AN 1007: Partial Reconfiguration Design Simulation Tutorial

Updated for Quartus® Prime Design Suite: **24.1**

Contents

819958 | 2024.05.17

Send Feedback

1. Partial Reconfiguration Design Simulation Tutorial Introduction

This application note demonstrates the simulation of a partial reconfiguration (PR) design in a third-party simulation tool.

Simulate a partial reconfiguration design to verify the functionality of the design with each PR persona implemented before placing the design in hardware. Simulation also provides valuable insight into what happens in your design when changing between PR personas and can show if a design is dependent on startup conditions.

Simulating your PR design helps you improve your design in the following ways:

- Verifying your design and PR persona functionality
- Identifying any design gaps when changing between the PR personas
- Identifying any design dependencies on startup conditions.

The implementation of the reference design discussed in this application note requires a basic familiarity with the Quartus® Prime Pro Edition software implementation flow, Partial Reconfiguration terminology, and knowledge of the Quartus Prime Pro Edition primary project files.

Related Information

- Partial Reconfiguration Terminology
- Partial Reconfiguration Design Flow
- Partial Reconfiguration Design Considerations
- Partial Reconfiguration Design Guidelines
- Partial Reconfiguration Design Simulation

2. Partial Reconfiguration Simulation Overview

The Quartus Prime Pro Edition software supports simulation of partial reconfiguration (PR) persona transitions through use of simulation multiplexers. You use the simulation multiplexers to change which persona drives logic inside the PR region during simulation. The simulation multiplexers can also drive the output with unknown values (X) during partial reconfiguration. You can use these multiplexers to simulate random values exiting the PR region, and to identify potentially problematic data.

The tutorial describes the following simulation methods:

- RTL Simulation of Static and PR Regions

  A simulation method similar to traditional RTL simulation, with the addition of simulation multiplexers. This method shows behavior of the static region during transitions. You can use RTL simulation to verify that unknown values exiting the PR region do not harm the static region functionality.

- PR Simulation with post-synthesis gate-level netlist (simulation model) of the persona

  A slower simulation method that provides the additional capability to inject unknown values into the registers inside the PR region. The gate-level PR simulation model allows for accurate simulation of registers in your design. You can use this simulation method to verify of reset sequences and detect adverse initial conditions for PR.

Similar to non-PR design simulations, preparing for a PR simulation involves setting up your simulator working environment, compiling simulation model libraries, and running your simulation.

3. Partial Reconfiguration Simulation Capabilities

The partial reconfiguration (PR) simulation reference design in this tutorial demonstrates the following key simulation capabilities:

- **PR Persona-swapping simulation**

  While traditional simulation methods can accurately simulate a single PR persona in one system, they are ineffective at simulating the transition between any two personas. Upon partial reconfiguration completion, the overall system must be aware of the PR region logic change and adapt correctly. Simulation of the persona swapping overcomes the limitation of the traditional simulation methods and allows to verify the transition between PR personas. This simulation includes all static region logic and PR region logic for handling the PR handshake interface.

- **Freeze logic simulation**

  All outputs of a PR region are in the unknown “X” state during reconfiguration and freeze logic protects the static region during PR from receiving random data. With an incorrect behavior of the freeze logic, the system can freeze or lockup in the hardware. The simulation of unknown "X" outputs of the PR persona during partial reconfiguration allows you to verify your freeze logic. This simulation ensures that all the logic in the PR region are in the unknown "X" state during reconfiguration, and then verifies that these unknown “X” states do not adversely affect the rest of the static design during PR.

- **PR Reset-sequence simulation**

  All registers in a PR region are in the unknown “X” state during reconfiguration. These registers in the PR region of the device are not guaranteed to be in any defined state after partial reconfiguration, in contrast to full chip configuration (registers are guaranteed to power-up at 0). The simulation of unknown "X" state of all registers in the PR persona helps to verify that the **reset sequence** for the PR region following the reconfiguration process works correctly and brings all the registers in the PR region to a known state prior to operation.

4. Types of Verification Achievable Through Simulation

Through simulation, you can achieve the following types of verification:

- Basic Functional Verification
- Partial Reconfiguration Reset Sequence Verification
- Persona Swapping Verification

Basic Functional Verification

- Is performed using the RTL of a design.
- Can test the persona in isolation using block level simulation, or the complete design with 1 persona instantiated.
- Would not simulate the switching of PR personas, or the reconfiguration activity

Requirements:

- The design under test RTL

Partial Reconfiguration Reset Sequence Verification

- Verifies the functionality of the persona starting right after PR has completed
- Can be done in the context of the whole design or as a block level simulation of just the persona.

Requirements:

- Needs to load the registers of the PR region with X, 0, 1 or random value
- Uses the "PR mode" of the EDA netlist writer: Generates the post synthesized netlist of the PR region
- Loads 0/1/X/rand into all registers of the persona

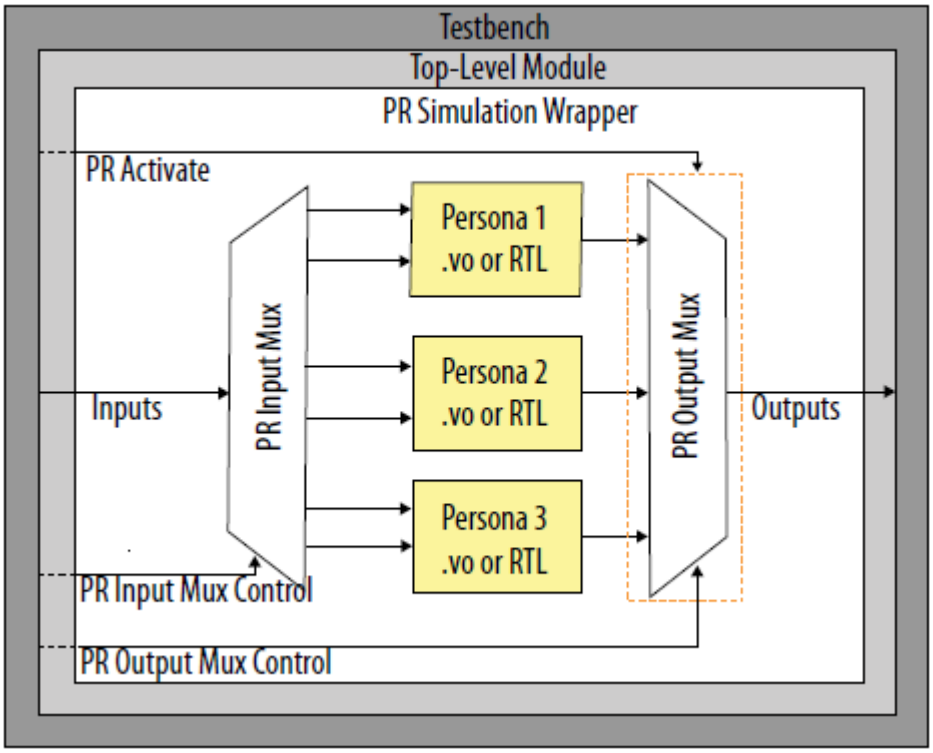
Persona Swapping Verification

- Simulates the swapping of PR personas for a given PR partition
- Can use RTL or gate level netlist models for personas
- When using the "PR mode" netlist models (post-synthesis gate-level netlists) the reset of the PR persona can also be simulated

Requirements:

- Input and output multiplexers to specify the active persona
- Simulation wrapper

Figure 1. Simulation of PR Persona Swapping



5. Partial Reconfiguration Simulation Requirements

"PR mode" of the EDA netlist writer (Required for PR Reset Sequence Verification)

"PR mode" of the EDA netlist is a post-synthesis technology mapped netlist for each persona. It uses PR simulation model of DFFEAS and models:

- The physical effect of registers powering up as unknown 'X' after PR
- Forcing the registers to unknown 'X' during PR
  - Requires the registers to be explicitly reset to a known state after PR
  - Causes all outputs of the PR region being unknown 'X' during PR
- Supports loading 0, 1, or random 0/1 during PR (instead of X)
- X-pessimistic LUTs models

PR Mode DFFEAS

Adds a new simulation only port on the model, **pr_activate**, to async load X into the register

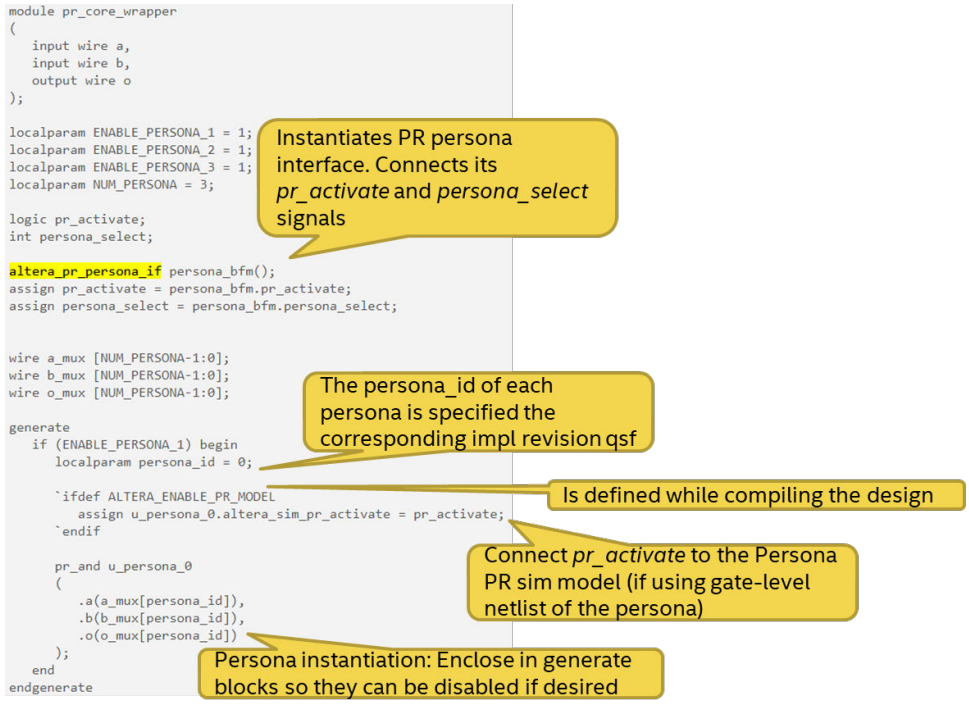
- This has the effect of making the register drive the unknown state
- Ties ***pr_activate*** to a top level signal of the simulation model for the PR persona and drive that signal from the testbench
- Generates reports for the register values if any of the corresponding events are triggered

PR Simulation Wrapper (Required for Persona Swapping Simulation)

PR Simulation Wrapper instantiate all personas in a single module, Personas can be RTL or PR Simulation model. It also includes

- Input and output multiplexers (altera_pr_wrapper_mux_in & altera_pr_wrapper_mux_out)
- PR region interface (altera_pr_persona_if)
  - Allows to connect the wrapper multiplexers to a testbench driver
- The PR select lines: PR Input/Output Mux Control
  - Simulation-only signals inside the PR region model: to switch between the different PR personas
  - Can be driven by the testbench

Figure 2. PR Simulation Wrapper file example



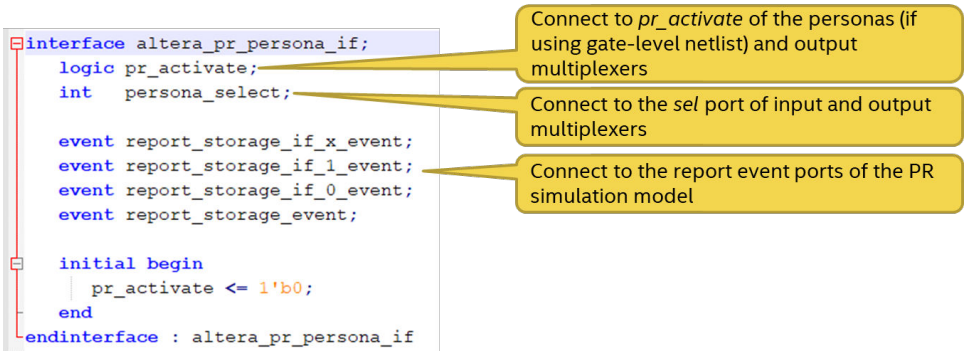

```
module pr_core_wrapper
(
   input wire a,
   input wire b,
   output wire o
);

localparam ENABLE_PERSONA_1 = 1;
localparam ENABLE_PERSONA_2 = 1;
localparam ENABLE_PERSONA_3 = 1;
localparam NUM_PERSONA = 3;

logic pr_activate;
int persona_select;

altera_pr_persona_if persona_bfm();
assign pr_activate = persona_bfm.pr_activate;
assign persona_select = persona_bfm.persona_select;


wire a_mux [NUM_PERSONA-1:0];
wire b_mux [NUM_PERSONA-1:0];
wire o_mux [NUM_PERSONA-1:0];

generate
   if (ENABLE_PERSONA_1) begin
      localparam persona_id = 0;

      `ifdef ALTERA_ENABLE_PR_MODEL
         assign u_persona_0.altera_sim_pr_activate = pr_activate;
      `endif

      pr_and u_persona_0
      (
         .a(a_mux[persona_id]),
         .b(b_mux[persona_id]),
         .o(o_mux[persona_id])
      );
   end
endgenerate
```

```
altera_pr_wrapper_mux_in #(.NUM_PERSONA(NUM_PERSONA), .WIDTH(1)) \
     u_a_mux(.sel(persona_select), .mux_in(a), .mux_out(a_mux));

altera_pr_wrapper_mux_in #(.NUM_PERSONA(NUM_PERSONA), .WIDTH(1)) \
     u_b_mux(.sel(persona_select), .mux_in(b), .mux_out(b_mux));

altera_pr_wrapper_mux_out #(.NUM_PERSONA(NUM_PERSONA), .WIDTH(1)) \
     u_o_mux(.sel(persona_select), .mux_in(o_mux), .mux_out(o), .pr_activate
     (pr_activate));

endmodule
```

PR Region Interface : altera_pr_persona_if

It is defined in <*QUARTUS_INSTALL_DIR*>/eda/sim_lib/altera_lnsim.sv

Figure 3. altera_pr_persona_if file



PR Input and Output Multiplexers

- Defined in <QUARTUS_INSTALL_DIR> /eda/sim_lib/altera_lnsim.sv
- Designate which persona is active in the simulation
- Used for swapping the personas in the design

Figure 4. altera_pr_wrapper_mux_in & altera_pr_wrapper_mux_out files

```
module altera_pr_wrapper_mux_in#(
    parameter NUM_PERSONA = 1,
    parameter WIDTH = 1,
    parameter [0:0] DISABLED_OUTPUT_VAL = 1'bx
) (
    input int sel,
    input wire [WIDTH-1:0] mux_in,
    output reg [WIDTH-1 : 0] mux_out [NUM_PERSONA-1:0]
);
    always_comb begin
        for (int i = 0; i < NUM_PERSONA; i++)
            if (i == sel)
                mux_out[i] = mux_in;
            else
                mux_out[i] = {WIDTH{DISABLED_OUTPUT_VAL}};
    end


endmodule : altera_pr_wrapper_mux_in
```

Specifies the active persona

```
module altera_pr_wrapper_mux_out #(
    parameter NUM_PERSONA = 1,
    parameter WIDTH = 1,
    parameter [0:0] DISABLED_OUTPUT_VAL = 1'bx
) (
    input int sel,
    input wire [WIDTH-1 : 0] mux_in [NUM_PERSONA-1:0],
    output reg [WIDTH-1:0]   mux_out,
    input wire               pr_activate
);

    always_comb begin
        if ((sel < NUM_PERSONA) && (!pr_activate))
            mux_out = mux_in[sel];
        else
            mux_out = {WIDTH{DISABLED_OUTPUT_VAL}};
    end

endmodule : altera_pr_wrapper_mux_out
```

Specifies the active persona

Allows to drive the mux output to X

Testbench

- Drives the pr_activate and persona_select signals
- Selects a persona, asserts pr_activate to start PR, deasserts pr_activate, triggers the report storage events, resets the PR region

Figure 5. PR Simulation Wrapper file example

```
    assign dut.pr_region.pr_activate = pr_activate;
    assign dut.pr_region.persona_select = persona_select;
```

```
//pr_counter persona
#`CLOCK_CYCLE
    pr_activate = 1;
    persona_select = 3'b001;
    $display ("STARTING PR: Switching to pr_counter  @time = %t", $time);
repeat(5) #`CLOCK_CYCLE //Partial Reconfiguration takes a non-trvial amount of time
    $display ("DURRING PR: o = %d                    @time = %t", o, $time);
#`CLOCK_CYCLE
    pr_activate = 1'b0;
    //asserting the report storage events will report any registers that are at 0, 1 or x
    //asserting report_storage_if_x_event is good for detecting inital conditions in
    you're pr region
    -> dut.pr_region.report_storage_if_0_event;
    -> dut.pr_region.report_storage_if_1_event;
    -> dut.pr_region.report_storage_if_x_event;
    $display ("FINISHING PR:                         @time = %t", $time);
//uncomment this to increase the time between PR and reset
//repeat(5) #`CLOCK_CYCLE
#`CLOCK_CYCLE
    reset_n = 1'b0;

#`CLOCK_CYCLE
    reset_n = 1'b1;
    $display ("COUNTER PERSONA: counter = o = %4d   @time = %t", o, $time);
repeat(15) #`CLOCK_CYCLE
    $display ("COUNTER PERSONA: counter = o = %4d   @time = %t", o, $time);
```

6. Partial Reconfiguration Simulation Reference Design Requirements

This tutorial requires the following software and design example files:

- SUSE* Linux Enterprise Server 12
- Quartus Prime Pro Edition Version 24.1
- Questa* Intel® FPGA Edition or Siemens* EDA Questa* Advanced Simulator Version 2023.4
- The example design project files for the design implementation. For details about these files, refer to Partial Reconfiguration Simulation Reference Design Files on page 16.

These requirements reflect the tested setup conditions. You might be able to adapt this reference design for later software versions and different operating systems (Linux or Windows).

7. Partial Reconfiguration Simulation Reference Design Overview

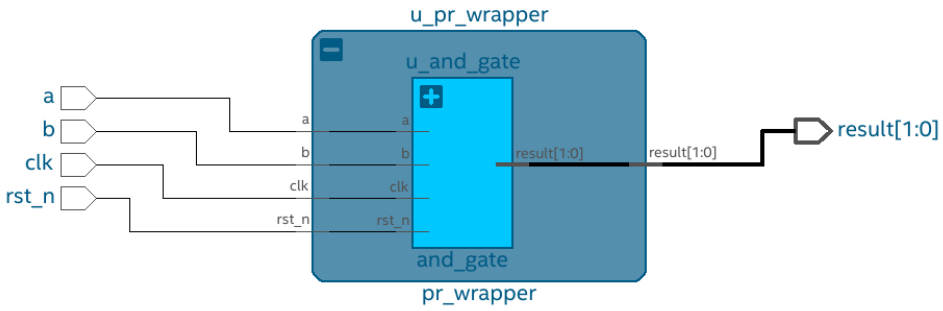
The reference design illustrates RTL and partial reconfiguration (PR) simulations in the Questa simulation software. It implements three PR personas inside a PR region. You can swap out the persona in the FPGA by using the partial reconfiguration process.

The personas in the reference design are as follows

- An AND gate
- A counter
- A finite state machine (FSM)

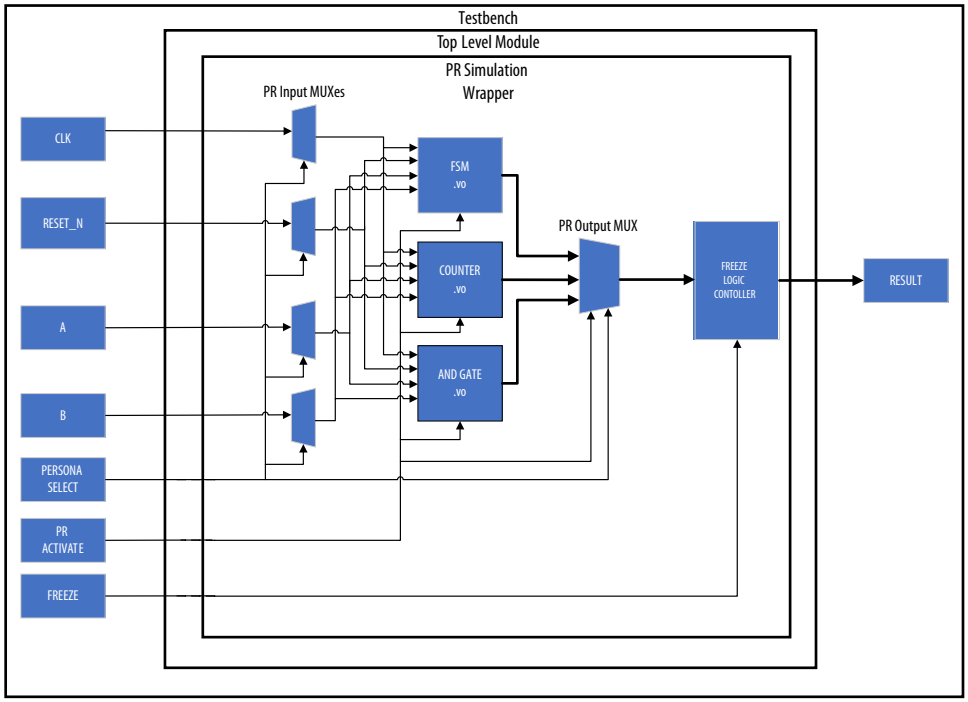
Each of these personas has a 100 MHz clock, an active low reset, two 1-bit inputs and one 2-bit output.

Figure 6. Partial Reconfiguration Reference Design Base Revision Block Diagram



The following figure shows the PR simulation design architecture. You can simulate the partial reconfiguration event by using simulation multiplexers on the input and output of the personas. These multiplexers are used to change which persona drives logic inside the PR region during simulation.

Figure 7. PR Simulation Design Block Diagram



The **PR Simulation Wrapper** is the main component of the simulation design. It contains instantiations of the gate-level PR simulation model of each persona, PR input and output multiplexers, and freeze logic controller module.

Simulation only signals **PR Activate**, **Persona Select** and **Freeze** do not pass through the **Top Simulation Module**, but rather pass directly from the **Testbench** to the **PR Simulation Wrapper**. This technique is required because the testbench and simulation wrapper are not synthesizable, while the RTL top-level modules and personas are synthesizable in a real design scenario. The **Testbench** controls the signals as they pass through to the **PR Simulation Wrapper** directly using a relative path.

You can use the **PR output MUX** to output unknown values (X), when PR is active, to verify that unknown values exiting the PR region do not propagate into the static region. The unknown (X) outputs of the PR persona allow you to verify your freeze strategy.

You can trigger the **PR simulation register model** to show unknown (X) value in all registers of the persona. This technique allows you to validate the reset sequence by showing that all registers return to a known state after PR completes and you apply reset.

The RTL simulation as well as the PR simulation (with gate-level PR simulation model of the personas), both use the same **Testbench** and **PR Simulation Wrapper**.

8. Partial Reconfiguration Simulation Reference Design Files

The reference design files required for this tutorial are available in the Intel FPGA Partial Reconfiguration Design Flow GitHub repository at https://github.com/intel/fpga-partial-reconfig.

To access the tutorial design:

1. Go the GitHub repository at https://github.com/intel/fpga-partial-reconfig.
2. On the repository page, click **Code ➤ Download ZIP**.
3. Unzip the `fpga-partial-reconfig-master.zip` file.
4. Navigate to the subfolder for your FPGA device:
   - Agilex™ 5: `tutorials/agilex5_pcie_devkit_pr_sim`
   - Agilex 7: `tutorials/agilex7_pcie_devkit_pr_sim`
   - Arria® 10: `tutorials/a10_pcie_devkit_pr_sim`
   - Stratix® 10: `tutorials/s10_pcie_devkit_pr_sim`

The device folder contains two subfolders:

- The `pr` folder contains the files that you need to follow this tutorial.
- The `pr_sim` folder contains the complete set of files that you create using this application note.

Table 1. PR Simulation Reference Design Files

File Name	Description
`top.sv`	Top-level file containing the flat implementation of the design. This file instantiates the `pr_wrapper` module.
`pr_wrapper.sv`	Wrapper file for instantiating personas in the PR region. Default persona: AND gate.
`and_gate.sv`	Persona Implementation: Synchronized AND gate with active low reset. Outputs the result of inputs ANDed together.
`counter.sv`	Persona Implementation: Counter with active low reset. Outputs the top two MSBs of the counter.
`fsm.sv`	Persona Implementation: FSM with active low reset.

The `pr_sim` directory contains the following additional design files:

Table 2. Simulation-Specific Reference Design Files

File Name	Description
top_tb.sv	Testbench file containing the simulation logic and the instantiation of top_sim module. The file also instantiates: • Gate-level PR simulation model of the personas • RTL version of the personas The top_sim module is the minimum requirement for the simulation. The personas are included to demonstrate the differences between using the gate-level PR simulation model versus the RTL for the personas.
top_sim.sv	Top-level file for the simulation. This file instantiates the pr_sim_wrapper module.
pr_sim_wrapper.sv	Wrapper file for instantiating the following items: • Gate-level PR simulation model of the personas • PR input and output multiplexers • freeze_logic_controller module The file also includes the define ALTERA_ENABLE_PR_MODEL definition that directs the compiler to create and use the PR simulation register model for each register within a persona.
freeze_logic_controller.sv	Control logic to freeze the PR region output while the PR transition is occurring. It allows the output to be set to a known value when activated and prevents any output from a persona during a PR transition to propagate into static portion of design.
gen_pr_models.do	Script for generating gate-level PR simulation model for the personas.
run_pr_sim.do	Script for generating gate-level PR simulation model for the personas and running the simulation.
run_pr_sim_only.do	Script for running simulation (skips gate-level PR simulation model generation for the personas).

Figure 8. gen_pr_models.do file

gen_pr_models.do

```
# Record all commands done in transcript window
transcript on

# Remove previous contents in pr directory
rm -rf ./pr

set project_name        pr_sim
set base_revision_name  top
set static_qdb_name     top_static.qdb

set root_partition      root_partition
set pr_partition        pr_partition

set persona_names {
    and_gate
    counter
    fsm
}

#####################################
# Make PR Persona Sim Models
#####################################

puts "Running Analysis & Synthesis on base revision"
# quartus_syn <project name> -c <base revision name>
quartus_syn $project_name -c $base_revision_name

puts "Exporting base partition from base revision"
#quartus_cdb <project name> -c <base revision name> --export_block root_partition --snapshot synthesized --file <static qdb name>
quartus_cdb $project_name -c $base_revision_name --export_block $root_partition --snapshot synthesized --file $static_qdb_name


#####################################
# Make Each Persona Sim Model
#####################################

foreach persona_name $persona_names {

    puts "Running Analysis & Synthesis on persona revision: $persona_name"
    # quartus_syn <project name> -c <persona revision name>
    quartus_syn $project_name -c $persona_name

    puts "Make PR sim model for $persona_name persona"
    #quartus_eda <project name> -c <persona revision name> --pr --simulation --tool=questasim --format=verilog --partition=<pr partition name> --module=<partition name>=<persona module name>
    quartus_eda $project_name -c $persona_name --pr --simulation --tool=questasim --format=verilog --partition=$pr_partition --module=$pr_partition=$persona_name
}
```

Figure 9. run_pr_sim.do file

run_pr_sim.do

```
# Record all commands done in transcript window
#transcript on

rm -rf work
vlib work

set quartus_dir             /p/psg/swip/releases/acds/24.1/115/linux64/quartus
set quartus_sim_lib_path    $quartus_dir/eda/sim_lib
set root_partition          root_partition
set pr_partition            pr_partition
set pr_sim_model_path       simulation/questa/pr

set testbench_files {
    top_tb.sv
}

set design_files {
    top.sv
    pr_wrapper.sv
    top_sim.sv
    pr_sim_wrapper.sv
    freeze_logic_controller.sv
}

# Used to compile persona pr sim models below
set persona_names {
    and_gate
    counter
    fsm
}

set library_files {
    altera_lnsim.sv
    tennm_atoms.sv
    altera_primitives.v
}

puts "============= Running PR Simulation ============="

puts "Generate PR simulation models"
do gen_pr_models.do

puts "Compiling testbench files"
foreach file $testbench_files {
    vlog -sv -work work $file
}

puts "Compiling design files"
foreach file $design_files {
    vlog -sv -work work $file
}

puts "Compiling pr persona files"
foreach persona_name $persona_names {
    vlog -sv -work work $pr_sim_model_path/$persona_name.$pr_partition.vo
}

puts "Compiling library files"
foreach library_file $library_files {
    vlog -sv -work work $quartus_sim_lib_path/$library_file
}

puts "Start simulation"
vsim -t lps -L work -voptargs="+acc" top_tb

puts "Running waveform file"
do wave.do

puts "Running sim"
run -all
puts "============= Finished PR Simulation ============="


```

9. PR Simulation Reference Design Walkthrough

This reference design walkthrough includes the following steps:

1. Get started.
2. Create a project revision for PR simulation.
3. Add RTL design files to the `pr_sim` revision.
4. Add PR simulation design files to the `pr_sim` revision.
5. Generate gate-level PR simulation models for the persona.
6. Prepare the setup simulation script.
7. Run PR simulation.

9.1. Step 1: Getting Started

Begin the reference design walkthrough by setting up the working environment to create the PR simulation reference design:

1. Create a `/pr` working directory in your system working environment.
2. Copy one of the following tutorial folders for your FPGA device to you new `/pr` working directory:
   - `tutorials/agilex5_pcie_devkit_pr_sim/pr`
   - `tutorials/agilex7_pcie_devkit_pr_sim/pr`
   - `tutorials/a10_pcie_devkit_pr_sim/pr`
   - `tutorials/s10_pcie_devkit_pr_sim/pr`
3. In the Quartus Prime Pro Edition software, click **File ➤ Open Project** and select the `/pr/pr_sim.qpf` project file.

9.2. Step 2: Creating a Project Revision for PR Simulation

You can simplify the setup of a PR simulation design by creating another project revision within your existing Quartus Prime project. This extra revision is helpful because it can help to simplify the generation and use of the gate-level PR simulation models for simulation of each persona.

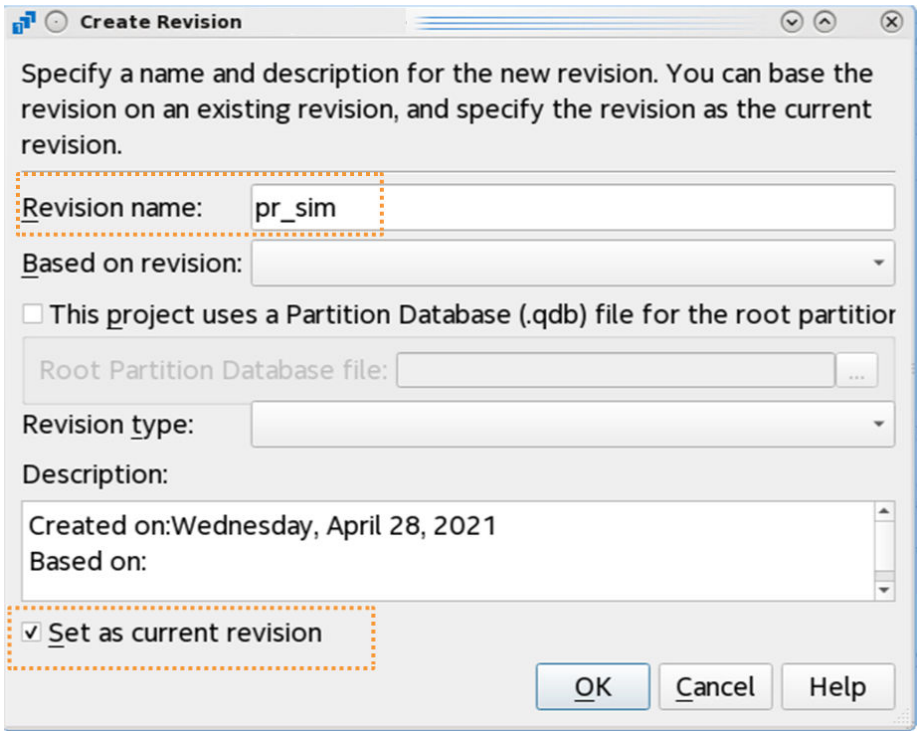
To create a project revision for the PR simulation design example:

1. In the Quartus Prime Pro Edition software, click **Project ➤ Revisions**.
2. In the **Revisions** dialog box, double-click **<<new revision>>**.
3. Specify the following values for the PR simulation project revision:

Table 3. PR Simulation Project Revision Settings

Setting	Value for Design Example
Revision name	`pr_sim`
Based on revision	Leave blank
Revision type	Leave blank
Set as current revision	Turn on (checked)

Figure 10. Creating `pr_sim` Project Revision



4. Click **OK** to save your settings.
5. Select the correct FPGA OPN for you project.
6. In the **Project Navigator** window, change the Top-Level Entity from `pr_sim` to `top_tb`.

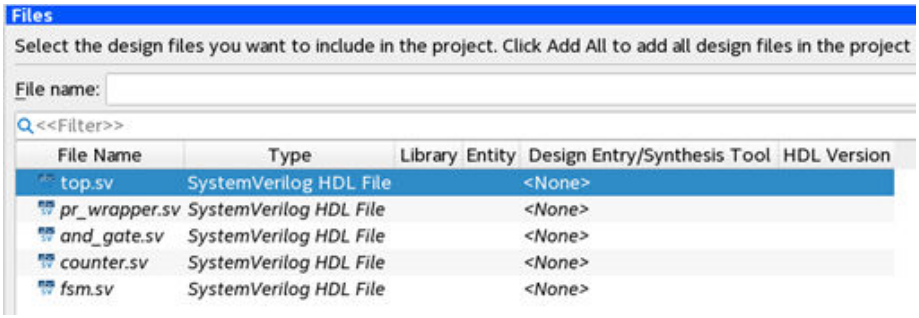
9.3. Step 3: Adding RTL Design Files to the pr_sim Revision

To complete the pr_sim revision, create and add the RTL design files to the revision to support functional simulation. For this design example, add design files from the other project revisions, rather than creating the files from scratch.

1. In the Quartus Prime Pro Edition software, click **Project ➤ Add/Remove Files in Project**.
2. In the **Files** tab, add the following design files for the **pr_sim** revision:

- top.sv
- pr_wrapper.sv
- and_gate.sv
- counter.sv
- fsm.sv

Figure 11. Adding RTL Design Files to the pr_sim Revision



3. Click **OK** to close the Settings dialog box.

9.4. Step 4: Adding PR Simulation Design Files to the pr_sim Revision

The PR Simulation design example project already includes the RTL design files. However, you must also add the following files from the Simulation-Specific Reference Design Files table in Partial Reconfiguration Simulation Reference Design Files on page 16 to the **pr_sim** revision to support simulation:

1. Copy the following files listed in the Simulation-Specific Reference Design Files table from the `pr_sim` folder to your `pr` folder:
   - `top_tb.sv`
   - `top_sim.sv`
   - `pr_sim_wrapper.sv`
   - `freeze_logic_controller.sv`
2. In the Quartus Prime Pro Edition software, click **Project ➤ Add/Remove Files in Project**.
3. In the **Files** tab, add the files for the **pr_sim** revision.
4. Click **OK** to close the Settings dialog box.

9.5. Step 5: Generating Gate-level PR Simulation Models for the Persona

Use the PR mode of **Intel Quartus Prime EDA Netlist Writer** through the command line to create the simulation model for a PR persona. The simulation model represents post-synthesis gate-level netlist for the PR persona.

When using the PR simulation model for the persona, the netlist includes a new **altera_sim_pr_activate** top-level signal for the model. You can asynchronously drive this signal to load all registers in the model with X. Use this feature to verify the reset sequence of the new persona on PR event completion.

By default, the PR simulation model asynchronously loads X into the register storage element on `pr_activate` signal assertion. You can parametrize this behavior on a per-register basis, or on a simulation-wide default basis. The simulation model supports four built-in modes:

- load X
- load 1
- load 0
- load rand

Before running a simulation, generate the gate-level simulation models (that is, `.vo` files) for the personas. The design example includes the `pr_sim/gen_pr_models.do` script that automates this process for you.

The following high-level steps summarize the process that the script automates:

1. Open the base revision of a PR project in Intel Quartus Prime Pro Edition, and then click **Processing ➤ Start ➤ Start Analysis & Synthesis**. Alternatively, run this command-line equivalent:

   ```
   quartus_syn <project name> -c <base revision name>
   ```

2. After synthesis is complete, click **Project ➤ Export Design Partition**, and then select the **root partition** for the Partition name, and select **synthesized** for the **Snapshot**. Click **OK**. Alternatively, run this command-line equivalent:

   ```
   quartus_cdb <project name> -c <base revision name> \
      "--export_block root_partition --snapshot synthesized \
      --file <static qdb name>
   ```

3. Click **Project ➤ Revisions** and switch the current revision to that of the persona you want to export. Click **Processing ➤ Start ➤ Start Analysis & Synthesis**. Alternatively, run this command-line equivalent:

   ```
   quartus_syn <project name> -c <persona revision name>
   ```

4. After synthesis of the persona revision completes, execute the following at the command line to generate the PR simulation model:

   ```
   quartus_eda <project name> –c <persona revision name> \
      "--pr–simulation --tool=questasim --format=verilog \
       --partition=<pr partition name> \
       --module=<partition name>=<persona module name>
   ```

5. Repeat steps 3 and 4 for all the personas that you want to simulate.

The EDA Netlist Writer GUI does not support PR Mode. Use command line to generate post-synthesis gate-level PR simulation model for the personas.

Related Information

- Generating the PR Persona Simulation Model
- Quartus Prime Pro Edition User Guide: Partial Reconfiguration

9.6. Step 6: Preparing the Setup Simulation Script

After design setup and PR simulation model generation, you can begin PR simulation. This design example includes the `run_pr_sim.do` simulator setup file. This script automatically compiles the testbench, the design, the PR persona model, and library files, and then starts and runs simulation. You can adapt this design example `.do` script for your own design.

To prepare the setup simulation script:

1. Copy the `run_pr_sim.do` file from the `pr_sim` directory to your `pr` project directory.
2. Open the `run_pr_sim.do` file in a text editor.
3. In the `run_pr_sim.do` file, replace the *quartus_dir* variable with the value of the `QUARTUS_ROOTDIR` environment variable. To determine the path of the `QUARTUS_ROOTDIR` environment variable, run the following command:

   ```
   env | grep QUARTUS_ROOTDIR
   ```

4. Save the modified `run_pr_sim.do` file.

9.7. Step 7: Run PR Simulation

Once the `run_pr_sim.do` setup script modifications are complete, follow these steps to run the script and PR simulation:

1. Open the Questa Intel FPGA Edition or QuestaSim*
2. In the transcript section, navigate to your `pr` directory.
3. Run the simulation with the following command:

   ```
   do run_pr_sim.do
   ```

The simulation runs according to your specifications.

10. PR Simulation Results

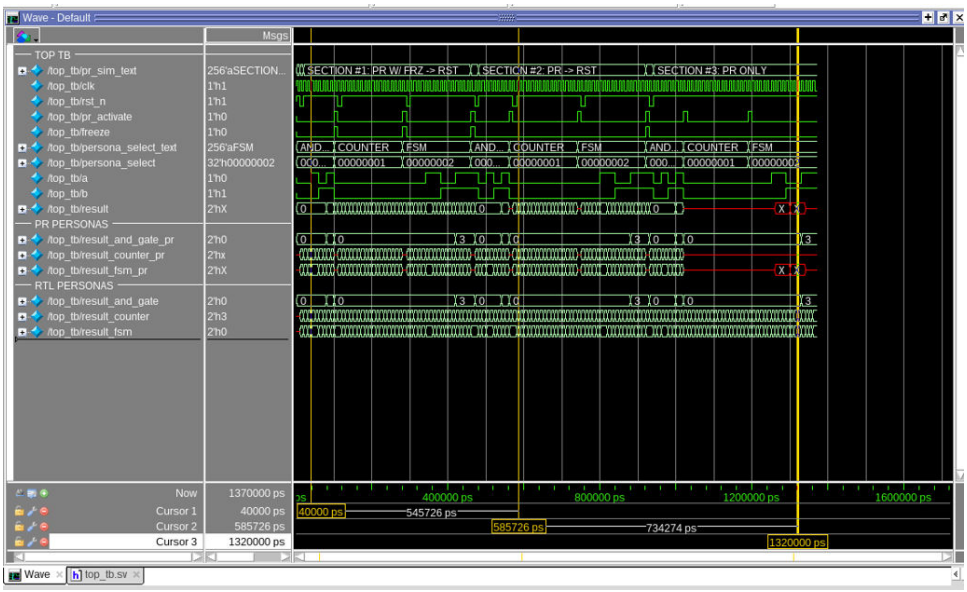
By looking at the Simulation waveforms, you can verify whether the signals behave consistently with your expectations. The simulation in this tutorial works as follows:

1. The simulation logic in the testbench `top_tb` drives the `pr_activate`, `persona_select` and `freeze` signals.
2. The testbench triggers a PR transition by asserting the `pr_activate` signal.
3. The `persona_select` signal selects the current persona. In addition, it also controls the PR input and output multiplexer select input lines to control the correct input and output of the persona.
4. The `pr_activate` signal forces the output of the PR output multiplexer to unknown (X) values (for freeze logic verification). In addition, it also triggers the PR simulation register model to show unknown (X) value in all registers of the persona (for reset sequence verification).
5. The `freeze` signal freezes the PR region output and sets to known value while the PR transition is occurring.
6. After some time transpires, the testbench de-asserts `pr_activate` signals and resets the PR persona.

The simulation of `top_sim` module is separated into the following sections:

- Initialization
- Initial Reset
- Section #1: PR with Freeze, then reset (PR W/ FRZ → RST)
- Section Reset
- Section #2: PR, then Reset (PR → RST)
- Section Reset
- Section #3: PR Only (PR ONLY)

Figure 12. Simulation Waveform



10.1. Persona Swapping Simulation

Persona swapping simulation verifies the swapping of PR personas for a given PR partition. For this simulation you can use RTL or gate level PR simulation model of the personas. The simulation requires PR input and output multiplexers to specify the active persona. The testbench sequence to run this simulation is as follows:

- Assert the `pr_activate` signal, while changing the `persona_select` signal on the PR input and output multiplexer select lines to select the desired persona.
- After finishing PR, the testbench deasserts the `pr_activate` signal.
- Reset the persona.

Refer to the `pr_sim_wrapper.sv` file for information on how to instantiate the PR multiplexers. The simulation library for these multiplexers is in `altera_lnsim.sv`.

The simulation steps are as follows:

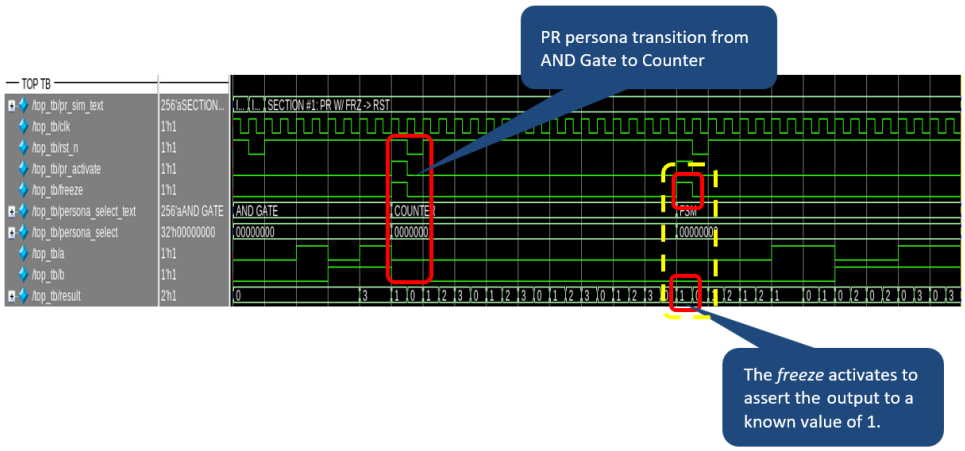
1. *Initialization* and *Initial Reset* occur within the first two cycles and transitions to **Section #1: PR with Freeze, then reset**.
2. The two 1-bit inputs, `a` and `b`, toggle through their four possible states, and the AND gate gives the correct response on the `result` output signal.
3. A PR transition occurs by activating the `pr_activate` signal and changing the `persona_select` to the *counter* persona. While this transition occurs, the `freeze` signal activates to assert the output to a known value of `1`.

   When PR is complete, after one cycle in this case, the `rst_n` reset signal asserts to reset the *counter* persona to a known value state.

For the reset to take effect, you may have to hold the `freeze` and `rst_n` signals for a longer period, depending on the persona pipeline depth delay. In this design example, each persona has a one cycle delay from input to output.

4. The 2-bit counter counts until the next PR transition occurs. The counter undergoes the same process as step 3 when transitioning to the *FSM* persona.
5. The inputs a and b toggle through its four possible states, and the FSM outputs the correct response while activated.

Figure 13. Persona Swapping Simulation Waveform



10.2. Reset Sequence Simulation

Reset sequence simulation verifies that persona functions start correctly after PR has completed. It can be done in the context of the whole design or as a block level simulation of just the persona. This simulation requires the post-synthesis gate-level PR simulation model of the persona. During the time when a PR operation occurs, you drive the `pr_activate` signal to set unknown (X) values into all registers inside the persona. After PR, the registers require to be explicitly reset to a known state.

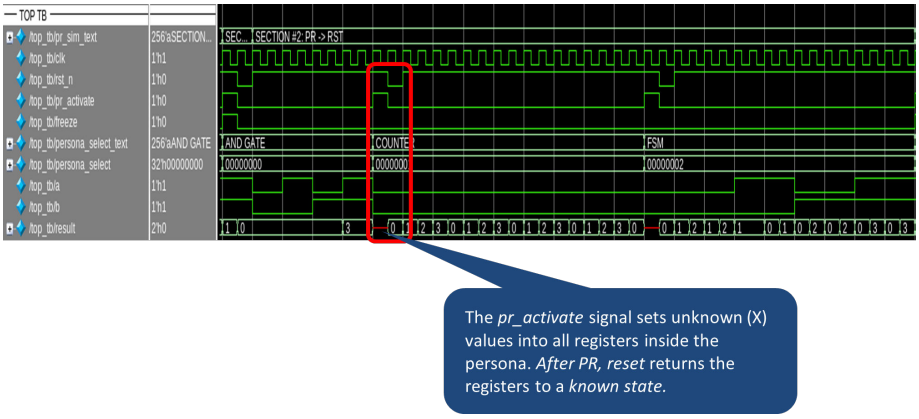
The testbench sequence to run this simulation consists of the following:

- Assert the `pr_activate` signal.
- After completing PR, deassert the `pr_activate` signal.
- Reset the persona.

The simulation steps are as follows:

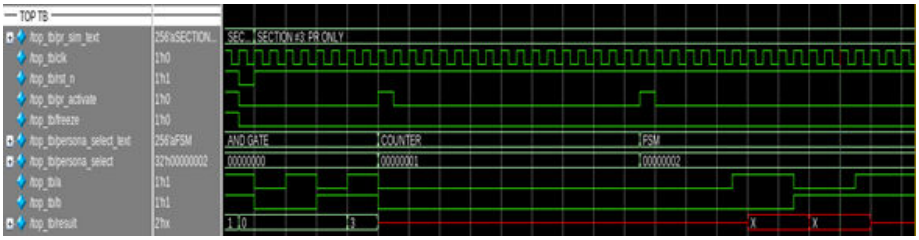
1. Section reset occurs over two cycles to reset the persona to the default AND gate persona and reset the persona to a known state to ensure proper initial functionality.
2. Simulation transitions to **Section #2: PR, then reset**.
3. The persona transition sequence occurs. However, during this PR transition, the `freeze` signal does not assert. As a result, the output is undefined (as X) during this time until the reset is asserted to reset the output and the persona to a known state. This undefined output is a feature of the post-synthesis gate-level PR simulation model to output an undefined value during PR transitions.

Figure 14. Reset Sequence Simulation Waveform: Persona Registers Reset to a Known State



4. *Section Reset* occurs again over two cycles to ensure the correct initial functionality. The simulation proceeds through the *AND gate* persona as expected.
5. Simulation transitions to **Section #3: PR Only**.
6. Once the PR transition occurs and completes the transition to the *counter* persona, the output remains undefined. This is a result of not applying the reset to registers within the persona to reset the registers to a known state. This feature of the generated post-synthesis gate-level PR simulation model allows the output to remain an unknown (X) value until entering the known reset state.
7. The undefined output continues throughout the *counter* persona, through the PR transition to the *FSM* persona, and through the *FSM* persona.

Figure 15. Reset Sequence Simulation Waveform: Persona Registers Do Not Reset to a Known State0

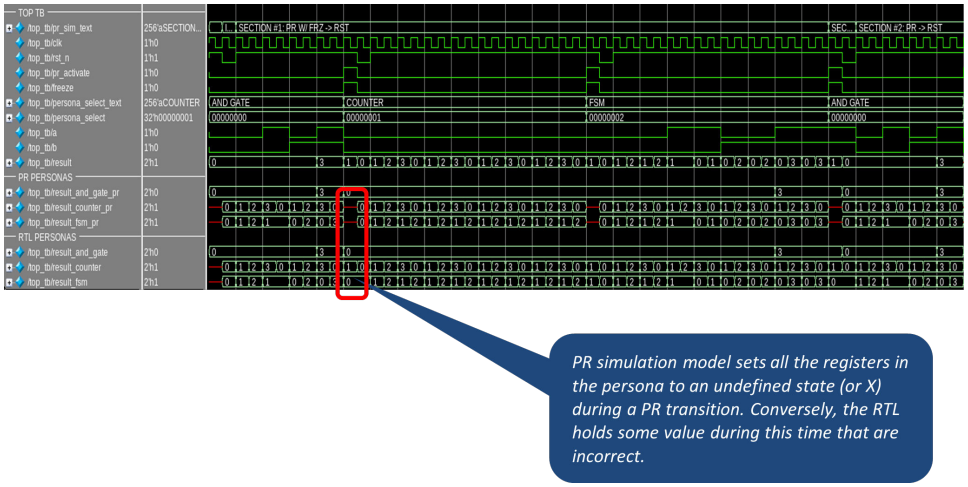


10.3. Persona using Post-Synthesis Gate-Level PR Simulation Model vs RTL

Along with the `top_sim` testbench simulation, the RTL personas and PR simulation model of the personas are additional testbench subsections of interest for demonstrative purposes only. These subsections demonstrate the differences between the PR simulation model and the RTL code.

Each testbench subsection displays the output of each of the personas using either the PR simulation models (generated earlier) or the RTL. The main differences in the use of the personas are when PR transactions occur, as the following figure illustrates:

Figure 16. Persona using Post-Synthesis Gate-Level PR Simulation Model vs RTL



When the first PR transition occurs in *Section #1: PR with Freeze, then reset*, there is a difference in the output of the PR simulation model and RTL persona simulations. This is a feature of the PR simulation model to set all the registers in the persona to an undefined state (or *X*) during a PR transition. Conversely, the RTL holds the value during this time.

While use of the RTL for simulation can be familiar and convenient, the implemented PR simulation models more closely emulate the behavior of the hardware during actual PR transitions.

11. Document Revision History for AN 1007: Partial Reconfiguration Design Simulation Tutorial

Document Version	Quartus Prime Version	Changes
2024.05.17	24.1	Initial release.