*Document naming convention: VAL\_DCN\_”PCR\_Name”\_”PCR\_Number”*

PCR Title: OCS CPM Refactor

PCR url:

Review State

|  |  |
| --- | --- |
| Owner | Linda Chung |
| Reviewers |  |

Revision

|  |  |  |  |
| --- | --- | --- | --- |
| PCR Revision Number | Val DCN Revision Number | Description | Revision Date |
|  |  |  |  |
|  |  |  |  |

Contents

[1 Affected Database 1](#_Toc162271469)

[1.1 Areas Affected 1](#_Toc162271470)

[2 Brief Description of the PCR 1](#_Toc162271471)

[3 Verif A-Spec Content Changes 1](#_Toc162271472)

[3.1 DUT Summary 1](#_Toc162271473)

[3.2 Block diagram of DUT (RTL centric) 1](#_Toc162271474)

[3.3 Block diagram (Verification centric) 1](#_Toc162271475)

[3.4 Dimensions of configurability for DUT 1](#_Toc162271476)

[3.4.1 Objective: 1](#_Toc162271477)

[3.4.2 Flow 1](#_Toc162271478)

[3.5 Verification area of focus 1](#_Toc162271479)

[3.6 Component Hierarchy 1](#_Toc162271480)

[3.6.1 Features’ Environments 1](#_Toc162271481)

[3.6.2 BFMs 1](#_Toc162271482)

[3.6.3 Scoreboards 1](#_Toc162271483)

[3.6.4 Monitors 1](#_Toc162271484)

[3.6.5 Clock and Reset 1](#_Toc162271485)

[3.6.6 RAL modeling 1](#_Toc162271486)

[3.7 Test Bench 1](#_Toc162271487)

[3.8 Test Island 1](#_Toc162271488)

[3.9 Abstraction Layers 1](#_Toc162271489)

[3.10 Stimulus Randomness Control 1](#_Toc162271490)

[3.11 Checking Control 1](#_Toc162271491)

[3.12 Checking strategy 1](#_Toc162271492)

[3.12.1 Scoreboard 1](#_Toc162271493)

[3.12.2 Assertions 1](#_Toc162271494)

[3.12.3 Test-based self-checking 1](#_Toc162271495)

[3.12.4 Compliance monitors checking 1](#_Toc162271496)

[3.12.5 Register checking 1](#_Toc162271497)

[3.13 Coverage Strategy 1](#_Toc162271498)

[3.14 Exceptions to the Published Coding Guidelines 1](#_Toc162271499)

[3.15 Development Process 1](#_Toc162271500)

[3.16 External dependencies 1](#_Toc162271501)

[3.17 Directory Structure 1](#_Toc162271502)

[3.18 Flows / Ladder diagrams / pseudocode 1](#_Toc162271503)

[3.18.1 HECI Host sending message to CSE flow 1](#_Toc162271504)

[4 Test Plan Content Changes 1](#_Toc162271505)

[4.1 Test Cases 1](#_Toc162271506)

[4.1.1 Test Scenario *<Name of the test that covers this test scenario – Name to match Unix environment test name>* 1](#_Toc162271507)

[4.1.2 Test Scenario *<Name of the test that covers this test scenario – Name to match Unix environment test name>* 1](#_Toc162271508)

[4.2 Checklist 1](#_Toc162271509)

[4.2.1 Register Validation 1](#_Toc162271510)

[4.2.2 Feature Vs. Flow Matrix 1](#_Toc162271511)

[4.2.3 Feature Vs. Feature Matrix 1](#_Toc162271512)

[4.2.4 Error Scenarios 1](#_Toc162271513)

[4.2.5 Fuse and Soft straps 1](#_Toc162271514)

[4.2.6 DFX and VISA 1](#_Toc162271515)

[4.2.7 Ad hoc Signals 1](#_Toc162271516)

[4.2.8 Power Flows 1](#_Toc162271517)

[4.2.9 Reset Flows 1](#_Toc162271518)

[4.2.10 Performance Measurement 1](#_Toc162271519)

[4.2.11 Clock Frequency coverage 1](#_Toc162271520)

[4.2.12 Coverage on multi iteration of scenario/feature/flow 1](#_Toc162271521)

[4.2.13 Misc. 1](#_Toc162271522)

[4.2.14 ValPlan AI checklist 1](#_Toc162271523)

[5 Security Test Plan Content Changes 1](#_Toc162271524)

[5.1 S3 for SDLe Tool 1](#_Toc162271525)

[5.1.1 <SKS engine> 1](#_Toc162271526)

[5.1.2 Validation report that Captures Security Verification Result 1](#_Toc162271527)

[5.1.3 <HCU DMA> 1](#_Toc162271528)

[5.1.4 Security and Validation Plan 1](#_Toc162271529)

[5.1.5 Validation report that Captures Security Verification Result 1](#_Toc162271530)

[5.1.6 Multiple Area 1](#_Toc162271531)

[5.1.7 Security and Validation Plan 1](#_Toc162271532)

[5.1.8 Validation report that Captures Security Verification Result 1](#_Toc162271533)

[5.2 Additional Consideration for Security Assets not covered in S3 1](#_Toc162271534)

[5.2.1 PUF Key 1](#_Toc162271535)

[5.3 Additional Consideration for Security Checks not covered in S3 1](#_Toc162271536)

[5.3.1 Unused byte lane 1](#_Toc162271537)

[6 Appendix 1](#_Toc162271538)

[6.1 Implementation Notes 1](#_Toc162271539)

[6.2 Bottoms up Effort 1](#_Toc162271540)

[6.3 Process on how to update the val dcn document? 1](#_Toc162271541)

# Affected Database

## Areas Affected

OCS

# Brief Description of the PCR

Develop standalone essential block to interact with CSE IPs including OCS. Essential block is responsible for the power states, clocks and resets. The goal is to have CSE\_TOP as integration testing and all the CSE IPs can fully validate functionality including power transitions. Most of OCS features does not get affect with power transitioning. For example, DMA will gracefully end before OCS accept the power transition. However, selftest and PUF have state retention and save and restore registers that will get affected from power states. OCS can only accept the power transition and not deny. As for validation, we will have focus tests for power transition along with each feature cross with different power states.

# Verif A-Spec Content Changes

## DUT Summary

## Block diagram of DUT (RTL centric)



Below is the diagram of the state transition mapped out through Pstate taken from the Essential Refactor Cluster Aspec 0p5 document.



PSTATE description and valid response but OCS can only accept all PSTATE:

|  |  |  |  |
| --- | --- | --- | --- |
| **PSTATE** | **State Description** | **Transition Actions** | **Valid Device Response** |
| **OFF** | Off State.  Security Engine is non-operational in an IP Inaccessible state (either cold/global reset, InAcc PG, or VNN Removal after Accessible PG occurred)  PACTIVE\_\*WAKEs from devices may be ignored  This is the default state of the subsystem | Request all devices to continue locking the external interface.  During P\_REQUEST window, all devices must turn off their SRAM, enable their CFI ISM to enter IDLE state, and then accept this PREQ. | Accept only |
| **ACTV** | Active State.  This is the normal operational state of the subsystem. The CFI ISMs will never be idle, all SRAMs are powered on, and external interfaces are unlocked (barring specific flows/configurations) | During P\_REQUEST window, all devices must prevent their CFI ISM to enter IDLE and then accept the PREQ. | Accept only |
| **RESTORE** | PG Restore  This is transitory state which indicates all context has been restored and CSE is ready to resume normal operation | Request all devices to continue locking their external interface.  During the P\_COMPLETE window, devices will perform a restore operation | Accept only |
| **ACCBLK** | IP-Accessible Blocking  All subIPs have locked their external interface and are prepared to enter IP-Accesible PG    Devices are permitted to upse PACTIVE\_\*WAKE to initiate wake from ACCBLK and return to ACTV | Request all devices to lock external interfaces. LMT does not need to be in the C2 state. Devices are permitted to decline this PREQ if they are unable to lock external interfaces immediately | Accept or  Deny |
| **ACCPG** | IP-Accessible PG  CSE is committed to entering IP-Accessible PG | Request all devices to continue locking the external interface.  All devices must save their context into HW Save and Restore Buffer, turning off their SRAM, enable their CFI ISM enter IDLE state, and then accept this PREQ.  Device is permitted to issue PACTIVE\_\*WAKE to request Essential to exit from ACCPG immediately. | Accept only |
| **INACCBLK** | IP-Inaccessible Blocking  Upon reaching this state, all subIPs have locked their external interface and are prepared to enter IP-Accesible PG  Devices are permitted to upse PACTIVE\_\*WAKE to initiate wake from ACCBLK and return to ACTV | Request all devices to continue locking the external interface.  All devices must save their context into HW Save and Restore Buffer, turning off their SRAM, enable their CFI ISM enter IDLE state, and then accept this PREQ.  PACTIVE\_\*WAKE from devices will be ignored | Accept or  Deny |
| **WRMRSTSRMOFF** | Warm Reset w/SRAM Off  Subsystem is prepared to enter Warm Reset. Device external interface remains locked, their CFI ISM is enabled to enter IDLE, and their SRAM is powered OFF. | Request all devices to continue locking the external interface  All devices must turn off their SRAM, enable their CFI ISM to enter IDLE state, and then accept this PREQ.  The PACTIVE\_\*WAKE from devcies will be ignored. | Accept only |
| **WRMRSTSRMON** | Warm Reset w/SRAM On  Subsystem is prepared to enter Warm Reset. Device external interface remains locked, their CFI ISM is enabled to enter IDLE, but their SRAM must remain powered ON. | Request all devices to continue locking the external interface  All devices must turn off their SRAM, enable their CFI ISM to enter IDLE state, and then accept this PREQ.  The PACTIVE\_\*WAKE from devcies will be ignored. | Accept only |
| **GLBLK** | Global Blocking  All subIPs have locked their external interface. LMT is in C2 state. Subsystem is preprared to enter TCG state. | Request all devices to lock external interface and have LMT in C2 state. Devices are permitted to decline this PREQ, if they are unable to lock external interface or have LMT in C2 state.  Once Device accepted the PREQ, they are permitted to use PACTIVE\_\*WAKE to initiate a wake from GLBLK. | Accept or  Deny |
| **TCG** | Func Clock Trunk Clock Gating (TCG)  Subsystem has deasserted the slowfast\_clkreq to SOC, thus fast/slow\_clk may be turned off at the trunk level. All devices continue to lock their external interface, LMT remains in C2 state. | Request all devices to continue locking external interface and have LMT in C2 state. Devices with stepping stone logic to manage the synchronous fast/slow\_clk crossing must disable it at this time.  Once Device accepted the PREQ, they are permitted to use PACTIVE\_\*WAKE to initiate a wake from TCG. | Accept only |
| **TCGEXITPREP** | TCG Exit Prep  Slowfast\_clkreq has been asserted and fast/slow\_clk is running.  Any stepping stones have been re-enabled and running correctly.  Subsystem is ready to re-enter ACTV state. | Request all devices to continue locking external interface and have LMT in C2 state, but to re-enable their stepping stone logic. Devices with stepping stones shall re-enable the stepping stone in the P\_REQUEST window then accept this PREQ. | Accept only |
| **CSERST** | Device Reset Isolation  All subIPs have quiesed all traffic and are prepared for a CSE Reset event (either CSE partition reset, or CSE internal reset). | Request all devices to stop propagating requests to internal CSE Fabric.  During the P\_REQUEST window, all devices must gracefully stop propagating request to internal CSE fabric, and then accept this PREQ.  Once the PREQ is de-asserted, during P\_COMPLETE window, it requests all devices to stop propagating requests to internal CSE Fabric, and ensure that all the request from internal CSE fabric has been gracefully terminiated or has been propagated to external interface.  All devices must gracefully drain all the requests from internal CSE fabric, and then de-assert its PACCEPT  In summary, during P\_REQUEST window, devices perform Device Isolation, and during P\_COMPLETE window, devices perform Bridge Isolation | Accept only |

## Block diagram (Verification centric)

A diagram of a computer program

AI-generated content may be incorrect.

## Dimensions of configurability for DUT

*Author requirement: Summarize on how your IP achieves configurable testbench.*

*Below is an summary of “Configuring a Test Environment” section from UVM cookbook with an example from cse sub-ip.*

### Objective:

*One of the key tenets of designing reusable testbenches is to make testbenches as configurable as possible. Doing this means that the testbench and its constituent parts can easily be reused and quickly modified (i.e. reconfigured) according to the DUT. This is achieved by following below steps.*

*For more information on Testbench Configuration and code examples, refer to “Configuring a Test Environment” chapter in UVM cookbook (*[*https://verificationacademy.com/cookbook/uvm*](https://verificationacademy.com/cookbook/uvm)*) (verificationacademy.com Account may need to be created to access the cookbook)*

### Flow

*Below process flow illustrates on how DUT configuration values are retrieved, stored, and distributed on the testbench side of the environment.*

1. *“Parameter package” contains named parameters with associated values to be shared by both the HDL/DUT side and testbench side of the environment.*

*Example: /subBlock/cse/units/cse/cfmia/ace/lib/csme\_mtl\_s/cfmia.json*

1. *Class based configuration object is implemented using Template Toolkit to encapsulate all related configuration variables for a DUT. configuration object variable values are retrieved using hierarchy reference to parameter package.*

*Example: /subBlock/cse/units/cse/cfmia/validation/cfmia/testbench/templates/cfmia\_cfg.sv.template*

1. *Test makes an instance of the configuration object and passes it on to the environment class.*
2. *Any object or component that has a handle to the environment class can access the variables and methods in the configuration object.*

## Verification area of focus

## Component Hierarchy

*Author requirement: Include path (file system path), and overview of parent environment. Include main components instantiated such as:*

* *Features’ environments*
* *BFMs*
* *List of all scoreboards along a brief description*
* *List of all monitors along a brief description*
* *Clock and reset modeling architecture*

*Also include a path and overview on base test. Include main common components such as background traffic generator(s).*

### Features’ Environments

*Author requirement: Include list of any features’ environments instantiated in the main IP env.*

### BFMs

*Author requirement: Include list of BFMs used in your IP. Include any treatment in your environment to switch between different ones, if applicable.*

### Scoreboards

P-channel scoreboard is responsible for checking the pactive signal correctness. The actual pactive signal is taken from the P-channel monitor which is monitoring the RTL DUT signals. The expected will be generated based on the idleness of each engine. Table below shows the event variable that can be used to indicate that the engines are busy and this includes the OCP fabric which has registers transcaction. If OCS is busy, then PACTIVE.PACTIVE\_FUNC should be 1. Scoreboard will not be clock cycle accuracy. There will be some buffer and the checking will confirm that it is correct within that window.

|  |  |
| --- | --- |
| **OCS Engine** | **Indicator used to know if it is busy** |
| DMA (AES\_A, AES\_P, HCU, GPDMA, RC4 ) | dma\_on\_e and dma\_off\_e event detected for each DMA |
| SKS |  |
| EAU | eau\_on\_e and eau\_off\_e event detected |
| ECC | ecc\_on\_e and ecc\_off\_e event detected |
| XMSS/LMS | xmss\_active variable |
| TRNG |  |
| PUF | ?? waiting for RTL to generate puf key to indicate puf is done (puf\_key\_valid\_e) |
| SELF\_TEST | ?? scoreboard does not calculate if it is done or not though |

Purpose of the scoreboard to confirm that the PACTIVE.PACTIVE\_FUNC is set correctly and predict whether or not the RTL will send the PACCEPT or PDENY. PDENY is only valid for GLBLK, ACCBLK, and INACCBLK state.

|  |  |  |
| --- | --- | --- |
| **Device Activity Indicator** | | |
| PACTIVE[6:0] | D2E | Each bit of PACTIVE indicates something regarding the activity of the device. Bits 0 and 1 are asynchronous wake events. Bits 2 through 6 indicate transactions pending/ongoing activity in a given clock domain. For bits 2 through 6, if device does not contain logic in that clock domain, the device shall tie that bit to 1’b0.  6: PACTIVE\_PRIVEPSIDE - Device's hint to indicate there is a transaction pending in priv\_side\_clk domain. This signal is synchronous to priv\_side\_clk domain.  5: PACTIVE\_LLEPSIDE - Device's hint to indicate there is a transaction pending in llep\_side\_clk domain. This signal is synchronous to llep\_side\_clk domain.  4: PACTIVE\_GPSIDE - Device's hint to indicate there is a transaction pending in side\_clk domain. This signal is synchronous to side\_clk domain.  3: PACTIVE\_PRIM - Device's hint to indicate there is a transaction pending in prim\_clk domain. This signal is synchronous to prim\_clk domain.  2: PACTIVE\_FUNC - Device's hint to indicate there is a transaction pending in fast/slow\_clk domain. This signal is synchronous to slow\_clk domain.  1: PACTIVE\_SPURIOUS\_WAKE – Device’s request to exit from low power state caused by a Spurious Wake source (see [HAS](https://docs.intel.com/documents/Security_IP/HAS/COMMON/IP%20HAS%20Chapters/ChapCSME02%20CSE/ChapCSME02%20CSE.html#spurious-power-wake-handling)). This signal can be asserted asynchronously, even when the clock is not running. Once asserted, it needs to remain asserted until CSE is in the OFF, ACTV, or TCGEXITPREP states.  0: PACTIVE\_FW\_WAKE - Device's request to exit from low power state caused by a FW Power Wake source (see [HAS](https://docs.intel.com/documents/Security_IP/HAS/COMMON/IP%20HAS%20Chapters/ChapCSME02%20CSE/ChapCSME02%20CSE.html#spurious-power-wake-handling)). This signal can be asserted asynchronusly, even when the clock is not running. Once asserted, it need to remain asserted until CSE is in the OFF, ACTV, or TCGEXITPREP states. |

Scoreboard to log the timeout signal assertion/deassertion timestamp. When a REQ comes in and the timestamp is within the timeout signal then expect a PACCEPT and not a PDENY. If the REQ is outside the timeout signal range, then clear out the timeout\_info variable.

Associated Array: timeout\_info = [“assert” : timestamp, “deassert” : timestamp]

For existing scoreboard, PUF and selftest scoreboard needs to be enhanced to understand the PSTATE transition. Selftest scoreboard needs to store the register values that are state retained and when there is a power transition to ACCPG, then add a check to make sure particular fields are correct upon exiting of ACCPG and going back to ACTV. Previously, this is done at CSE\_TOP with a focus test. Now, the retention is done at OCS level because OCS will have its own UPF. Also, need to do those field for save and restore flow.

As for PUF scoreboard, we need to make sure the PUF key is always retained through all power flow because the key is store in the AON domain.

The save and restore scoreboard will need to save the content when RTL issues the save write requests and restore the content when it sees RTL issues the restore read requests. RTL will set the address it wants to write/read to and the scoreboard will create a snoop memory area to keep track of all those transactions in order to reply to the read requests RTL issue during RESTORE state. The content will get used for functional testing as well. For example, selftest indicators that decide if selftest will rerun, pass, or fail.

### Monitors

P-channel monitor monitors all p-channel interface siganls and broadcast them through analysis port.

We will be using the IOSF BFM export port to monitor downstream/upstream traffic in order to validate the pactive signal

Save and restore monitor monitors all its interface signals to create a save and restore snoop memory for the predictor in the scoreboard and drive the input signals to DUT.

### Clock and Reset

*Author requirement: Include description on how your IP models clocks and reset. Additionally, the author is recommended to detail here, the strategy used to send reset information to components such as scoreboards, to aid in accurate reference-modeling of the DUT.*

Drive side\_rst (used for factory signing driven before cse\_rst and for all beside TCG and cserst) and powergood\_rst (only in OFF stage) accordingly. Same as baseline and driven at TOP level (find out which configurations)

Factory signing with side reset with FW reads/writes through APB but it depends on Dfx so the testing is still within TOP level.

### RAL modeling

*Author requirement:*

* *Describe implementation choice of implicit prediction or explicit prediction.*
* *Treatment of /V and lock attributes, etc.*
* *HDL paths checking, including documentation of each exception case.*

## Test Bench

*Author requirement: Include path to tb file, in which DUT is instantiated. Also include memory model if any. Describe UPF switching DUT instantiating hierarchy, if applicable. Describe the collection of tied-off signals with justification.*

There is no special UPF validation beside testbench driving the pfet and other signals to connect UPF correctly. UPF is used for state retention and we will validate the registers through testing and scoreboard for selftest.

Create clocking block for all the save and restore signals within the interface file. Drive the output signals within the test island through always\_comb block.

## Test Island

Include P-Channel interface to add in test island and rename the reset and clock signal.

|  |  |  |
| --- | --- | --- |
| **Signal Name** | **Direction** | **Description** |
| slow\_vnnpgd\_gclk | Input | Vnnpg version of slow clock |
| cse\_rst\_vnnpgd\_slow\_b | Input | Vnnpgd version cse reset |
| cse\_side\_rst\_b | Input |  |
| powergood\_rst\_b | Input |  |
| ess\_ocs\_pchnl\_pstate[M-1:0] | Intput | The power state to which ESS is requesting the device to transition to |
| ess\_ocs\_pchnl\_preq | Input | Active High Request to transtion to the power state indicated by PSTATE |
| ocs\_ess\_pchnl\_paccept | Output | Active High Accept of the PREQ, indicating the device will transition to the state indicated by PSTATE |
| ocs\_ess\_pchnl\_pdeny | Output | Active High Deny of the PREQ, indicating the device will not transition to the state indicated by PSTATE, and will remain in its current state |
| ocs\_ess\_pchnl\_pactive[6:0] | Output | Each bit of PACTIVE indicates something regarding the activity of the device. Bits 0 and 1 are asynchronous wake events. Bits 2 through 6 indicate transactions pending/ongoing activity in a given clock domain. For bits 2 through 6, if device does not contain logic in that clock domain, the device shall tie that bit to 1’b0.  6: PACTIVE\_PRIVEPSIDE - Device's hint to indicate there is a transaction pending in priv\_side\_clk domain. This signal is synchronous to priv\_side\_clk domain.  5: PACTIVE\_LLEPSIDE - Device's hint to indicate there is a transaction pending in llep\_side\_clk domain. This signal is synchronous to llep\_side\_clk domain.  4: PACTIVE\_GPSIDE - Device's hint to indicate there is a transaction pending in side\_clk domain. This signal is synchronous to side\_clk domain.  3: PACTIVE\_PRIM - Device's hint to indicate there is a transaction pending in prim\_clk domain. This signal is synchronous to prim\_clk domain.  2: PACTIVE\_FUNC - Device's hint to indicate there is a transaction pending in fast/slow\_clk domain. This signal is synchronous to slow\_clk domain.  1: PACTIVE\_SPURIOUS\_WAKE – Device’s request to exit from low power state caused by a Spurious Wake source (see [HAS](https://docs.intel.com/documents/Security_IP/HAS/COMMON/IP%20HAS%20Chapters/ChapCSME02%20CSE/ChapCSME02%20CSE.html#spurious-power-wake-handling)). This signal can be asserted asynchronously, even when the clock is not running. Once asserted, it needs to remain asserted until CSE is in the OFF, ACTV, or TCGEXITPREP states.  0: PACTIVE\_FW\_WAKE - Device's request to exit from low power state caused by a FW Power Wake source (see [HAS](https://docs.intel.com/documents/Security_IP/HAS/COMMON/IP%20HAS%20Chapters/ChapCSME02%20CSE/ChapCSME02%20CSE.html#spurious-power-wake-handling)). This signal can be asserted asynchronusly, even when the clock is not running. Once asserted, it need to remain asserted until CSE is in the OFF, ACTV, or TCGEXITPREP states. |
| ess\_ocs\_pchnl\_timeout | Input | P-Channel Timeout  Indicates a timeout for a given flow has occurred and devices *must* respond immediately.  This signal can assert prior to a PREQ being sent, with the expectation that the device will accept in a timely fashion  This signal can assert after PREQ has been asserted, but before all devices have responded. In the event that it asserts during an active handshake, any device that has not responded must accept immediately.  See section 1.4.4 for more details |

Add save and restore interface to test island:

|  |  |  |
| --- | --- | --- |
| **Signal Name** | **Direction** | **Description** |
| ocs\_gsk\_snr\_put | Out | SNR Command Put. Devices track an internal credit count and may only issue an snr\_put if it has credits available. Devices must decrement their internal credit count after each snr\_put. |
| ocs\_gsk\_snr\_addr[9:0] | Out | SNR Address. 0-based.  Each device should start at address “0”. |
| ocs\_gsk\_snr\_wxrb | Out | SNR Command Write/Read indication 1- write (save) 0 - read (restore) |
| ocs\_gsk\_snr\_wdata[63:0] | Out | SNR Command Write Data (Save) |
| gsk\_ocs\_snr\_rdata[63:0] | In | Read data for Restore Operation  Returned in order of read requests. Valid only during snr\_cpl |
| gsk\_ocs\_snr\_rdata\_cpl | In | Completion indicator. 1-clock valid signal (each clock valid indicates a single completion).  For reads, the snr\_rdata is valid in the cycle snr\_cpl is asserted.  For writes, it indicates the SNR has been delivered to the AON SNR Buffer.  Devices must only receive a snr\_cpl after issuing an snr\_put. There is a 1-to-1 relationship with snr\_put and snr\_cpl.  The snr\_cpl also operates as a credit return. Devices will increment their internal credit count upon receiving snr\_cpl. |

typedef enum logic [3:0] {

OFF = 4'h0,

ACTV = 4'h1,

RESTORE = 4'h2,

ACCBLK = 4'h3,

ACCPG = 4'h4,

INACCBLK = 4'h5,

WRMRSTSRMOFF = 4'h6,

WRMRSTSRMON = 4'h7,

GLBLK = 4'h8,

TCG = 4'h9,

TCGEXITPREP = 4'hA,

CSERST = 4'hB

} pchnl\_pstate\_t;

## Abstraction Layers

*Author requirement: detail, through additional sub-sections, the available abstraction layers used within the IP environment (including sequences.) Examples that warrant descriptions here, include the use of Saola’s “Interrupt Manager”, “System Manager” (Saola’s memory-management abstraction which plays the role of the C language’s malloc() and free() APIs), “Fuse Abstraction Layer”, as well as IP-specific abstraction layers, if any significant developments are anticipated/proposed or already exist. The goal of this section is twofold: to provide an overview of how the IP’s validation collaterals intend to use the abstraction layer (such as partially, or only for one specific use-case, or heavily in a variety of different ways, all summarized), to help reviewers provide feedback on potential gotchas or hazards, and so on. The other goal is to serve as general documentation to benefit new IP team members, and to help when the IP team plans to migrate from an old foundation (e.g. OVM) to a newer one (e.g. UVM.) Each of these abstraction layers that are planned to be used, or are actually used, should be described in a corresponding sub-section named after that abstraction layer. In general, the author is reminded that abstraction layers represent a double-edged sword: they may help in the short term, but if not approached carefully, over the long term timeframe, reliance on abstraction layers incur nontrivial migration and maintenance costs. When using an abstraction, it is strongly encouraged that IP validation collaterals should minimize their use of each abstraction, and to use the abstraction in a way that facilitates migrating from one abstraction implementation, to the next. Relevant examples include migrating from Saola’s “System Manager” memory-management facilities, to UVM’s very different facilities; similarly, Saola’s RAL to UVM’s RAL. Documenting the strategy employed when approaching the use of such abstractions here, helps to force the author to keep these points in mind, and to document requirements and reasons for using such facilities.*

## Stimulus Randomness Control

*Author requirement: Include a brief table to describe how to silence certain parts of the environment randomness. Example is a knob to guard BFM randomization. Since some of the collaterals may have been listed in different parts of this document, feel free to reference to the appropriate section for the particular knob.*

|  |  |
| --- | --- |
| *Knob* | *Description* |
| *pvc\_rand\_en* | *Turn on PVC randomization available in the BFM* |
|  |  |

## Checking Control

*Author requirement: Summarize the mechanism to enable/disable various components (i.e. scoreboards, reg predictors etc) and briefly describe the implications of doing so. Please feel free to provide reference to specific section within this document for the detailed description of the component.*

*Example:*

|  |  |  |  |
| --- | --- | --- | --- |
| *Field name* | *Instance name* | *Default*  *val* | *Description*  *(Please refer to* [*section 7.1*](#_Component_Hierarchy) *for further details on a specific scoreboard component)* |
| *gsk\_component\_en* | *\*Gsk\_env.Gsk\_agents\_top.IntPsf2SB* | *1* | *This knob when set, enables the scoreboard that checks mIA initiated traffic targeted to go out over sideband interface (i.e. SB ATT, LTR). When cleared, the scoreboard is disabled.* |
| *Enable* | *\*Gsk\_env.spiral\_env.Gsk\_ebb\_comparator* | *1* | *This knob when set, enables Spiral/boot-guard scoreboard. When cleared, the scoreboard is disabled.* |

*Below are some examples of setting these knobs using OVM/UVM “set\_config\_int” API call. The user is expected to use this API in the build function of their respective test/top level env.*

*set\_config\_int("\*Gsk\_env.Gsk\_agents\_top.IntPSF2SB","gsk\_component\_en", 0) //Disabling mIAàSB traffic scoreboarding*

*In case the user wants to disable ALL the scoreboards (this may be needed while writing a self-checking focused test for error scenario testing etc), below is the mechanism to achieve it (using “\*” wildcard)*

*set\_config\_int("\*","gsk\_component\_en", 0) //Disable all gasket scoreboards*

*\*Cautionary Note: The existing Gasket validation infrastructure has a known limitation of not being able to disable a scoreboard component completely through above mechanism, certain portions of the component may still be active and spit out error messages. Similarly, many of the gasket scoreboard components also perform register modeling (i.e. RAL shadow copy update) which would not be impacted by disabling the respective scoreboard. So, please be mindful to check if is this matching your intention before updating the settings.*

## Checking strategy

Author requirement: Call attention to any internal signals/interfaces/bridges used as observation points

### Scoreboard

Scoreboard will be connected to the PChannel monitor and save and restore monitor.

### Assertions

Write assertions to check the follow PChannel requirements:

1. PREQ can only assert when both PACCEPT and PDENY are both deasserted
2. PACCEPT can only assert when PREQ is asserted.
3. PDENY can only assert when PREQ is asserted.
4. PACCEPT and PDENY must never both be asserted at the same time
5. PREQ can only deassert when either PACCEPT or PDENY is asserted
6. PACCEPT and PDENY can only deassert when PREQ is deasserted.
7. PSTATE must remain stable while PREQ is asserted

### Test-based self-checking

The ocs\_illegal\_power\_flow\_test will have self-checking to make sure that RTL always accept the PSTATE transition.

### Compliance monitors checking

N/A

### Register checking

N/A

## Coverage Strategy

We will collect CSpec coverage inside scoreboard. We will have coverage for all the interface signals. There are no new register introduced from this DCN. We need to cover all PSTATE.

## Exceptions to the Published Coding Guidelines

## Development Process

Author requirement: Note if there is a plan to use a unit-level testing like the vunit. Or using emacs capability to stitch top-level modules, instead of collage. Other option is just hand-crafted with no special tools.

## External dependencies

Author requirement: Such as libraries, Saola

## Directory Structure

## Flows / Ladder diagrams / pseudocode

Author requirement: Depict understanding of all generic flows. Example below

### HECI Host sending message to CSE flow

The below diagram shows basic HECI Host sending message to CSE flow.

(embed Wikipedia page for ladder diagram guidelines)



# Test Plan Content Changes

## Test Cases

### ocs\_basic\_power\_flow\_test

|  |
| --- |
| Objective  Test will transition from Cold Boot to Accessible PG Entry/Exit to Inaccessible PG Entry/Exit |
| Description   1. Boot up sequence (ocs\_hard\_reset\_sequence)    1. assert cse\_rst\_vnnpgd\_slow\_b and cse\_side\_rst\_b    2. Fork:       1. random delay , then deassert cse\_side\_rst\_b       2. random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACTV    3. random delay, then deassert cse\_rst\_vnnpgd\_slow\_b    4. 50% of time call init sequence to set up OCS BAR to see what happens because OCS shouldn’t respond to this until h.    5. wait for RTL to set ocs\_ess\_paccept = 1    6. random delay, then set ess\_ocs\_preq = 0    7. wait for RTL to set ocs\_ess\_paccept = 0 2. Call ocs initialization sequence if it was not called in 1e. 3. Call accessible PG entry sequence (TODO: Save operation needs to be done)    1. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACCBLK    2. Wait for RTL to set ocs\_ess\_paccept = 1    3. Random delay, then set ess\_ocs\_preq = 0    4. Wait for RTL to set ocs\_ess\_paccept = 0    5. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACCPG    6. Wait for RTL to set ocs\_ess\_paccept = 1    7. Random delay, then set ess\_ocs\_preq = 0    8. Wait for RTL to set ocs\_ess\_paccept = 0    9. Random delay, assert cse\_rst\_vnnpgd\_slow\_b    10. Random delay, assert the cse\_side\_rst\_b 4. Call accessible PG exit sequence (TODO: Check for selftest register retention for OCS UPF testing )    1. Fork:       1. Random delay, then deassert cse\_side\_rst\_b       2. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACTV    2. Random delay, then deassert cse\_rst\_vnnpgd\_slow\_b    3. Wait for RTL to set ocs\_ess\_paccept = 1    4. Random delay, then set ess\_ocs\_preq = 0    5. Wait for RTL to set ocs\_ess\_paccept = 0 5. Random delay, call inaccessible PG entry sequence    1. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = INACCBLK    2. Wait for RTL to set ocs\_ess\_paccept = 1    3. Random delay, then set ess\_ocs\_preq = 0    4. Wait for RTL to set ocs\_ess\_paccept = 0    5. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = WRMRST SRM ON    6. Wait for RTL to set ocs\_ess\_paccept = 1    7. Random delay, then set ess\_ocs\_preq = 0    8. Wait for RTL to set ocs\_ess\_paccept = 0    9. Random delay, assert cse\_rst\_vnnpgd\_slow\_b    10. Random delay, then assert cse\_side\_rst\_b 6. Random delay, call inaccessible PG exit sequence    1. Fork:       1. Random delay, then deassert cse\_side\_rst\_b       2. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACTV    2. Random delay, deassert cse\_rst\_vnnpgd\_slow\_b.    3. Wait for RTL to set ocs\_ess\_paccept = 1    4. Random delay, then set ess\_ocs\_preq = 0    5. Wait for RTL to set ocs\_ess\_paccept = 0     Open: Check GSC and DMR sip variations if we have side reset.  Note: Side reset is on a deeper domain so it needs to be deasserted before CSE reset and it needs to be asserted after CSE reset. |
| Checking  N/A |
| Coverage  Cover following PSTATE: OFF, ACTV, INACCBLK, WRMRST SRMON, WRMRST SRMOFF, ACCBLK, ACCPG  Cover preq, paccept, and pactive signals |
| Useful Resources  <Example: Block diagrams, waves, snippets from PCR/HAS etc.> |

### ocs\_reset\_isolation\_test

|  |
| --- |
| Objective (3rd)  Test will transition from ACTV -> CSERST ->ACTV |
| Description   1. Boot up sequence (ocs\_hard\_reset\_sequence)    1. assert cse\_rst\_vnnpgd\_slow\_b and cse\_side\_rst\_b    2. random delay , then deassert cse\_side\_rst\_b    3. random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACTV    4. random delay, then deassert cse\_rst\_vnnpgd\_slow\_b    5. wait for RTL to set ocs\_ess\_paccept = 1    6. random delay, then set ess\_ocs\_preq = 0    7. wait for RTL to set ocs\_ess\_paccept = 0 2. Call ocs initialization sequence 3. Reset isolation entry sequence    1. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = CSERST    2. Wait for RTL to set ocs\_ess\_paccept = 1    3. Random delay, then set ess\_ocs\_preq = 0    4. Wait for RTL to set ocs\_ess\_paccept = 0    5. Random delay, then assert cse\_rstisoen\_slow\_b    6. Random delay, then assert cse\_rst\_vnnpgd\_slow\_b    7. Short random delay of 2-3 clock cycle only, then deassert cse\_rstisoen\_slow\_b    8. Random delay, then deassert cse\_rst\_vnnpgd\_slow\_b 4. Reset isolation exit sequence    1. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACTV    2. Wait for RTL to set ocs\_ess\_paccept = 1    3. Random delay, then set ess\_ocs\_preq = 0    4. OPEN: how to make sure group 2 fuses are deliveried??? Check the MRA interface    5. Wait for RTL to set ocs\_ess\_paccept = 0 |
| Checking  N/A |
| Coverage  Cover following PSTATE: OFF, ACTV, CSERST  Cover preq, paccept |
| Useful Resources  <Example: Block diagrams, waves, snippets from PCR/HAS etc.> |

### ocs\_accessible\_pg\_test

|  |
| --- |
| Objective (2nd)  Test will transition from ACTV -> ACCBLK ->ACTV or ACCPG->ACTV or remains in ACTV if ACCBLK gets deny |
| Description   1. Boot up sequence (ocs\_hard\_reset\_sequence)    1. assert cse\_rst\_vnnpgd\_slow\_b and cse\_side\_rst\_b    2. random delay , then deassert cse\_side\_rst\_b    3. random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACTV    4. random delay, then deassert cse\_rst\_vnnpgd\_slow\_b    5. wait for RTL to set ocs\_ess\_paccept = 1    6. random delay, then set ess\_ocs\_preq = 0    7. wait for RTL to set ocs\_ess\_paccept = 0 2. Call ocs initialization sequence 3. Call accessible PG entry sequence with accblk\_exit variable = 1 or 0 with even distribution and deny variable with 10% distribution    1. If deny ==1   Fork   * + - 1. **Thread 1**: Issue OCS traffic   Wait for ocs\_ess\_pactive.pactive\_func = 1   * + - 1. **Thread 2**: Random delay enough that ocs\_ess\_pactive.pactive\_func = 1, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACCBLK.   Wait for RTL to set ocs\_ess\_pdeny = 1  Random delay, then set ess\_ocs\_preq = 0 and set ess\_ocs\_pstate = ACTV  Skip Step 4 (Call accessible PG exit sequence) and test ends.   * 1. Else      1. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACCBLK      2. Wait for RTL to set ocs\_ess\_paccept = 1      3. Random delay, then set ess\_ocs\_preq = 0      4. Wait for RTL to set ocs\_ess\_paccept = 0      5. If accblk\_exit =1,         1. Go to Step 4 ii then 4 c. Do not assert any resets.   Else if accblk\_exit = 0,   * + - 1. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACCPG       2. *Fork / join\_any*          1. *Thread 1: RTL will trigger save write requests to GSK so scoreboard will detect those writes and save the contents that are getting written. Every time there is a ocs\_gsk\_snr\_put signal, testbench will drive gsk\_ocs\_snr\_cpl signal with some small random clock delay. This thread will loop forever until the other thread completes to indicate RTL is done with saving.*          2. Thread 2: Wait for RTL to set ocs\_ess\_paccept = 1       3. Random delay, then set ess\_ocs\_preq = 0       4. Wait for RTL to set ocs\_ess\_paccept = 0       5. Random delay, assert cse\_rst\_vnnpgd\_slow\_b       6. Random delay, assert the cse\_side\_rst\_b  1. Call accessible PG exit sequence    1. Fork:       1. Random delay, then deassert cse\_side\_rst\_b if it was asserted       2. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACTV    2. Random delay, then deassert cse\_rst\_vnnpgd\_slow\_b if it was asserted    3. Wait for RTL to set ocs\_ess\_paccept = 1    4. Random delay, then set ess\_ocs\_preq = 0    5. Wait for RTL to set ocs\_ess\_paccept = 0 |
| Checking  N/A |
| Coverage  Cover following PSTATE: OFF, ACTV, ACCBLK, ACCPG  Cover preq, paccept, and pactive signals |
| Useful Resources  <Example: Block diagrams, waves, snippets from PCR/HAS etc.> |

### ocs\_inaccessible\_pg\_test

|  |
| --- |
| Objective (1st)  Test will transition from ACTV -> INACCBLK ->ACTV or WARM\* or OFF->ACTV |
| Description   1. Boot up sequence (ocs\_hard\_reset\_sequence)    1. assert cse\_rst\_vnnpgd\_slow\_b and cse\_side\_rst\_b    2. random delay , then deassert cse\_side\_rst\_b    3. random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACTV    4. random delay, then deassert cse\_rst\_vnnpgd\_slow\_b    5. wait for RTL to set ocs\_ess\_paccept = 1    6. random delay, then set ess\_ocs\_preq = 0    7. wait for RTL to set ocs\_ess\_paccept = 0 2. Call ocs initialization sequence 3. Random delay, call inaccessible PG entry sequence    1. If deny ==1   Fork   1. **Thread 1**: Issue OCS traffic   Wait for ocs\_ess\_pactive.pactive\_func =1   1. **Thread 2**: Random delay enough that ocs\_ess\_pactive.pactive\_func = 1, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = INACCBLK.   Wait for RTL to set ocs\_ess\_pdeny = 1  Random delay, then set ess\_ocs\_preq = 0 and set ess\_ocs\_pstate = ACTV  Skip Step 4 (call inaccessible PG exit sequence) and test ends.   * 1. Else  1. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = INACCBLK 2. Wait for RTL to set ocs\_ess\_paccept = 1 3. Random delay, then set ess\_ocs\_preq = 0 4. Wait for RTL to set ocs\_ess\_paccept = 0 5. If inaccblk\_exit =1,    * 1. Go to step 4 ii and then 4d. Do not assert any resets.   Else if inaccblk\_exit = 0,   * + 1. Case statement to pick next power state transition of WRMRST SRM ON, WRMRST SRM OFF, or OFF. For OCS, it is all the same functionality.        1. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = WRMRST SRM ON   Wait for RTL to set ocs\_ess\_paccept = 1  Random delay, then set ess\_ocs\_preq = 0  Wait for RTL to set ocs\_ess\_paccept = 0  Random delay, assert cse\_rst\_vnnpgd\_slow\_b  Random delay, then assert cse\_side\_rst\_b   * + - 1. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = WRMRST SRM OFF   Wait for RTL to set ocs\_ess\_paccept = 1  Random delay, then set ess\_ocs\_preq = 0  Wait for RTL to set ocs\_ess\_paccept = 0  Random delay, assert cse\_rst\_vnnpgd\_slow\_b  Random delay, then assert cse\_side\_rst\_b   * + - 1. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = OFF   Wait for RTL to set ocs\_ess\_paccept = 1  Random delay, then set ess\_ocs\_preq = 0  Wait for RTL to set ocs\_ess\_paccept = 0  Random delay, assert cse\_rst\_vnnpgd\_slow\_b.  Random delay, then assert cse\_side\_rst\_b  Random delay, then Assert the powergood\_rst\_b.  (note: all reset can assert same clock cycle)   1. Random delay, call inaccessible PG exit sequence    1. Deassert powergood\_rst\_b if it was asserted before in the OFF state    2. Fork:       1. Random delay, then deassert cse\_side\_rst\_b       2. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACTV    3. Random delay, deassert cse\_rst\_vnnpgd\_slow\_b.    4. Wait for RTL to set ocs\_ess\_paccept = 1    5. Random delay, then set ess\_ocs\_preq = 0    6. Wait for RTL to set ocs\_ess\_paccept = 0 |
| Checking  N/A |
| Coverage  Cover following PSTATE: OFF, ACTV, INACCBLK, WRMRST SRMON, WRMRST SRMOFF  Cover preq, paccept, and pactive signals |
| Useful Resources  <Example: Block diagrams, waves, snippets from PCR/HAS etc.> |

### ocs\_save\_and\_restore\_test

|  |
| --- |
| Objective (4th)  Test will transition from ACTV -> ACCBLK -> ACCPG -> (internally park OCS IP to OFF without preq) -> RESTORE -> ACTV.  Testbench environment needs to understand that after ACCPG, internally PSTATE will default back to OFF state without a preq because the reset are asserted.  OCS does the save during ACCPG and restore in the restore state. |
| Description   1. Boot up sequence (ocs\_hard\_reset\_sequence)    1. assert cse\_rst\_vnnpgd\_slow\_b and cse\_side\_rst\_b    2. random delay , then deassert cse\_side\_rst\_b    3. random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACTV    4. random delay, then deassert cse\_rst\_vnnpgd\_slow\_b    5. wait for RTL to set ocs\_ess\_paccept = 1    6. random delay, then set ess\_ocs\_preq = 0    7. wait for RTL to set ocs\_ess\_paccept = 0 2. Call ocs initialization sequence 3. Random delay, call accessible PG entry sequence    1. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACCBLK    2. Wait for RTL to set ocs\_ess\_paccept = 1    3. Random delay, then set ess\_ocs\_preq = 0    4. Wait for RTL to set ocs\_ess\_paccept = 0    5. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACCPG    6. *Fork / join\_any*       * *Thread 1: RTL will trigger save write requests to GSK so scoreboard will detect those writes and save the contents that are getting written. Every time there is a ocs\_gsk\_snr\_put signal, testbench will drive gsk\_ocs\_snr\_cpl signal with some small random clock delay. This thread will loop forever until the other thread completes to indicate RTL is done with saving.*       * Thread 2: Wait for RTL to set ocs\_ess\_paccept = 1    7. Random delay, then set ess\_ocs\_preq = 0    8. Wait for RTL to set ocs\_ess\_paccept = 0    9. Random delay, assert cse\_rst\_vnnpgd\_slow\_b    10. Random delay, assert the cse\_side\_rst\_b 4. Random delay, call restore entry sequence    1. Random delay, then deassert cse\_side\_rst\_b    2. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = RESTORE    3. Random delay, then assert vnn\_restore wire 🡨 OPEN: is this done before the preq=1    4. Random delay, then deassert cse\_rst\_vnnpgd\_slow\_b    5. Wait for RTL to set ocs\_ess\_paccept = 1    6. Random delay, then set ess\_ocs\_preq = 0    7. *Fork / join\_any*       * *Thread 1: RTL send restore read requests. Testbench drives gsk\_ocs\_snr\_rdata[63:0] and gsk\_ocs\_snr\_cpl signal accordingly as when they see a ocs\_gsk\_snr\_put*   Open for check: Check the address if it’s in the right range. Count the put and a particular address. Discuss with Bindu further.   * + - Thread 2:Wait for RTL to set ocs\_ess\_paccept = 0   1. Random delay, then deassert vnn\_restore wire  1. Random delay, call restore exit sequence    1. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACTV    2. Wait for RTL to set ocs\_ess\_paccept = 1    3. Random delay, then set ess\_ocs\_preq = 0    4. Wait for RTL to set ocs\_ess\_paccept = 0 |
| Checking  The save and restore scoreboard will make sure the restored data is correct. |
| Coverage  Cover following PSTATE: OFF, ACTV, INACCBLK, RESTORE  Cover preq, paccept |
| Useful Resources  <Example: Block diagrams, waves, snippets from PCR/HAS etc.> |

### ocs\_save\_and\_restore\_err\_test

|  |
| --- |
| Objective (4th)  Test will transition from ACTV -> ACCBLK -> ACCPG -> (internally park OCS IP to OFF without preq) -> RESTORE -> ACTV.  Testbench environment needs to understand that after ACCPG, internally PSTATE will default back to OFF state without a preq because the reset are asserted.  OCS does the save during ACCPG and restore in the restore state. During the save, testbench will intentionally **not** issue the cpl signal. 🡨 this will hang until until the cpl signal get send out right? Colin: Yes will hang. Real system will not happen so no need timeout concept. Bindu: move us back to ACTV after a long wait  During the restore, testbench will fork off two thread. Thread 1 to generate background read data without cpl signal. Thread 2 will issue good read data with cpl signal.  Open: Is there value in forcing internal credit signals to different values? |
| Description   1. Boot up sequence (ocs\_hard\_reset\_sequence)    1. assert cse\_rst\_vnnpgd\_slow\_b and cse\_side\_rst\_b    2. random delay , then deassert cse\_side\_rst\_b    3. random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACTV    4. random delay, then deassert cse\_rst\_vnnpgd\_slow\_b    5. wait for RTL to set ocs\_ess\_paccept = 1    6. random delay, then set ess\_ocs\_preq = 0    7. wait for RTL to set ocs\_ess\_paccept = 0 2. Call ocs initialization sequence 3. Random delay, call accessible PG entry sequence    1. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACCBLK    2. Wait for RTL to set ocs\_ess\_paccept = 1    3. Random delay, then set ess\_ocs\_preq = 0    4. Wait for RTL to set ocs\_ess\_paccept = 0    5. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACCPG    6. *Fork / join\_any*       * *Thread 1: RTL will trigger save write requests to GSK so scoreboard will detect those writes and save the contents that are getting written. Every time there is a ocs\_gsk\_snr\_put signal, testbench will drive gsk\_ocs\_snr\_cpl signal sometime. This thread will loop forever until the other thread completes to indicate RTL is done with saving.*       * Thread 2: Wait for RTL to set ocs\_ess\_paccept = 1    7. Random delay, then set ess\_ocs\_preq = 0    8. Wait for RTL to set ocs\_ess\_paccept = 0    9. Random delay, assert cse\_rst\_vnnpgd\_slow\_b    10. Random delay, assert the cse\_side\_rst\_b 4. Random delay, call restore entry sequence    1. Random delay, then deassert cse\_side\_rst\_b    2. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = RESTORE    3. Random delay, then assert vnn\_restore wire    4. Random delay, then deassert cse\_rst\_vnnpgd\_slow\_b    5. Wait for RTL to set ocs\_ess\_paccept = 1    6. Random delay, then set ess\_ocs\_preq = 0    7. *Fork / join\_any*       * *Thread 1: RTL send restore read requests. Testbench drives gsk\_ocs\_snr\_rdata[63:0] and gsk\_ocs\_snr\_cpl signal accordingly as when they see a ocs\_gsk\_snr\_put*       * Thread 2: Wait for RTL to set ocs\_ess\_paccept = 0       * Thread 3: Testbench drives *gsk\_ocs\_snr\_rdata[63:0]* without gsk\_ocs\_snr\_cpl    8. Random delay, then deassert vnn\_restore wire 5. Random delay, call restore exit sequence    1. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACTV    2. Wait for RTL to set ocs\_ess\_paccept = 1    3. Random delay, then set ess\_ocs\_preq = 0    4. Wait for RTL to set ocs\_ess\_paccept = 0   A diagram of a diagram  AI-generated content may be incorrect.      A screenshot of a computer  AI-generated content may be incorrect. |
| Checking  The save and restore scoreboard will make sure the restored data is correct. |
| Coverage  Cover following PSTATE: OFF, ACTV, INACCBLK, RESTORE  Cover preq, paccept |
| Useful Resources  <Example: Block diagrams, waves, snippets from PCR/HAS etc.> |

### ocs\_tcg\_test

|  |
| --- |
| Objective  Test will transition from ACTV -> GLBLK ->VNN - > VNNEXITPREP -> ACTV |
| Description   1. Boot up sequence (ocs\_hard\_reset\_sequence)    1. assert cse\_rst\_vnnpgd\_slow\_b and cse\_side\_rst\_b    2. random delay , then deassert cse\_side\_rst\_b    3. random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACTV    4. random delay, then deassert cse\_rst\_vnnpgd\_slow\_b    5. wait for RTL to set ocs\_ess\_paccept = 1    6. random delay, then set ess\_ocs\_preq = 0    7. wait for RTL to set ocs\_ess\_paccept = 0 2. Call ocs initialization sequence 3. Random delay, call global block entry sequence    1. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = GLBLK    2. Wait for RTL to set ocs\_ess\_paccept = 1    3. Random delay, then set ess\_ocs\_preq = 0    4. Wait for RTL to set ocs\_ess\_paccept = 0 4. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = TCG 5. Wait for RTL to set ocs\_ess\_paccept = 1 6. Random delay, then set ess\_ocs\_preq = 0 7. Wait for RTL to set ocs\_ess\_paccept = 0 8. With 90% distribution, gate the clock then random delay, enable clock 9. Random delay, call tcg exit sequence    1. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = TCGEXITPREP    2. Wait for RTL to set ocs\_ess\_paccept = 1    3. Random delay, then set ess\_ocs\_preq = 0    4. Wait for RTL to set ocs\_ess\_paccept = 0    5. Issue DMA Traffic    6. Wait for RTL to set ocs\_ess\_pactive.pactive\_func = 1    7. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACTV    8. Wait for RTL to set ocs\_ess\_paccept = 1    9. Random delay, then set ess\_ocs\_preq = 0    10. Wait for RTL to set ocs\_ess\_paccept = 0 |
| Checking  Scoreboard will check if the pactive is set correctly. |
| Coverage  Cover following PSTATE: OFF, ACTV, GLBLK, VNN, VNNEXITPREP  Cover preq, paccept, and pactive signals |
| Useful Resources  <Example: Block diagrams, waves, snippets from PCR/HAS etc.> |

### ocs\_global\_blocking\_test

|  |
| --- |
| Objective  Test will transition from ACTV -> GLBLK -> ACTV or Remains in ACTV because GLBLK got deny |
| Description   1. Boot up sequence (ocs\_hard\_reset\_sequence)    1. assert cse\_rst\_vnnpgd\_slow\_b and cse\_side\_rst\_b    2. random delay , then deassert cse\_side\_rst\_b    3. random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACTV    4. random delay, then deassert cse\_rst\_vnnpgd\_slow\_b    5. wait for RTL to set ocs\_ess\_paccept = 1    6. random delay, then set ess\_ocs\_preq = 0    7. wait for RTL to set ocs\_ess\_paccept = 0 2. Call ocs initialization sequence 3. Random delay, call global block entry sequence with deny variable = 1 with 10% distribution    1. If deny ==1   Fork   * + - 1. **Thread 1**: Issue OCS traffic   Wait for ocs\_ess\_pactive.pactive\_func = 1   * + - 1. **Thread 2**: Random delay enough that ocs\_ess\_pactive.pactive\_func = 1, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = GLBLK.   Wait for RTL to set ocs\_ess\_pdeny = 1  Random delay, then set ess\_ocs\_preq = 0 and set ess\_ocs\_pstate = ACTV  Skip Step 4 and test ends.   * 1. Else      1. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = GLBLK      2. Wait for RTL to set ocs\_ess\_paccept = 1      3. Random delay, then set ess\_ocs\_preq = 0      4. Wait for RTL to set ocs\_ess\_paccept = 0      5. Continue to Step 4  1. Random delay, call tcg exit sequence    1. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACTV    2. Wait for RTL to set ocs\_ess\_paccept = 1    3. Random delay, then set ess\_ocs\_preq = 0    4. Wait for RTL to set ocs\_ess\_paccept = 0 |
| Checking  N/A |
| Coverage  Cover following PSTATE: OFF, ACTV, GLBLK  Cover preq, paccept, and pactive signals |
| Useful Resources  <Example: Block diagrams, waves, snippets from PCR/HAS etc.> |

### ocs\_pchnl\_timeout\_with\_cserst\_test

|  |
| --- |
| Objective  Test will test out CSERST with timeout asserted |
| Description   1. Boot up sequence (ocs\_hard\_reset\_sequence)    1. assert cse\_rst\_vnnpgd\_slow\_b and cse\_side\_rst\_b    2. random delay , then deassert cse\_side\_rst\_b    3. random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACTV    4. random delay, then deassert cse\_rst\_vnnpgd\_slow\_b    5. wait for RTL to set ocs\_ess\_paccept = 1    6. random delay, then set ess\_ocs\_preq = 0    7. wait for RTL to set ocs\_ess\_paccept = 0 2. Call ocs initialization sequence 3. Fork    1. Thread 1: Random delay and assert the pchnl\_timeout signal. The point it to catch the timeframe where the timeout signal is asserted before the PREQ and during PREQ. Only deassert the pchnl\_timeout once RTL deassert ocs\_ess\_paccept. Deassertion of pchnl\_timeout can happen right after the first ocs\_ess\_paccept deassertion or multiple ocs\_ess\_paccept deassertion.    2. Thread 2: Within a loop of 5 rounds and with random delay, call the reset isolation entry and exit flow. |
| Checking  Assertions will make sure that the RTL response with PACCEPT within a reasonable timeframe. |
| Coverage |
| Useful Resources  <Example: Block diagrams, waves, snippets from PCR/HAS etc.> |

### ocs\_pchnl\_timeout\_with\_accpg\_test

|  |
| --- |
| Objective  Test will test out accessible\_pg with timeout asserted. Timeout can be asserted only in ACCPG. |
| Description   1. Boot up sequence (ocs\_hard\_reset\_sequence)    1. assert cse\_rst\_vnnpgd\_slow\_b and cse\_side\_rst\_b    2. random delay , then deassert cse\_side\_rst\_b    3. random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACTV    4. random delay, then deassert cse\_rst\_vnnpgd\_slow\_b    5. wait for RTL to set ocs\_ess\_paccept = 1    6. random delay, then set ess\_ocs\_preq = 0    7. wait for RTL to set ocs\_ess\_paccept = 0 2. Call ocs initialization sequence 3. Fork    1. Thread 1: Wait on pstate=ACCBLK and PREQ=1 and then 0. Issue Random delay and assert the pchnl\_timeout signal. The point it to catch the timeframe where the timeout signal is asserted before the PREQ with PSTATE= ACCPG and during PREQ with PSTATE= ACCPG. Deassert the pchnl\_timeout once RTL deassert ocs\_ess\_paccept. Deassertion of pchnl\_timeout can happen right after the first ocs\_ess\_paccept deassertion or multiple ocs\_ess\_paccept deassertion.    2. Thread 2: Within a loop of 5 rounds and with random delay, call the accessible\_pg entry and exit sequence |
| Checking  Assertions will make sure that the RTL response with PACCEPT within a reasonable timeframe. |
| Coverage |
| Useful Resources  <Example: Block diagrams, waves, snippets from PCR/HAS etc.> |

### ocs\_pchnl\_timeout\_with\_inaccpg\_test

|  |
| --- |
| Objective  Test will test out inaccessible\_pg with timeout asserted. Timeout can be asserted only during OFF, WRMRSTSRAMOFF, WRMRSTSRAMON states |
| Description   1. Boot up sequence (ocs\_hard\_reset\_sequence)    1. assert cse\_rst\_vnnpgd\_slow\_b and cse\_side\_rst\_b    2. random delay , then deassert cse\_side\_rst\_b    3. random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACTV    4. random delay, then deassert cse\_rst\_vnnpgd\_slow\_b    5. wait for RTL to set ocs\_ess\_paccept = 1    6. random delay, then set ess\_ocs\_preq = 0    7. wait for RTL to set ocs\_ess\_paccept = 0 2. Call ocs initialization sequence 3. Fork    1. Thread 1: Wait on pstate=INACCBLK and PREQ=1 and then 0. Issue Random delay and assert the pchnl\_timeout signal. The point it to catch the timeframe where the timeout signal is asserted before the PREQ with PSTATE= OFF, WRMRSTSRAMOFF, or WRMRSTSRAMON and during PREQ with PSTATE= OFF, WRMRSTSRAMOFF, or WRMRSTSRAMON. Deassert the pchnl\_timeout once RTL deassert ocs\_ess\_paccept. Deassertion of pchnl\_timeout can happen right after the first ocs\_ess\_paccept deassertion or multiple ocs\_ess\_paccept deassertion.    2. Thread 2: Within a loop of 5 rounds and with random delay, call the inaccessible\_pg entry and exit sequence |
| Checking  Assertions will make sure that the RTL response with PACCEPT within a reasonable timeframe. |
| Coverage |
| Useful Resources  <Example: Block diagrams, waves, snippets from PCR/HAS etc.> |

### ocs\_ecc\_power\_flow\_test

|  |
| --- |
| Objective  Test will test out all power flows with ECC traffic |
| Description   1. Boot up sequence (ocs\_hard\_reset\_sequence)    1. assert cse\_rst\_vnnpgd\_slow\_b and cse\_side\_rst\_b    2. random delay , then deassert cse\_side\_rst\_b    3. random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACTV    4. random delay, then deassert cse\_rst\_vnnpgd\_slow\_b    5. wait for RTL to set ocs\_ess\_paccept = 1    6. random delay, then set ess\_ocs\_preq = 0    7. wait for RTL to set ocs\_ess\_paccept = 0 2. Call ocs initialization sequence 3. Fork    1. Thread 1: Issue ECC traffic    2. Thread 2: Random delay, then randomly pick one of the power flow entry and exit to get back to ACTV state (accessible pg, inaccessible pg, save and restore, TGC, or reset isolation) 4. Check to make sure all ECC registers are back to default values for accessible/inaccessible pg/ reset isolation. 5. Issue ECC traffic again because after the fork/join, test will be in ACTV power state |
| Checking  Scoreboard will check if the pactive is set correctly.  Test will check if the ECC registers are reset back to default values for all power transition. |
| Coverage |
| Useful Resources  <Example: Block diagrams, waves, snippets from PCR/HAS etc.> |

### ocs\_eau\_power\_flow\_test

|  |
| --- |
| Objective  Test will test out all power flows with EAU traffic |
| Description   1. Boot up sequence (ocs\_hard\_reset\_sequence)    1. assert cse\_rst\_vnnpgd\_slow\_b and cse\_side\_rst\_b    2. random delay , then deassert cse\_side\_rst\_b    3. random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACTV    4. random delay, then deassert cse\_rst\_vnnpgd\_slow\_b    5. wait for RTL to set ocs\_ess\_paccept = 1    6. random delay, then set ess\_ocs\_preq = 0    7. wait for RTL to set ocs\_ess\_paccept = 0 2. Call ocs initialization sequence 3. Fork    1. Thread 1: Issue EAU traffic    2. Thread 2: Random delay, then randomly pick one of the power flow entry and exit to get back to ACTV state (accessible pg, inaccessible pg, save and restore, TGC or reset isolation) 4. Check to make sure all EAU registers are back to default values for accessible/inaccessible pg/ reset isolation. 5. Issue EAU traffic again because after the fork/join, test will be in ACTV power state |
| Checking  Scoreboard will check if the pactive is set correctly.  Test will check if the EAU registers are reset back to default values for all power transition. |
| Coverage  <Define CP’s that are when hit considered intent met>  <mention seeds required to hit the CP> |
| Useful Resources  <Example: Block diagrams, waves, snippets from PCR/HAS etc.> |

### ocs\_sks\_power\_flow\_test

|  |
| --- |
| Objective  Test will test out all power flows with SKS traffic |
| Description   1. Boot up sequence (ocs\_hard\_reset\_sequence)    1. assert cse\_rst\_vnnpgd\_slow\_b and cse\_side\_rst\_b    2. random delay , then deassert cse\_side\_rst\_b    3. random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACTV    4. random delay, then deassert cse\_rst\_vnnpgd\_slow\_b    5. wait for RTL to set ocs\_ess\_paccept = 1    6. random delay, then set ess\_ocs\_preq = 0    7. wait for RTL to set ocs\_ess\_paccept = 0 2. Call ocs initialization sequence 3. Fork    1. Thread 1: Issue SKS traffic    2. Thread 2: Random delay, then randomly pick one of the power flow entry and exit to get back to ACTV state (accessible pg, inaccessible pg, save and restore, TGC or reset isolation) 4. Check to make sure all SKS registers are back to default values for accessible/inaccessible pg/ reset isolation. 5. Issue SKS traffic again because after the fork/join, test will be in ACTV power state |
| Checking  Scoreboard will check if the pactive is set correctly.  Test will check if the SKS registers are reset back to default values for all power transition. |
| Coverage  <Define CP’s that are when hit considered intent met>  <mention seeds required to hit the CP> |
| Useful Resources  <Example: Block diagrams, waves, snippets from PCR/HAS etc.> |

### ocs\_dma\_power\_flow\_test

|  |
| --- |
| Objective  Test will test out all power flows with DMA traffic |
| Description   1. Boot up sequence (ocs\_hard\_reset\_sequence)    1. assert cse\_rst\_vnnpgd\_slow\_b and cse\_side\_rst\_b    2. random delay , then deassert cse\_side\_rst\_b    3. random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACTV    4. random delay, then deassert cse\_rst\_vnnpgd\_slow\_b    5. wait for RTL to set ocs\_ess\_paccept = 1    6. random delay, then set ess\_ocs\_preq = 0    7. wait for RTL to set ocs\_ess\_paccept = 0 2. Call ocs initialization sequence 3. Fork    1. Thread 1: Issue DMA traffic    2. Thread 2: Random delay, then randomly pick one of the power flow entry and exit to get back to ACTV state (accessible pg, inaccessible pg, save and restore, TGC or reset isolation) 4. Check to make sure all DMA registers are back to default values for accessible/inaccessible pg/ reset isolation. 5. Issue DMA traffic again because after the fork/join, test will be in ACTV power state |
| Checking  Scoreboard will check if the pactive is set correctly.  Test will check if the DMA registers are reset back to default values for all power transition. |
| Coverage  <Define CP’s that are when hit considered intent met>  <mention seeds required to hit the CP> |
| Useful Resources  <Example: Block diagrams, waves, snippets from PCR/HAS etc.> |

### ocs\_trng\_power\_flow\_test

|  |
| --- |
| Objective  Test will test out all power flows with TRNG traffic |
| Description   1. Boot up sequence (ocs\_hard\_reset\_sequence)    1. assert cse\_rst\_vnnpgd\_slow\_b and cse\_side\_rst\_b    2. random delay , then deassert cse\_side\_rst\_b    3. random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACTV    4. random delay, then deassert cse\_rst\_vnnpgd\_slow\_b    5. wait for RTL to set ocs\_ess\_paccept = 1    6. random delay, then set ess\_ocs\_preq = 0    7. wait for RTL to set ocs\_ess\_paccept = 0 2. Call ocs initialization sequence 3. Fork    1. Thread 1: Issue TRNG traffic    2. Thread 2: Random delay, then randomly pick one of the power flow entry and exit to get back to ACTV state (accessible pg, inaccessible pg, save and restore, TGC or reset isolation) 4. Check to make sure all TRNG registers are back to default values for accessible/inaccessible pg/ reset isolation. 5. Issue TRNG traffic again because after the fork/join, test will be in ACTV power state |
| Checking  Scoreboard will check if the pactive is set correctly.  Test will check if the TRNG registers are reset back to default values for all power transition. |
| Coverage  <Define CP’s that are when hit considered intent met>  <mention seeds required to hit the CP> |
| Useful Resources  <Example: Block diagrams, waves, snippets from PCR/HAS etc.> |

### ocs\_xmss\_power\_flow\_test

|  |
| --- |
| Objective  Test will test out all power flows with XMSS traffic |
| Description   1. Boot up sequence (ocs\_hard\_reset\_sequence)    1. assert cse\_rst\_vnnpgd\_slow\_b and cse\_side\_rst\_b    2. random delay , then deassert cse\_side\_rst\_b    3. random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACTV    4. random delay, then deassert cse\_rst\_vnnpgd\_slow\_b    5. wait for RTL to set ocs\_ess\_paccept = 1    6. random delay, then set ess\_ocs\_preq = 0    7. wait for RTL to set ocs\_ess\_paccept = 0 2. Call ocs initialization sequence 3. Fork    1. Thread 1: Issue XMSS traffic    2. Thread 2: Random delay, then randomly pick one of the power flow entry and exit to get back to ACTV state (accessible pg, inaccessible pg, save and restore, TGC or reset isolation) 4. Check to make sure all XMSS registers are back to default values for accessible/inaccessible pg/ reset isolation. 5. Issue XMSS traffic again because after the fork/join, test will be in ACTV power state |
| Checking  Scoreboard will check if the pactive is set correctly.  Test will check if the XMSS registers are reset back to default values for all power transition. |
| Coverage  <Define CP’s that are when hit considered intent met>  <mention seeds required to hit the CP> |
| Useful Resources  <Example: Block diagrams, waves, snippets from PCR/HAS etc.> |

### ocs\_selftest\_power\_flow\_test

|  |
| --- |
| Objective  Test will test out all power flows with selftest traffic |
| Description   1. Boot up sequence (ocs\_hard\_reset\_sequence)    1. Override selftest fuse en to 1 to run selftest. Look into the algorithm disable registers to randomize its register programming to validate the state retention.    2. assert cse\_rst\_vnnpgd\_slow\_b and cse\_side\_rst\_b    3. random delay , then deassert cse\_side\_rst\_b    4. random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACTV    5. random delay, then deassert cse\_rst\_vnnpgd\_slow\_b    6. wait for RTL to set ocs\_ess\_paccept = 1    7. random delay, then set ess\_ocs\_preq = 0    8. wait for RTL to set ocs\_ess\_paccept = 0 2. Wait for selftest to complete 3. Call ocs initialization sequence 4. Random delay, then randomly pick one of the power flow entry and exit to get back to ACTV state (accessible pg, inaccessible pg, save and restore, TGC or reset isolation) 5. Check selftest registers. Depending on the power state, check the state retention and save and restore registers and/or default values. 6. For save and restore, make sure that selftest does not get rerun and it boot up OCS again either with selftest passed or selftest failed.   **Focus on the list below for the state retention and save and restore registers:**  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: SELFTEST\_RERUN.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: SELFTEST\_PASS.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: SELFTEST\_FAIL.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: SELFTEST\_STAGE1\_PASS.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: SELFTEST\_STAGE1\_FAIL.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: SELFTEST\_STAGE2\_PASS.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: SELFTEST\_STAGE2\_FAIL.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: SELFTEST\_STAGE3\_PASS.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: SELFTEST\_STAGE3\_FAIL.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: AES\_ECB\_DISABLE.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: AES\_CBC\_DISABLE.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: AES\_CTR\_DISABLE.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: AES\_CFB\_DISABLE.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: AES\_OFB\_DISABLE.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: AES\_GCM\_DISABLE.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: AES\_CCM\_DISABLE.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: AES\_CTS\_DISABLE.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: RC4\_DISABLE.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: MD5\_DISABLE.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: SHA1\_DISABLE.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: SHA224\_DISABLE.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: SHA256\_DISABLE.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: SHA384\_DISABLE.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: SHA512\_DISABLE.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: HMAC\_MD5\_DISABLE.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: HMAC\_1\_DISABLE.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: HMAC\_224\_DISABLE.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: HMAC\_256\_DISABLE.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: HMAC\_384\_DISABLE.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: HMAC\_512\_DISABLE.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: DRNG\_DISABLE.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: ECDSA\_SIGN\_VERIFY\_DISABLE.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: EAU\_DISABLE.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: EAU\_SMALL\_KEYSZ\_DISABLE.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: SM2\_DISABLE.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: SM3\_DISABLE.set\_sr(1);  ./saola/ESEXX/self\_test\_regs\_ESEXX\_regs.svh: SM4\_DISABLE.set\_sr(1); |
| Checking  Scoreboard will check if the pactive is set correctly.  Test will check if the selftest registers functionality.  Selftest scoreboard will understand the save and restore power state and expect to skip selftest run. |
| Coverage  <Define CP’s that are when hit considered intent met>  <mention seeds required to hit the CP> |
| Useful Resources  <Example: Block diagrams, waves, snippets from PCR/HAS etc.> |

### ocs\_puf\_power\_flow\_test

|  |
| --- |
| Objective  Test will test out all power flows with production flow PUF |
| Description   1. Boot up sequence (ocs\_hard\_reset\_sequence)    1. assert cse\_rst\_vnnpgd\_slow\_b and cse\_side\_rst\_b    2. random delay , then deassert cse\_side\_rst\_b    3. random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACTV    4. random delay, then deassert cse\_rst\_vnnpgd\_slow\_b    5. wait for RTL to set ocs\_ess\_paccept = 1    6. random delay, then set ess\_ocs\_preq = 0    7. wait for RTL to set ocs\_ess\_paccept = 0 2. Call ocs initialization sequence 3. Wait for PUF key to get generated. 4. Random delay, then randomly pick one of the power flow entry and exit to get back to ACTV state (accessible pg, inaccessible pg, save and restore, TGC or reset isolation) 5. Check to make sure all PUF registers are back to default values for accessible/inaccessible pg/ reset isolation. Depending on the power flow, check that the PUF key is retained because it is stored in the AON domain.   Note: DO multiple resets and have to assert the puf helper data to check if the ocs rtl completes the group 2 fuse pulling each before it sets the paccept =0 once its ACTV state |
| Checking  Scoreboard will check if the pactive is set correctly.  Test will check if the PUF registers are reset back to default values for all power transition. |
| Coverage  <Define CP’s that are when hit considered intent met>  <mention seeds required to hit the CP> |
| Useful Resources  <Example: Block diagrams, waves, snippets from PCR/HAS etc.> |

### ocs\_illegal\_power\_flow\_test

|  |
| --- |
| Objective  Test will test out illegal power flow transition and expect RTL to issue PACCEPT but not do anything and OCS is will remain in currrect state.  ACTV -> ACCBLK -> INACCBLK ->ACTV |
| Description   1. Boot up sequence (ocs\_hard\_reset\_sequence)    1. assert cse\_rst\_vnnpgd\_slow\_b and cse\_side\_rst\_b    2. random delay , then deassert cse\_side\_rst\_b    3. random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACTV    4. random delay, then deassert cse\_rst\_vnnpgd\_slow\_b    5. wait for RTL to set ocs\_ess\_paccept = 1    6. random delay, then set ess\_ocs\_preq = 0    7. wait for RTL to set ocs\_ess\_paccept = 0 2. Call ocs initialization sequence 3. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = ACCBLK 4. Wait for RTL to set ocs\_ess\_paccept = 1 5. Random delay, then set ess\_ocs\_preq = 0 6. Wait for RTL to set ocs\_ess\_paccept = 0 7. Random delay, then set ess\_ocs\_preq = 1 and set ess\_ocs\_pstate = INACCBLK. 8. Wait for RTL to set ocs\_ess\_paccept = 1 🡨 ensure get paccept or pdeny 9. Random delay, then set ess\_ocs\_preq = 0 10. Wait for RTL to set ocs\_ess\_paccept = 0 |
| Checking  <Details on how checking will be performed for the stimulus present; Ex: Assertions, Scoreboard Checks, Self-checking test, etc.?> |
| Coverage  <Define CP’s that are when hit considered intent met>  <mention seeds required to hit the CP> |
| Useful Resources  <Example: Block diagrams, waves, snippets from PCR/HAS etc.> |

## Checklist

### Register Validation

*The author is required and expected to comply with the Intel-wide “Register MRC” team’s recommendations for register attribute validation. Please refer to the website alias “goto/registers,” clicking the “Documents” item from the right-hand frame, then “*[*Register Validation Requirements*](https://docs.intel.com/documents/arch_register_spec/Documents/RegisterValidationRequirements/RegisterValidationRequirements.html)*.” (At some future point in time, this documentation shall be transferred to some location within our development repository, for long-term maintenance by our own team. While the CREST tool described in the provided link is not required within our team, it is a requirement that an equivalent be used.*

*<Fill in below table for every special attribute field that is not covered by recursive testing or automated tools>*

*<Field Name>*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute | Data | Stimulus present in test | Check Name | Cover point Name |
| *Access type* | *<RW/1C>* | *<Test Name>* | *<Check Name>* | *<CP Name>* |

### Feature Vs. Flow Matrix

*<List the test names that covers given Feature X Flow validation>*

*<Purpose: This table organizes high-level information facilitating the management and tracking of individual tests which are involved in validating the specified flows and features.>*

*<Fill in below table for every new feature and flow added>*

*<Below are few examples given; author and reviewer should agree (pre-review) upon an appropriate list of “flows” and “features” for treatment in a tabular fashion as depicted here. >*

|  |  |  |  |
| --- | --- | --- | --- |
| Feature X Flow | *<Power Flows>* | *<Reset Flows>* | *<Clock Gating>* |
| *<Feature A>* | *<Test Name>* | *<Test Name>* | *<Test Name>* |
| *<Feature B>* | *<Test Name>* | *<Test Name>* | *<Test Name>* |
| *<Feature C>* | *<Test Name>* | *<Test Name>* | *<Test Name>* |
| *<Feature D>* | *<Test Name>* | *<Test Name>* | *<Test Name>* |

### Feature Vs. Feature Matrix

*<List the test names that covers given Feature X Feature validation>*

*<Purpose: This table organizes high-level information facilitating the management and tracking of individual tests which are involved in validating the specified flows and features.>*

*<Fill in below table for every new feature and flow added>*

*<Below are few examples given; author and reviewer should agree (pre-review) upon an appropriate list of “feature” and “feature” for treatment in a tabular fashion as depicted here. >*

|  |  |  |  |
| --- | --- | --- | --- |
| Feature X Feature | *<Feature A>* | *<Feature B>* | *<Feature C>* |
| *<Feature A>* | *<Test Name>* | *<Test Name>* | *<Test Name>* |
| *<Feature B>* | *<Test Name>* | *<Test Name>* | *<Test Name>* |
| *<Feature C>* | *<Test Name>* | *<Test Name>* | *<Test Name>* |
| *<Feature D>* | *<Test Name>* | *<Test Name>* | *<Test Name>* |

### Error Scenarios

*<List the test names that covers error scenario validation>*

*<Fill in below table for every new feature added>*

*<For example, things like malformed requests (e.g., requests violating some requirement of the bus specification or IP HAS), and violations to the IP’s programming model>*

*<Author and reviewer to decide on the content that can be considered as Error Scenarios for your IP>*

|  |  |  |  |
| --- | --- | --- | --- |
| Scenario | Stimulus present in test | Check Name | Cover point Name |
| *<unsupported Cmd type>* | *<Test Name>* | *<Check Name>* | *<CP Name>* |
| *<unsupported length>* | *<Test Name>* | *<Check Name>* | *<CP Name>* |
| *< unsupported Address>* | *<Test Name>* | *<Check Name>* | *<CP Name>* |
| *<Spurious cmpl>* | *<Test Name>* | *<Check Name>* | *<CP Name>* |

### Fuse and Soft straps

*<List the test names that covers given fuses/soft straps validation>*

|  |  |  |  |
| --- | --- | --- | --- |
| Fuse/Soft Strap Name | Stimulus present in test | Check Name | Cover point Name |
| *<Fuse Name>* | *<Test Name>* | *<Check Name>* | *<CP Name>* |

### DFX and VISA

*<List the test names that covers given DFX and VISA validation>*

|  |  |  |  |
| --- | --- | --- | --- |
| DFX/VISA Signal or Flow | Stimulus present in test | Check Name | Cover point Name |
| *<Signal Name>* | *<Test Name>* | *<Check Name>* | *<CP Name>* |

### Ad hoc Signals

*<List the test names that covers given ad hoc signal validation>*

*<Long Term Plan: Maintain documentation about Ad Hoc signals within our testbench collaterals where the DUT is instantiated—imagine embedding information there, extracted by a script, and resulting in a text file report or a PDF file, and here we could cite that path to file generated in the repo.>*

|  |  |  |  |
| --- | --- | --- | --- |
| Ad hoc Signal | Stimulus present in test | Check Name | Cover point Name |
| *<Signal Name>* | *<Test Name>* | *<Check Name>* | *<CP Name>* |

### Power Flows

*<Below are few examples given, author and reviewer to decide on the content that can be considered as power flows for your IP>*

|  |  |  |  |
| --- | --- | --- | --- |
| Power Flow | Stimulus present in test | Check Name | Cover point Name |
| *<Acc PG entry-exit>* | *<Test Name>* | *<Check Name>* | *<CP Name>* |

### Reset Flows

*<Below are few examples given, author and reviewer to decide on the content that can be considered as reset flows for your IP>*

|  |  |  |  |
| --- | --- | --- | --- |
| Reset Flow | Stimulus present in test | Check Name | Cover point Name |
| *<Global Reset>* | *<Test Name>* | *<Check Name>* | *<CP Name>* |
| *<Host Partition reset>* | *<Test Name>* | *<Check Name>* | *<CP Name>* |

### Performance Measurement

*<Add tests related to performance measurement of feature>*

### Clock Frequency coverage

*<Add frequencies that are covered as part of PSV validation>*

|  |  |
| --- | --- |
| Clock Name | Frequency range covered. |
| *<cse clk>* | *< [180-360] MHz>* |
| *<prim clk>* | *< [180-360] MHz>* |

### Coverage on multi iteration of scenario/feature/flow

*<List down the scenario/feature/flow that requires a cover point to be written in such a way that the CP is hit only when the flows is performed 2 times or more in the same test>*

|  |  |
| --- | --- |
| Feature/Flow/Scenario | Cover Point Name |
| *<LMT Halt>* | *<>* |
|  |  |

### Misc.

*<Author to use one of the below options when filling the table>*

NA – If the scenario is not applicable.

YES – Provide test name where this scenario is covered.

NO – Provide reason for “no” for not covering the scenario.

|  |  |  |  |
| --- | --- | --- | --- |
| Scenario | Stimulus present in test | Check Name | Cover point Name |
| Exhaust credits on IP interfaces (e.g., back-pressure scenarios caused by insufficient credits) | *<Test Name>* | *<Check Name>* | *<CP Name>* |

### ValPlan AI checklist

*<Author to list down the valid rules generated by ValPlan AI and make sure they are all getting validated>*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Section | Rule | Stimulus present in test | Check Name | Cover point Name |
| (Section 3.1.3.4.6) Gen Graphics | KVM related RAVDMs are limited to Reg\_Wr\_Req only. | *<Test Name>* | *<Check Name>* | *<CP Name>* |

# Security Test Plan Content Changes

## S3 for SDLe Tool

*< Every product leaving Intel must go through SDL. For each project, the security risk is assessed by a Product Security Expert (PSE), based on which he/she defines the security assurance plan for the project. The security assurance plan includes both SDL tasks, which are focused on tasks performed by the product team, as well as activities involving evaluation by team(s) of external security experts. SDLe is a tool to keep track of all the SDL tasks assigned to the product team. Product team will proceed with SDL task execution and once all the SDL tasks are completed, product team will acquire SDL approval for their release >*

### <SKS engine>

##### Security and Validation Plan

|  |  |  |  |
| --- | --- | --- | --- |
| PCR | Requirement Validated | Validated Detailed Test Description | Owner |
| *<HSD number and title of the PCR>*  *<Revision of the requirement defined>* | *<called out requirement from SDL HSD>* | *<Background Description>*  *<Test name>*  *<Execution Sequence>*  *<Pass Condition>* |  |

### Validation report that Captures Security Verification Result

|  |  |  |  |
| --- | --- | --- | --- |
| Evaluation Area | Coverage Name and Explanation | Evidence that coverage is getting hit | Owner |
| *<HSD number and title of the PCR>* | *<CSpec / Whitebox>* | *<Snip of the coverage collection>* |  |

### <HCU DMA>

### Security and Validation Plan

|  |  |  |  |
| --- | --- | --- | --- |
| PCR | Requirement Validated | Validated Detailed Test Description | Owner |
| *<HSD number and title of the PCR>*  *<Revision of the requirement defined>* | *<called out requirement from SDL HSD>* | *<Background Description>*  *<Test name>*  *<Execution Sequence>*  *<Pass Condition>* |  |

### Validation report that Captures Security Verification Result

|  |  |  |  |
| --- | --- | --- | --- |
| Evaluation Area | Coverage Name and Explanation | Evidence that coverage is getting hit | Owner |
| *<HSD number and title of the PCR>* | *<CSpec / Whitebox>* | *<Snip of the coverage collection>* |  |

### Multiple Area

### Security and Validation Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PCR | Requirement Validated | Validated Detailed Test Description | Affected area | Owner |
| *<HSD number and title of the PCR>*  *<Revision of the requirement defined>* | *<called out requirement from SDL HSD>* | *<Background Description>*  *<Test name>*  *<Execution Sequence>*  *<Pass Condition>* |  |  |
| [14012033569](https://hsdes.intel.com/resource/14012033569)  Support Gkey testmode toggle prevention until reset  **HSD Rev**: 7 | For defense in depth, ensure there is a test to verify that the HW could hide the production GKEY until the next reset, after entering the GKEY testmode. | **Background:**  OCS is required to protect the production Gkey and soft strap re-pull capability should not get access to the production secret. To further protect the production GKEY, OCS is adding additional logic for the testmode\_gkey\*\_fuse\_en signals by creating an internal latched testmode\_gkey\*\_fuse\_en signals. OCS will be using the internal latched testmode\_gkey\*\_fuse\_en signals to indicate production versus testmode Gkey and the latched signal holds its value until a reset.  **Objective:**  Test the internal latched signal that RTL will used to determine whether to use testmode GKEY or not.  **Test name:**   * ocs\_sks\_gkey\_reset\_test * ocs\_aes\_gkey\_reset\_test   **Execution Sequence:**   1. Generate Gkey\* with testmode\_gkey\*\_fuse\_en = 1 2. Perform AES encrypt to use the Gkey1 3. Generate Gkey\* with testmode\_gkey\*\_fuse\_en = 0 4. Perform AES encrypt to use the Gkey\* with the same input values as Step 2. 5. Assert the reset 6. Generate Gkey\* with random testmode\_gkey\*\_fuse\_en value 7. Perform AES encrypt to use the Gkey\*   **Pass Conditions:**  No mismatch between the expected and actual data of the encrypted data. | SKS, AES\_A DMA, AES\_P DMA | Linda |

### Validation report that Captures Security Verification Result

|  |  |  |  |
| --- | --- | --- | --- |
| Evaluation Area | Coverage Name and Explanation | Evidence that coverage is getting hit | Owner |
| *<HSD number and title of the PCR>* | *<CSpec / Whitebox>* | *<Snip of the coverage collection>* |  |
| [14012033569](https://hsdes.intel.com/resource/14012033569)  Support Gkey testmode toggle prevention until reset | **Whitebox coverages**: testmode\_gkey1\_fuse\_en\_stk, testmode\_gkey0\_fuse\_en\_stk, testmode\_gkey2\_fuse\_en\_stk  Make sure that the testmode\_gkey\*\_fuse\_en input signal is 1 while the internal sticky signal is 0 and GKEY generation bit. |  | Linda |

## Additional Consideration for Security Assets not covered in S3

*<Document any verification done for security asset defined from the threat model or identify by the architect or designer. If there are many testcases that verify the mitigation, then pick 1 to 3 tests to document in the table format. Any testcases that are describe in the security testplan do not need to be included in the functional testplan. Try to avoid duplication when possible. Only document assets that are not called out as a requirement in S3.>*

### PUF Key

|  |  |  |
| --- | --- | --- |
| **ASSET:** | PUF key | |
| **ACCESS PATH:** | *<List out all the physical pathway that needs to be considered and stimulus to verify that pathway>*  List out all the pathway:   * IOSF-Primary: all the testing is done through this pathway by using register accessing * MRA: this is not possible with IP testing but can be considered using FPV testing. | |
| **THREAT:** | FW uses PUF key to decrypt UDS (unique device secrets) | |
| **HARDWARE MITIGATION:** | ROM invalidates PUF KEY in SKS | |
|  | Test Name: | ocs\_puf\_invalidate\_test |
| Description: | 1. This test will generate the PUF key and store it in slot 30 of SKS. 2. Invalidate the slot 30 to make sure the PUF key is not accessible anymore. 3. Try using the key in slot 30 with a set key command in SKS to confirm that the slot is invalidated. |
| Checking: | RTL assertions to make sure HW writes zeros to SKS slot when invalidation command is detected. |
| Coverage: | Coverage group name: *sks\_go\_command\_cg*  Cross the invalidate command with the SKS slot address |
| **THREAT:** | PUF is transferred to ROM/FW readable register | |
| **HARDWARE MITIGATION:** | PUF key cannot be transferred to FW visible register through SKS programming | |
|  | Test Name: | ocs\_sks\_slot\_attr\_test |
| Description: | 1) This test was written as part of testplan for PCR:14011194333: to check that if parameter OCS\_SKS\_GKEY\_PUF\_SECURE is set, then SKS\_ATR\_1.SECURE\_KEY is \*always\* set to 1 after GKEY1 or PUF is successfully generated.  2) However, the scope of this test is broadened to check ALL slot attributes, for both the ways in which slot can be populated.  3) The test needs to check both things:   1. that the correct value is loaded into SKS\_ATR\_X AND 2. that the value is loaded at the right time (that is right after slot is valid).   4) Make sure that SKS\_ATR\_1.SECURE\_KEY is programmable from SKS\_CMDR.SECURE\_KEY when slot is not populated by GKEY1/PUF..or in other words, make sure SKS\_ATR\_1.SECURE\_KEY programming by GKEY1/PUF is not sticky. |
| Checking: | Scoreboard updates the expected secure attribute for slot 1/30 to always be 1 if the ocs\_sks\_gkey\_puf\_secure parameter is set. Then, scoreboard will compare it with the actual secure attribute field set by HW. |
| Coverage: | Cover group name: *read\_sks\_puf\_secure\_att\_reg\_cg*  Cross the generation of PUF key, secure key field in the SKS\_CMDR register, and secure key attribute setting in SKS\_ATR[30] register |

## Additional Consideration for Security Checks not covered in S3

*<Document any verification done for security checks that might have multiple assets and/or cannot be document in a table format as in Chapter 3. The security checks can be documented in paragraph form. Only document checks that are not called out as a requirement in S3.>*

### Unused byte lane

*<Checks to ensure no data leakage through inactive byte lanes.*

*Check to ensure HW only responds with data size that was requested for.>*

# Appendix

## Implementation Notes

*<Any further details that the Validation DCN owner wants to include that are not needed to be part of overall documents>*

## Bottoms up Effort

*<T-Shirt Sizing>*

|  |  |  |
| --- | --- | --- |
| ***<Validation Environment>***  ***<Ex: MISA>***  **Effort Sizing** | ***<Validation Environment>***  ***<Ex: GASKET>***  **Effort Sizing** | ***<Validation Environment>***  ***<Ex: CSE TOP>***  **Effort Sizing** |
| *<S>* | *<M>* | *<XL>* |

*<Provide overall effort needed to execute this PCR, effort to be provided per IP environment, below is an example template, owner can choose to use a different template than below>*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Area of Impact** | **Comments** | ***<Validation Environment>***  ***<Ex: MISA>***  **Effort in Weeks** | ***<Validation Environment>***  ***<Ex: GASKET>***  **Effort in Weeks** | ***<Validation Environment>***  ***<Ex: CSE TOP>***  **Effort in Weeks** |
| Documentation | Val DCN (Test Plan, Verif A-spec, Security Test Plan) |  |  |  |
| Environment | Example: TB, TI, Interface, monitors, cfg files, env.sv |  |  |  |
| Test Development/Execution |  |  |  |  |
| Checkers | Scoreboards, Assertions, Protocol checkers etc.. |  |  |  |
| C-Spec Coverage |  |  |  |  |
| OpenBox Coverage |  |  |  |  |
| L0/L1 regression | Baselining regression and coverage |  |  |  |
| Misc | Example: Turnin, Collage, Etc.. |  |  |  |
| Reviews | Reviews (Val DCN, Code, Paranoia) and Feedback incorporation |  |  |  |
| **Total Effort in Weeks** |  |  |  |  |

*<Provide Details on work that can execute in parallel>*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Workstream | WW1 | WW2 | WW3 | WW4 | WW5 | WW6 | Total |
| 1 | *<Documentation>* | *<Environment changes>* | *<Test Development>* |  |  |  |  |
| 2 |  | *<Environment changes>* | *<Test Development>* |  |  |  |  |
| 3 |  |  | *<Test Development>* |  |  |  |  |
|  |  |  |  |  |  |  |  |

## Process on how to update the val dcn document?

**Option 1:** When an existing content needs a update, follow below steps

1. Enable “track changes” feature in overall document(s)
2. Add your changes
3. Copy paste the delta into this document for the review
4. Once the review is complete and all the changes are approved, port the feedback back to the overall document(s)

local copy of overall document(s)

Val DCN

Val DCN Review

**Option 2:** When new content is added, follow below steps

1. Add your changes into this document
2. Once the review is complete and all the changes are approved, port the changes to the overall document(s)

Val DCN

Val DCN Review

overall document(s)