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 # 2015~~3~~WW25~~30~~

June~~July~~ 19~~24~~th 2015~~3~~

Copyright and Disclaimer Information

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

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- The agent now reports error if test sends save request when pmc_wake is asserted. • 4796471 – tracker now prints Accessible flow properly • 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 – [Enhancement] drive fet_en_ack_b from the PGCB BFM – Future Fix 4966274 – PowerGatingMonitorSeqItem toString function returns empty string – Enhancement Request 5076719 – [Enhancement] FET protocol checks missing – Bug 5076832 – missing package import in source/CC/CCAgentPkg.sv – Enhancement Request

Formatted Table

<table>
<tr><td>2013WW26</td><td>WW26</td><td>
Added coverage model (see tracker/monitor userguide).
Made following bug fixes.
<table>
<tr><td>5077365</td><td>Assertions are not using fab_pmc_pg_rdy_ack_b, fab_pmc_pg_rdy_nak_b synced to PMC clockdomain</td><td>Bug</td></tr>
<tr><td>5077849</td><td>pok values in the monitor not reset correctly during global reset event</td><td>Bug</td></tr>
</table>
</td></tr>
<tr><td>2013WW30</td><td>WW30</td><td>
The following changes were made
<table>
<tr><td>5077770</td><td>[Chassis ECN] make default value of restore_b configurable as per Chassis PM ECN 1570775</td><td>Does not affect any SPT IP</td></tr>
<tr><td></td><td>Changes made for performance speed-up</td><td>Not a functional change</td></tr>
<tr><td></td><td>Enhancement to drive fabric nack in the PGCBAgent. Please see userguide.</td><td>To be used by PMC only.</td></tr>
</table>
</td></tr>
<tr><td>2015WW25</td><td></td><td>Replace script compiling and running of model/tests with ace flows.
<table>
<tr><th>HSDes</th><th>Description</th><th>Comment</th></tr>
<tr><td>1204719334</td><td>Unexpect assertion firing when reset deasserts.</td><td>One more term added to assertion to qualify rising/faling edge.</td></tr>
<tr><td>1404190277</td><td>Change to support chassis reset messages for non PGCB IPs in chassis_rst_pkg random mode.</td><td>Added additional argument when getting SB registration.</td></tr>
</table>
</td></tr>
</table>

Formatted: No bullets or numbering

1 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.

1.1 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.

1.2 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~~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	~~Alamelu Ramaswamy~~Danny Valdez	~~Aramaswad~~bvalde1	FM		916-356-8485~~377-2848~~
Secondary Owner					
Original Developer	Alamelu Ramaswamy	~~aramaswa~~	~~FM~~		~~916-377-2848~~
Manager	~~Haitham Abul-Haija~~Anurag Tyagi				

2 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 :

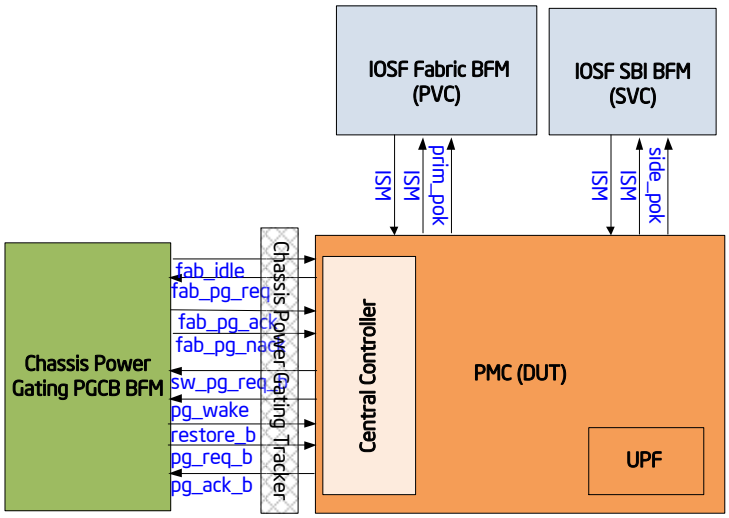
- Applications
- Features
- Operations
- Control
- Bibliography

2.1 Applications

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

2.1.1 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

2.2 Features

2.2.1 PGCB Agent

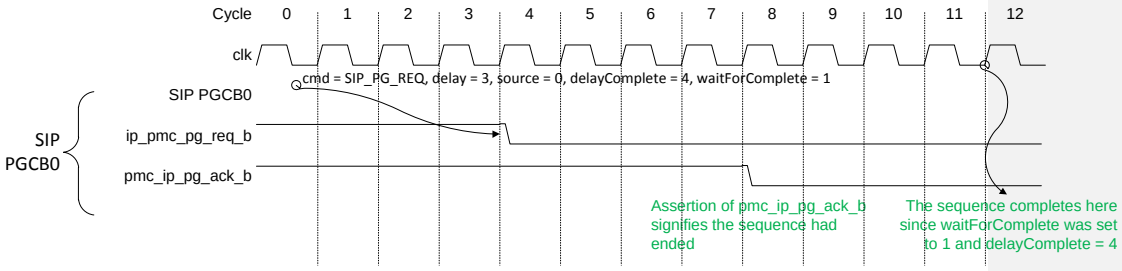
Feature	Supported (Yes/No)	Expected release date
Supports upto maximum of 128 SIP, PGD, fabric and FET blocks each.	Yes	
Mastering capabilities with configurable delay 1. Send gate request SIP_PG_REQ 2. Send ungate request SIP_UG_REQ 3. Fabric enter idle - FAB_IDLE 4. Fabric exit idle - FAB_IDLE_EXIT 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	

2.2.1.1 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.

2.2.1.2 Waveforms

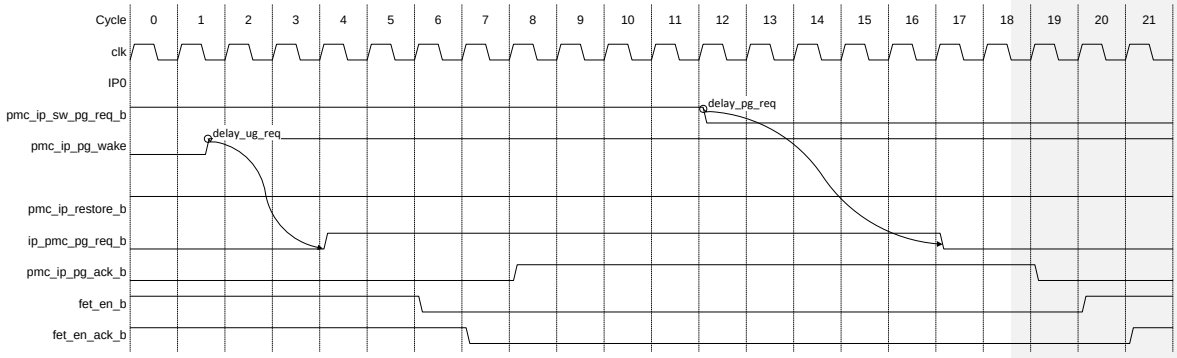
2.2.1.2.1 SIP master command



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.

2.2.1.2.2 SIP PG and UG response

SIP PG and UG flow



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.

2.2.1.2.3 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.

a. For the BFM, the fab_pmc_pg_req_b is the commit point.
b. The nack/nack distribution I (send_fab_nack variable in the PGCBResponseSequenceItem) s as follows
    i. 30% – assert nack after random delay irrespective of any other signal value. The random delay can be constrained (delay_fab_nack variable in PGCBResponseSequenceItem)
    ii. 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.

2.3 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.

2.4 Requirements

2.4.1 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

2.4.2 Compute Environment

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

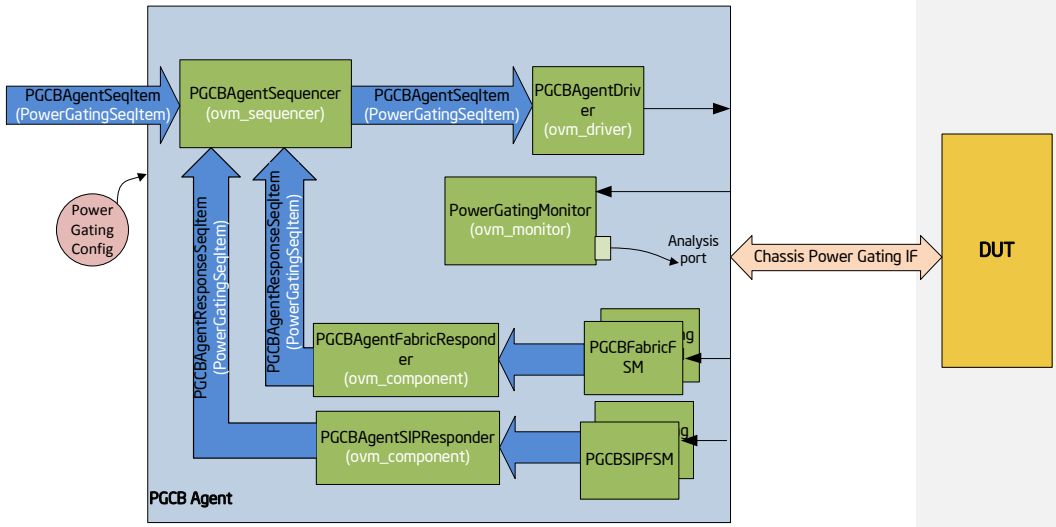
- Simulators the package can be used with.
  - Synopsis VCS
- It is uploaded into IRR. It is called Chassis Power Gating VIP.

2.4.3 System Verilog Packages

ovm_pkg – System Verilog base framework
sla_pkg – Saola package

2.5 Architecture

2.5.1 Class/Component Descriptions



Power Gating PGCBAgent block diagram.

3 Getting Started

See integration guide

4 Setting Up a Testbench Environment

See integration guide

5 Implementing Test Scenarios

See integration guide

6 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

7 Agent Packages

The package can be imported as shown below.

```
import PGCBAgentPkg::*;
```

8 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	

9 Agent Interface

List the signal interface supported in this Agent.

9.1 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;
```

10 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.

10.1 Configuring the Agent

10.1.1 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

Function Name	Parameter(s)	Description
	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

Function Name	Parameter(s)	Description
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.

11 Transaction Sequence Item

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

11.1 PGCBAgentSeqItem

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

11.1.1 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.

11.1.2 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
	}

11.1.3 PGCBAgentBaseSequence

11.1.3.1 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

11.1.4 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

11.1.5 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.

Command	Parameters	Description	WaitForComplete
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.

11.2 PGCBAgentResponseSeqItem

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

11.2.1 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)

11.2.2 Constraints

Constraint Name and Hierarchy	Description
source_c	source >= 0; source < 128;
cmd_c	none
noResponse_c	noResponse == 0;

Constraint Name and Hierarchy	Description

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};
	}