to	Description
Primary_if	i_DfxSecurePlugin_TapTestIsland	Passes the DfxSecurePlugin pin interface to the instance of Test Island.

5.2.2 Collage or Sandbox Files

Not applicable to this IP

5.2.3 IP Environment

Table 7. Testbench Environment Files

List of File	Description
`DfxSecurePlugin_TestIsland.sv`	Sets the interface string.

List of File	Description
DfxSecurePlugin_Env.sv	Contains instances of other components of the Env (Monitors, Scoreboard, Coverage Collector, etc.).
DfxSecurePlugin_Agent.sv	Contains the Monitors, Driver and Sequencer.
DfxSecurePlugin_Driver.sv	Driver file that applies all of the stimuli to the DUT.
DfxSecurePlugin_Seqr.sv	Passes the stimulus packet from Sequence to Driver.
DfxSecurePlugin_InpMonitor.sv	Monitors the input pins going to DUT.
DfxSecurePlugin_OutMonitor.sv	Monitors the output pins going to DUT.
DfxSecurePlugin_OutMonTxn.sv	Contains the packet that to be sent to Scoreboard from the Output Monitor.
DfxSecurePlugin_InpMonTxn.sv	Contains the packet that to be sent to Scoreboard from the Input Monitor.
DfxSecurePlugin_TbDefines.svh	Contains the various parameters described in Section 4.4.
DfxSecurePlugin_pin_if.sv	Contains the definition of all pins that the Env drives and samples.
DfxSecurePlugin_Pkg.sv	Contains the consolidated Env files packaged into this IP.
DfxSecurePlugin_Scoreboard.sv	Models the DUT/RTL at transaction level.
DfxSecurePlugin_Coverage.sv	Contains the functional coverage.
DfxSecurePlugin_SeqDrvTxn.sv	Packet that goes between the Sequencer and Driver; the input constraints on the test are captured here.
DfxSecurePlugin_SeqLib.sv	Collection of all the sequences in the IP.
DfxSecurePlugin_Tbtop.sv	The Top of the Testbench, contains the instantiation of DUT, TestIsland, Program Block, and PinIf.

5.2.3.1 Configuring the IP Environment

In order to configure the IP Environment, follow the below steps:

- Instantiate the agent and pin interface in the `DfxSecurePlugin_Env.sv` file:

```
DfxSecurePlugin_Agent       i_DfxSecurePlugin_Agent;
```

  Set this below variable in build phase of Env.

```
set_config_int("stap_DfxSecurePlugin_Agent", "is_active", is_active);
stap_DfxSecurePlugin_Agent =
DfxSecurePlugin_Agent::type_id::create("stap_DfxSecurePlugin_Agent",this);

protected virtual DfxSecurePlugin_pin_if pins;
```

- Instantiate the pin interface in Tbtop. Pass the DfxSecurePlugin_pin_if to TestIsland in `Top.sv`.

```
DfxSecurePlugin_pin_if pif(soc_clock);
DUT_TestIsland i_DUT_TestIsland (pif, DUT_IF);
//-------------------------------------------------------------
//DFX SECURE PLUG IN PIN INTERFACE
//-------------------------------------------------------------
assign pif.fdfx_powergood       = 1;
assign `DUT_IF.tpsb_fdfx_secure_policy = pif.fdfx_secure_policy;
assign `DUT_IF.tpsb_fdfx_earlyboot_exit = pif.fdfx_earlyboot_exit;
assign `DUT_IF.tpsb_fdfx_policy_update  = pif.fdfx_policy_update;
```

- Instantiate vifcontainer and DfxSecurePlugin TestIsland in TestIsland and import DfxSecurePlugin_Pkg:

```
import DfxSecurePlugin_Pkg::*;
DfxSecurePlugin_VifContainer vif_container;
DfxSecurePlugin_TestIsland i_TapTestIsland(pif);

defparam i_TestIsland.DFXSECUREPLUGINVIF =
"ovm_test_top.Env.stap_DfxSecurePlugin_Agent.*";
```

Use the DfxSecurePlugin_VIF as a parameter for TestIsland Parameter.

Once the components are instantiated in Env, you need to write sequences to drive the tests. The sequences are discussed in section 5.2.4.

5.2.3.2 Saola Environment Walkthrough

There are no RAL.

5.2.3.3 Saola/RAL Components

Not applicable to this IP

5.2.3.4 System Manager

Not applicable to this IP

5.2.3.5 Fuse

Not applicable to this IP

5.2.4 Sequences

5.2.4.1 Sequence for Bringing up the IP

Not applicable to this IP

5.2.4.2 Initialization Sequences

The sequence DfxSecurePlugin_DrivePowerGoodSeq, available in `verif/tb/DfxSecurePlugin_SeqLib.sv`, can be used to get the DUT out of reset state.

5.2.4.3 BFM Sequences

Not applicable to this IP

5.2.4.4 IOSF Primary/Sideband BFM Sequences

Not applicable to this IP

5.2.4.5 Other Reusable Sequences

Not applicable to this IP

5.2.4.6 IP Test Sequences

Test Sequence Name	Parameters	Function
DfxSecurePlugin_BaseSeq	DFX_USE_SB_OVR DFX_SECURE_POLICY_MATRIX	Places the DUT in default mode, drives the fdfx_powergood reset asynchronously and drives the signals based on DFX_USE_SB_OVR. When DFX_USE_SB_OVR =0 ▪ fdfx_secure_policy = 'b0 ▪ sb_policy_ovr_value = 'b0 ▪ oem_secure_policy = 'b0 When DFX_USE_SB_OVR =1 ▪ fdfx_secure_policy = 'b0 ▪ sb_policy_ovr_value = random value (where the range depends on DFX_NUM_OF_FEATURES_TO_SECURE) ▪ oem_secure_policy = 0 to 15
DfxSecurePlugin_DrivePowerGoodSeq	Not Applicable	Issues a low to high transition on the fdfx_powergood signal.
DfxSecurePlugin_PolicySweepSeq	DFX_SECURE_POLICY_MATRIX	Places the DUT in DfxSecurePlugin_PolicySweepSeq mode. Drives the fdfx_powergood reset asynchronously and sweeps all 16 policies over fdfx_secure_policy irrespective of DFX_USE_SB_OVR. The driving of sb_policy_ovr_value and oem_secure_policy depends on DFX_USE_SB_OVR as follows: When DFX_USE_SB_OVR =0 ▪ sb_policy_ovr_value = 'b0 ▪ oem_secure_policy = 'b0 When DFX_USE_SB_OVR =1 ▪ sb_policy_ovr_value = random value (where the range depends on DFX_NUM_OF_FEATURES_TO_SECURE) ▪ oem_secure_policy = 0 to 15
DfxSecurePlugin_DriveUserPolicySeq	DFX_SECURE_POLICY_MATRIX	Places the DUT in DfxSecurePlugin_DriveUserPolicySeq mode. Drives the user defined policies over fdfx_secure_policy (ranging from 0 to 15). The driving of sb_policy_ovr_value and oem_secure_policy depends on DFX_USE_SB_OVR as follows: When DFX_USE_SB_OVR =0 ▪ sb_policy_ovr_value = 'b0 ▪ oem_secure_policy = 'b0 When DFX_USE_SB_OVR =1 ▪ sb_policy_ovr_value = random value (where the range depends on DFX_NUM_OF_FEATURES_TO_SECURE) ▪ oem_secure_policy =Strap Value.

Test Sequence Name	Parameters	Function
DfxSecurePlugin_DriveUserPolicyViaSBOVRSeq	DFX_USE_SB_OVR DFX_SECURE_POLICY_MATRIX	Places the DUT in DfxSecurePlugin_DriveUserPolicyViaSBOVRSeq mode and drives the signals based on DFX_USE_SB_OVR as follows: When DFX_USE_SB_OVR =0 ▪ fdfx_secure_policy = 'b0 ▪ sb_policy_ovr_value = 'b0 ▪ oem_secure_policy = 'b0 When DFX_USE_SB_OVR =1 ▪ fdfx_secure_policy = 'b0 ▪ sb_policy_ovr_value = random value (where the range depends on DFX_NUM_OF_FEATURES_TO_SECURE) ▪ oem_secure_policy = 0 to 15

5.2.4.7 SoC Requirements for Sequence Reuse

Not applicable to this IP

5.2.4.8 Sequence File Dependencies

All the sequences used for testing the Dfx Secure Plugin are stored at: `$IP_ROOT/verif/tb/DfxSecurePlugin_SeqLib.sv` file. The classes in this file extend from ovm_sequence.

5.2.5 Driver

The driver models the behavior of the Security Aggregator block in the SoC. It drives the signals as shown in Figure 4 and Figure 5.

Figure 4. Driver Timing (DFX_USE_SB_OVR = 0)

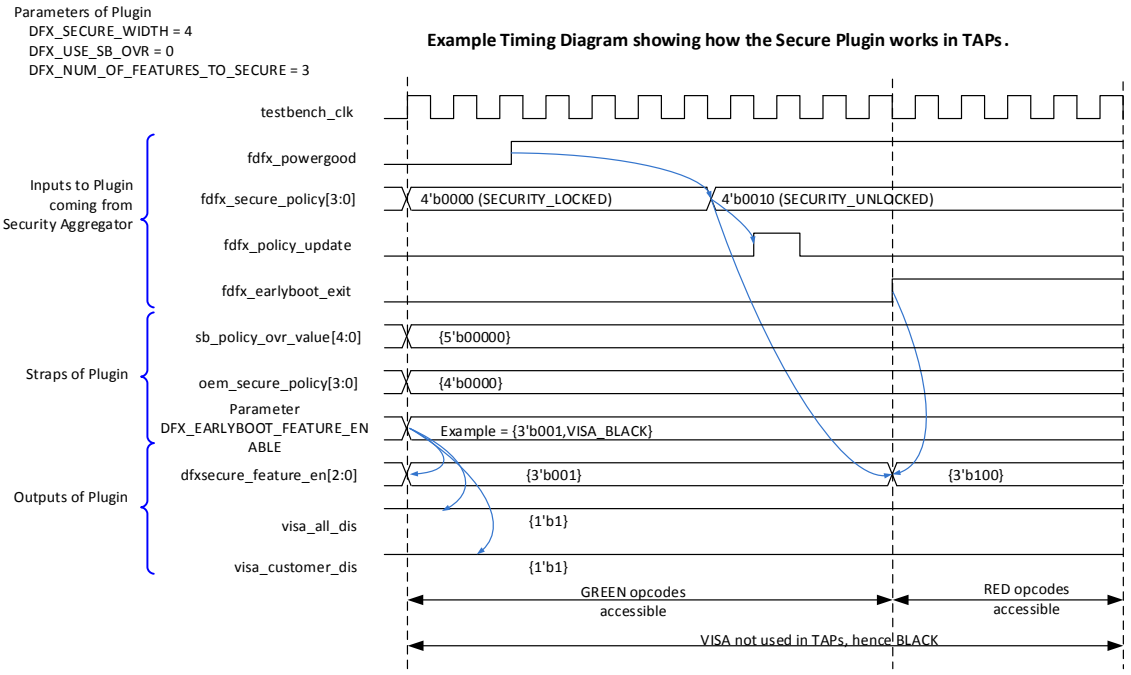
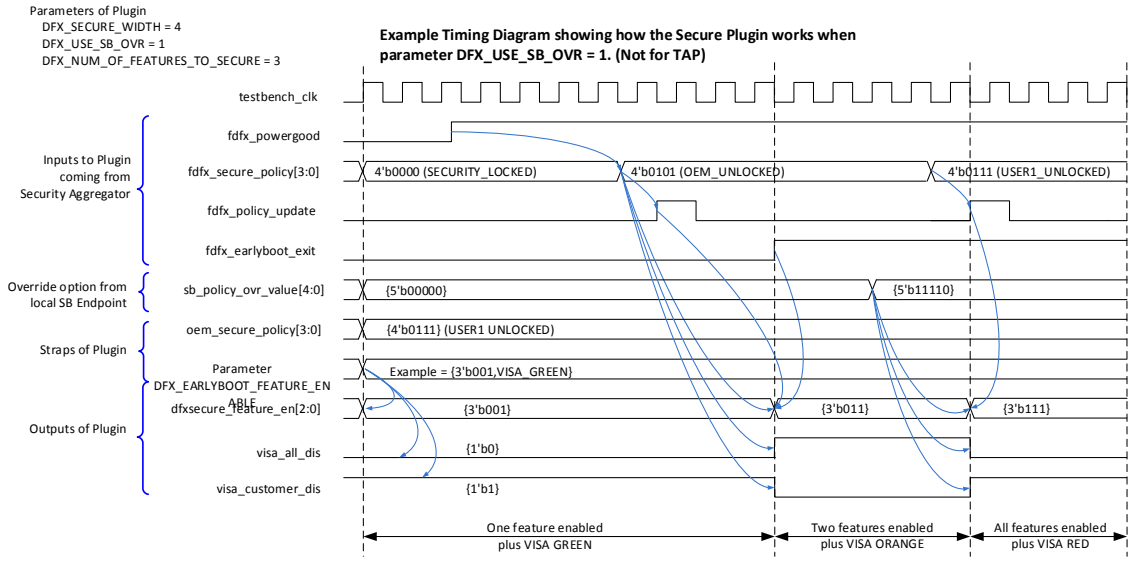


Figure 5. Driver Timing (FX_USE_SB_OVR = 1)



5.2.6 Miscellaneous

5.2.6.1 Using the Runtime or Post-Processing Checkers

For Bypass mode, the input and output monitors snoop the interface and constructs a packet that is sent on every cycle to Scoreboard. Then, checks for the packet integrity and flags an error if they do not match.

5.2.6.2 Environment Files

Refer to section 5.2.3 for more information.

5.2.6.3 Coverage

Code Coverage is measured with Line Coverage, Conditional Coverage, Branch Coverage, and FSM Coverage. Functional coverage is measured with data bins. Both Functional coverage and Code coverage are generated with using "urg" command.

Run command to get Coverage:

```
make run_script SCRIPT=source scripts/get_coverage
make run_script SCRIPT=source scripts/get_coverage_use1
```

5.3 IP-Level Information Required for Sequence Writing

There are no requirements for this IP's verification Env that can be used at integration level.

5.4 Environment Settings and Files

5.4.1 Base Test

The base test sets the following configuration variables in the environment:

```
set_config_int
("i_DfxSecurePlugin_Env.i_DfxSecurePlugin_Agent.i_DfxSecurePlugin_Seqr",
"count", 0);
set_config_int ("i_DfxSecurePlugin_Env.*", "is_active", OVM_ACTIVE);
set_config_int ("i_DfxSecurePlugin_Env*", "has_scoreboard", OVM_ACTIVE);
set_config_int ("i_DfxSecurePlugin_Env*", "has_cov_collector", OVM_ACTIVE);
```

5.4.2 Configuration Object

The base test sets the following configuration object in the environment:

```
i_DfxSecurePlugin_Env = DfxSecurePlugin_Env#()::type_id::create
("i_DfxSecurePlugin_Env", this);
```

5.4.3 API

Not applicable to this IP

5.4.4 Steps to Compile and Run Unit Tests

The following command runs for all configs:

```
make run_vcs_regress
```

5.5 Description of Reusable Tests

There are no reusable tests.

5.6 Description of Reusable Automation Scripts

Not applicable to this IP

5.7 Supported Compiler Options for Simulation

The below table summarizes the supported options.

Argument	Input
"-sverilog"	"-sverilog"
"-debug_all"	"-debug_all"
"-lca"	"-lca"
"+vcsd"	"+vcsd"
"+memcbk"	"+memcbk"
"+vpi"	"+vpi"
"-timescale"	"1ps/1ps"

5.8 Reusable Simulation RUNMODEs

Not applicable to this IP

5.9 RTL Verification Libraries

Not applicable to this IP

5.10 SoC-Specific Validation

Not applicable to this IP

6 Tools and Methodology for Integration

6.1 Supported Tools

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

- VCSMX
- OVM
- ACE
- Lintra
- Emulation
- Synthesis
- Conformal

6.2 Environment Variables

As per HDK flow, there will be no environment variables needed.

6.3 Directory Structure

```
|-- README.txt
|
|-- ace
|   `-- lib
|       `-- dfxsecure_plugin_model
|-- doc
|-- scripts
|   |-- ARN
|   |-- dfx_parameter_gui
|   |-- lib
|   `-- qa
|-- source
|   `-- rtl
|       |-- dfxsecure_plugin
|       |-- ctech_lib
|       |-- include
|           |-- assertions
|           `-- configs
|-- subIP
|-- tools
|   |-- cdc
|   |-- collage
|   |-- lec
|   |-- lint
|   |   `-- scripts
|   |-- emulation
|   `-- syn
|
`-- verif
    |-- lib
    |-- tb
    |   `-- include
    |       `-- configs
    |           |-- default
    |           |-- use0
    |           `-- use1
    `-- tests
```

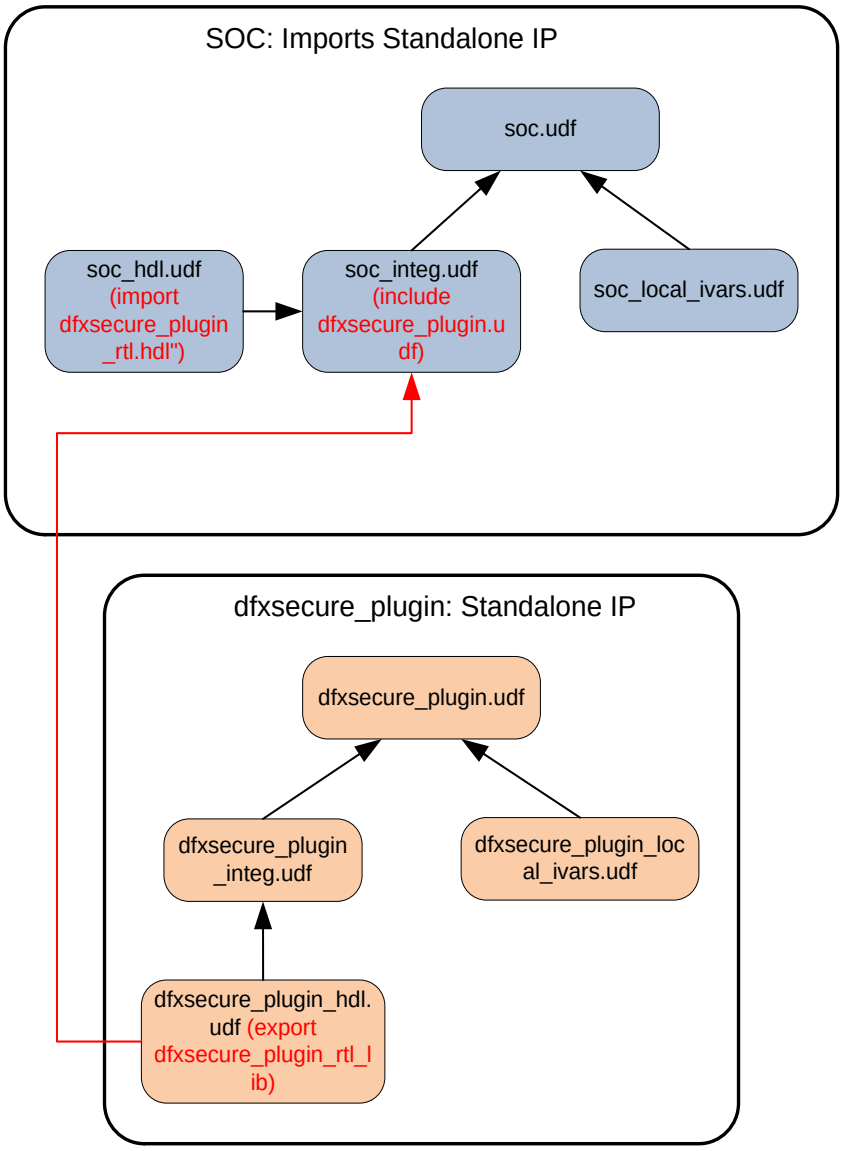
6.4 Ace

Paths to acerc: ace/<ip-name>.acerc

Location of udf file: `ace/<ip-name>.udf`

Required content in sip_shared_libs: Not Applicable

Figure 6. Integration Block Diagram



The SoC must import the dfxsecure_plugin UDFs in the Ace setup. Following are the steps to do this:

1. Place dfxsecure_plugin in the sub-IP directory of the project in which it will be used.
2. Define dfxsecure_plugin as a sub-scope in the `soc.acerc` file of the SoC. In this file, add the path of the dfxsecure_plugin IP to the search path of the SoC.

3. In `soc_integ.udf`, add dfxsecure_plugin_hdl.udf as an included UDF.
4. In `soc_hdl.udf`, import and use the dfxsecure_plugin_rtl_lib. Declare these as relevant dependent libs in the higher-level RTL libs where this module will be instantiated.
5. To check if the `dfxsecure_plugin_hdl.udf` shows up, and if the model of the SoC is clearly shown as the current scope, run the following command:

```
ace -show_models
```

6.5 Lintra

To run Lintra:

This is HDK based approach. HDK simbuild 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_compile CUST=<CUST_NAME>

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

#To run lintra compile
make run_sglint_compile CUST=<CUST_NAME>

#To run lintra elab
make run_sglint_elab CUST=<CUST_NAME>
```

6.6 Synthesis

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

```
make run_lint_dc_fv CUST=ADL
```

6.6.1 Clocks

The environment setup file and clock file names are available in the following directory:

`$IP_ROOT/tools/syn/inputs/<CUST>/`

Note: This is purely combinatorial design. The virtual clock "ref_clk" is used for the constraints.

Table 8. Virtual Clock

Sl. No.	Clock Name	Clock Period	Clock Waveform	Clock Source
1	ref_clk	10000 ns	{0 5000}	Not Applicable

Table 9. Primary Clocks

Not applicable to this IP

Table 10. Generated Clocks

Not applicable to this IP

6.6.2 Clock Diagram

There are no clock pins in the design.

6.6.3 Constraint Files

dfxsecure_plugin doesn't have any clocks. So, for all the outputs we need to give the following constraint.

set_max_delay delay_value [-from from_list] [-to to_list].

delay_value should be one destination clock period.

from_list = dfxsecure_feature_en, visa_all_dis, visa_customer_dis.

to_list = destination whoever is consuming.

Synthesis constraints are in the following file:

`$IP_ROOT/tools/syn/<project_name>/dfxsecure_plugin/scripts/dfxsecure_plugin_io_constraints.tcl`

6.6.4 BKMs

The following line will waive the error in the RDT2 flow.

-W- default_case_unreach .*dfxsecure_plugin.sv:184: DEFAULT branch of CASE statement cannot be reached. \(ELAB-311\)

6.6.5 Scan Insertion

Not applicable to this IP

6.6.6 Latches

There are four latches present in the design. Refer to the HAS document for more information.

6.7 Formal Verification

No special constraints are used.

6.8 CDC

Not applicable to this IP as there are no clocks in the design. However, it is run on Dfx Secure Clocks

```
make run_cdc CUST=ADL

For spyglass:
make run_sgcdc_comp CUST=ADL
make run_sgcdc_test CUST=ADL
```

6.9 SCAN

Not applicable to this IP

6.10 Emulation

Emulation is enabled based on ace flow.

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

6.11 Xprop

Xprop is enabled by default for this IP and can be run as follow.

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

6.12 Collage

This IP contains a CoreKit for Collage to aid in an SoC RTL integration. There is a single group of signals that are part of collage interface. The rest of the signals need to be connected at the SoC level (CoreAssembler). Core Kit is located at:

```
tools/collage/builder/builder.dfxsecure_plugin.tcl
```

```
To run collage:

make run_collage CUST=<CUST_NAME>
```

Table 11. Partial Collage Interface Port List

Interface Group	Remark
iosf_dfx_dfxsecure_plugin	Contains input ports dfx_secure_policy, fdfx_earlyboot_exit and fdfx_policy_update that are driven by the security aggregator.

7 Physical Integration

This chapter is intended to capture the aspect ratio requirements and any fixed size impact, etc., of memories that will be used in the IP. It is not intended to be "accurate" so much as an indication of what the impact and limitations might be. As this information will be based on the current memories, it would be only as accurate as the current design.

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 power draw limitations for array testing	Not Applicable

8 Integration Test Plan

8.1 Integration Considerations

The Scoreboard and Coverage Collector from this IP are not meant to be used at the SoC Level. Instead, the Test Island assertions and checkers cover this functionality.

1. Generate the parameters for your instance of DFx Secure Plugin (see section 4.6, “Generating Configuration Parameters”). If the name of your IP/cluster is "SOC", then the parameter generation tool will provide the following files.
   - dfxsecure_params_include.vh
   - dsp_tb_params.vh
   - dfxsecure_params_input.vh
   - SOC_dfxsecureplugin_param_values.vh
   - SOC_dfxsecureplugin_param_overide.vh
   - SOC_DfxSecurePlugin_TbDefines.svh
2. In the Top-level TestBench:
   - Include the following file (TestBench parameter defines) above the module definition of top-level testbench:

```
`include "DfxSecurePlugin_TbDefines.svh"

module SoCTopTb #(`DSP_TB_PARAMS_DECL)();
```

   - Instantiate DfxSecurePlugin_pin_if with the correct parameters clocks and pass it to the SoC Test Island:

```
DfxSecurePlugin_pin_if #(`DSP_TB_PARAMS_INST)
                        i_dfxsecure_pins(Primary_if.jtagbfm_clk));
```

   - Make the DfxSecurePlugin_pin_if available for SoCTestIsland:

```
SoCTestIsland #(`STAP_DSP_TB_PARAMS_INST)
              i_SoCTestIsland(

                              //Inputs
                              .pins       (i_dfxsecure_pins)
                               );
```

   - Create a defparam for VIF:

```
defparam i_SoCTestIsland.DSPVIF =
"ovm_test_top.Env.soc_DfxSecurePlugin_Agent.*";
……
endmodule
```

     The reference file will be available at:

```
verif/tb/DFxSecurePlugin_Tbtop.sv
```

3. In SoC Test Island:

- Include the parameter file above the module definition of SoCTestIsland:

```
`include "SOC_dfxsecureplugin_param_values.vh"
```

- Add defined parameter and pin interface of DfxSecurePlugin to SoCTestIsland:

```
module SoCTestIsland #( `DSP_TB_PARAMS_DECL,
                        parameter DfxSecurePlugin_VIF = "*"
                      )
                     (DfxSecurePlugin_pin_if pins);
```

  - Import DfxSecurePlugin_Pkg:

```
import DfxSecurePlugin_Pkg::*;
```

  - Instantiate DfxSecurePlugin_VifContainer:

```
DfxSecurePlugin_VifContainer            dfx_vif_container;
```

  - Create an object for vif container and set the config object:

```
set_config_object(DfxSecurePlugin_VIF, "V_DfxSecurePlugin_PIN_IF",
dfx_vif_container,0);
```

  - Instantiate DfxSecurePlugin_TestIsland:

```
DfxSecurePlugin_TestIsland
     #(`STAP_DSP_TB_PARAMS_INST,
       `include "STAP_dfxsecureplugin_param_overide_temp.vh")
      pri_DfxSecurePlugin_TestIsland(pins);
```

  - Create defparam for VIF:

```
defparam pri_DfxSecurePlugin_TestIsland.DfxSecurePlugin_VIF =
DfxSecurePlugin_VIF;
```

4. In the SoC Env:
   - Instantiate DfxSecurePlugin_Agent:

```
DfxSecurePlugin_Agent soc_DfxSecurePlugin_Agent;
```

   - Set config for DfxSecurePlugin_Agent:

```
set_config_int("soc_DfxSecurePlugin_Agent", "is_active", OVM_ACTIVE);
```

   - Create the agent:

```
soc_DfxSecurePlugin_Agent =
DfxSecurePlugin_Agent#()::type_id::create("soc_DfxSecurePlugin_Agent",this);
```

     The reference file will be available at:

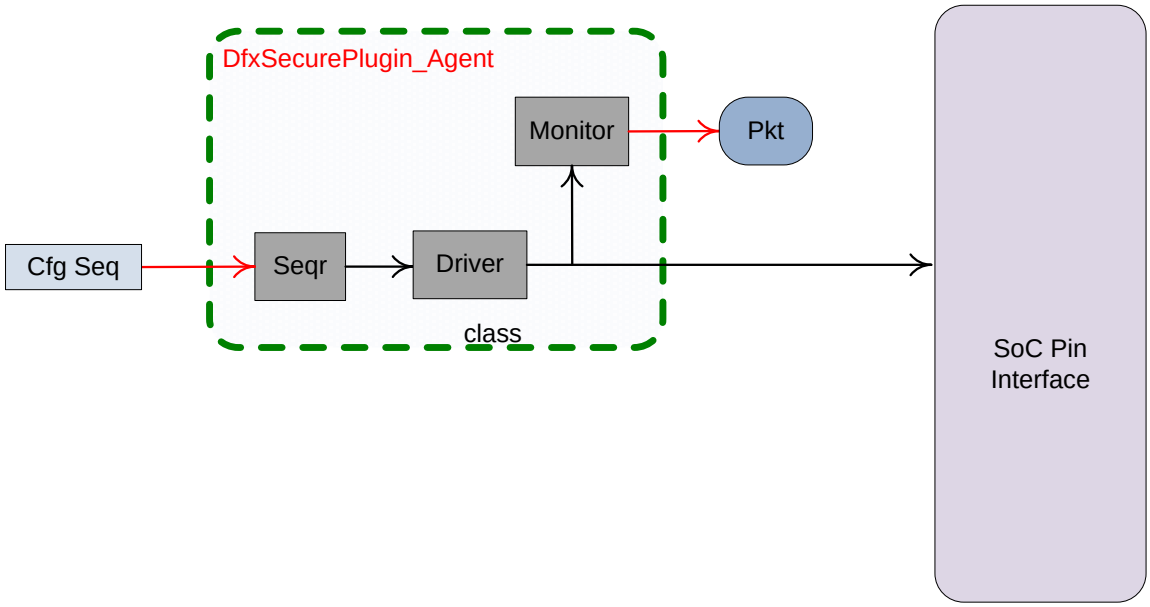
```
$IP_ROOT/verif/tb/DfxSecurePlugin_Env.sv
```

5. Write a new sequence that extends ovm_sequence and register to factory with DfxSecurePlugin_Pkg:: DriveDfxSecurePolicySeq.

8.2 Environment Usage

You can define and pass any policy, such as SECURITY_LOCKED, through the task DriveDfxSecurePolicy. This will drive the fdfx_secure_policy of dfxsecure_plugin with your defined policy value.

Figure 7. Integration Block Diagram of dfxsecure_plugin in an SoC



The code snippet of the SoC sequences to drive SECURITY_LOCKED is shown below:

```
class SecurityLocked extends DfxSecurePlugin_Pkg::DriveDfxSecurePolicySeq;

   //-----------------------------
   // Binding with the sequencer
   //-----------------------------
   `ovm_sequence_utils (SecurityLocked, DfxSecurePlugin_Pkg::DfxSecurePlugin_Seqr)

   //--------------
   // New Function
   //--------------
   function new (string name = "SecurityLocked");
      super.new (name);
   endfunction : new

   //---------------
   // OVM Body Task
   //---------------
   virtual task body ();
       DriveDfxSecurePolicy(SECURITY_LOCKED);
   endtask : body

endclass : SecurityLocked
```

The code snippet from a test in SoC Tests that drives the sequence is shown below:

```
SecurityLocked SL;
SL = new("Security Locked Mode");
SL.start(Env.soc_DfxSecurePlugin_Agent.i_DfxSecurePlugin_Seqr);
```

8.3 APIs

The below table describes the available APIs for controlling the DFx Secure Plugin stimulus.

Task Name	Usage	Remark
Powergood_Reset	Powergood_Reset;	Drives the fdfx_powergood signal to LOW and later one cycle drives it to HIGH for 10 clock cycles.
DriveFdfxEarlybootExit	DriveFdfxEarlybootExit(VALUE);	Drives the fdfx_earlyboot_exit signal to VALUE VALUE = 1'b1 or 1'b0
DriveFdfxPolicyUpdate	DriveFdfxPolicyUpdate(VALUE);	Drives the fdfx_policy_update signal to VALUE VALUE = 1'b1 or 1'b0
DriveSbPolicyOvrValue	DriveSbPolicyOvrValue (VALUE);	Drives the sb_policy_ovr_value signal to VALUE = {fdfx_feature_en, visa_all_dis, visa_customer_dis.} VALUE must be TB_DFX_SECURE_WIDTH+1 bits wide.
DriveFdfxSecurePolicy	DriveFdfxSecurePolicy (VALUE);	Drives the fdfx_secure_policy signal to VALUE. VALUE is TB_DFX_SECURE_WIDTH wide.
DriveOemSecurePolicyStrap	DriveOemSecurePolicyStrap (VALUE);	Drives the oem_secure_policy signal to VALUE. VALUE is TB_DFX_SECURE_WIDTH wide.
DriveFdfxEarlydebugStrap	DriveFdfxEarlydebugStrap (VALUE)	Drives a random value on fdfx_earlydebug_strap. VALUE must be TB_DFX_SECURE_WIDTH+1 bits wide.