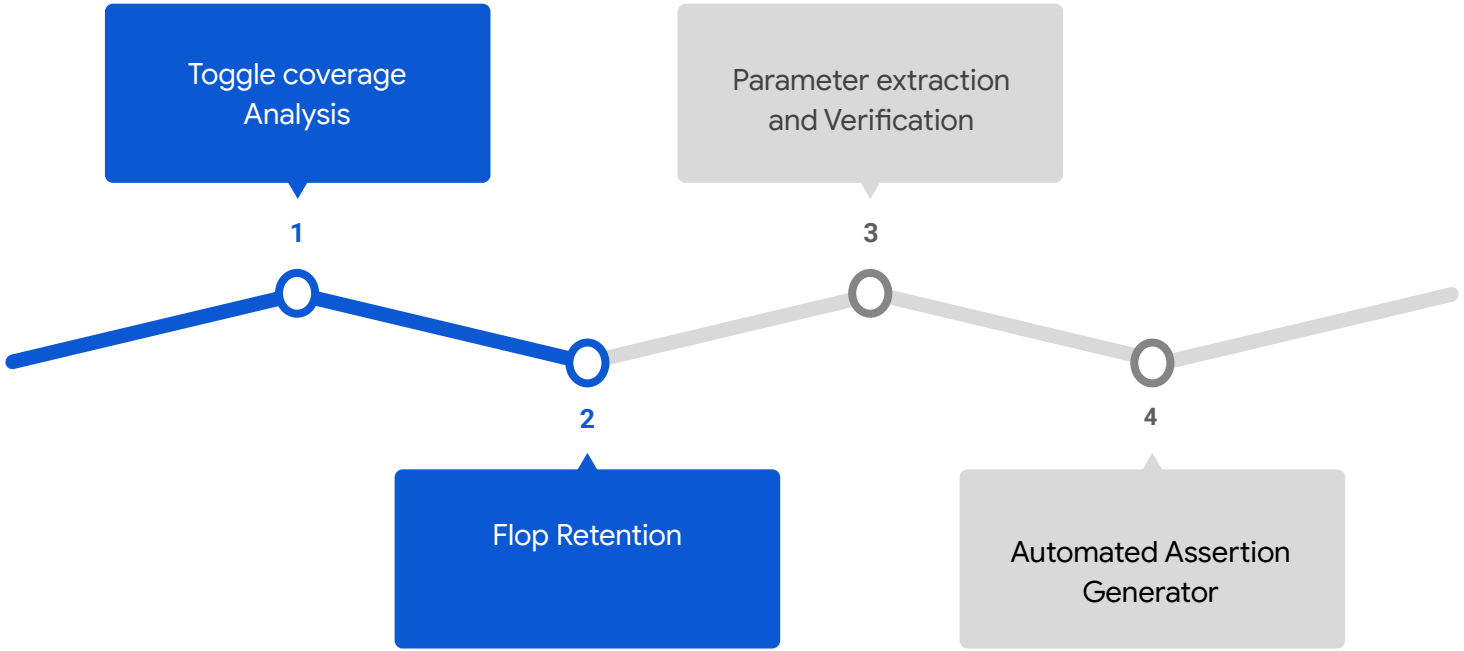
Internship Project

# Toggle Coverage Analysis and Flop Retention

Mentored by: Ravi Mangal and Shamsheer Shaik

Vansh Kapoor, Jul ’23

# Internship Assignments



# Toggle Coverage Analysis

- Problem Statement
- Approach
- Accomplishments

# Problem Statement

```
Module: sswrp_gmc.u_aux.u_pg.u_dftss_gmc_pg (dftss_gmc_pg)
-----------------------------------------------------------------------------
      1) idft_dftss_scan_in (1'b0)
      2) idft_kill_clk_en (1'b0)
      3) idfd_scan_part0_so (1'b0)
      4) gateway_sib_from_so_0 (1'b0)
      5) gateway_sib_from_so_1 (1'b0)
      6) gateway_sib_EDT_top_from_so (1'b0)
      7) idft_spare_port1 (1'b0)
      8) idft_spare_port2 (1'b0)

Module:  sswrp_gmc.u_aux.u_pg.u_gspan_fsw_rcvr
-----------------------------------------------------------------------------

Module:  sswrp_gmc.u_aux.u_pg.u_gspan_rcvr
-----------------------------------------------------------------------------

Module: sswrp_gmc.u_aux.u_pg.u_gspan_atb_initiator_to_memss (gmc_gspan_atb_initiator)
-----------------------------------------------------------------------------
      1) atclken (1'b1)
```

Signals that don't toggle

Module name indicating hierarchy

- Carry out toggle coverage analysis
- Python based automation flow for exclusion file extraction
- Exclusion file comprises of tied off or floating signals in the DUT
- Exclusions are thoroughly reviewed with RTL team and are waived during toggle analysis

# Approach

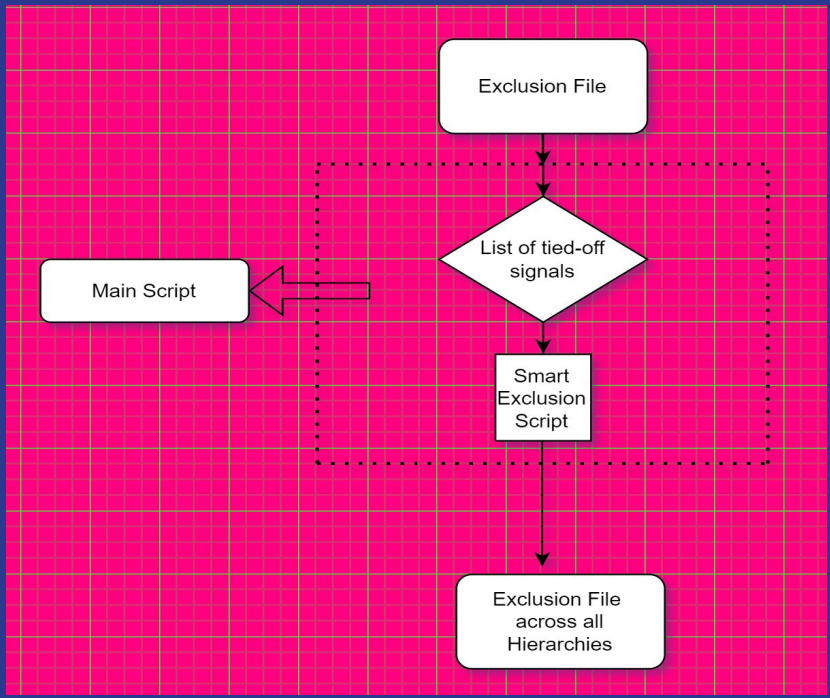
```
Module: sswrp_gmc.u_aux.u_aon.u_nic400_gmc (nic400_sswrp_gmc)
-----------------------------------------------------------------------------
        1) coreclken (1'b1)

Module: sswrp_gmc.u_aux.u_aon.u_axi4_to_apb5_xfw_cfg (AXITOAPB5_ifabric_AXI_TO_APB5)
-----------------------------------------------------------------------------
        1) qch_ifabric_clk_QReqn (1'b1)
```

```
<rule ccType="inst" domain="icc" entityName="sswrp_gmc_top_tb/&quot;lane[0]&quot;/dut/u_aux/u_aon/u_nic400_gmc/coreclken" entityType="togg
le" excTime="1684834159" name="exclude" reviewer="Vansh" user="0" vscope="default"></rule>
<rule ccType="inst" domain="icc" entityName="sswrp_gmc_top_tb/&quot;lane[0]&quot;/dut/u_aux/u_aon/u_axi4_to_apb5_xfw_cfg/qch_ifabric_clk_Q
Reqn" entityType="toggle" excTime="1684834159" name="exclude" reviewer="Vansh" user="0" vscope="default"></rule>
```

- Created a **top-level Python execution file** that runs three Python scripts sequentially
- Each script takes a specific log file as input and generates a vRefine file

# Smart Exclusion File Creation

- User creates an exclusion file at the top level
- Automation dumps an exclusion file across all the hierarchies for the connected signals using the design elaboration database.



# Accomplishments

- The flow generates exclusion files for automated identification of constant or unconnected signals in the design
- Flow saves time spent by user to exclude the connected signals across all hierarchies via smart exclusion process
- These vRefine files would be used to exclude the respective signals during the toggle coverage analysis, thereby **optimizing the design verification**

# Flop Retention

- Problem Statement
- Approach
- Accomplishments

# Problem Statement

```
always_ff @(posedge clk or negedge rst_n) begin
  if(~rst_n) begin
    frequency_index_set_q   <= '0;
  end else begin
    frequency_index_set_q   <= frequency_index_set_q_next;
  end
end
```

Declaring **frequency_index_set_q** as the flop output

- Generate a list of internal flops declared in multiple system Verilog files located in various possible paths
- Locate the output signals of flip flops within multiple RTL files written in system Verilog whose path is provided
- Integrated the file into the existing CL and troubleshoot the errors

# Approach

```
always_ff @(posedge clk or negedge rst_n) begin
  if(~rst_n) begin
    sw_sr_state[`$i`] <= '0;
  end else if (csr_sw_sr_state_change_trig[`$i`]) begin
    sw_sr_state[`$i`] <= csr_sw_sr_state_change_value[`$i`];
  end else begin
    sw_sr_state[`$i`] <= sw_sr_state_next[`$i`];
  end
end
```

Flop output signal

Declaring a flip flop output signal

```
sw_pd_state, sswrp_gmc_top_tb.lane[0].dut.u_gmc_top.u_gmc_lane.u_gmc_core.u_gmc_core_low_power.u_lp_retention.sw_pd_state
sw_sr_state, sswrp_gmc_top_tb.lane[0].dut.u_gmc_top.u_gmc_lane.u_gmc_core.u_gmc_core_low_power.u_lp_retention.sw_sr_state
sw_cs_state, sswrp_gmc_top_tb.lane[0].dut.u_gmc_top.u_gmc_lane.u_gmc_core.u_gmc_core_low_power.u_lp_retention.sw_cs_state
lp_data_fsm_cs, sswrp_gmc_top_tb.lane[0].dut.u_gmc_top.u_gmc_lane.u_gmc_core.u_gmc_core_low_power.u_lp_retention.lp_data_fsm_cs
lp_ctrl_fsm_cs, sswrp_gmc_top_tb.lane[0].dut.u_gmc_top.u_gmc_lane.u_gmc_core.u_gmc_core_low_power.u_lp_retention.lp_ctrl_fsm_cs
cs_fsm_cs, sswrp_gmc_top_tb.lane[0].dut.u_gmc_top.u_gmc_lane.u_gmc_core.u_gmc_core_low_power.u_lp_retention.cs_fsm_cs
lp2_fsm_cs, sswrp_gmc_top_tb.lane[0].dut.u_gmc_top.u_gmc_lane.u_gmc_core.u_gmc_core_low_power.u_lp_retention.lp2_fsm_cs
```

Output csv file

- Searched through all available directories and utilized the syntax of flip flops in System Verilog to identify and extract the relevant output signals

# Accomplishments

- Added automated checkers for flop retention during teardown
- Second phase of the task encompassed integrating this file into the existing CL, executing the test cases, and troubleshooting any errors
- Obtained valuable insights into the process of running test cases and the **debugging methodologies** involved

# Parameter Extraction and Verification

- Problem Statement
- Approach
- Accomplishments

# Problem Statement: Part 1

```
sswrp_gmc_top_tb.u_gmc_top_gspan_fsw_sender_inst_2...instance of module gmc_top_gspan_payload_initiator
ReqSSClocks...parameter int = 1
ReqChanWidth...parameter int = 3
AckSSClocks...parameter int = 1
SyncDepth...parameter int = 3
MaxRetimers...parameter int = 10
DataWidth...parameter int = 3
sswrp_gmc_top_tb.u_gmc_top_gspan_fsw_sender_inst_1...instance of module gmc_top_gspan_payload_initiator
ReqSSClocks...parameter int = 1
ReqChanWidth...parameter int = 3
AckSSClocks...parameter int = 1
SyncDepth...parameter int = 3
MaxRetimers...parameter int = 10
DataWidth...parameter int = 3
sswrp_gmc_top_tb.u_gmc_top_gspan_fsw_sender_inst_0...instance of module gmc_top_gspan_payload_initiator
ReqSSClocks...parameter int = 1
ReqChanWidth...parameter int = 3
AckSSClocks...parameter int = 1
SyncDepth...parameter int = 3
MaxRetimers...parameter int = 10
DataWidth...parameter int = 3
```

Input file containing parameters from different instances

1. Create a python script which takes a list of parameters from different instances and dumps it into a csv output

# Approach

```
sswrp_gmc_top_tb.u_gmc_top_cti_targ_if...instance of interface req_ack_target_if

i_payload...variable type parameter PAYLOAD_TYPE = 1'h0

sswrp_gmc_top_tb.u_gmc_top_cti_init_if...instance of interface req_ack_initiator_if

i_payload...variable type parameter PAYLOAD_TYPE = 2'h0
```

Input log file

Parameter name

value

```
sswrp_gmc_top_tb.u_gmc_top_cti_targ_if
i_payload,1'h0
sswrp_gmc_top_tb.u_gmc_top_cti_init_if
i_payload,2'h0
```

Output csv file

- Utilized the parameter name, its hierarchy and its value from input log file to generate the output csv file

# Problem Statement: Part 2

```
sswrp_gmc_top_tb.lane[0].dut.u_gmc_top
SYNC_DEPTH,2
DV_STUB_PHY,1
sswrp_gmc_top_tb.lane[0].dut.u_aux.u_aon.u_phy_axi_xfw_awrap_MFW_T_G0R32M35
AXUIPLSB,32'h0000000f
AXUIPLEN,32'h00000008
AXUGLLSB,32'h0000001e
AXUGLLEN,32'h0000000a
PUIPLSB,32'h0000000f
PUIPLEN,32'h00000008
PUGLLSB,32'h0000001e
PUGLLEN,32'h0000000a
MAX_OT,32'h00000010
```

Input file containing parameters from different instances and their expected values

1. Create a script which generates sv file that flags an error if there is a mismatch between the parameter values in the RTL file and csv

# Approach

```
sswrp_gmc_top_tb.lane[0].dut.u_gmc_top
SYNC_DEPTH,2
DV_STUB_PHY,1
```

```
initial begin
  if(sswrp_gmc_top_tb.lane[0].dut.u_gmc_top.SYNC_DEPTH != 2) begin
    `uvm_error("PARAM_VALUE_CHECK",$sformat("Param value mismatch!!, RTL param value = %h and expected value from CSV= %h",sswrp_gmc_top_tb.lane[0].dut.u_gmc_top.SYNC_DEPTH,2))
  end
  if(sswrp_gmc_top_tb.lane[0].dut.u_gmc_top.DV_STUB_PHY != 1) begin
    `uvm_error("PARAM_VALUE_CHECK",$sformat("Param value mismatch!!, RTL param value = %h and expected value from CSV= %h",sswrp_gmc_top_tb.lane[0].dut.u_gmc_top.DV_STUB_PHY,1))
```

Input csv and output sv file for parameter verification

- Utilized the hierarchy of the modules of different instances and their expected values to flags errors in case of any mismatch

# Accomplishments

```
UVM_ERROR @ 0 ps: reporter [sswrp_gmc_top_tb.lane[0].dut.u_gmc_top.DV_STUB_PHY : PARAM_VALUE_C
HECK] Param value mismatch!!, RTL param value = 00000000 and expected value from CSV= 00000001
```

Error in run.sh.log file of one of the test cases

- Developed a script for automated parameter verification flow
- The created script generated a sv file that helped in finding **mismatches** of **parameter values** between the RTL file and the CSV file
- Detected the discrepancy between the value of the parameter **DV_STUB_PHY** in the RTL file and the value it was supposed to have

# Automated Assertion Generator

- Problem Statement
- Approach
- Accomplishments

# Problem Statement

```
logic u_lpddr5x_ram_gs0_ram_0_0_SD;
logic u_lpddr5x_ram_gs0_ram_0_0_DSLP;

assign u_lpddr5x_ram_gs0_ram_0_0_ISOSRM = sswrp_gmc_top_tb.lane[0].dut.u_gmc_top.u_gmc_scan_box.u_gmc_phy_wrapper.GEN_PHY.u_lpddr5x_ram_gs0.ram_0_0.ISOSRM;
assign u_lpddr5x_ram_gs0_ram_0_0_ISORET = sswrp_gmc_top_tb.lane[0].dut.u_gmc_top.u_gmc_scan_box.u_gmc_phy_wrapper.GEN_PHY.u_lpddr5x_ram_gs0.ram_0_0.ISORET;
assign u_lpddr5x_ram_gs0_ram_0_0_SD = sswrp_gmc_top_tb.lane[0].dut.u_gmc_top.u_gmc_scan_box.u_gmc_phy_wrapper.GEN_PHY.u_lpddr5x_ram_gs0.ram_0_0.SD;
assign u_lpddr5x_ram_gs0_ram_0_0_DSLP = sswrp_gmc_top_tb.lane[0].dut.u_gmc_top.u_gmc_scan_box.u_gmc_phy_wrapper.GEN_PHY.u_lpddr5x_ram_gs0.ram_0_0.DSLP;

  property sswrp_gmc_sram_u_lpddr5x_ram_gs0_ram_0_0_ISOSRM_cg_check_P;
    @(posedge clk) disable iff (ISOSRM_assert_disable)
      (gmc_init_done === 1) |-> $stable(u_lpddr5x_ram_gs0_ram_0_0_ISOSRM);
  endproperty
  property sswrp_gmc_sram_u_lpddr5x_ram_gs0_ram_0_0_ISORET_cg_check_P;
    @(posedge clk) disable iff (ISORET_assert_disable)
      (gmc_init_done === 1) |-> $stable(u_lpddr5x_ram_gs0_ram_0_0_ISORET);
  endproperty
  property sswrp_gmc_sram_u_lpddr5x_ram_gs0_ram_0_0_SD_cg_check_P;
    @(posedge clk) disable iff (SD_assert_disable)
      (gmc_init_done === 1) |-> $stable(u_lpddr5x_ram_gs0_ram_0_0_SD);
  endproperty
  property sswrp_gmc_sram_u_lpddr5x_ram_gs0_ram_0_0_DSLP_cg_check_P;
    @(posedge clk) disable iff (DSLP_assert_disable)
      (gmc_init_done === 1) |-> $stable(u_lpddr5x_ram_gs0_ram_0_0_DSLP);
  endproperty

  sswrp_gmc_sram_u_lpddr5x_ram_gs0_ram_0_0_ISOSRM_cg_check_A  : assert property (sswrp_gmc_sram_u_lpddr5x_ram_gs0_ram_0_0_ISOSRM_cg_check_P)
  else
    begin
   `ASSERT_RPT(
      "sswrp_gmc_sram_u_lpddr5x_ram_gs0_ram_0_0_ISOSRM_cg_check_P",
      $sformatf("ISOSRMToggled unexpectedly")
    )
    end
  sswrp_gmc_sram_u_lpddr5x_ram_gs0_ram_0_0_ISORET_cg_check_A  : assert property (sswrp_gmc_sram_u_lpddr5x_ram_gs0_ram_0_0_ISORET_cg_check_P)
  else
    begin
   `ASSERT_RPT(
      "sswrp_gmc_sram_u_lpddr5x_ram_gs0_ram_0_0_ISORET_cg_check_P",
      $sformatf("ISORETToggled unexpectedly")
    )
```

Snippet of the generated assertion file

- Create a Python script that accepts specific signal paths as input and generates a file containing system Verilog assertions that assert certain properties of these signals

# Approach

```
sswrp_gmc_top_tb.lane[0].dut.u_gmc_top.u_gmc_scan_box.u_gmc_phy_wrapper.GEN_PHY.u_lpddr5x_ram_gs0.ram_0_0.ISOSRM
sswrp_gmc_top_tb.lane[0].dut.u_gmc_top.u_gmc_scan_box.u_gmc_phy_wrapper.GEN_PHY.u_lpddr5x_ram_gs0.ram_0_0.ISORET
sswrp_gmc_top_tb.lane[0].dut.u_gmc_top.u_gmc_scan_box.u_gmc_phy_wrapper.GEN_PHY.u_lpddr5x_ram_gs0.ram_0_0.SD
```

Input Signals

```
logic u_lpddr5x_ram_gs0_ram_0_0_SD;
logic u_lpddr5x_ram_gs0_ram_0_0_DSLP;

assign u_lpddr5x_ram_gs0_ram_0_0_ISOSRM = sswrp_gmc_top_tb.lane[0].dut.u_gmc_top.u_gmc_scan_box.u_gmc_phy_wrapper.GEN_PHY.u_lpddr5x_ram_gs0.ram_0_0.ISOSRM;
assign u_lpddr5x_ram_gs0_ram_0_0_ISORET = sswrp_gmc_top_tb.lane[0].dut.u_gmc_top.u_gmc_scan_box.u_gmc_phy_wrapper.GEN_PHY.u_lpddr5x_ram_gs0.ram_0_0.ISORET;
assign u_lpddr5x_ram_gs0_ram_0_0_SD = sswrp_gmc_top_tb.lane[0].dut.u_gmc_top.u_gmc_scan_box.u_gmc_phy_wrapper.GEN_PHY.u_lpddr5x_ram_gs0.ram_0_0.SD;
assign u_lpddr5x_ram_gs0_ram_0_0_DSLP = sswrp_gmc_top_tb.lane[0].dut.u_gmc_top.u_gmc_scan_box.u_gmc_phy_wrapper.GEN_PHY.u_lpddr5x_ram_gs0.ram_0_0.DSLP;

  property sswrp_gmc_sram_u_lpddr5x_ram_gs0_ram_0_0_ISOSRM_cg_check_P;
    @(posedge clk) disable iff (ISOSRM_assert_disable)
      (gmc_init_done === 1) |-> $stable(u_lpddr5x_ram_gs0_ram_0_0_ISOSRM);
  endproperty
  property sswrp_gmc_sram_u_lpddr5x_ram_gs0_ram_0_0_ISORET_cg_check_P;
    @(posedge clk) disable iff (ISORET_assert_disable)
      (gmc_init_done === 1) |-> $stable(u_lpddr5x_ram_gs0_ram_0_0_ISORET);
  endproperty
  property sswrp_gmc_sram_u_lpddr5x_ram_gs0_ram_0_0_SD_cg_check_P;
    @(posedge clk) disable iff (SD_assert_disable)
      (gmc_init_done === 1) |-> $stable(u_lpddr5x_ram_gs0_ram_0_0_SD);
  endproperty
  property sswrp_gmc_sram_u_lpddr5x_ram_gs0_ram_0_0_DSLP_cg_check_P;
    @(posedge clk) disable iff (DSLP_assert_disable)
      (gmc_init_done === 1) |-> $stable(u_lpddr5x_ram_gs0_ram_0_0_DSLP);
  endproperty

  sswrp_gmc_sram_u_lpddr5x_ram_gs0_ram_0_0_ISOSRM_cg_check_A  : assert property (sswrp_gmc_sram_u_lpddr5x_ram_gs0_ram_0_0_ISOSRM_cg_check_P)
  else
    begin
    `ASSERT_RPT(
      "sswrp_gmc_sram_u_lpddr5x_ram_gs0_ram_0_0_ISOSRM_cg_check_P",
      $sformatf("ISOSRMToggled unexpectedly")
    )
    end
  sswrp_gmc_sram_u_lpddr5x_ram_gs0_ram_0_0_ISORET_cg_check_A  : assert property (sswrp_gmc_sram_u_lpddr5x_ram_gs0_ram_0_0_ISORET_cg_check_P)
  else
    begin
    `ASSERT_RPT(
      "sswrp_gmc_sram_u_lpddr5x_ram_gs0_ram_0_0_ISORET_cg_check_P",
      $sformatf("ISORETToggled unexpectedly")
    )
```

Generated assertion file

- The python script declared signals taking hierarchy into account to define the path
- Defined the properties of these signals using the script and asserted them using system Verilog assertions

# Accomplishments

- With the help of the script generated system Verilog assertions for signals can be automated
- The script has a generalized approach in utilizing the hierarchy of the signals present in the input file

# Thank You!