Chassis Power Gating Central Controller Verification IP

INTEGRATION GUIDE

Synopsis:

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

IP Rev # 2017WW25~~12~~

June~~March~~ 20~~3~~th 2017

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

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

- Running a Standalone Demo

```
Example:
Cd into Chassis_PowerGatingVC and setenv MODEL_ROOT $cwd
source ace/ace.env
To complile and elaborate: ace -c
To run a test : ace –x –t <test_name>
Note tests can be found in test directory: 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.

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 Parameterized vs non-parameterized interfaces

The PowerGatingIF is a parameterized interface which cannot be stitiched using the collage. Therefore IPs were asked not to expose the PowerGatingIF to the SOC and make all the connections internally as follows. See next section for example.

Users can now choose to instantiate a set of non-parameterized interface which are corekit/collage-compliant. This way the interface can be part of the IP's TI and SOC will connect these interfaces to the design using collage flows. See section called "Non-parameterized Interface" . With these interfaces the IP's TI will look something like this…

```
module ip_ti(PowerGatingSIPIF pg_if, iosf_sb_intf iosf_sb_if, );
      PowerGatingSIPIF pg_if;
      PowerGatingSIPTI pg_ti(pg_if);
endmodule
```

NOTE: This flow has not been validated by the Chassis Sandbox team yet. Once it is validated, IPs can make use of this interface.

IMPORTANT NOTE: If you are integrating the VC for the first time, then use the non-parameterized interface and TI.

2.1.1 Connecting Agents in the Testbench/Test-island with parameterized interface PowerGatingIF

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 PowerGatingIF **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 ip_ti(iosf_sb_intf iosf_sb_if);
```

```
        PowerGatingIF pg_if;
        CCAgentTI 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
```

IPs that are instantiated multiple times at FC (like PXP) are required to make the power gating interface ports to the TI and SOC will make the connections. Example:

```
    module ip_ti(PowerGatingIF pg_if, iosf_sb_intf iosf_sb_if);
        CCAgentTI pg ti(pg if);
        ccu_ti ccu_ti(ccu_if);
        .....
    endmodule
```

IMPORTANT NOTE:

1. 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>.
2. In the examples shown below, say the env is instantiated in the base test as follows

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

3. The CCAgent is instantiated in the env as follows

   ```
   ccAgent = CCAgent::type_id::create("gpio_pg_agent",this)
   ```

4. So the parameter should be set to - ***.gpio_env.gpio_pg_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),
    .NUM_D3(NUM_D3),
     .NUM_VNN_ACK_REQ(NUM_VNN_ACK_REQ),
    .NUM_D0I3(NUM_D0I3)

     ) 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 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;

               assign pgIF.pmc_ip_vnn_ack =
{temp_pmc_ip_vnn_ack,powerGatingIF.pmc_ip_vnn_ack[NUM_VNN_ACK_REQ-1:0]};

               assign pgIF.ip_pmc_vnn_req =
{temp_ip_pmc_vnn_req,powerGatingIF.ip_pmc_vnn_req[NUM_VNN_ACK_REQ-1:0]};
               assign pgIF.fdfx_pgcb_bypass = `IP_TOP.fdfx_pgcb_bypass;
               assign pgIF.fdfx pgcb ovr = `IP TOP.fdfx pgcb ovr;
               assign pgIF.ip_pmc_d3 = `IP_TOP.ip_pmc_d3;
               assign pgIF.ip pmc d0i3 = `IP TOP.ip pmc d0i3;



       end: ASSIGN PASSIVE BLK
       else begin: ASSIGN_ALL_BLK
               assign pgIF.clk = `IP TOP.clk;
               assign pgIF.reset_b = `IP_TOP.reset_b;
               assign `IP_TOP.pmc_ip_sw_pg_req_b = pgIF.pmc_ip_sw_pg_req_b;
               assign pgIF.ip_pmc_pg_req_b = `IP_TOP.ip_pmc_pg_req_b;
               assign `IP TOP.pmc ip pg ack b = pgIF.pmc ip pg ack b;
               assign `IP_TOP.pmc_ip_pg_wake = pgIF.pmc_ip_pg_wake;

               assign `IP_TOP.pmc_ip_restore_b = pgIF.pmc_ip_restore_b;

               assign pgIF.prim pok = `IP TOP.prim pok;
               assign pgIF.side pok = `IP TOP.side pok;

               assign `IP_TOP.pmc_ip_vnn_ack = pgIF.pmc_ip_vnn_ack[NUM_VNN_ACK_REQ-1:0];

               assign pgIF.ip_pmc_vnn_req = {temp_ip_pmc_vnn_req,
`IP_TOP.ip_pmc_vnn_req[NUM_VNN_ACK_REQ-1:0]};
               assign pgIF.fab pmc idle= `IP TOP.fab pmc idle;
               assign `IP_TOP.pmc_fab_pg_rdy_req_b = pgIF.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_rdy_nack_b;
               assign `IP_TOP.fet_en_b = pgIF.fet_en_b;
               assign pgIF.fet_en_ack_b = `IP_TOP.fet_en_ack_b;

               assign `IP_TOP.fdfx_pgcb_bypass = pgIF. fdfx_pgcb_bypass;
               assign `IP TOP.fdfx pgcb ovr = pgIF.fdfx pgcb ovr;
               assign pgIF.ip_pmc_d3 = `IP_TOP.ip_pmc_d3;
               assign pgIF.ip_pmc_d0i3 = `IP_TOP.ip_pmc_d0i3;

       end: ASSIGN ALL BLK
       endgenerate

       CCAgentTI #(
       .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),
        .NUM VNN ACK REQ(NUM VNN ACK REQ),
```

```
        .IS_ACTIVE(IS_ACTIVE),
        .IP_ENV_TO_CC_AGENT_PATH("*.gpio_env.gpio_pg_agent")

        )ccTI(pgIF);
```

This is an example of the testbench connections where there is no Fabric in the test environment. The user needs to set NO_FAB_PGCB parameter to 1. If there is no SIP interface in the environment, the user needs to set NO_SIP_PGCB to 1.

```
        PowerGatingIF#(
        .NUM SIP PGCB(NUM SIP PGCB),
        .NUM_FET(NUM_FET),
        .NUM SW REQ(NUM SW REQ),
        .NUM_PMC_WAKE(NUM_PMC_WAKE),
         .NUM_VNN_ACK_REQ(NUM_VNN_ACK_REQ),
        .NO_FAB_PGCB(1),

         ) ccIF ();

        CCAgentTI #(
        .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),
        .NO_FAB_PGCB(1),
        .IS_ACTIVE(IS_ACTIVE),
         .NUM VNN ACK REQ(NUM VNN ACK REQ),
        .IP_ENV_TO_CC_AGENT_PATH("*.gpio_env.gpio_pg_agent")


        )ccTI(ccIF);
NOTE: NUM_VNN_ACK_REQ is added for TGPLP VNN removal requirement.
```

2.1.2 Naming and Instantiating Testbench Components with parameterized interface

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

        CCAgent ccAgent;
        PowerGatingConfig cc_cfg;

        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("*gpio_pg_agent", "is_active", 1);
                set_config_int("*gpio_pg_agent*", "hasPrinter", 1);

                super.build();
```

```
            ccAgent = CCAgent::type_id::create("gpio_pg_agent", this);
            cc_cfg = new({"gpio_pg_agent","ConfigObject"});


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

            /*********
            CC
            **********/
            cc_cfg.SetTrackerName("GPIO");
            //cc_cfg.DisableConfigPrinting();

            cc_cfg.AddFETBlock(.index(0), .name("DOM0"));
            cc_cfg.AddFETBlock(.index(1), .name("DOM1"));
            cc_cfg.AddFETBlock(.index(2), .name("DOM2"));

            cc_cfg.AddFabricPGCB(.index(0), .name("FAB0"),
.fet_index(0),.hys(300ns), .ungate_priority(4));
            cc_cfg.AddFabricPGCB(.index(1), .name("FAB1"),
.fet_index(0),.hys(400ns), .ungate_priority(4));
            cc_cfg.AddFabricPGCB(.index(2), .name("FAB2"),
.fet_index(0),.hys(600ns), .ungate_priority(4));

            cc_cfg.AddSIPPGCB(.index(0), .name("PGD1"), .fet_index(0),
.ungate_priority(1), .pmc_wake_index(0), .sw_ent_index(1), .SB_array({0}));
            cc_cfg.AddSIPPGCB(.index(1), .name("PGD2"), .fet_index(1),
.ungate_priority(2), .pmc_wake_index(1), .sw_ent_index(0), .SB_array({1}));
            cc_cfg.AddSIPPGCB(.index(2), .name("PGD3"), .fet_index(2),
.ungate_priority(3), .pmc_wake_index(2), .sw_ent_index(2), .SB_array({2}),
fabric_index(1));


            cc_cfg.AddSIP(.name("GPIO"), .sip_type(PowerGating::HOST),
.pgcb_array({0, 1}), .AON_prim_array({0}), .d3[{0}], .d0i3[{0}]);
            cc_cfg.AddSBEP(.index(0), .source_id('hE8));
            cc_cfg.AddSBEP(.index(1), .source_id('hB5));
            cc_cfg.AddSBEP(.index(2), .source_id('hA2));
            cc_cfg.AddPrimEP(.index(0), .AON_EP(1), .pmc_wake_index(3)
            .vnn_ack_req_index(0),//index of vnn_ack_req of this SIP
);



            //User set config object using {<name>, "ConfigObject"}

            set_config_object("*", "gpio_pg_agentConfigObject",cc_cfg,0);

            `ovm_info(get_full_name(), "Set config object CC", OVM_HIGH)
```

2.1.3 Connecting Agents in Testbenc/TI with non-parameterized interface

Users can now choose to instantiate a set of non-parameterized interface which are corekit/collage-compliant. This way the interface can be part of the IP's TI and SOC will connect these interfaces to the design using collage flows. See section called "Non-parameterized Interface" . With these interfaces the IP's TI will look something like this…

```
      module ip_ti(PowerGatingSIPIF kvm_if, PowerGatingSIPIF ptio_if,
PowerGatingResetIF reset_if, ….iosf_sb_intf iosf_sb_if, );
            PowerGatingSIPIF kvm_if;
            PowerGatingSIPTI kvm_ti(kvm_if);
```

```
            ….…
      endmodule
```

Here is a real example of a test-island connection

```
      PowerGatingResetIF  reset_if();
      PowerGatingResetTI #(
            .IP_ENV_TO_AGENT_PATH("*csme_env.ccAgentCsme1_col")
      ) reset_cc_ti(reset_if);

      PowerGatingSIPIF kvm_if();
      PowerGatingSIPIF ptio_if();
      PowerGatingFabricIF fab1_if();

      PowerGatingSIPTI #(
            .NAME("KVM"),
            .FET_NAME("FET1"),
            .IP_ENV_TO_AGENT_PATH("*csme_env.ccAgentCsme1_col"))
      cc_kvm_ti(kvm_if);

      PowerGatingSIPTI #(
            .NAME("PTIO"),
            .FET_NAME("FET23"),
            .IP_ENV_TO_AGENT_PATH("*csme_env.ccAgentCsme1_col"))
      cc_ptio_ti(ptio_if);

      PowerGatingFabricTI #(
            .NAME("PTIO_F"),
            .IP_ENV_TO_AGENT_PATH("*csme_env.ccAgentCsme1_col"))
      cc_fab1_ti(fab1_if);
```

IMPORTANT NOTE:

1. 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>.
2. In the examples shown above, say the env is instantiated in the base test as follows

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

3. The CCAgent is instantiated in the env as follows

   ```
   ccAgent = CCAgent::type_id::create("ccAgent_csme1_col",this)
   ```

4. So the parameter should be set to - ***.csme_env.ccAgent_csme1_col**

2.1.4 Naming and instantiating TB components with non-parameterized interface

Here is an example of how to configure the VC if the non-parameterized interfaces are used.

```
            cc_cfg_col.AddSIPPGCB(
                        .name("KVM"),
                        .ungate_priority(1),
                        .side_pid({'hE2})
```

```
                );
            cc_cfg_col.AddSIPPGCB(
                        .name("PTIO"),
                        .ungate_priority(1),
                        .side_pid({'hE3}),
                        .fabric_name("PTIO_F")
                );
            cc_cfg_col.AddFabricPGCB(
                .name("PTIO_F"),
                .hys(300ns),
                .ungate_priority(4));
            cc_cfg_col.AddSIP(.name("CSME1"),
                        .pgcb_name({"KVM"})
                );
            cc_cfg_col.AddSBEP(.source_id('hE2), .ip_ready('hA0));
            cc_cfg_col.SetTrackerName("PGCBAgentTracker_csme1");
            set_config_object("*",
"ccAgentCsme1_colConfigObject",cc_cfg_col,0);
```

2.2 Steps to compile and run test

ace –c –x -t <test_name>

2.3 hdl files

The hdl files are in the ace folder.

2.4 Driving fet_en_ack_b

Please see document below (if link don't work file is now embedded)

https://sharepoint.amr.ith.intel.com/sites/ChipsetHWSoC_PowerIP/Power%20WIKI%20Documents/Steps%20to%20drive%20IP%20ack_port.docx

Steps to drive IP ack_port.docx

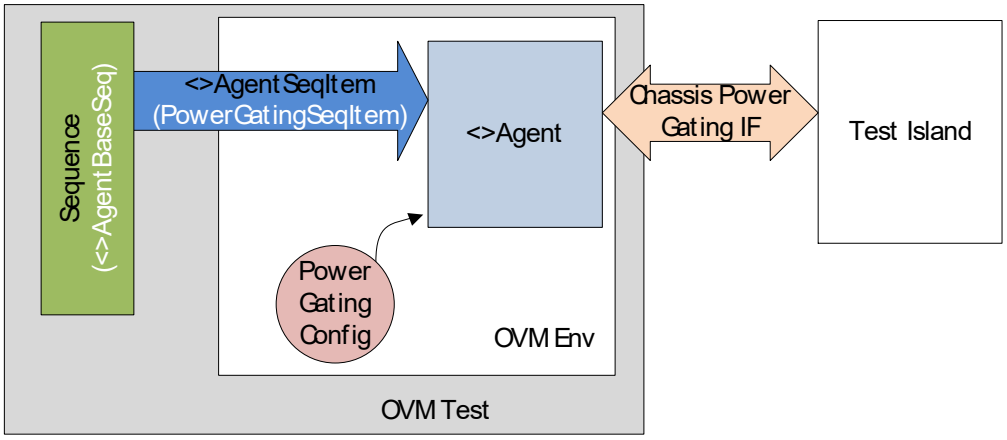
2.5 UDF files

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. The IP will have to add a vlog_opt to there udf file for compiling ChassisPowerGatingVIP. Refer to below for the 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",
}
```

2.6 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("*gpio_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 (Controlling Timing Delays)

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

Example

```
class SIPSWPGReqSequence extends ovm sequence;
        //=============================================================================
        // PUBLIC VARIABLES
        //=============================================================================
        CCAgentBaseSequence seq;
        //=============================================================================
        // OVM Macros for public variables
        //=============================================================================
        `ovm_sequence_utils_begin(SIPSWPGReqSequence, CCAgentSequencer)
        `ovm_sequence_utils_end

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

        task body();
                `ovm_do_with(seq,  {seq.cmd  ==  PowerGating::SW_PG_REQ;  seq.source  ==
0;seq.delay == 3;})

                //if using the parameterized interface, sourceName can be used
                `ovm_create(seq);
                seq.sourceName = "KVM";
```

```
            `ovm_rand_send_with(seq, {cmd == PowerGating::PMC_SIP_WAKE;})

            //Or PowerGatingConfig::getSIPPGCBIndex can be used to specify the source
            `ovm_do(seq, {cmd == PowerGating::SIP_PMC_WAKE; source =
PowerGatingConfig::getSIPPGCBIndex("KVM");})

      endtask

endclass : SIPSWPGReqSequence
```

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.

```
class SIPPMCWakeAllSequence extends ovm sequence;
      //=============================================================================
      // PUBLIC VARIABLES
      //=============================================================================
      CCAgentBaseSequence seq;
      //=============================================================================
      // OVM Macros for public variables
      //=============================================================================
      `ovm sequence utils begin(SIPPMCWakeAllSequence, CCAgentSequencer)
      `ovm_sequence_utils_end

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

      task body();
            `ovm do with(seq, {seq.cmd == PowerGating::PMC SIP WAKE ALL; seq.delay ==
3; seq.waitForComplete == 1;})
      endtask


endclass : SIPPMCWakeAllSequence
```

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 CCAgentResponseSeqItemNew extends CCAgentResponseSeqItem;

      //=============================================================================
      // OVM Macros for public variables
      //=============================================================================
      `ovm object utils begin(CCAgentResponseSeqItemNew)
      `ovm_object_utils_end
      /*****************************************************************************
      *  @brief Constructor.
      *****************************************************************************/
      function new(string name="PGCBAgentXaction");
            super.new(name);
      endfunction : new
      constraint noResponse_c {
```

```
                noResponse == 0;
        }
        constraint delay_pg_ack_c {
                delay pg ack >= 0; delay pg ack < 6;
                }

        constraint delay_ug_ack_c {
                delay_ug_ack == 25;
                }

        constraint delay_fab_ug_req_c {
                delay fab ug req >= 0; delay fab ug req < 6;
                }
        constraint delay_fet_en_c {
                delay_fet_en >= 0; delay_fet_en < 6;
                }
        constraint delay_fet_dis_c {
                delay fet dis >= 0; delay fet dis < 6;
                }
endclass: CCAgentResponseSeqItemNew
```

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

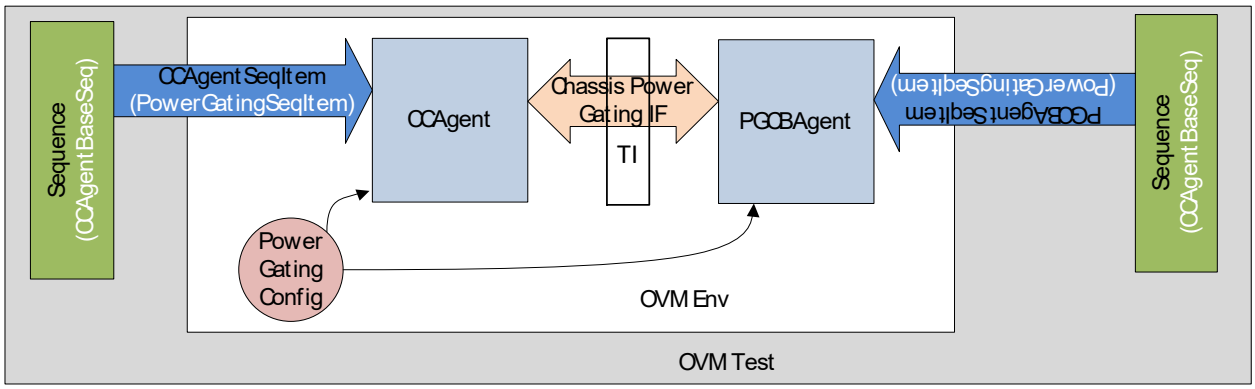
```
class ArbitrationTest extends PowerGatingBaseTest;
                SIPPMCWakeAllSequence sipWakeAll;
                SIPPMCWakeSequence sipPMCWake;
                SIPSWPGReqSequence sipSWPGReqSeq;
                SIPHWUGReqSequence sipHWUGReqSeq;

                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 Standalone testbench

4.1 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



Block diagram of standalone testbench

4.2 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	1. Fabric exit idle command 2. Make sure req and ack assert 3. Fabric enter idle 4. 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

Test Name	Description
FabricResponseOverrideTest	In this test, we use the OVM type overrise 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
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.

Test Name	Description

DfxTest	Drives the DFX signals.
	Plesae refer to this as an example.
	
	`CCAgentBaseSequence seq;`
	`//start test`
	`#2ns; //dont want to be in sync with clock`
	`//asset bypass and ovr at the same time for PGCB no : 1. Set delay to 0`
	`seq = new();`
	`seq.randomize() with {cmd == PowerGating::FDFX_BYPASS_ASD; source == 0; delay == 0;};`
	`seq.start(sla_env.ccAgent.sequencer);`
	`seq.randomize() with {cmd == PowerGating::FDFX_OVR_ASD; source == 0; delay == 0;};`
	`seq.start(sla_env.ccAgent.sequencer);`
	
	`//IP to fill in code here - wait for IP signals to power gate here.`
	
	`//now deassert ovr. IP must wake up`
	`seq.randomize() with {cmd == PowerGating::FDFX_OVR_DSD; source == 0; delay == 10;};`
	`seq.start(sla_env.ccAgent.sequencer);`
	
	`//IP to fill in code here - wait for IP signals to wake up here.`