Chassis Power Gating PGCB Verification IP

INTEGRATION 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~~4~~WW25~~46~~June~~November~~ 19~~3~~th 2015~~4~~

Copyright and Disclaimer Information

Contents

1 Getting Started

Here is a guide to getting started quickly explained in the following sub-sections:

See the next section for more detailed instructions

- Installing for the First Time

  ~~cd into ChassisPowerGatingVIP.~~

  ~~The comp_elab file sources the env and has the compile and elaboration vcs commands.~~

- Running a Standalone Demo

  `Example:`

~~Example:~~

```
Cd into Chassis_PowerGatingVC and setenv MODEL_ROOT $cwd
source ace/ace.env
To complile and elaborate: ace -c~~scripts/compile~~
To run a~~one~~ test : ace -x -t <test_name>~~scripts/comp_elab <test_name> -gui~~
~~To run all tests: scripts/run_tests -test_src verif/tb/test/~~
```

2 Setting Up a Testbench Environment

This section describes how to setup and configure OVM envirnonments for this Agents. The previous section explained Agent package installation, prerequisite tools, and example Testbenches.

This section show how Agent can be connected in two types of testbench :

- Standalone Testbench
- Cluster and Chip Testbench

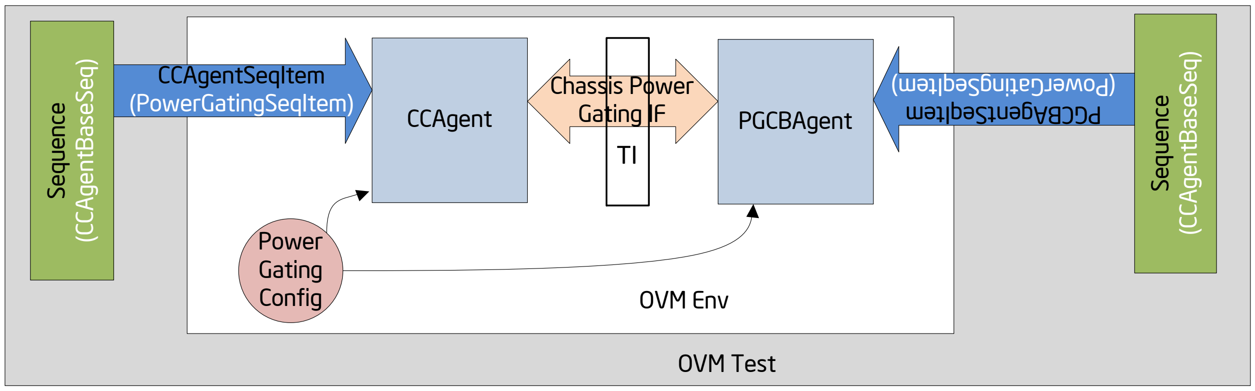
The Standalone testbench connects an agent to another agent and is used to model the connection of this agent to unit-level DUT. Cluster and Chip Testbench connects an agent to a unit-level DUT in Cluster and Chip environment.

The section after this one describes how configure Agents and implement test scenarios.

2.1 Creating a Standalone Testbench

This section focuses on how to connect Agents in Standalone Testbench, with example codes.

Here is a block diagram of the standalone testbench which was used to validate the agents. The CC and PGCB agents are connected to each other


Sequence (CCAgentBaseSeq)
CCAgentSeqItem (PowerGatingSeqItem)
CCAgent
Chassis Power Gating IF
TI
PGCBAgent
PGCBAgentSeqItem (PowerGatingSeqItem)
Sequence (CCAgentBaseSeq)
Power Gating Config
OVM Env
OVM Test

Block diagram of standalone testbench

Refer to each item mentioned below:

- Connecting Agents in the Testbench
- Naming and Instantiating Testbench Components
- Extending the Testbench for Test Execution
- Steps to compile and run test

2.1.1 Connecting Agents in the Testbench

Shown below is an example of the test island/test-bench file.

The power gating VC need to be in passive mode at SOC. Therefore it need to be instantiated in the IP's TI. But the power gating VC interface **does not support TB automation flow**. So please follow the guidelines below while integrating these collaterals.

IPs that have only one instance at SOC level (most IPs fall under this category) are required to instantiate the VCs in their TI and make the connections to the interface internally inside the TI using `defines. Please note that the assign statements need to done with care. Otherwise, it could cause checker/coverage to be disabled or SOC integration issues. Example:

```
module pmc_ti(iosf_sb_intf iosf_sb_if);
      PowerGatingIF pg_if;
      PGCBAgentTI pg_ti(pg_if);

      generate if(IS_ACTIVE)
          assign pg_if.ip_pmc_pg_req_b = `IP_TOP.abc_pmc_pg_req_b;
          assign `IP_TOP.pmc_abc_pg_ack_b = pg_if.pmc_ip_pg_ack_b;
      .....
      generate if(!IS_ACTIVE)
          assign pg_if.ip_pmc_pg_req_b = `IP_TOP.abc_pmc_pg_req_b;
          assign pg_if.pmc_ip_pg_ack_b = `IP_TOP.pmc_abc_pg_ack_b;
      .....
endmodule
```

IMPORTANT NOTE:

1. IP_ENV_TO_PGCB_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>.<PGCBAgent OVM name>.

2. In the examples shown below, say the env is instantiated in the base test as follows

   `env = <GPIO env type>::type_id::create("`**`pmc_env`**`", this);`

3. The CCAgent is instantiated in the env as follows

   `pgcbAgent = PGCBAgent::type_id::create("`**`pmc_pgcb_agent`**`",this)`

4. So the parameter should be set to - ***.pmc_env.pmc_pgcb_agent**

```
PowerGatingIF#(
.NUM_SIP_PGCB(NUM_SIP_PGCB),
.NUM_FET(NUM_FET),
.NUM_FAB_PGCB(NUM_FAB_PGCB),
.NUM_SW_REQ(NUM_SW_REQ),
.NUM_PMC_WAKE(NUM_PMC_WAKE),
.NUM_PRIM_EP(NUM_PRIM_EP),
.NUM_SB_EP(NUM_SB_EP)
```

```
         ) pgIF ();
        generate if (!IS_ACTIVE) begin: ASSIGN_PASSIVE_BLK

        //This is jsut for the monitor
                assign pgIF.clk = `IP_TOP.clk;
                assign pgIF.reset_b = `IP_TOP.reset_b;
                assign pgIF.pmc_ip_sw_pg_req_b = `IP_TOP.pmc_ip_sw_pg_req_b;
                assign pgIF.ip_pmc_pg_req_b = `IP_TOP.ip_pmc_pg_req_b;
                assign pgIF.pmc_ip_pg_ack_b = `IP_TOP.pmc_ip_pg_ack_b;
                assign pgIF.pmc_ip_pg_wake =  `IP_TOP.pmc_ip_pg_wake;

                assign pgIF.pmc_ip_restore_b = `IP_TOP.pmc_ip_restore_b;

                assign pgIF.prim_pok = `IP_TOP.prim_pok;
                assign pgIF.side_pok = `IP_TOP.side_pok;

                assign pgIF.fab_pmc_idle= `IP_TOP.fab_pmc_idle;
                assign pgIF.pmc_fab_pg_rdy_req_b = `IP_TOP.pmc_fab_pg_rdy_req_b;
                assign pgIF.fab_pmc_pg_rdy_ack_b = `IP_TOP.fab_pmc_pg_rdy_ack_b;
                assign pgIF.fab_pmc_pg_rdy_nack_b = `IP_TOP.fab_pmc_pg_nack;
                assign pgIF.fet_en_b = `IP_TOP.fet_en_b;
                assign pgIF.fet_en_ack_b = `IP_TOP.fet_en_ack_b;

        end: ASSIGN_PASSIVE_BLK
        else begin: ASSIGN_ALL_BLK
                assign pgIF.clk = `IP_TOP.clk;
                assign pgIF.reset_b = `IP_TOP.reset_b;
                assign pgIF.pmc_ip_sw_pg_req_b = `IP_TOP.pmc_ip_sw_pg_req_b;
                assign `IP_TOP.ip_pmc_pg_req_b = pgIF.ip_pmc_pg_req_b ;
                assign pgIF.pmc_ip_pg_ack_b = `IP_TOP.pmc_ip_pg_ack_b;
                assign pgIF.pmc_ip_pg_wake = `IP_TOP.pmc_ip_pg_wake;

                assign pgIF.pmc_ip_restore_b = `IP_TOP.pmc_ip_restore_b;

                assign `IP_TOP.prim_pok = pgIF.prim_pok;
                assign `IP_TOP.side_pok = pgIF.side_pok;

                assign `IP_TOP.side_pok = pgIF.fab_pmc_idle;
                assign pgIF.pmc_fab_pg_rdy_req_b = `IP_TOP.pmc_fab_pg_rdy_req_b;
                assign `IP_TOP.fab_pmc_pg_rdy_ack_b = pgIF.fab_pmc_pg_rdy_ack_b ;
                assign `IP_TOP.fab_pmc_pg_rdy_nack_b = pgIF.fab_pmc_pg_rdy_nack_b;
                assign pgIF.fet_en_b = `IP_TOP.fet_en_b;
                assign pgIF.fet_en_ack_b = `IP_TOP.fet_en_ack_b;

        end: ASSIGN_ALL_BLK
        endgenerate
        PGCBAgentTI #(
        .NUM_SIP_PGCB(NUM_SIP_PGCB),
        .NUM_FET(NUM_FET),
        .NUM_FAB_PGCB(NUM_FAB_PGCB),
        .NUM_SW_REQ(NUM_SW_REQ),
        .NUM_PMC_WAKE(NUM_PMC_WAKE),
        .NUM_PRIM_EP(NUM_PRIM_EP),
        .NUM_SB_EP(NUM_SB_EP),
        .IS_ACTIVE(IS_ACTIVE),
        //.NO_FAB_PGCB(1),
        .IP_ENV_TO_PGCB_AGENT_PATH({IP_ENV, ".pmc_pgcb_agent"})


        )pgcbTI(pgIF);
```

2.1.1.1 Driving fet_en_ack_b from the testbench

Note that the fet_en_ack_b needs to be driven from the testbench in response to fet_en_b with some delay. In the real system, the ack would be driven by the last fet in the fet chain.

2.1.2 Naming and Instantiating Components

Shown below is an example of how to instantiate the agents and configure it using the PowerGatingConfigObject.

Note that the arguments need to be passed by name while configuring the agent using the config object methods. See system Verilog LRM for details on passing arguments by name.

```
class PowerGatingSaolaEnv extends ovm_env;

        `ovm_component_utils_begin(PowerGatingSaolaEnv)
        `ovm_component_utils_end

        PGCBAgent pgcbAgent;
        PowerGatingConfig cc_cfg_pgcb;

        function new(string name, ovm_component parent);
                super.new(name, parent);
        endfunction

        function void build();

                // Turn off all the sequencers by default
                set_config_int("*sequencer", "count", 0);
                set_config_int("*pmc_pg_agent", "is_active", 1);
                set_config_int("*pgcbcAgent01*", "hasPrinter", 1);

                super.build();
                //this.set_level(SLA_TOP);

                pgcbAgent = PGCBAgent::type_id::create("pmc_pg_agent", this);
                cc_cfg_pgcb = new({"pmc_pg_agent","ConfigObject"});

                `ovm_info(get_full_name(), "Agents created", OVM_HIGH)

                /*****************
                PGCB
                ******************/
                cc_cfg_pgcb.SetTestIslandName("pmc_testisland");
                cc_cfg_pgcb.SetTrackerName("PGCBAgentTracker");

                cc_cfg_pgcb.AddFETBlock(.index(0), .name("FET0"));
                cc_cfg_pgcb.AddFETBlock(.index(1), .name("FET1"));
                cc_cfg_pgcb.AddFETBlock(.index(2), .name("FET2"));

                cc_cfg_pgcb.AddFabricPGCB(.index(0), .name("FAB0"), .fet_index(2));
                //int .index, string .name, int .fet_index = 0, time .hys = 0ps, int
sip_mapping{$}, int fabric_mapping{$}, bit initial_state

                cc_cfg_pgcb.AddSIPPGCB(.index(0), .name("SIP0"), .fet_index(0),
.pmc_wake_index(0), .sw_ent_index(1), .SB_array({0}));
                cc_cfg_pgcb.AddSIPPGCB(.index(1), .name("SIP1"), .fet_index(1),
.pmc_wake_index(1), .sw_ent_index(0), .SB_array({1}));
                cc_cfg_pgcb.AddSIPPGCB(.index(2), .name("FSIP"), .fet_index(2),
.pmc_wake_index(2), .sw_ent_index(2), .SB_array({2}), .fabric_index(0));

                cc_cfg_pgcb.AddSIP(.name("ADSP"), .sip_type(PowerGating::HOST),
.pgcb_array({0, 1}), .AON_prim_array({0}));

                cc_cfg_pgcb.AddSBEP(.index(0), .source_id('hE8));
                cc_cfg_pgcb.AddSBEP(.index(1), .source_id('hB5));
                cc_cfg_pgcb.AddSBEP(.index(2), .source_id('hA2));
                cc_cfg_pgcb.AddPrimEP(.index(0), .AON_EP(1), .pmc_wake_index(3));
                //User set config object using {<name>, “ConfigObject”}

                set_config_object("*", "pmc_pg_agentConfigObject",cc_cfg_pgcb,0);
```

```
            `ovm_info(get_full_name(),  "Set  config  object  for  PGCBAgent",
OVM_HIGH)
```

2.1.3 Extending the Testbench for Test Execution

Here is a list of existing tests that can be run on this standalone test-bench.

Test Name	Description
FabricBasicTest	1. Fabric exit idle command 2. Make sure req and ack deassert 3. Fabric enter idle 4. Make sure req and ack assert
FabricAbortTest	5. Fabric exit idle command 6. Make sure req and ack assert 7. Fabric enter idle 8. Make sure req asserts 5. Fabric exit idle again before ack 6. Make sure the NACK asserts 7. Here make fabric enter idle again 8. Make sure the req deassert first and only then asserts again.
FabricAgentWakeTest	For this test the SetAgentWakeModel finction needs to be called See section10.1
FabricResponseOverrideTest	In this test, we use the OVM type overdise capabilities to override the delay constraints to >0 and <5. The response seq item also contains a noResponse bit that can be used to prevent the agent from responding. See section 11.2
SIPBasicTest	1. Wake up the SIP using PMC wake 2. Make sure req and ack deassert 3. Assert SW PG request 4. Make sure req and ack assert 7. Assert HW UG req 8. Make sure req and ack assert 9. Assert HW PG req 10. Make sure req and ack deassert

Test Name	Description
SIPAbortTest	1. Wake up the SIP using PMC wake 2. Make sure req and ack deassert 3. Assert HW Save req 4. Deassert HW Save req 5. Make sure the flow is aborted
ResetTest	Assert reset in the middle of the test
ArbitrationTest	In this test, the arbitration logic is tested
FETModeTest	This test is to validate the FET ON mode
WaitForComplete	This test is to validate waitforcomplete function for a pmc wake sequence
IPInaccTest	Tests all the state machines for IP Inacc state
Acc_IPInaccTest	Tests all state machine arcs and to verify agent going from Acc PG state to In acc PG state
AssertionFailTest	Tests failure cases of all assertions
ConcurrentTest	Random test to test arbitration and make sure all requests get granted eventually.
ErrorPMCWakeTest	Test the error cases requested by PMC
FETModeTest	Tests the mode where fets are not turned off and also tests resetting the mode.
RestoreTest	Test basic retore flow
RestoreErrorTest	Test errorenous scenario where IP asserts pg_req_b while in the restore window.

2.1.4 Steps to compile and run test

ace –c –x -t~~scripts/comp_elab~~ <test_name>~~ –gui~~

2.1.5 hdl files

The hdl files are in the ace folder.

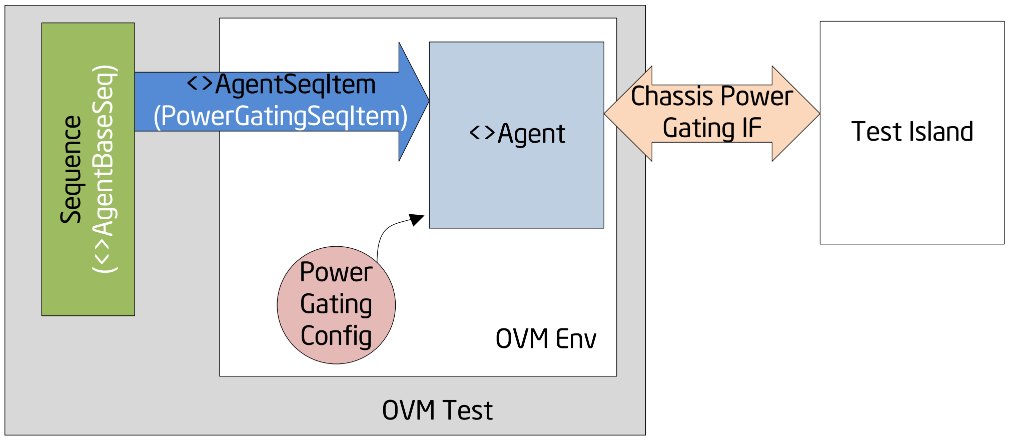
2.1.6 UDF files

There are no UDF files provided but in a typical IP udf file, you would have to speficy the path to the hdl and the dependent libs as follows. Note: Post VCS version 2013.06 is causing error after removing the VCS Backward compatibility switch (BC mode switch) XLRM PACKAGE IMPORT COMPAT = TRUE. This switch is getting deprecated in the upcoming VCS release i.e. VCS 2014.03. Previously this error was being masked due to presence of the BC mode switch. A fix is provided in the udf file and ChassisPowerGatingVIP Pkg files.~~ I tried to make the fix transparent for the IP if they are using $ENV{VCS_VER} for determining the VCS version they are using. If the IP is not using this variable, then they~~ The IP will have to add a vlog_opt to there udf file for compiling ChassisPowerGatingVIP. Refer to below for the vlog_opts switch "VCS EXPORT SUPPORT".

```
ChassisPowerGatingVIP => {
-hdl_spec =>
['verif/lib/shared/ChassisPowerGatingVIP/ace/ChassisPowerGatingVIP.hdl],
-dependent_libs => ['ovm_pkg', 'sla_pkg',],
-vlog_opts => [
                "-sverilog +define+VCS_EXPORT_SUPPORT",
}
```

2.2 Creating a Cluster and Full Chip Testbench

Shown below is a block diagram of the cluster test environment.



In the full-chip/SOC environment, use “is_active” config in the Agent use set the agent in passive mode.

```
set_config_int("*pmc_pg_agent", "is_active", 0);
```

Make sure it matches the IS_ACTIVE in the test-island. Otherwise, you will get a warning and the CCAgent will override with IS_ACTIVE specified in the TI.

3 Implementing Test Scenarios

This section shows how to write tests. The previous section explained how to setup and configure a simulation envirnonment. In this section, we focus on features that allow the Agents to create test scenarios.

- Configuring the Agents
- Sending Specific Transaction Sequences
- Extending Transaction Constraints

This section describes how to stimulate and control a DUT, and the section that follows this one describes how to observe and verify a DUT. It discusses how configure the interface assertions, monitor, Scoreboard, and coverage collector.

3.1 Configuring the Agents

This section shows how to configure Agents from a test or Testbench. Add in an example of how to configure Agents during the configure phase of OVM simulation using Agent configuration methods.

As shown in the previous sections, the PowerGatingConfig object can be used to configure the agent.

3.2 Sending Specific Transaction Sequences (Using the Base Sequence)

This section shows how to create a customized Stimulus Generator to send directed transactions to the Agents.

The **PGCBAgentBaseSequence** must be used to create any sequence with the necessary constaints.

Note that the base sequence has a rand bit waitForComplete which can be set to 1 if the user wants to wait till the sequence compeltes. Please see the userguide to know what wait for compelte means for different sequences.

Example

```
class SIPHWSaveReqSequence extends ovm_sequence;
       //=========================================================
       // PUBLIC VARIABLES
       //===========================================================
       PGCBAgentBaseSequence seq;
       //=============================================================
       // OVM Macros for public variables
       //==============================================================
       `ovm_sequence_utils_begin(SIPHWSaveReqSequence, PGCBAgentSequencer)
       `ovm_sequence_utils_end

       /*********************************************************************
       *  @brief Constructor.
       *******************************************************************/
       function new(string name = "SIPHWSaveReqSequence");
              super.new(name);
       endfunction : new
```

```
        task body();
                `ovm_do_with(seq,     {seq.cmd     ==     PowerGating::SIP_SAVE_REQ;
seq.source == 1;seq.delay == 3; waitForComplete == 1;})
        endtask


endclass : SIPHWSaveReqSequence
```

3.3 Extending Transaction Constraints (Controlling Timing Delays)

Complete list of constraints supported in this Agent is described in section 11.

The response sequence item's constraints can be changed as shown below.

Once the new sequence item is created, the ovm factory function set_type_override_by_type can be used to override by type in the test.

```
class PGCBAgentResponseSeqItemNew1 extends PGCBAgentResponseSeqItem;


        //=========================================================================
        // OVM Macros for public variables
        //=========================================================================
        `ovm_object_utils_begin(PGCBAgentResponseSeqItemNew1)
                //`ovm_field_int(delay , OVM_ALL_ON)
        `ovm_object_utils_end
        /*************************************************************************
        *  @brief Constructor.
        *************************************************************************/
        function new(string name="PGCBAgentXaction");
                super.new(name);
        endfunction : new

        /*constraint delay_c {
                delay == 5;
        }
        */
        constraint delay_ug_req_c {
                delay_ug_req == 5;
        }

        //Setting noRespnse to 1. So the respnder will not send any responses.
        constraint noResponse_c {
                noResponse == 1;
        }

endclass: PGCBAgentResponseSeqItemNew1
```

Here is an example if a test where the type override is done

```
class ArbitrationTest extends PowerGatingBaseTest;
                SIPPMCWakeAllSequence sipWakeAll;
                SIPPMCWakeSequence sipPMCWake;
                SIPSWPGReqSequence sipSWPGReqSeq;
                SIPHWUGReqSequence sipHWUGReqSeq;

                SIPHWSaveReqSequence sipHWSaveReqSeq;
                SIPRandHWSaveReqSequence randsipHWSaveReqSeq;
                SIPRandHWUGReqSequence randsipHWUGReqSeq;
                `ovm_component_utils(ArbitrationTest)
                function  new(string  name  =  "ArbitrationTest",  ovm_component  parent  =
null);
                        super.new(name, parent);
                endfunction

                function void build();
                        set_config_int("*", "count", 0);
                        set_config_int("*","recording_detail", OVM_FULL);
                        ovm_top.set_report_verbosity_level(OVM_FULL);

                        super.build();
                        set_type_override_by_type(PGCBAgentResponseSeqItem::get_type(),
PGCBAgentResponseSeqItemNew1::get_type());
                        set_type_override_by_type(CCAgentResponseSeqItem::get_type(),
CCAgentResponseSeqItemNew::get_type());

                endfunction
```

4 Agent Parameters

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

```
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 bit NO_PRIM_EP = 0;

parameter bit IS_ACTIVE = 1;


parameter string IP_ENV_TO_PGCB_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 save and 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
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.

5 Agent Interface

List the signal interface supported in this Agent.

5.1 PowerGatingIF signals

```
input logic clk;
input logic reset_b;
input logic[NUM_SW_REQ-1:0] pmc_ip_sw_pg_req_b;

output logic[NUM_SIP_PGCB-1:0] ip_pmc_save_req_b;
input logic[NUM_SIP_PGCB-1:0] pmc_ip_save_ack_b;

input logic[NUM_SIP_PGCB-1:0] pmc_ip_restore_b;

output logic[NUM_SIP_PGCB-1:0] ip_pmc_pg_req_b;
input logic[NUM_SIP_PGCB-1:0] pmc_ip_pg_ack_b;
input 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;


output logic[NUM_FAB_PGCB-1:0] fab_pmc_idle;
input logic[NUM_FAB_PGCB-1:0] pmc_fab_pg_rdy_req_b;
output logic[NUM_FAB_PGCB-1:0] fab_pmc_pg_rdy_ack_b;
output logic[NUM_FAB_PGCB-1:0] fab_pmc_pg_rdy_nack_b;

input logic[NUM_FET-1:0] fet_en_b;
input logic[NUM_FET-1:0] fet_en_ack_b;
```