Chassis Power Gating Central Controller Verification IP

User Guide

**Synopsis:**

This component should be used by all IPs (SIP,Fabric) that use the PGCB to validate its Chassis defined power gating interface. It is a System Verilog OVM component. The user can configure the number of SIP, Fabric and delays using pamameters, configuarion objects as well as contrainted-random transactions. This VC will also consists of a monitor that scoreboards can subscribe to, a checker to check the Chassis defined power gating protocols and coverage collector.

IP Rev 2017WW25

June 20th 2017

Copyright and Disclaimer Information

Copyright © 2012, Intel Corporation. All rights reserved.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

\* Other names and brands may be claimed as the property of others.

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

Information in this document is provided in connection with Intel products. No license, express or implied, by estoppel or otherwise, to any Intellectual property rights is granted by this document. Except as provided in Intel’s terms and conditions of sale for such products, Intel assumes no liability whatsoever and Intel disclaims any express or implied warranty, relating to sale and/or use of Intel products including liability or warranties relating to fitness for a particular purpose, merchantability, or infringement of any patent, copyright, or other Intellectual property right.

Unless otherwise agreed in writing by Intel, the Intel products are not designed or intended for any application in which the failure of the Intel product could create a situation where personal injury or death may occur.

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

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

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

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

Contents

[1 Introduction 8](#_Toc351451964)

[1.1 Terminology 8](#_Toc351451965)

[1.2 Tool Support 8](#_Toc351451966)

[2 Overview 10](#_Toc351451967)

[2.1 Applications 10](#_Toc351451968)

[2.1.1 IP and Fabric test environment 10](#_Toc351451969)

[2.2 Features 12](#_Toc351451970)

[2.3 Control 18](#_Toc351451971)

[2.4 Requirements 19](#_Toc351451972)

[2.4.1 Specifications and Reference 19](#_Toc351451973)

[2.4.2 Compute Environment 19](#_Toc351451974)

[2.4.3 System Verilog Packages 19](#_Toc351451975)

[2.5 Architecture 20](#_Toc351451976)

[2.5.1 Class/Component Descriptions 20](#_Toc351451977)

[3 Getting Started 21](#_Toc351451978)

[4 Setting Up a Testbench Environment 22](#_Toc351451979)

[5 Implementing Test Scenarios 23](#_Toc351451980)

[6 Monitoring and Checking the Protocol 24](#_Toc351451981)

[7 Agent Packages 27](#_Toc351451982)

[8 Agent Parameters 28](#_Toc351451983)

[9 Agent Interface 30](#_Toc351451984)

[9.1 PowerGatingIF signals 30](#_Toc351451985)

[10 Configuration Methods 31](#_Toc351451986)

[10.1 Configuring the Agent 32](#_Toc351451987)

[10.1.1 PowerGatingConfig 32](#_Toc351451988)

[11 Transaction Sequence Item and Base Sequence 41](#_Toc351451989)

[11.1 CCAgentSeqItem 41](#_Toc351451990)

[11.1.1 Members 41](#_Toc351451991)

[11.1.2 Constraints 41](#_Toc351451992)

[11.1.3 CCAgentBaseSequence 42](#_Toc351451993)

[11.1.4 waitForComplete 43](#_Toc351451994)

[11.1.5 Commands 43](#_Toc351451995)

[11.2 CCAgentResponseSeqItem 45](#_Toc351451996)

[11.2.1 Members 45](#_Toc351451997)

[11.2.2 Constraints 46](#_Toc351451998)

|  |  |  |
| --- | --- | --- |
| Revision | Date | Description |
| 0.51 | WW39 | Initial version compliant to 0.7 version of the spec |
| 0.51\_v1 | WW40 | Added integration guide. |
| 0.6 | WW41 | * Moved section 3,4,5 of this UG into the integration guide * Modified TI to directly pass in the interface (to follow SIP methodology). * Added IS\_ACTIVE parameter in the TI which needs to be set to 0 in SOC level. * Added a mode where the user can keep a particular fet ON even when all the conditions for turning off the fet is satisfied. * Added waveform to show example of a master command ans waitForComplete. * Added documentationon waitForComplete bit in the base sequence. * Added two new master commands to deassert pmc wake signal. * Added two parameters NO\_SIP and NO\_FAB for environments that may have only SIP or only Fabric interface |
| 0.6\_v1 | WW41.2 | * Updated HDL file and added FAQ section in integration guide. |
| 0.7 | WW43.3 | * Added tracker. See tracker userguide for details. * Added configuration needed for tracker. Please see section 10 for the changes. * Please see integration guide for examples on configuration. * Added pok ports. |
| 0.71 | WW46.1 | * Enhancements/Bug fixes * [4796471](https://vthsd.intel.com/hsd/seg_softip/bug/default.aspx?bug_id=4796471) – tracker now prints Accessible flow properly * [4796548](https://vthsd.intel.com/hsd/seg_softip/bug/default.aspx?bug_id=4796548) – user can now specify any tradker name. * Updated block diagram. * Added clarifications and fixed typos. See change bars |
| 0.72 | WW47.1 | * Added a parameter IP\_ENV\_TO\_CC\_AGENT\_PATH to avoid integration issues/name conflicts in FC. * Bug fix 4796728 – printer fifo instance name is made unique now to avoid collision * NOTE: in the previous version the paramters NO\_FAB and NO\_SIP were changes but not noted in the change bar. |
| 0.8 | WW01 | * Support for new restore flow. * IMPORTANT NOTES   + There is still support for ip\_pmc\_save\_req\_b and pmc\_ip\_save\_ack\_b for backward compatibility. Ips that have implemented the new restore flow should leave these signals unconnected.   + But the delay constraint parameters have been removed. Please ses details in the userguide.   + IP that have implemented the new restore flow changes should update to this version, make TI changes to remove save req/ack and connect restore and remove any reference to delay\_save\_ack and delay\_restore\_ack. No other changes are necessary. * Bug fix 4797397 – delay distribution now has been changed in favor of smaller values. * Warm reset flow in the monitor/tracker. * Pok flow changes in monitor/tracker. |
| 0.82 | WW11 | * Bug fixes for AON Ips and fixed a typo in the config onject class. * Also enforced a rule to make sure all Ips add a sideband EP using the AddSBEP method. |
| 0.85 | WW11 | * Changed fabric power gating signal behavior and polarity as per Chassis 0.9 PG HAS. * Added config to specify which SIP belong to which fabric. |
| 2013WW24 | WW24 | Bug fixes and documentation updates   |  |  |  | | --- | --- | --- | | [4966274](https://vthsd.intel.com/hsd/seg_softip/bug/default.aspx?bug_id=4966274) | [PowerGatingMonitorSeqItem toString function returns empty string](https://vthsd.intel.com/hsd/seg_softip/bug/default.aspx?bug_id=4966274) | [Enhancement Request](https://vthsd.intel.com/hsd/seg_softip/bug/default.aspx?bug_id=4966274) | | [5076719](https://vthsd.intel.com/hsd/seg_softip/bug/default.aspx?bug_id=5076719) | [[Enhancement] FET protocol checks missing](https://vthsd.intel.com/hsd/seg_softip/bug/default.aspx?bug_id=5076719) | Bug | | [5076832](https://vthsd.intel.com/hsd/seg_softip/bug/default.aspx?bug_id=5076832) | [missing package import in source/CC/CCAgentPkg.sv](https://vthsd.intel.com/hsd/seg_softip/bug/default.aspx?bug_id=5076832) | [Enhancement Request](https://vthsd.intel.com/hsd/seg_softip/bug/default.aspx?bug_id=4966274) | |
| 2013WW25 | WW25 | * DFX support for fdfx\_pgcb\_bypass and fdfx\_pgcb\_ovr signals in driver and tracker/monitor (no checks added but coverage will be added for these signals) * D3/D0i3 support in tracker/monitor (no checks added) |
| 2013WW26 | WW26 | * Added coverage model (see tracker/monitor userguide). * Made following bug fixes.  |  |  |  | | --- | --- | --- | | [5077365](https://vthsd.intel.com/hsd/seg_softip/bug/default.aspx?bug_id=5077365) | [Assertions are not using fab\_pmc\_pg\_rdy\_ack\_b, fab\_pmc\_pg\_rdy\_nak\_b synced to PMC clockdomain](https://vthsd.intel.com/hsd/seg_softip/bug/default.aspx?bug_id=5077365) | [Bug](https://vthsd.intel.com/hsd/seg_softip/bug/default.aspx?bug_id=5077365) | | [5077849](https://vthsd.intel.com/hsd/seg_softip/bug/default.aspx?bug_id=5077849) | [pok values in the monitor not reset correctly during global reset event](https://vthsd.intel.com/hsd/seg_softip/bug/default.aspx?bug_id=5077849) | Bug | |
| 2013WW30 | WW30 | * The following changes have been made  |  |  |  | | --- | --- | --- | | [5077770](https://vthsd.intel.com/hsd/seg_softip/bug/default.aspx?bug_id=5077770) | [[Chassis ECN] make default value of restore\_b configurable as per Chassis PM ECN 1570775](https://vthsd.intel.com/hsd/seg_softip/bug/default.aspx?bug_id=5077770) | Does not affect any SPT IP | |  | Changes made for performance speed-up | Not a functional change | |
| 2014WW12 |  | NOTE: Collage (non-parameterized interface) changes have been implemented and documented   |  |  |  | | --- | --- | --- | | **HSD** | **Description** | **Comment** | | [1013373972](https://hsdes.intel.com/home/default.html/article?id=1013373972) | [BXT.D0i2]. pgcb bfm should not report assertion error when test ends in PG\_HS state. | New hook to disable end of test checking in PowerGatingConfig object called disable\_eot\_check. | | [1019110072](https://hsdes.intel.com/home/default.html/article?id=1019110072) | Inacc pg test fails in DNV as the PGVC trackers skip printing of the INACC\_PON message | Bug fix for race condition. | |  | All assertions are now disabled when reset\_b !== 1 (instead of reset\_b === 0) |  | | [1013560870](https://hsdes.intel.com/home/default.html/article?id=1013560870) | Chassis reset package false scoreboard error: Clarification required |  | |
| 2015WW25 |  | Replace script compiling and running of model/tests with ace flows.   |  |  |  | | --- | --- | --- | | **HSDes** | **Description** | **Comment** | | 1204719334 | Unexpect assertion firing when reset deasserts. | One more term added to assertion to qualify rising/faling edge. | | 1404190277 | Change to support chassis reset messages for non PGCB IPs in chassis\_rst\_pkg random mode. | Added additional argument when getting SB registration. | |
| 2017WW12 |  | Summary:   * Adding VVN\_ACK/VNN\_REQ port and driving capability   Adding Ip-pmc-vnn-req/pmc-ip-vnn-ack interface per ADDSIP  For driving above signals adding cmd: VNN\_ACK\_DSD/VNN\_ACK\_ASD/VNN\_REQ\_ASD/VNN\_REQ\_DSD |
| 2017WW25 |  | Summary:  -HSD: [1405863579](https://hsdes.intel.com/appstore/article/" \l "/1405863579)  -Added feature to responde VNN\_ACK as a part of auto response |

# Introduction

The ChassisPowerGatingVIP verification component should be used to validate an IPs (SIP, Fabric) Chassis defined power gating interface. It is a System Verilog OVM component. It consists of CCAgent (central controller agent) to emulate the PMC’s power gating central controller.

The user can configure the number of SIP, Fabric and delays using pamameters, configuarion objects as well as contrainted-random transactions. This VIP will also consists of a monitor that scoreboards can subscribe to, a checker to check the Chassis defined power gating protocols and coverage collector.

This agent does not assert/deassert prim and side resets to the IP.

## Terminology

List the term with specific meanings used in this specification. This section can be found in respective design specification.

The following terms have specific meanings in the PowerGating VC specification and Agent.

|  |  |
| --- | --- |
| Terminology | Meaning |
| IP and SIP | IP and SoftIP are used interchangeably in this document |
| CC | Power Gating Central Controller in the PMC of the SOC |
| PGCB | Power Gating Control Block as mentioned in the Chassis PM Arch spec |
| BFM | Bus Functional Model of an IP. |
| ***Agent*** | ***It is an ovm\_agent that consists of the BFM and Monitor. The BFM can be set to active or passive mode using the is\_active.***  ***This should not be confused with IOSF Agents. The doc specifies them as IOSF Agent wherever applicable.*** |
| PG | Power Gate |
| UG | Power Ungate |
| PGD | Power Gated Domain. It refers to a SIP or fabric domain with an instance of the PGCB – it has a unique interface with the PMC.  Multiple PGDs can be under the same FET block.  Multiple PGDs can be under the same SW visible entity and therefore controlled by the same bit in PMC. |

## Tool Support

To file request on new features, report problems, raise issues, please take a minute to fill up the HSD form here :

**Issue Reporting:** <https://vthsd.intel.com/hsd/seg_softip/#bug/default.aspx?ldudef=1>

* **Unit Name:** Chassis VIP.Power Gating CCAgent
* **Owner:**  aramaswa

You can call or e-mail a support representative to fill out a ticket for you, but response time may be slower.

Support Contacts

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Role | Name | User ID | Location | E-Mail | Telephone Number |
| Primary Owner | Danny Valdez | dbvalde1 | FM |  | 916-356-8485 |
| Secondary Owner | None |  |  |  |  |
| Original Developer | Alamelu Ramaswamy |  |  |  |  |
| Manager | Anurag Tyagi |  |  |  |  |

# Overview

This section provides an overview of how ChassisPowerGatingVIP is used in an OVM/AVM SystemVerilog Testbench. It shows how this component’s features allow tests written for low level Testbenches to be re-used at chip-level.

Explain by referring to following items :

* Applications
* Features
* Operations
* Control
* Bibliography

## Applications

Agents, interfaces, monitors, and coverage collectors are used to perform verification of Intellectual Property designs. Applications of Agents and related Verification IP include:

### IP and Fabric test environment

The CC BFM should be used to emulate the PMC behavior while validating an IP’s (including fabric’s) Chassis defined power gating interface. Please see diagram below. The driver and sequencer will be active only at the cluster level and will be passive in the chip/full-chip level. The monitor and checker will be active at both cluster and full-chip level.

The figure below shows an IP validation environment which also uses the CCU BFM for clocking and vcc modeling BFM for UPF. The Power gating CC BFM will respond with ack when the IP(DUT) asserts/deasserts the pg\_req. It also drives the fet\_en\_b that should be used in UPF. The fet\_en\_ack is an input to the BFM which needs to be driven by the ack port in UPF.

Note that the environment needs to exercise randomizing the delay in ISM handshake to emulate the scenario where the fabric also was power gated when the IP requested ungating. (Check with the IOSF Fabric BFM owner on how to do this).

Figure 2 shows the usage model in a fabric test environment. The figure below shows an fabric validation environment which also uses the CCU BFM for clocking and vcc modeling BFM for UPF.



**Figure 1 Example usage model for the CCAgent BFM in the SIP env**



**Figure 2 Example usage model for the CCAgent BFM in the Fabric env**

## Features

|  |  |  |
| --- | --- | --- |
| Feature | Supported in this version? | Expected release date |
| Supports upto maximum of 128 SIP, fabric and FET blocks each. | Yes |  |
| Give user the ability to specify  Initial states of IP.  PGCB to FET block mapping since multiple PGCBs can belong to one FET block.  Ungate request priority for arbitration  Total number of SW visible entities  Total number if PMC wakes driven by PMC. | Yes |  |
| Mastering capabilities with configurable delay  Send SW power gating request - SW\_PG\_REQ  Send PMC wake - PMC\_SIP\_WAKE  FAB\_PG\_REQ, FAB\_UG\_REQ – Fabric PG/UG req | Yes |  |
| Mastering capabilities   1. PMC\_SIP\_WAKE\_ALL – this command would assert the pmc\_ip\_pg\_wake signal for all the SIPs. 2. SET\_FET\_ON\_MODE – this command is used to give the user the ability to keep a particular FET from turned off. 3. RESET\_FET\_ON\_MODE | Yes |  |
| Slave response to   1. SIP PG/UG req 2. Restore commands 3. Fabric enter/exit idle | Yes |  |
| Assertion and deasserton of fet\_en\_b | Yes |  |
| * SIP/fabric dependencies mapping for waking the fabric from PG state when there is a SIP wake. | Yes |  |
| Randomly take away power to a PGD if it requests UG soon after its PG request is acked.  See figure 5 for details on the flow. | Yes |  |
| Mode where the user can keep a particular fet ON even when all the conditions for turning off the fet are satisfied. This can be set using the command SET\_FET\_ON\_MODE. | Yes |  |
| Arbitration of requests  Please see the next section and the timing diagram for details on arbitration. | Yes |  |
| Assertion of resets for IP Inaccessible flow.  There is no plan to handle assertion and deassertion of the ip\_prim\_rst and ip\_side\_rst in this BFM since the assertion/de-assertion flow is SOC dependent.  Are are specific/known models that can be given as options in the BFM to drive the resets? | No | none |
| Monitor and tracker | Yes | ww43 |
| VNN ack assertion and deassertion | No | TBD |

#### Assumptions

|  |  |
| --- | --- |
|  | Assumption |
|  | Tests will always end after a SIP is either in a PG state or UG state. This assumption is made by the checker. |

#### Arbitration

Maintain 4 queues that get populated when requests are seen by the FSM

Fabric UG request

SIP UG request

Fabric PG requests

SIP PG request

1. Wait for driver to finish previous command (that is the arbitration point).
   1. Please see below details on the different modes for arbitration.
2. Check the queue in the order mentioned above and move to step 3 if not empty. If all are empty, go to step 1. Note that UG requests get priority over PG requests and fabric request get priority over SIP requests.
3. Select a request randomly from the queue.
4. Send to the sequencer and delete the request.
5. Go to step 1.

PCH arbitration model (this would be used in BXT also)

* 1. In this mode, the CCAgent will only process one PG/UG request at a time.
  2. A request starts with the SIP/Fabric requesting it and ends with power down the FET for power down and deassertion of ack for SIP or pmc\_fab\_pg\_rdy\_req\_b for Fabric for power up.

|  |  |  |
| --- | --- | --- |
|  | Start of request | End of request (next request is selected here) |
| SIP UnPG | Deeasertion of pg\_req\_b | Deassertion of pg\_ack\_b |
| SIP PG | Assertion of pg\_req\_b | Deassertion of fet\_en\_ack\_b  Or  Assertion of pg\_ack\_b if the FET block is ot ready to be turned off |
| Fabric UnPG | Fabric exit idle  or  test command to ungate the fabric | Deasertion of pmc\_fab\_pg\_rdy\_req\_b |
| Fabric PG | fab\_pmc\_pg\_rdy\_ack\_b assertion | If FET block is not ready to be turned off, then the next request can be processed immediately  Or  Deassertion of fet\_en\_ack\_b |

* 1. No other pg\_ack\_b or fet\_en\_b are asserted/deasserted in between.
  2. User still has the ability to override the delays. See section 4 for details on how to override the response delays.
  3. Gate requests are serviced First-in-first-done including the fabric.
  4. Ungate requests get priority over gate requests.
  5. Among ungate requests, the user has the ability to program the priority per PGCB (see section 10 for details on configuration object).

#### Fabric gating and wake conditions

Using configuration objects, user can specify which SIP PGCB are mapped to which Fabric. The CCAgent uses this mapping information to gate or ungate the fabric by asserting/deassering pmc\_fab\_pg\_rdy\_req\_b as follows.

Assert pmc\_fab\_pg\_rdy\_req\_b (power gate)– if the fabric is in idle and all the SIPs that are mapped to the fabric is also power gated. (SIP is power gated when the ip\_pmc\_pg\_req\_b is asserted)

Deassert pmc\_fab\_pg\_rdy\_req\_b (power ungate) – if fabric gets out of idle or if any of the SIP that is mapped to this fabric requests power ungating. (SIP requesr ungatung by deassertion of ip\_pmc\_pg\_req\_b)

#### Waveforms

##### SIP Master command



**Figure 3 SIP master command example**

The above waveform shows the following command.

The pmc wake signal get driven 1 clock after the command is sent. The sequence ends when (see section 11 to see what wait for compelte means for each command) pmc\_ip\_pg\_ack\_b deasserts. But it finishes and returns after counting down delayComplete which is set to 2.

cmd == PMC\_SIP\_WAKE; source ==1, delay ==1, waitForComplete ==1, delayComplete = 2

Note that the BFM does not automatically deassert pmc\_ip\_pg\_wake. The test needs to send a command DEASSERT\_PMC\_WAKE.

##### Restore flow



The above waveform shows the following command.

cmd == SIP\_RESTORE\_NEXT\_WAKE; source ==1

The SIP\_RESTORE\_NEXT\_WAKE command will assert pmc\_ip\_restore\_b during the next wake; that is after deassertion of ip\_pmc\_pg\_req\_b and before pmc\_ip\_pg\_ack\_b. The delay\_restore parameter in the response sequence item is applied as shown in the waveform.

Note that the testbench/test needs to send restore cycles on IOSF sideband or primary and deassert restore signal.

Restore flow that should be implemented in the test/env

* + 1. Sample pmc\_ip\_restore\_b when pmc\_ip\_pg\_ack\_b deasserts. If pmc\_ip\_restore\_b is asserted, continue.
    2. Wait for side\_pok and prim\_pok to assert, side\_rst\_b and prim\_rst\_b to deassert and also wait for the IP to send IP\_READY message (if applicable) or early\_boot\_done signal to assert (if applicable).
    3. Then send restore cycles on IOSF sideband or IOSF primary.
    4. Deassert pmc\_ip\_restore\_b using the bfm command DEASSERT\_SIP\_RESTORE.

Note that the sequence with command PMC\_SIP\_WAKE ends when (see section 11 to see what wait for compelte means for each command) pmc\_ip\_pg\_ack\_b deasserts. But it finishes and returns after counting down delayComplete which is set to 2.

There is also a SIP\_RESTORE command that the user can use to assert pmc\_ip\_restore\_b anytime.

##### SIP requests arbitration



**Figure 4 SIP arbitration timing diagrams**

Description of figure 4

1. Consider 3 SIP PGCBs under 3 separate FET blocks.
2. PGCB0 requests PG.
3. While the CCAgent is in the process of acking it, test sends a command to wake PGCB1 using the pmc wake signal.
4. PGCB2 also requests PG.
5. The CCAgent finishes (by waiting for fet\_en\_ack\_b) the PG flow for PGCB0. This is the arbitration point where the agent arbitrates between the requests. See previous section for arbitration.
6. The UG request from PGCB1 is choosen first. The fet\_en\_b for PGCB1 is asserted on the next clock.
7. The arbitration point for UG flow is deassertion on pmc\_ip\_pg\_ack\_b.
8. The CCagent asserts pmc\_ip\_pg\_ack\_b for PGCB2 on the next clock.
9. See section 10 for details on how to configure the delays mentioned in the waveform

##### DFX support

The VC provides commands to assert and deassert the dfx signals fdfx\_pgcb\_bypass and fdfx\_pgcb\_ovr. There are no checks on these signals. User can call these commands to emulate the flow mentioned in the PGCB integration guide. The VC also provides coverage to ensure IPs test the combinations 10 and 11.

FDFX\_BYPASS\_ASD,

FDFX\_BYPASS\_DSD,

FDFX\_OVR\_ASD,

FDFX\_OVR\_DSD,

Please refer to the example test DfxTest in verif/tests area.

##### Power down FET block after UG request if received

**Figure 5 Power down the FET block after ungate request is received**



The same applies to fabric power gating request also. If the fabric exits idle right after it sends ack, the CCAgent will choose to randomly deassert the fet\_en\_b.

##### Fabric requests arbitration

TBD

1. Consider a Fabric interface where the fabric exits idle.
2. The BFM will wait for delay\_fab\_ug\_req and deassert pmc\_fab\_pg\_rdy\_req\_b.
3. It will then wait for the corresponding ip\_pmc\_pg\_req\_b to deassert.
4. After that the fet\_en is sequenced and pmc\_ip\_pg\_ack\_b is deasserted.
5. If waitForComplete is set, the BFM will wait for fab\_pmc\_pg\_rdy\_ack\_b to deassert.

##### Fabric wake due to sip wake

Consider the case where SIP PGCB0 is mapped to Fabric PGCB0 (using config object. See section 10) and the SIP PGCB has higher priority than the Fabric PGCB. The CCAgent first grants SIP PGCB and then also wakes the fabric.

TBD

## Control

This verification component has three basic levels of control:

* Parameters
  + Parameters are used to set the number of SW entities, SIP, Fabric PGDs and FET blocks.
* Configuration objects
  + Configuration object PowerGatingConfig is used to configure the SIP and fabric behaviors.
* Transactions
  + Transactions are sent by test during runtime to assert/deassert signals.

In general, parameters and configuarion objects are set once per testbench, configuration methods are set once per test, and transactions are set many times during a test.

## Requirements

### Specifications and Reference

|  |  |
| --- | --- |
| Document | Description |
| Chassis Power Gating PM Arch spec | <https://sharepoint.amr.ith.intel.com/sites/MDGArchMain/Converged/chassisWG/_layouts/WordViewer.aspx?id=/sites/MDGArchMain/Converged/chassisWG/HAS%20Releases/Chassis%20Power%20Managment%20uArch%20Rev%200.70Final.docx&Source=https%3A%2F%2Fsharepoint%2Eamr%2Eith%2Eintel%2Ecom%2Fsites%2FMDGArchMain%2FConverged%2FchassisWG%2FHAS%2520Releases%2FForms%2FAllItems%2Easpx&DefaultItemOpen=1> |

### Compute Environment

In order to use the package the following tools, software and operating systems are required.

* + Simulators the package can be used with.
    - Synopsis VCS
  + It is called Chassis Power Gating VIP.

### System Verilog Packages

*ovm\_pkg – System Verilog base framework*

*sla\_pkg – Saola package*

## Architecture

### Class/Component Descriptions



**Power Gating CCAgent block diagram**

# Getting Started

See integration guide

# Setting Up a Testbench Environment

See integration guide

# Implementing Test Scenarios

See integration guide

# Monitoring and Checking the Protocol

This section shows how to monitor and check bus protocol using assertions, monitor/tracker and coverage collector. The previous section explained how to create random transaction sequences and test scenarios. The focus of this section is how to ensure tests and transactions behave properly.

1. **Signal-level interface compliance:** These are the checks that have been implemented in the interface file as assertions to check interface protocols.

|  |  |  |  |
| --- | --- | --- | --- |
| **Check Category** | **Rules** | **Covered where?** | **Implemented in Power Gating Checker in the latest release?** |
| SIP h/s | <ip>\_pmc\_pg\_req\_b must deassert only if pmc\_<ip>\_pg\_ack\_b is asserted. | PowerGating Checker assertion | Yes |
| SIP h/s | <ip>\_pmc\_pg\_req\_b must assert only if pmc\_<ip>\_pg\_ack\_b is deasserted. | PowerGating Checker assertion | Yes |
| SIP h/s | pmc\_<ip>\_pg\_ack\_b must deassert only if <ip>\_pmc\_pg\_req\_b is deasserted. | PowerGating Checker assertion | Yes |
| SIP h/s | pmc\_<ip>\_pg\_ack\_b must assert only if <ip>\_pmc\_pg\_req\_b is asserted. | PowerGating Checker assertion | Yes |
| Future ack | Any ip\_pmc\_pg\_req\_b assertion must be followed by pmc\_ip\_pg\_ack\_b assertion by end of test | PowerGating Checker assertion | Yes |
| Future ack | Any ip\_pmc\_pg\_req\_b deassertion must be followed by pmc\_ip\_pg\_ack\_b deassertion by end of test | PowerGating Checker assertion | Yes |
| pmc\_wake | If not already deasserted, ip\_pmc\_pg\_req\_b must deassert in response to pmc\_ip\_pg\_wake. | PowerGating Checker S/M check | Yes |
| ~~pmc\_wake~~ | ~~ip\_pmc\_pg\_req\_b must assert only if pmc\_ip\_pg\_wake is deasserted.~~ | PowerGating Checker S/M check | ~~No~~ |
| Inaccessible | If IP is in Inaccessible PG state (ip\_pmc\_pg\_req\_b & pmc\_ip\_pg\_ack\_b == 0 && all pok = 0), ip\_pmc\_pg\_req\_b must deassert only in response to pmc\_ip\_pg\_wake assertion. | PowerGating Checker S/M check | Yes |
| Restore | If pmc\_ip\_restore\_b was asserted when pmc\_ip\_pg\_ack\_b deasseted, subsequently ip\_pmc\_pg\_req\_b must assert only after pmc\_ip\_restore\_b is deasserted. | PowerGating Checker assertion | Yes |
| Fet h/s | fet\_en\_b must deassert only if fet\_en\_ack\_b is asserted. | PowerGating Checker assertion | Yes |
| Fet h/s | fet\_en\_b must assert only if fet\_en\_ack\_b is deasserted. | PowerGating Checker assertion | Yes |
| Fet h/s | fet\_en\_ack\_b must deassert only if fet\_en\_b is deasserted. | PowerGating Checker assertion | Yes |
| Fet h/s | fet\_en\_ack\_b must assert only if fet\_en\_b is asserted. | PowerGating Checker assertion | Yes |
| Fet | fet\_en\_b must assert only if at least one ip\_pmc\_pg\_req\_b in that Fet block is deasserted. | PowerGating Checker S/M check | Yes |
| Fet | If pmc\_ip\_pg\_ack\_b is deasserted, the corresponding fet\_en\_b and fet\_en\_ack\_b must be asserted. | PowerGating Checker S/M check | Yes |
| Fet | fet\_en\_b must deassert only if all PGDs in that fet block has asserted ip\_pmc\_pg\_ack\_b. | PowerGating Checker S/M check | Yes |
| ~~Fabric h/s~~ | ~~PMC must assert pmc\_fab\_pg\_rdy\_req\_b only when fab\_pmc\_idle is 1~~ | ~~PowerGating Checker assertion~~ | ~~No~~ |
| Fabric h/s | fab\_pmc\_pg\_rdy\_ack\_b and fab\_pmc\_pg\_rdy\_nack\_b must not be asserted at the same time. | PowerGating Checker assertion | Yes |
| Fabric h/s | pmc\_fab\_pg\_rdy\_req\_b must deassert only if fab\_pmc\_pg\_rdy\_ack\_b or fab\_pmc\_pg\_rdy\_nack\_b is asserted. | PowerGating Checker assertion | Yes |
| Fabric h/s | pmc\_fab\_pg\_rdy\_req\_b must assert only if fab\_pmc\_pg\_rdy\_ack\_b and fab\_pmc\_pg\_rdy\_nack\_b are deasserted. | PowerGating Checker assertion | Yes |
| Fabric h/s | fab\_pmc\_pg\_rdy\_ack\_b must deassert only if pmc\_fab\_pg\_rdy\_req\_b is deasserted. | PowerGating Checker assertion | Yes |
| Fabric h/s | fab\_pmc\_pg\_rdy\_ack\_b must assert only if pmc\_fab\_pg\_rdy\_req\_b is asserted. | PowerGating Checker assertion | Yes |
| Fabric h/s | fab\_pmc\_pg\_rdy\_nack\_b must deassert only if pmc\_fab\_pg\_rdy\_req\_b is deasserted. | PowerGating Checker assertion | Yes |
| Fabric h/s | fab\_pmc\_pg\_rdy\_nack\_b must assert only if pmc\_fab\_pg\_rdy\_req\_b is asserted. | PowerGating Checker assertion | Yes |
| Fabric h/s | Any pmc\_fab\_pg\_rdy\_req\_b assertion must be followed by fab\_pmc\_pg\_rdy\_ack/nack\_b assertion by end of test | PowerGating Checker assertion | Yes |
| Fabric h/s | Any pmc\_fab\_pg\_rdy\_req\_b deassertion must be followed by fab\_pmc\_pg\_rdy\_ack/nack\_b deassertion by end of test | PowerGating Checker assertion | Yes |

# Agent Packages

The package can be imported as shown below.

import CCAgentPkg::\*;

# Agent Parameters

Show examples on using parameters to configure the signal width, etc in the PowerGatingIF.

parameter int NUM\_SIP\_PGCB = 1;

parameter int NUM\_FET = 1;

parameter int NUM\_SW\_REQ = 1;

parameter int NUM\_PMC\_WAKE = 1;

parameter int NUM\_FAB\_PGCB = 1;

parameter bit NO\_SIP\_PGCB = 0;

parameter bit NO\_FAB\_PGCB = 0;

parameter int NUM\_SB\_EP = 1;

parameter int NUM\_PRIM\_EP = 1;

parameter int NUM\_VNN\_ACK\_REQ = 1;

parameter int NUM\_D3 = 1;

parameter int NUM\_D0I3 = 1;

parameter bit NO\_PRIM\_EP = 0;

parameter bit IS\_ACTIVE = 1;

parameter string IP\_ENV\_TO\_CC\_AGENT\_PATH = “”;

|  |  |
| --- | --- |
| parameter | Description |
| NUM\_SIP\_PGCB | Total number of SIP PGCBs. If there are no SIP PGCBs in the test env, then set NO\_SIP to 1.  The pg handshakes will be one per SIP PGCB |
| NUM\_FET | This is the numbe of FET blocks. Fet\_en\_b and fet\_en\_ack\_b will be one per FET block  Using configuration object, the user can map different PGCBs to FET blocks. |
| NUM\_SW\_REQ | Number of SW PG requests |
| NUM\_PMC\_WAKE | Number is PMC wake signals |
| NUM\_FAB\_PGCB | Total number of fabric PGCBs. If there are no SIP PGCBs in the test env, then set NO\_FAB\_PGCB to 1.  The fabric pg req, ack and nack will be one per fabric PGCB |
| NO\_SIP\_PGCB | Set to 1 if there are not SIP PGCBs in the test env. |
| NO\_FAB\_PGCB | Set to 1 if there are not Fabric PGCBs in the test env. |
| IS\_ACTIVE | This parameter should be set to 0 when the agent is promoted to an environment where the actual PMC is present. This should match the Agent’s is\_active config. |
| NUM\_SB\_EP | The number of sideband endpoints |
| NUM\_PRIM\_EP | The number of primary endpoints |
| NUM\_VNN\_ACK\_REQ | The numer of VNN\_ACK/REQ sigs |
| NUM\_D3 | The number of ip\_pmc\_d3 signals |
| NUM\_D0I3 | The number of ip\_pmc\_d0i3 signals |
| NO\_PRIM\_EP | Set this to 1 if there are no primary endpoints |
| IP\_ENV\_TO\_CC\_AGENT\_PATH | This parameter specifies the full hierarchy of the CCAgent instance starting from the IP’s env name. The hierarchy should be specified in the form \*<Env’s OVM name>.<CCAgent OVM name>.. Please see integration guide for more details. |

# Agent Parameterized Interface

List the signal interface supported in this Agent.

## PowerGatingIF signals

logic clk;

logic jtag\_tck;

logic reset\_b;

logic[NUM\_SW\_REQ-1:0] pmc\_ip\_sw\_pg\_req\_b;

logic[NUM\_SIP\_PGCB-1:0] ip\_pmc\_save\_req\_b;

logic[NUM\_SIP\_PGCB-1:0] pmc\_ip\_save\_ack\_b;

logic[NUM\_SIP\_PGCB-1:0] pmc\_ip\_restore\_b;

logic[NUM\_SIP\_PGCB-1:0] ip\_pmc\_pg\_req\_b;

logic[NUM\_SIP\_PGCB-1:0] pmc\_ip\_pg\_ack\_b;

logic[NUM\_PMC\_WAKE-1:0] pmc\_ip\_pg\_wake;

logic[NUM\_SB\_EP-1:0] side\_pok;

logic[NUM\_PRIM\_EP-1:0] prim\_pok;

logic[NUM\_FAB\_PGCB-1:0] fab\_pmc\_idle;

logic[NUM\_FAB\_PGCB-1:0] pmc\_fab\_pg\_rdy\_req\_b;

logic[NUM\_FAB\_PGCB-1:0] fab\_pmc\_pg\_rdy\_ack\_b;

logic[NUM\_FAB\_PGCB-1:0] fab\_pmc\_pg\_rdy\_nack\_b;

logic[NUM\_FET-1:0] fet\_en\_b;

logic[NUM\_FET-1:0] fet\_en\_ack\_b;

logic[NUM\_SIP\_PGCB-1:0] fdfx\_pgcb\_bypass;

logic[NUM\_SIP\_PGCB-1:0] fdfx\_pgcb\_ovr;

logic[NUM\_VNN\_ACK\_REQ-1:0] ip\_pmc\_vnn\_req;

logic[NUM\_VNN\_ACK\_REQ-1:0] pmc\_ip\_vnn\_ack;

logic[NUM\_D3-1:0] ip\_pmc\_d3;

logic[NUM\_D0I3-1:0] ip\_pmc\_d0i3;

# Configuration Methods for Parameterized Interface

This section describes methods (functions) in Agents that can be called by tests or Testbenches. Most methods are intended to be called once at during the configure phase of a test, and most return a single bit one (1) upon success. As SystemVerilog functions, they execute in zero simulation time.

The Agents generally follow a rule that a function called without parameters applies to all possible values of the parameters.

For examples of how to call the configuration methods from a test see Section 5.

## Configuring the Agent

### PowerGatingConfig

The PowerGatingConfig ovm\_object is used to configure the CC agents.

List of config object APIs and their parameters.

|  |  |  |
| --- | --- | --- |
| Function Name | Parameter(s) | Description |
| AddFETBlock | int index  string name | The FET block index number and the name of the fet block.This function needs to be called before the AddSIPPGCB and AddFabricPGCB function |
| SetTrackerName | string name | Name of the tracker.The printer will add .out to the name. .  The default name for the tracker is PG\_TRACKER. |
| DisableConfigPrinting | -- | The tracker prints out configuration information at the beginning of the test.  This function disables printing the configuration information. |
| SetRandomPriorityMode | - | This will ignore the priority and randomly select ungate requests. |
| AddFabricPGCB | int num | The fabric index number |
| string name | The fabric name which would be used in the printer while printing into the tracker.  To keep the tracker formatting clean, the name should be restricted to 4 letters. |
| int fet\_index = 0 | The FET block index number this fabric PGD is associated with.  The default is 0.  Note that this is redundant now since the fet\_index is inferred from the SIP interface thati s associated with this fabric. |
| PowerGating::InitialState initial\_state = PowerGating::POWER\_GATED | POWER\_GATED – default. Initial state of PGCB is power gated state.  POWER\_UNGATED – initial state of IP/PGCB is un-gated state.  The CCAgent will drive the correct reset values on all its output signals based on the initial state. |
| time hys = 0ps | This specifies the amount of time between seeing the fabric enter idle and asserting the pg\_req to the fabric. If the fabric exits idle in the meantime, the agent will not assert pg\_req.  Default is 0ps |
| int array sip\_pgcb\_dependency = [] | Specify list of agents on which the fabric has a dependency.  See section 2.2.1.3 to know how the CCAgent uses this information. |
| int ungate\_priority | The priority of this PGCB’s ungate request.  1 <= ungate\_priority <= NUM\_SIP\_PGCB + NUM\_FAB\_PGCB - 1  1 – highest priority  NUM\_SIP\_PGCB + NUM\_FAB\_PGCB – lowest priority  Each ungate\_priority should be unique. |
| AddSIPPGCB | int index | The PGCB index number |
| string name | The PGCB name which would be used in the printer while printing into the tracker.  To keep the tracker formatting clean, the name should be restricted to 4 letters. |
| PowerGating::InitialState initial\_state = PowerGating::POWER\_GATED | POWER\_GATED – default. Initial state of IP/PGCB is IP-inacceessible state state.  POWER\_UNGATED – initial state of IP/PGCB is un-gated state.  The CCAgent will drive the correct reset values on all its output signals based on the initial state. |
| int fet\_index = 0 | The FET block index number this PGCB is associated with.  The default is 0. |
| int ungate\_priority | The priority of this PGCB’s ungate request.  1 <= ungate\_priority <= NUM\_SIP\_PGCB + NUM\_FAB\_PGCB - 1  1 – highest priority  NUM\_SIP\_PGCB + NUM\_FAB\_PGCB – lowest priority  Each ungate\_priority should be unique. |
| int sw\_ent\_index | The index number of the pmc\_ip\_sw\_pg\_req\_b the PGCB is connected to |
| int pmc\_wake\_index | The index number of the pmc\_ip\_pg\_wake the PGCB is connected to |
| int array SB\_array | The index array of all the sideband endpoints inside the PGD tha tis controlled by this PGCB |
| int array prim\_array | The index array of all the sideband endpoints inside the PGD tha tis controlled by this PGCB |
| int fabric\_index = -1 | This specifies if this SIP interface is part of a fabric interface. If left at -1 (default value), then this SIP interface is not part of a fabric interface.  Otherwise, user needs to specify the index of the fabric interface this SIP interface belongs to.  NOTE: Currently no error is reported if the fabric is configured without a corresponding SIP. Note that the PSF still allows for the old fabric interface. |
|  | bit initial\_restore\_asserted = 0 | Set this to 1 if you want pmc\_ip\_restore\_b to be asserted by default. |
| AddSIP | string name | The SIP name which would be used in the printer while printing into the tracker.  To keep the tracker formatting clean, the name should be restricted to 4 letters. |
| PowerGating::SIPType | CSME  HOST  DUAL  TODO: clarify usage model |
| int array PGCB\_array | The index array of all the PGCBs in this SIP |
| int array AON\_SB\_array | The index array of all the AON Sideband endpoints if any |
| int array AON\_prim\_array | The index array of all the AON Primary endpoints if any |
| int array d3[] | Specifies the interface indices of all the ip\_pmc\_d3 signals that belong ot this SIP |
| int array d0i3[] | Specifies the interface indices of all the ip\_pmc\_d30i signals that belong ot this SIP |
|  | vnn\_ack\_req\_index | Pass index of ip-pmc-vnn-req/pmc-ip-vnn-ack index |
| AddSBEP | int index | The signal index of this sideband endpoint pok signal |
| bit[7:0] source\_id | No usage model as of now |
| bit AON\_EP | Set to 1 if the endpoint is in AON domain |
| int pmc\_wake\_index | * Only applicable if the EP belong to an AON domain. * Species the pmc\_wake signal index that is connected to this EP. |
| bit boot\_prep\_early = 0, | This can be used to specify if this endpoint subscribes to this message b setting it to 1 |
| bit ip\_ready = 0 | This can be used to specify if this endpoint subscribes to this message b setting it to 1 |
| bit boot\_prep\_general = 0, | This can be used to specify if this endpoint subscribes to this message b setting it to 1 |
| bit reset\_prep\_reset\_start = 0, | This can be used to specify if this endpoint subscribes to this message b setting it to 1 |
| bit reset\_prep\_general = 0, | This can be used to specify if this endpoint subscribes to this message b setting it to 1 |
| bit reset\_prep\_link\_turnoff = 0 | This can be used to specify if this endpoint subscribes to this message b setting it to 1 |
| AddPrimEP | int index | The signal index of this primary endpoint pok signal |
| bit[15:0] req\_id | No usage model as of now |
| bit AON\_EP | Set to 1 if the endpoint is in AON domain |
| int pmc\_wake\_index | Only applicable if the EP belong to an AON domain.  Species the pmc\_wake signal index that is connected to this EP. |

# Agent Non-Parameterized Interface and test-island

interface PowerGatingResetIF;

logic clk;

logic reset\_b;

endinterface

interface PowerGatingSIPIF;

parameter int NUM\_SIDE = 1;

parameter int NUM\_PRIM = 1;

parameter int NUM\_D3 = 1;

parameter int NUM\_D0I3 = 1;

logic clk;

logic reset\_b;

logic pmc\_ip\_sw\_pg\_req\_b;

logic ip\_pmc\_pg\_req\_b;

logic pmc\_ip\_pg\_ack\_b;

logic pmc\_ip\_restore\_b;

logic pmc\_ip\_pg\_wake;

logic[NUM\_SIDE-1:0] side\_pok;

logic[NUM\_PRIM-1:0] prim\_pok;

logic[NUM\_SIDE-1:0] side\_rst\_b;

logic[NUM\_PRIM-1:0] prim\_rst\_b;

logic[NUM\_D3-1:0] ip\_pmc\_d3;

logic[NUM\_D0I3-1:0] ip\_pmc\_d0i3;

logic fdfx\_pgcb\_bypass;

logic fdfx\_pgcb\_ovr;

logic jtag\_tck;

logic restore\_next\_wake;

logic fet\_en\_b;

logic fet\_en\_ack\_b;

endinterface: PowerGatingSIPIF

interface PowerGatingFabricIF;

logic clk;

logic reset\_b;

logic fab\_pmc\_idle;

logic pmc\_fab\_pg\_rdy\_req\_b;

logic fab\_pmc\_pg\_rdy\_ack\_b;

logic fab\_pmc\_pg\_rdy\_nack\_b;

endinterface: PowerGatingFabricIF

Each instance of these interfaces must have a corresponding test-island instantiation. The TI module defition is as follows. User must set the parameters including the ‘NAME’ and make sure it matches the configuration.

module PowerGatingResetTI(PowerGatingResetIF intf);

parameter string IP\_ENV\_TO\_AGENT\_PATH = "";

module PowerGatingSIPTI(PowerGatingSIPIF intf);

parameter string NAME = "";

//parameter int INDEX = 0; //for backward compatibility

parameter int NUM\_SIDE = 1;

//TODO: add a NO\_PRIM parameter

parameter int NUM\_PRIM = 1;

parameter int NUM\_D3 = 1;

parameter int NUM\_D0I3 = 1;

parameter string FET\_NAME = "";

parameter string FABRIC\_NAME = "";

//parameter bit BFM\_DRIVES\_POK = 1; //deprecated

parameter string IP\_ENV\_TO\_AGENT\_PATH = "";

module PowerGatingFabricTI(PowerGatingFabricIF intf);

parameter string NAME = "";

parameter string IP\_ENV\_TO\_AGENT\_PATH = "";

# Configuration Methods for Non-Parameterized Interface

### PowerGatingConfig

The PowerGatingConfig ovm\_object is used to configure the CC agents.

List of config object APIs and their parameters.

|  |  |  |
| --- | --- | --- |
| Function Name | Parameter(s) | Description |
| SetTrackerName | string name | Name of the tracker.The printer will add .out to the name. .  The default name for the tracker is PG\_TRACKER. |
| SetRandomPriorityMode | - | This will ignore the priority and randomly select ungate requests. |
| AddFabricPGCB | string name | The fabric name as specified in the PowerGating FabricTI instance  To keep the tracker formatting clean, the name should be restricted to 4 letters. |
| PowerGating::InitialState initial\_state = PowerGating::POWER\_GATED | POWER\_GATED – default. Initial state of PGCB is power gated state.  POWER\_UNGATED – initial state of IP/PGCB is un-gated state.  The CCAgent will drive the correct reset values on all its output signals based on the initial state. |
| time hys = 0ps | This specifies the amount of time between seeing the fabric enter idle and asserting the pg\_req to the fabric. If the fabric exits idle in the meantime, the agent will not assert pg\_req.  Default is 0ps |
| int ungate\_priority | The priority of this PGCB’s ungate request.  1 <= ungate\_priority <= NUM\_SIP\_PGCB + NUM\_FAB\_PGCB - 1  1 – highest priority  NUM\_SIP\_PGCB + NUM\_FAB\_PGCB – lowest priority  Each ungate\_priority should be unique. |
| AddSIPPGCB | string name | The SIP PGCB name as specified in the PowerGatingSIPTI instance.  To keep the tracker formatting clean, the name should be restricted to 4 letters. |
| PowerGating::InitialState initial\_state = PowerGating::POWER\_GATED | POWER\_GATED – default. Initial state of IP/PGCB is IP-inacceessible state state.  POWER\_UNGATED – initial state of IP/PGCB is un-gated state.  The CCAgent will drive the correct reset values on all its output signals based on the initial state. |
| int ungate\_priority | The priority of this PGCB’s ungate request.  1 <= ungate\_priority <= NUM\_SIP\_PGCB + NUM\_FAB\_PGCB - 1  1 – highest priority  NUM\_SIP\_PGCB + NUM\_FAB\_PGCB – lowest priority  Each ungate\_priority should be unique. |
| logic[7:0] side\_pid[] | The index array of all the sideband endpoints inside the PGD tha tis controlled by this PGCB |
| int fabric\_name = “” | This specifies if this SIP interface is part of a fabric interface. If left unassigned, then this SIP interface is not part of a fabric interface.  Otherwise, user needs to specify the name of the fabric interface this SIP interface belongs to.  NOTE: Currently no error is reported if the fabric is configured without a corresponding SIP. Note that the PSF still allows for the old fabric interface. |
|  | bit initial\_restore\_asserted = 0 | Set this to 1 if you want pmc\_ip\_restore\_b to be asserted by default. |
| AddSBEP | logic[7:0] source\_id | No usage model as of now |
| bit boot\_prep\_early = 0, | This can be used to specify if this endpoint subscribes to this message b setting it to 1 |
| bit ip\_ready = 0 | This can be used to specify if this endpoint subscribes to this message b setting it to 1 |
| bit boot\_prep\_general = 0, | This can be used to specify if this endpoint subscribes to this message b setting it to 1 |
| bit reset\_prep\_reset\_start = 0, | This can be used to specify if this endpoint subscribes to this message b setting it to 1 |
| bit reset\_prep\_general = 0, | This can be used to specify if this endpoint subscribes to this message b setting it to 1 |
| bit reset\_prep\_link\_turnoff = 0 | This can be used to specify if this endpoint subscribes to this message b setting it to 1 |
| AddSIP | string name | The SIP name which would be used in the printer while printing into the tracker.  To keep the tracker formatting clean, the name should be restricted to 4 letters. |
| string pgcb\_name[] | The name of all the PGCBs in this SIP |

# Transaction Sequence Item and Base Sequence

This section describes transaction classes / OVM Sequence Item and tasks available to program the Agent.

## CCAgentSeqItem

Here is a description of the CCAgentSeqItem used to initiate and transmit cycles.

### Members

List the variable/parameter name of this class.

|  |  |  |
| --- | --- | --- |
| Variable Name | Type | Description |
| cmd | PowerGating::Event\_e | Specifies the command.  Possible values are specified below in the constraints |
| source | int | This is the index number of the SIP or Fabric PGD where the command should be executed.  The value should be < NUM\_SIP\_PGCB or NUM\_FAB\_PGCB or NUM\_SW\_REQ or NUM\_PMC\_WAKE  If the non-parameterized interfaces are used, then user can use the static method called PowerGatingConfig::getSIPPGCBIndex(string name) to convert the name of the SIP PGCB to index.  Example:  `ovm\_do(seq, {cmd == PowerGating::SIP\_PMC\_WAKE; source = PowerGatingConfig::getSIPPGCBIndex(“KVM”);}) |
| sourceName | string | If the user does not use the source, sourceName can be specified.  Example:  `ovm\_create(seq);  seq.sourceName = "KVM";  `ovm\_rand\_send\_with(seq, {cmd == PowerGating::PMC\_SIP\_WAKE;}) |
| delay | int | **Number of clock cycles to wait before executing the command** |
| delayComplete | int | After the sequence ends, wait this many number of clocks. Please see the waveforms for details on how **delayComplete is used.** |

### Constraints

|  |  |
| --- | --- |
| Constraint Name and Hierarchy | Description |
| delay\_c | delay >=0; delay < 20; |
| source\_c | source >= 0; source < 128; |
| cmd\_c | cmd inside  {  PowerGating::SW\_PG\_REQ,  PowerGating::DEASSERT\_SW\_PG\_REQ,  PowerGating::SIP\_PMC\_WAKE,  PowerGating::DEASSERT\_SIP\_PMC\_WAKE,  PowerGating::FAB\_PG\_REQ,  PowerGating::FAB\_UG\_REQ,  PowerGating::SIP\_PMC\_WAKE\_ALL,  PowerGating::DEASSERT\_SIP\_PMC\_WAKE\_ALL,  PowerGating::SET\_FET\_ON\_MODE,  PowerGating::RESET\_FET\_ON\_MODE ,  PowerGating::SIP\_RESTORE\_NEXT\_WAKE,  PowerGating::DEASSERT\_SIP\_RESTORE,  PowerGating::SIP\_RESTORE  PowerGating::FDFX\_BYPASS\_ASD,  PowerGating::FDFX\_BYPASS\_DSD,  PowerGating::FDFX\_OVR\_ASD,  PowerGating::FDFX\_OVR\_DSD  } |

### CCAgentBaseSequence

#### Members

List the variable/parameter name of this class.

|  |  |  |
| --- | --- | --- |
| Variable Name | Type | Description |
| cmd | PowerGating::Event\_e | Specifies the command.  Possible values are specified below in the constraints |
| source | int | This is the index number of the SIP or Fabric PGD where the command should be executed.  The value should be < NUM\_SIP\_PGCB or NUM\_FAB\_PGCB or NUM\_SW\_REQ or NUM\_PMC\_WAKE |
| delay | int | **Number of clock cycles to wait before executing the command** |
| delayComplete | int | **After the sequence ends, wait this many number of clocks. Please see the waveforms for details on how delayComplete is used.** |
| waitForComplete | Bit | **Wait for sequence to complete before proceeding. See the tabled below to know what wait for complete means for different commands** |

### waitForComplete

As mentioned above, users can also optionally set waitForComplete in the base sequence **to 1 if they want to wait for sequence to complete before proceeding. See the tabled below to know what wait for complete means for different commands**

### Commands

|  |  |  |  |
| --- | --- | --- | --- |
| Command | Parameters | Description | Wait for complete |
| SW\_PG\_REQ | int source  int delay  int delayComplete | This command will assert the pmc\_ip\_sw\_pg\_req\_b signal for the source specified after <delay> number of clocks.  Source can be >= 0 and < NUM\_SW\_REQ. | Returns after the signal is driven. |
| DEASSERT\_SW\_PG\_REQ | int source  int delay  int delayComplete | This command will deassert the pmc\_ip\_sw\_pg\_req\_b signal for the source specified after <delay> number of clocks.  Source can be >= 0 and < NUM\_SW\_REQ. | Returns after the signal is driven and delayComplete expires. |
| PMC\_SIP\_WAKE | int source  int delay  int delayComplete | This command will assert the pmc\_ip\_pg\_wake signal for the source specified after <delay> number of clocks.  Source can be >= 0 and < NUM\_PMC\_WAKE. | Returns after all the SIP PGCB connected to the specified pmc wake signal is in ungated state; that is the pmc\_ip\_pg\_ack\_b is deasserted. |
| DEASSERT\_PMC\_SIP\_WAKE | int source  int delay  int delayComplete | This command will assert the pmc\_ip\_pg\_wake signal for the source specified after <delay> number of clocks.  Source can be >= 0 and < NUM\_PMC\_WAKE. | Returns once all the signals are driven and delayComplete expires.. |
| PMC\_SIP\_WAKE\_ALL | int delayComplete | Will assert all the pmc\_ip\_pg\_wake signals with random delay between 0 to 20 clocks. | Returns after all the SIP PGCBs are in ungated state; that is the pmc\_ip\_pg\_ack\_b is deasserted. |
| DEASSERT\_PMC\_SIP\_WAKE\_ALL | int delayComplete | Will deassert all the pmc\_ip\_pg\_wake signals with random delay between 0 to 20 clocks. | Returns once all the signals are driven and delayComplete expires.. |
| PMC\_SIP\_WAKE\_TYPE  (NOT IMPLEMENTED YET) | PowerGating::SIPType sipType  int delayComplete | Specify which type of devices should be woken up – CSME, HOST or DUAL | Returns once all the signals are driven and delayComplete expires.. |
| SIP\_RESTORE\_NEXT\_WAKE | int source | This will assert pmc\_ip-restore\_b during the next wake (after ip\_pmc\_pg\_req\_b deassertion and before pmc\_ip\_pg\_ack\_b deassertion) for the source number specified. | Returns immediately |
| SIP\_RESTORE | int source  int delay  int delayComplete | This will assert pmc\_ip\_restore\_b | Returns after the signal is driven and delayComplete expires. |
| DEASSERT\_SIP\_RESTORE | int source  int delay  int delayComplete | This will deassert pmc\_ip\_restore\_b | Returns after the signal is driven and delayComplete expires. |
| VNN\_ACK\_ASD | int source  int delay  int delayComplete | This will deassert pmc\_ip\_vnn\_ack | Returns after the signal is driven and delayComplete expires. |
| VNN\_ACK\_DSD | int source  int delay  int delayComplete | This will assert pmc\_ip\_vnn\_ack | Returns after the signal is driven and delayComplete expires. |
| VNN\_REQ\_DSD | int source  int delay  int delayComplete | This will deassert pmc\_ip\_vnn\_ack | Returns after the signal is driven and delayComplete expires. |
| VNN\_REQ\_ASD | int source  int delay  int delayComplete | This will assert pmc\_ip\_vnn\_ack | Returns after the signal is driven and delayComplete expires. |
| FDFX\_BYPASS\_ASD | int source  int delay | Assert fdfx\_pgcb\_bypass  Source can be >= 0 and <= NUM\_SIP\_PGCB | Returns immediately after signal is driven |
| FDFX\_BYPASS\_DSD | int source  int delay | Dessert fdfx\_pgcb\_bypass  Source can be >= 0 and <= NUM\_SIP\_PGCB | Returns immediately after signal is driven |
| FDFX\_OVR\_ASD | int source  int delay | Assert fdfx\_pgcb\_ovr  Source can be >= 0 and <= NUM\_SIP\_PGCB | Returns immediately after signal is driven |
| FDFX\_OVR\_DSD | int source  int delay | Dessert fdfx\_pgcb\_ovr  Source can be >= 0 and <= NUM\_SIP\_PGCB | Returns immediately after signal is driven |
| FAB\_PG\_REQ | int source  int delay  int delayComplete | This command will assert the pmc\_fab\_pg\_rdy\_req\_b signal for the source specified after <delay> number of clocks  source can be >= 0 and < NUM\_FAB\_PGCB. | Returns after the signal is driven and delayComplete expires. |
| FAB\_UG\_REQ | int source  int delay  int delayComplete | This command will deassert the pmc\_fab\_pg\_rdy\_req\_b signal for the source specified after <delay> number of clocks.  source can be >= 0 and < NUM\_FAB\_PGCB. | Returns after the signal is driven and delayComplete expires. |
| SET\_FET\_ON\_MODE | int source | This command will set the mode where the FET block index specified using ‘source’ will never be turned off even if all the PGCBs under that FET block is in power gated state.  source can be >= 0 and < NUM\_FET | Returns immediately |
| RESET\_FET\_ON\_MODE | int source | This command will reset the FET on mode for the FET block index specified using ‘source’. At this point, if the FET is ready to be turned off, it will be turned off.  source can be >= 0 and < NUM\_FET | Returns immediately. |

## CCAgentResponseSeqItem

The response seq item can be used to control the behavior of the responses sent by the CCAgent.

### Members

List the variable/parameter name of this class.

Note that the arguments need to be passed by name. See system Verilog LRM for details on passing arguments by name.

|  |  |  |
| --- | --- | --- |
| Variable Name | Type | Description |
| cmd | PowerGating::Event\_e | Specifies the command. |
| source | int | This is the index number of the SIP or Fabric PGD where the command should be executed.  The value should be < NUM\_PGCB or NUM\_FAB |
| noResponse | bit | **When set, the responder will not send any responses.** |
| delay\_restore | Bit | **If the test writer sent a command to assert restore during the next wake, this is the number of clocks the driver waits before asserting pmc\_ip\_restore\_b** |
|  |  |  |
| delay\_pg\_ack | int | This is the delay in number of clocks the driver waits before asserting the pmc\_ip\_pg\_ack\_b in response to a pmc\_ip\_pg\_req\_b assertion |
|  |  |  |
| delay\_ug\_ack | int | This is the delay in number of clocks the driver waits before deasserting the pmc\_ip\_pg\_ack\_b in response to a pmc\_ip\_pg\_req\_b assertion |
| delay\_fab\_ug\_req | int | This is the delay in number of clocks the driver waits before deasserting the pmc\_fab\_pg\_rdy\_req\_b in response to fabric exiting idle.  The hysteresis is part of the config object. |
| delay\_fet\_en | int | This is the delay in number of clocks the driver waits before asserting fet\_en\_b after all the PGDs in that FET block is in PG state. |
| delay\_fet\_dis | int | This is the delay in number of clocks the driver waits before deasserting fet\_en\_b after any the PGDs in that FET block sends UG request. |

### Constraints

|  |  |  |
| --- | --- | --- |
| Constraint Name and Hierarchy | Description | |
| source\_c | source >= 0; source < 128; | |
| cmd\_c | none | |
| noResponse\_c | noResponse == 0; | |
| delay\_restore | dist {  [0:1] :/ 10,  [2:10] :/ 80,  [11:1000] :/ 10};  } | |
|  | | :/:/:/ |
| delay\_pg\_ack | | dist {  [0:1] :/ 10,  [2:10] :/ 80,  [11:1000] :/ 10};  } |
|  | |  |
| delay\_ug\_ack | | dist {  [0:1] :/ 10,  [2:10] :/ 80,  [11:1000] :/ 10};  } |
| delay\_fab\_ug\_req | | dist {  [0:1] :/ 10,  [2:10] :/ 80,  [11:1000] :/ 10};  } |
| delay\_fet\_en | | dist {  [0:1] :/ 10,  [2:10] :/ 80,  [11:1000] :/ 10};  } |
| delay\_fet\_dis | | dist {  [0:1] :/ 10,  [2:10] :/ 80,  [11:1000] :/ 10};  } |