Chassis Power Gating PGCB Verification IP

User Guide

**Synopsis:**

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

IP Rev # 2015WW25

June 19th 2015

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 7](#_Toc337589361)

[1.1 Terminology 7](#_Toc337589362)

[1.2 Tool Support 8](#_Toc337589363)

[2 Overview 9](#_Toc337589364)

[2.1 Applications 9](#_Toc337589365)

[2.1.1 PMC test environment 9](#_Toc337589366)

[2.2 Features 11](#_Toc337589367)

[2.2.1 PGCB Agent 11](#_Toc337589368)

[2.3 Control 13](#_Toc337589369)

[2.4 Requirements 13](#_Toc337589370)

[2.4.1 Specifications and Reference 13](#_Toc337589371)

[2.4.2 Compute Environment 14](#_Toc337589372)

[2.4.3 System Verilog Packages 14](#_Toc337589373)

[2.5 Architecture 14](#_Toc337589374)

[2.5.1 Class/Component Descriptions 14](#_Toc337589375)

[3 Getting Started 15](#_Toc337589376)

[4 Setting Up a Testbench Environment 16](#_Toc337589377)

[5 Implementing Test Scenarios 17](#_Toc337589378)

[6 Monitoring and Checking the Protocol 18](#_Toc337589379)

[7 Agent Packages 21](#_Toc337589380)

[8 Agent Parameters 22](#_Toc337589381)

[9 Agent Interface 23](#_Toc337589382)

[9.1 PowerGatingIF signals 23](#_Toc337589383)

[10 Configuration Methods 24](#_Toc337589384)

[10.1 Configuring the Agent 25](#_Toc337589385)

[10.1.1 PowerGatingConfig 25](#_Toc337589386)

[11 Transaction Sequence Item 28](#_Toc337589387)

[11.1 PGCBAgentSeqItem 28](#_Toc337589388)

[11.1.1 Members 28](#_Toc337589389)

[11.1.2 Constraints 28](#_Toc337589390)

[11.1.3 PGCBAgentBaseSequence 29](#_Toc337589391)

[11.1.4 waitForComplete 29](#_Toc337589392)

[11.1.5 Commands 29](#_Toc337589393)

[11.2 PGCBAgentResponseSeqItem 31](#_Toc337589394)

[11.2.1 Members 31](#_Toc337589395)

[11.2.2 Constraints 31](#_Toc337589396)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 waveform to show example of a master command ans waitForComplete. * Added documentationon waitForComplete bit in the base sequence. * 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. | |
| 07 | WW43.3 | * Added tracker. See tracker userguide for details. * Added configuration needed for tracker. Please see section 10 for the changes. * Added pok ports. * Please see integration guide for examples on configuration. * Added command for in-accessible flow. | |
| 0.71 | WW46.2 | * Enhancements/Bug fixes * [4796594](https://vthsd.intel.com/hsd/seg_softip/bug/default.aspx?bug_id=4796594)- The agent now reports error if test sends save request when pmc\_wake is asserted. * [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 tracker name. * Updated block diagram. * Added clarifications and fixed typos. See change bars | |
| 0.72 | WW47.1 | * Added a parameter IP\_ENV\_TO\_PGCB\_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. * Save req/ack have been removed * Bug fix 4797397 – delay distribution now has been changes 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   |  |  |  | | --- | --- | --- | | [4796550](https://vthsd.intel.com/hsd/seg_softip/bug/default.aspx?bug_id=4796550) | [[Enhancement] drive fet\_en\_ack\_b from the PGCB BFM](https://vthsd.intel.com/hsd/seg_softip/bug/default.aspx?bug_id=4796550) | [Future Fix](https://vthsd.intel.com/hsd/seg_softip/bug/default.aspx?bug_id=4796550) | | [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) | | |
| 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 were 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 | |  | Enhancement to drive fabric nack in the PGCBAgent. Please see userguide. | To be used by PMC only. | | |
| 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. | | |

# Introduction

The ChassisPowerGatingVIP verification component should be used to validate an PMCs Chassis defined power gating interface. It is a System Verilog OVM component. It consists PGCBAgent to emulate PGCB functionality.

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.

NOTE: Even though the agent is called PGCBAgent, it handles the pmc\_ip\_sw\_pg\_req and pmc\_wake signal.

## 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.*** |
| PGCBAgent | This is the behavioral model of the PGCB that should be used in the PMC’s validation env to emulate PGCB behavior  NOTE: Even though the agent is called PGCBAgent, it handles the pmc\_ip\_sw\_pg\_req and pmc\_wake signal. |
| PG | Power Gate |
| UG | Power Ungate |
| PGD | Power Gated Domain. It refers to a SIP or fabric domain iwht an instance of the PGCB.  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.Chassis Power Gating PGCBAgent
* **Owner:**  dbvalde1

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 |  |  |  |  |  |
| 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:

### PMC test environment

* + - The PGCB agent shoud be used to emulate an IP and Fabric’s PGCB behavior while validating PMC’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.
    - IMPORTANT: It is the responsibility of the test to make sure that it emulates the the correct behavior of the fab\_pmc\_idle signal.
      * In the real system, before PMC sees any traffic on the IOSF sideband or primary interface, the corresponding nodes of the fabric would already be power-ungated. So before the test sends any cycle either on the IOSF sideband or primary interface, the appropriate fab\_pmc\_idle signals needs to be deasserted and those fabric interfaces should be in power-ungated state.



**Usage model for the PGCBAgent BFM in the PMC env**

## Features

### PGCB Agent

|  |  |  |
| --- | --- | --- |
| Feature | Supported (Yes/No) | Expected release date |
| Supports upto maximum of 128 SIP, PGD, fabric and FET blocks each. | Yes |  |
| Mastering capabilities with configurable delay  Send gate request SIP\_PG\_REQ  Send ungate request SIP\_UG\_REQ  Fabric enter idle - FAB\_IDLE  Fabric exit idle - FAB\_IDLE\_EXIT  See section 10 for more details on the commands | Yes |  |
| Slave response to   1. SW PG/UG request 2. PMC wake 3. Fabric PG/UG req   See the timing diagrams in the next section. | Yes |  |
| Assertion of ip\_pmc\_d3i3 and d0i3. | No | TBD |
| Assertion of vnn\_req | No | TBD |
| Monitor and tracker | Yes |  |

#### Assumptions

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

#### Waveforms

##### SIP master command



The waveform shows an example command SIP\_PG\_REQ. The pg req signal is driven after 1 clock since delay = 3. Since waitForComplete is set, the sequence completes 4 clocks (completeDelay = 4) after the sequence ends which is the assertion of pg ack from PMC.

##### SIP PG and UG response



The above waveform is to show the how the PGCBAgent reponds to a pmc wake signal along with the delays. These delays by extending the PGCBAgentResponseSeqItem. Please see the integration guide.

##### Fabric ACK and NACK responses

The BFM drives ack and nack as follows.

There are cases where fab\_pmc\_idle is 0 and fab\_pmc\_pg\_rdy\_ack\_b asserts. That is a possibility if the idle changes value after fabirc’s commit point.

* 1. For the BFM, the fab\_pmc\_pg\_req\_b is the commit point.
  2. The nack/nack distribution I (send\_fab\_nack variable in the PGCBResponseSequenceItem) s as follows
     1. 30% – assert nack after random delay irrespective of any other signal value. The random delay can be constrained (delay\_fab\_nack variable in PGCBResponseSequenceItem)
     2. 70%
        1. Assert ack after random delay if fab\_pmc\_idle = The random delay can be constrained (delay\_fab\_\_pg\_ack variable in PGCBResponseSequenceItem)
        2. Assert nack if fab\_pmc\_idle goes to 0 while counting the random delay.

## Control

This verification component has three basic levels of control:

* Parameters
  + Parameters are used to set the number of SW entities, SIP, Fabric PGCBd 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 uploaded into IRR. 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 PGCBAgent 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 PGCBAgentPkg::\*;

# Agent Parameters

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

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 int NUM\_SB\_EP = 1;

parameter int NUM\_PRIM\_EP = 1;

parameter bit NO\_PRIM\_EP = 0;

parameter bit IS\_ACTIVE = 1;

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

parameter bit BFM\_DRIVES\_POK = 1;

parameter bit BFM\_DRIVES\_FET\_EN\_ACK = 0;

|  |  |
| --- | --- |
| parameter | Description |
| NUM\_SIP\_PGCB | Total number of SIP PGCBs  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  The fabric pg req, ack and nack will be one per fabric PGCB |
| IS\_ACTIVE | This parameter should be set to 0 when the agent is promoted to an environment where the actual PGCBs 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 |
| NO\_PRIM\_EP | Set this to 1 if there are no primary endpoints |
| IP\_ENV\_TO\_PGCB\_AGENT\_PATH | This is the hierarchy of the PGCBAgent instance in the IP’s env. The hierarchy should be specified in the form \*<Env OVM name>.<PGCBAgent OVM name>. Please see integration guide for more details. |
| BFM\_DRIVES\_POK |  |
| BFM\_DRIVES\_FET\_EN\_ACK |  |

# Agent Interface

List the signal interface supported in this Agent.

## PowerGatingIF signals

logic clk;

logic reset\_b;

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

# Configuration Methods

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 PGCB agents.

List of config object APIs and their parameters.

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

|  |  |  |
| --- | --- | --- |
| Function Name | Parameter(s) | Description |
| DisableFetGateCheck | None | This method can be used by the user to disable all fet gating checks in cases where fet override bits are tested. |
| DisableFetUnGateCheck | None | This method can be used by the user to disable all fet ungating checks in cases where fet override bits are tested. |
| AddFETBlock | int index  string name | The FET block index number  This function needs to be called before the AddSIPPGCB and AddFabricPGCB function |
| SetTrackerName | string name | Name of the tracker. The printer will add PG\_TRACKER and .out to the name.  Example. If the name is set to GPIO1, the tracker name wil be PG\_TRACKER\_GPIO1.out |
| DisableConfigPrinting | -- | The tracker prints out configuration information at the beginning of the test.  This function disables printing the configuration information. |
| 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.  Double check on the issue with same name and 4 letter name. |
| 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. |
| 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 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. |
| 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 |
| 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. |
| 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. |

# Transaction Sequence Item

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

## PGCBAgentSeqItem

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\_FAB\_PGCB or NUM\_SIP\_PGCB |
| 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::SIP\_PG\_REQ,  PowerGating::SIP\_UG\_REQ,  PowerGating::FAB\_IDLE,  PowerGating::FAB\_IDLE\_EXIT,  PowerGating::SIDE\_POK\_ASD,  PowerGating::SIDE\_POK\_DSD,  PowerGating::PRIM\_POK\_ASD,  PowerGating::PRIM\_POK\_DSD  } |

### PGCBAgentBaseSequence

#### 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\_FAB\_PGCB or NUM\_SIP\_PGCB |
| 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

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 | WaitForComplete |
|  |  |  |  |
|  |  |  |  |
| SIP\_PG\_REQ | int source  int delay  int delayComplete | This command will asserts the ip\_pmc\_pg\_req\_b signal for the source specified after <delay> number of clocks  source can be >= 0 and < NUM\_SIP\_PGCB. | Wait till pmc\_ip\_pg\_ack\_b is asserted and delayComplete expires. |
| SIP\_UG\_REQ | int source  int delay  int delayComplete | This command will deassert the ip\_pmc\_pg\_req\_b signal for the source specified after <delay> number of clocks  source can be >= 0 and < NUM\_SIP\_PGCB. | If pmc\_ip\_restore\_b is deasserted, wait till pmc\_ip\_pg\_ack\_b is deasserted and delayComplete expires.  If pmc\_ip\_restore\_b is asserted, wait for pmc\_ip\_restore\_b to deassert and delayComplete to expire. |
| FAB\_IDLE | int source  int delay  int delayComplete | This command will assert the fab\_pmc\_idle signal for the source specified after <delay> number of clocks  source can be >= 0 and < NUM\_FAB\_PGCB. | Wait till the signals are driven and delayComplete expires. |
| FAB\_IDLE\_EXIT | int source  int delay  int delayComplete | This command will deassert the fab\_pmc\_idle signal for the source specified after <delay> number of clocks  source can be >= 0 and < NUM\_FAB\_PGCB. | Wait till the signals are driven and delayComplete expires. |

## PGCBAgentResponseSeqItem

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.

|  |  |  |
| --- | --- | --- |
| 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\_SIP\_PGCB or NUM\_SIP\_PGCB |
| noResponse | bit | **When set, the responder will not send any responses.** |
| delay\_pg\_req | Int | This is the delay in number if clocks te driver waits before asserting ip\_pmc\_pg\_req\_b in response to a master command, SW PG Request. |
| delay\_ug\_req | int | This is the delay in number of clocks the driver waits before deasserting the ip\_pmc\_pg\_req\_b in response PMC wake or SW UG request. |
| delay\_fab\_pg\_ack | Int | This is the delay in number of clocks the driver waits before asserting fab\_pmc\_pg\_rdy\_ack\_b in response to a pmc\_fab\_pg\_rdy\_req\_b assertion |
| delay\_fab\_nack | Int | This is the delay in number of clocks the driver waits before asserting fab\_pmc\_pg\_rdy\_nack\_b after fab\_pmc\_idle deassertion |
| delay\_fab\_ug\_ack | int | This is the delay in number of clocks the driver waits before deasserting fab\_pmc\_pg\_rdy\_ack\_b in response to a pmc\_fab\_pg\_rdy\_req\_b deassertion |
| delay\_fet\_en\_ack | int | Delay in number of clocks the driver waits before assertion/deassertion of fet\_en\_ack\_b in desponse to fet\_en\_b (only applicable if BFM\_DRIVES\_FET\_EN\_ACK is set to 1) |

### 

### Constraints

|  |  |  |
| --- | --- | --- |
| Constraint Name and Hierarchy | Description | |
| source\_c | source >= 0; source < 128; | |
| cmd\_c | none | |
| noResponse\_c | noResponse == 0; | |
|  | |  |
| delay\_pg\_req | | dist {  [0:1] :/ 10,  [2:10] :/ 80,  [11:1000] :/ 10};  } |
| delay\_ug\_req | | dist {  [0:1] :/ 10,  [2:10] :/ 80,  [11:1000] :/ 10};  } |
| delay\_fab\_pg\_ack | | dist {  [0:1] :/ 10,  [2:10] :/ 80,  [11:1000] :/ 10};  } |
| delay\_fab\_nack | | dist {  [0:1] :/ 10,  [2:10] :/ 80,  [11:1000] :/ 10};  } |
| delay\_fab\_ug\_ack | | dist {  [0:1] :/ 10,  [2:10] :/ 80,  [11:1000] :/ 10};  } |
| delay\_fet\_en\_ack | | constraint delay\_fet\_en\_ack\_c {  delay\_fet\_en\_ack dist{  [0:1] :/ 10,  [2:10] :/ 80,  [11:1000] :/ 10};  } |
| send\_fab\_nack | | constraint send\_fab\_nack\_c {  send\_fab\_nack dist{  [0:0] :/ 70,  [1:1] :/ 30};  } |