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 2017~~5~~WW25

June~~March~~ 20~~23~~th 2017

Copyright and Disclaimer Information

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 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 – tracker now prints Accessible flow properly • 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 | 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
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 | Assertions are not using fab_pmc_pg_rdy_ack_b, fab_pmc_pg_rdy_nak_b synced to PMC clockdomain | Bug 5077849 | pok values in the monitor not reset correctly during global reset event | Bug
2013WW30	WW30	• The following changes have been made 5077770 | [Chassis ECN] make default value of restore_b configurable as per Chassis PM ECN 1570775 | 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	[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	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	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 -Added feature to responde VNN_ACK as a part of auto response

1 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.

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.***
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.

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.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				

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 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.

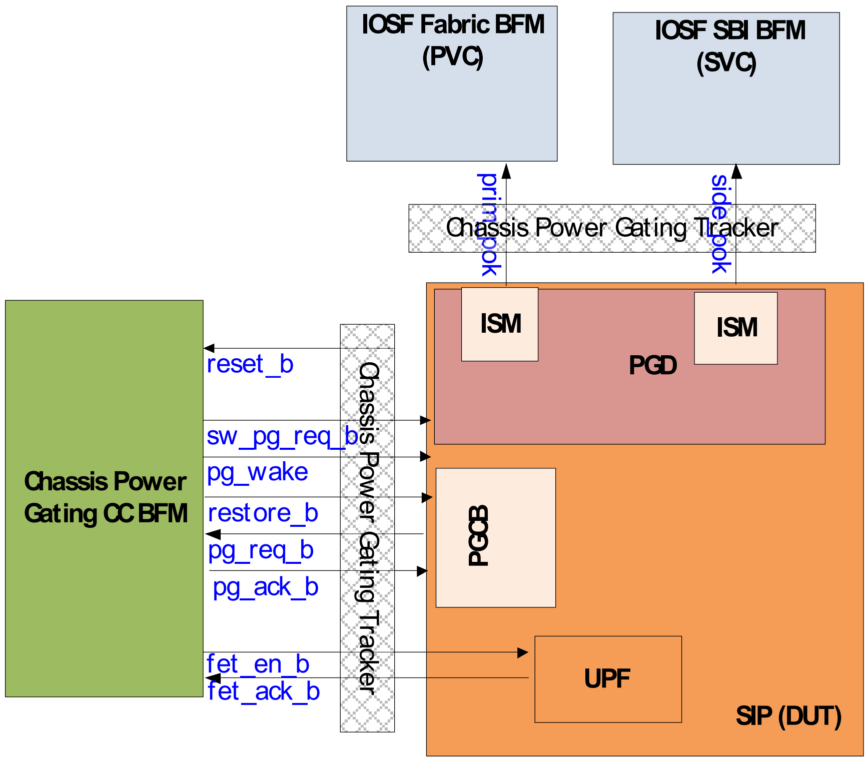


Figure 1 Example usage model for the CCAgent BFM in the SIP env

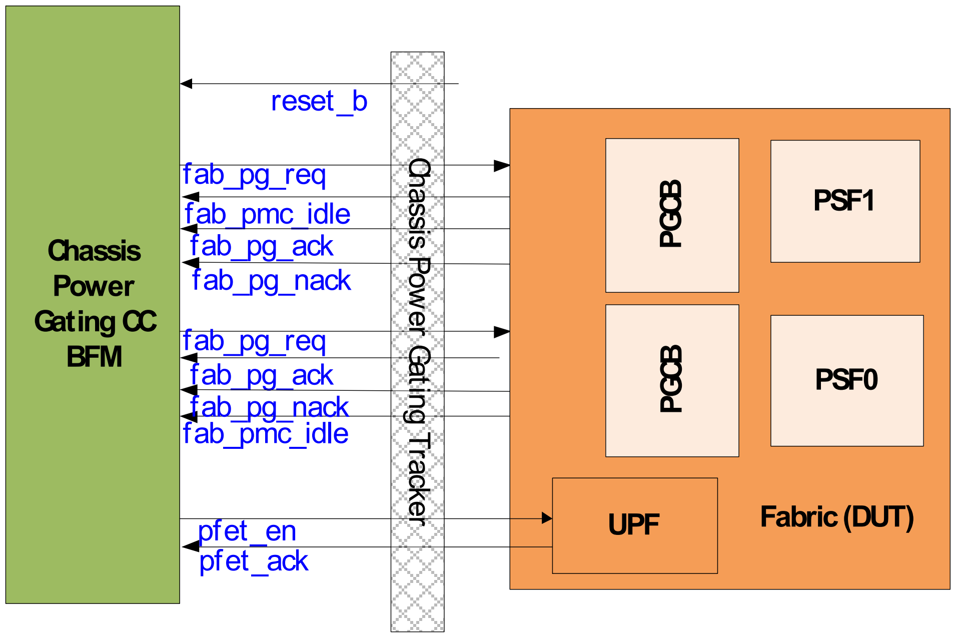


Figure 2 Example usage model for the CCAgent BFM in the Fabric env

2.2 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 1. Initial states of IP. 2. PGCB to FET block mapping since multiple PGCBs can belong to one FET block. 3. Ungate request priority for arbitration 4. Total number of SW visible entities 5. Total number if PMC wakes driven by PMC.	Yes	
Mastering capabilities with configurable delay 1. Send SW power gating request - SW_PG_REQ 2. Send PMC wake - PMC_SIP_WAKE 3. 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

2.2.1.1 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.

2.2.1.2 Arbitration

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

1. Fabric UG request
2. SIP UG request
3. Fabric PG requests
4. SIP PG request

1. Wait for driver to finish previous command (that is the arbitration point).
   a. 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)

a. In this mode, the CCAgent will only process one PG/UG request at a time.
b. 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

c. No other pg_ack_b or fet_en_b are asserted/deasserted in between.
d. User still has the ability to override the delays. See section 4 for details on how to override the response delays.
e. Gate requests are serviced First-in-first-done including the fabric.
f. Ungate requests get priority over gate requests.
g. Among ungate requests, the user has the ability to program the priority per PGCB (see section 10 for details on configuration object).

2.2.1.3 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)

2.2.1.4 Waveforms

2.2.1.4.1 SIP Master command

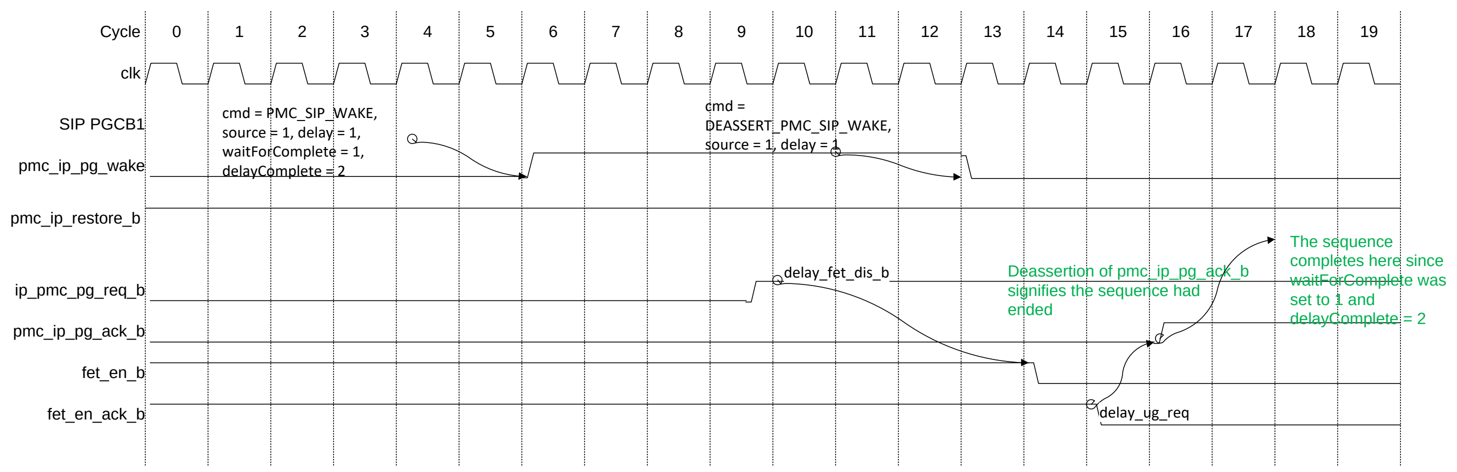

Cycle
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
clk
SIP PGCB1
pmc_ip_pg_wake
pmc_ip_restore_b
ip_pmc_pg_req_b
pmc_ip_pg_ack_b
fet_en_b
fet_en_ack_b
cmd = PMC_SIP_WAKE, source = 1, delay = 1, waitForComplete = 1, delayComplete = 2
cmd = DEASSERT_PMC_SIP_WAKE, source = 1, delay = 1
delay_fet_dis_b
delay_ug_req
Deassertion of pmc_ip_pg_ack_b signifies the sequence had ended
The sequence completes here since waitForComplete was set to 1 and delayComplete = 2

Figure 3 SIP master command example

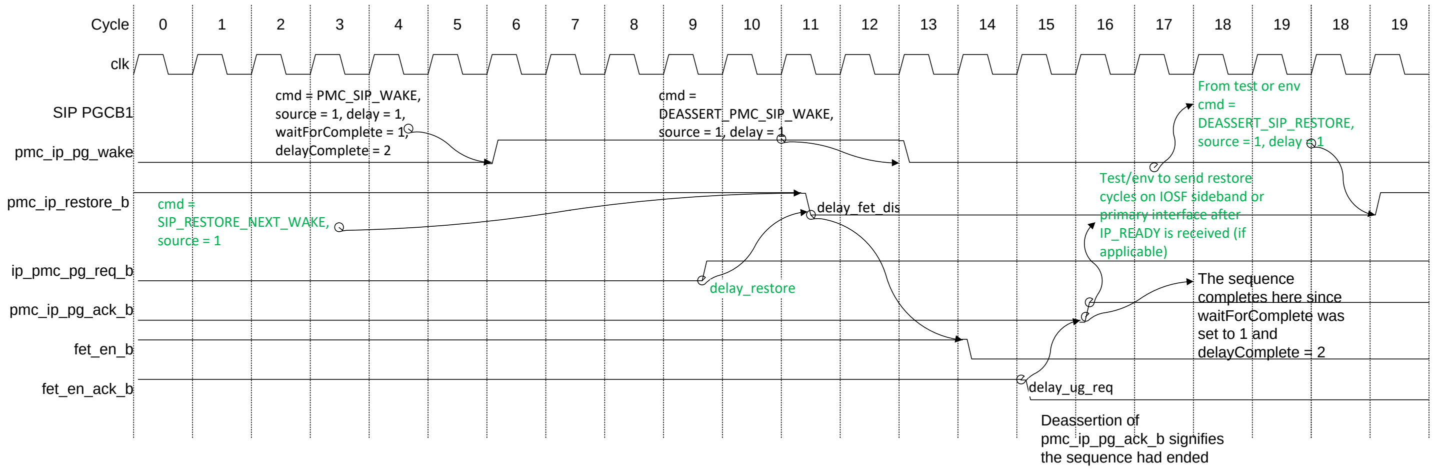
The above waveform shows the following command.

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.

2.2.1.4.2 Restore flow


Cycle
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
18
19
clk
SIP PGCB1
pmc_ip_pg_wake
pmc_ip_restore_b
ip_pmc_pg_req_b
pmc_ip_pg_ack_b
fet_en_b
fet_en_ack_b
cmd = PMC_SIP_WAKE, source = 1, delay = 1, waitForComplete = 1, delayComplete = 2
cmd = DEASSERT_PMC_SIP_WAKE, source = 1, delay = 1
From test or env cmd = DEASSERT_SIP_RESTORE, source = 1, delay = 1
cmd = SIP_RESTORE_NEXT_WAKE, source = 1
delay_fet_dis
Test/env to send restore cycles on IOSF sideband or primary interface after IP_READY is received (if applicable)
delay_restore
The sequence completes here since waitForComplete was set to 1 and delayComplete = 2
delay_ug_req
Deassertion of pmc_ip_pg_ack_b signifies the sequence had ended

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

i. Sample pmc_ip_restore_b when pmc_ip_pg_ack_b deasserts. If pmc_ip_restore_b is asserted, continue.
ii. 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).
iii. Then send restore cycles on IOSF sideband or IOSF primary.
iv. 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.

2.2.1.4.3 SIP requests arbitration

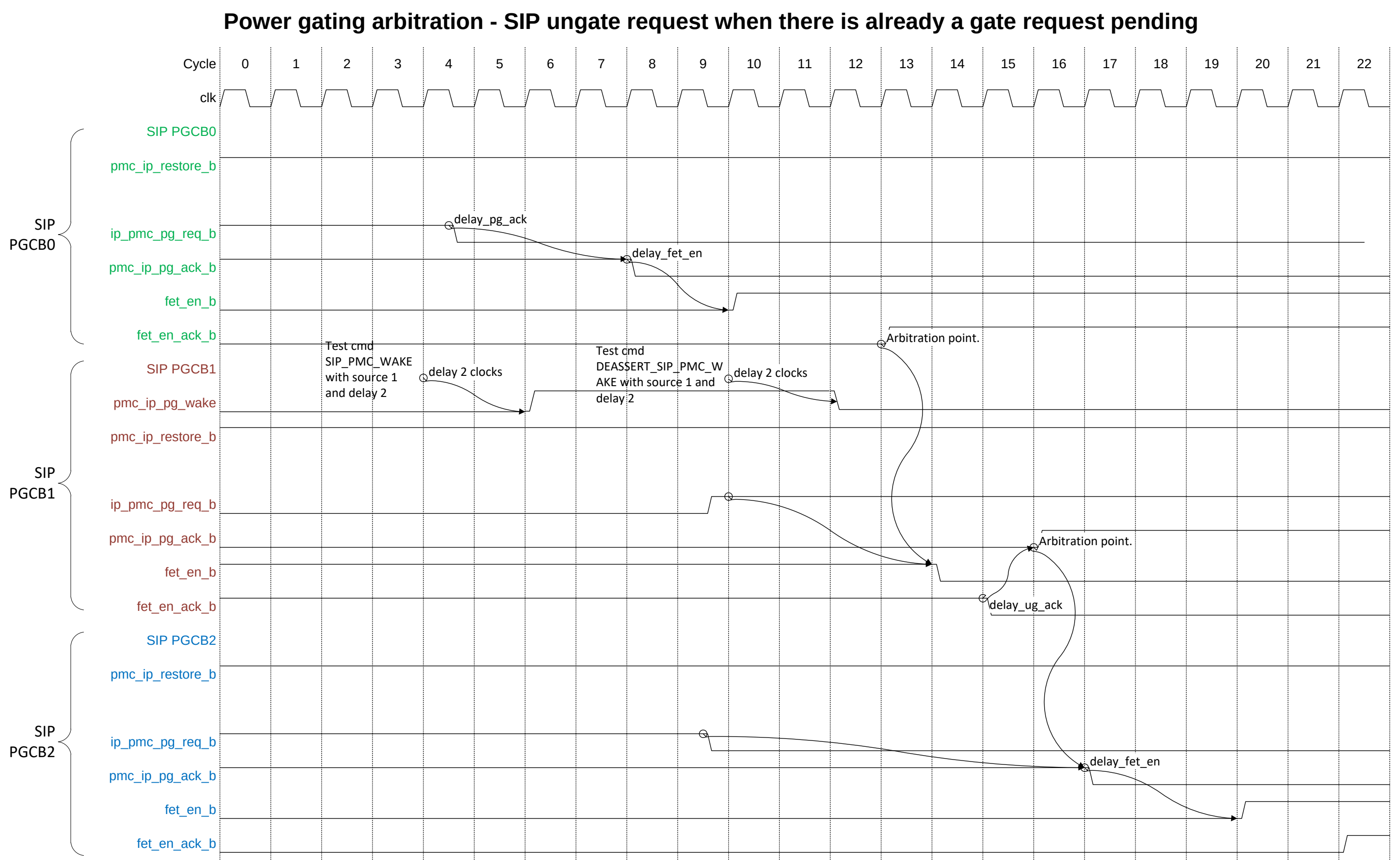


Figure 4 SIP arbitration timing diagrams

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

2.2.1.4.4 DFX support

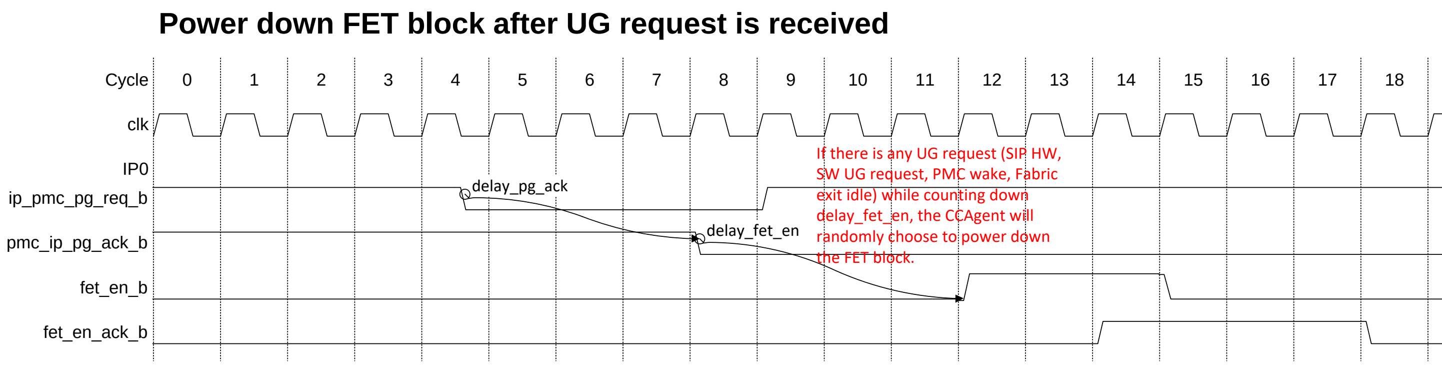
The VC provides commands to assert and deassert the dfx signals fdfx_pgcb_bypass and fdfx_pgcb_ovr. There are no checks on these signals. User can call these commands to emulate the flow mentioned in the PGCB integration guide. The VC also provides coverage to ensure IPs test the combinations 10 and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

FDFX_BYPASS_ASD,
FDFX_BYPASS_DSD,
FDFX_OVR_ASD,
FDFX_OVR_DSD,

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

2.2.1.4.5 Power down FET block after UG request if received

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


Power down FET block after UG request is received
Cycle
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
clk
IP0
ip_pmc_pg_req_b
pmc_ip_pg_ack_b
fet_en_b
fet_en_ack_b
delay_pg_ack
delay_fet_en
If there is any UG request (SIP HW, SW UG request, PMC wake, Fabric exit idle) while counting down delay_fet_en, the CCAgent will randomly choose to power down the FET block.

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.

2.2.1.4.6 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.

2.2.1.4.7 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

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 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.

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.

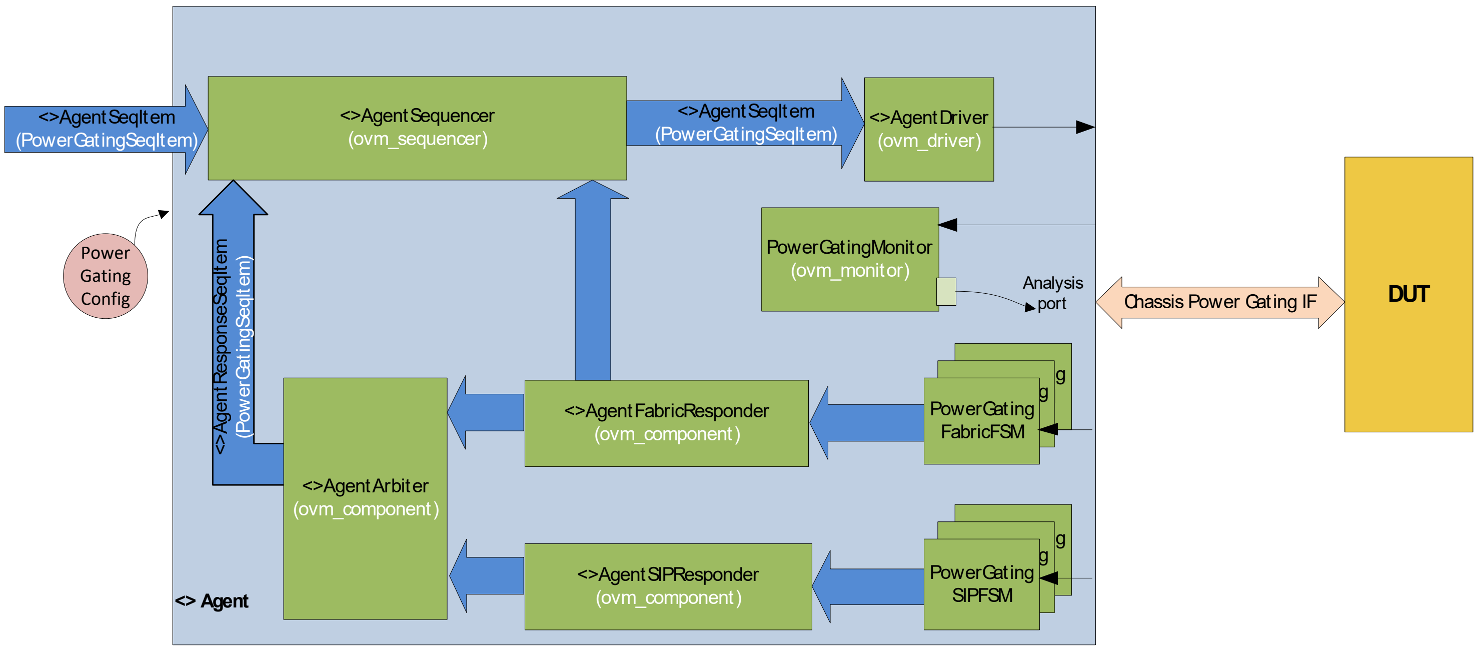
- *Simulators the package can be used with.*
    - *Synopsis VCS*
- *It is called Chassis Power Gating VIP.*

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


<>AgentSeqItem (PowerGatingSeqItem)
<>AgentSequencer (ovm_sequencer)
<>AgentSeqItem (PowerGatingSeqItem)
<>AgentDriver (ovm_driver)
Power Gating Config
PowerGatingMonitor (ovm_monitor)
Analysis port
Chassis Power Gating IF
DUT
<>AgentResponseSeqItem (PowerGatingSeqItem)
<>AgentFabricResponder (ovm_component)
PowerGating FabricFSM
<>AgentArbiter (ovm_component)
<>AgentSIPResponder (ovm_component)
PowerGating SIPFSM
<> Agent

Power Gating CCAgent 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 CCAgentPkg::*;
```

8 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

parameter	Description
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.

9 Agent Parameterized Interface

List the signal interface supported in this Agent.

9.1 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;
```

10 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.

10.1 Configuring the Agent

10.1.1 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.

Function Name	Parameter(s)	Description
	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-inacceeissible 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.

Function Name	Parameter(s)	Description
	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

Function Name	Parameter(s)	Description
	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.

11 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 = "";
```

12 Configuration Methods for Non-Parameterized Interface

12.1.1 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.

Function Name	Parameter(s)	Description
	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

Function Name	Parameter(s)	Description
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

13 Transaction Sequence Item and Base Sequence

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

13.1 CCAgentSeqItem

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

13.1.1 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_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.

13.1.2 Constraints

Constraint Name and Hierarchy	Description
delay_c	delay >=0; delay < 20;
source_c	source >= 0; source < 128;

Constraint Name and Hierarchy	Description

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
	}

13.1.3 CCAgentBaseSequence

13.1.3.1 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_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

13.1.4 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

13.1.5 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..

Command	Parameters	Description	Wait for complete
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.

Command	Parameters	Description	Wait for complete
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.

13.2 CCAgentResponseSeqItem

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

13.2.1 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

Variable Name	Type	Description
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.

13.2.2 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};
	}

Constraint Name and Hierarchy	Description

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};
	    }