**Stratix10 Sector-Oriented PR Demonstrator**

**Intel Labs**

**Revision History:**

| Revision | Date | Comments |
|---|---|---|
| 0.1 | 03/25/2019 | Initial draft |
| | | |
| | | |
| | | |
| | | |

## FPGA Network-on-Chip (NoC)[1]

### Overview

This document aims to describe the implementation of 3x3 sector-aligned NoC design on Stratix10 FPGA. Each sector consists of user region (reconfigurable) and NoC region.

Master of a sector can send packets to any other 8 sectors by forwarding to its neighbor sector (north/ east/ south/ west). The data will be stored in FIFO of the destination sector. The longer the data path, the more sectors the packet will be rerouted to, as each sector can only communicate with its immediate neighbor.

In this design, sector 0 will act as a master and JTAG interface for debugging. It can also enable/ disable security filter in each sector to block/ unblock data transfer.

| | | |
|---|---|---|
| 0 | 1 | 2 |
| 3 | 4 | 5 |
| 6 | 7 | 8 |

*Sectors naming*

Sector-sector bandwidth is roughly 133 MHz / 12 cycles latency * 4 bytes/transfer = 44 MB/s.

Software Requirement: Intel® Quartus® Prime Design Suite 18.0.0/219

[1] Legal Disclaimers: Intel provides these materials as-is, with no express or implied warranties. All products, dates, and figures specified are preliminary, based on current expectations, and are subject to change without notice. Intel processors, chipsets, and desktop boards 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. Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No product or component can be absolutely secure. Check with your system manufacturer or retailer or learn more at http://intel.com. Some results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance. Intel, Quartus, Stratix, the Intel logo and the Stratix logo are trademarks of Intel Corporation in the United States and other countries. Other names and brands may be claimed as the property of others.
© Intel Corporation 2020

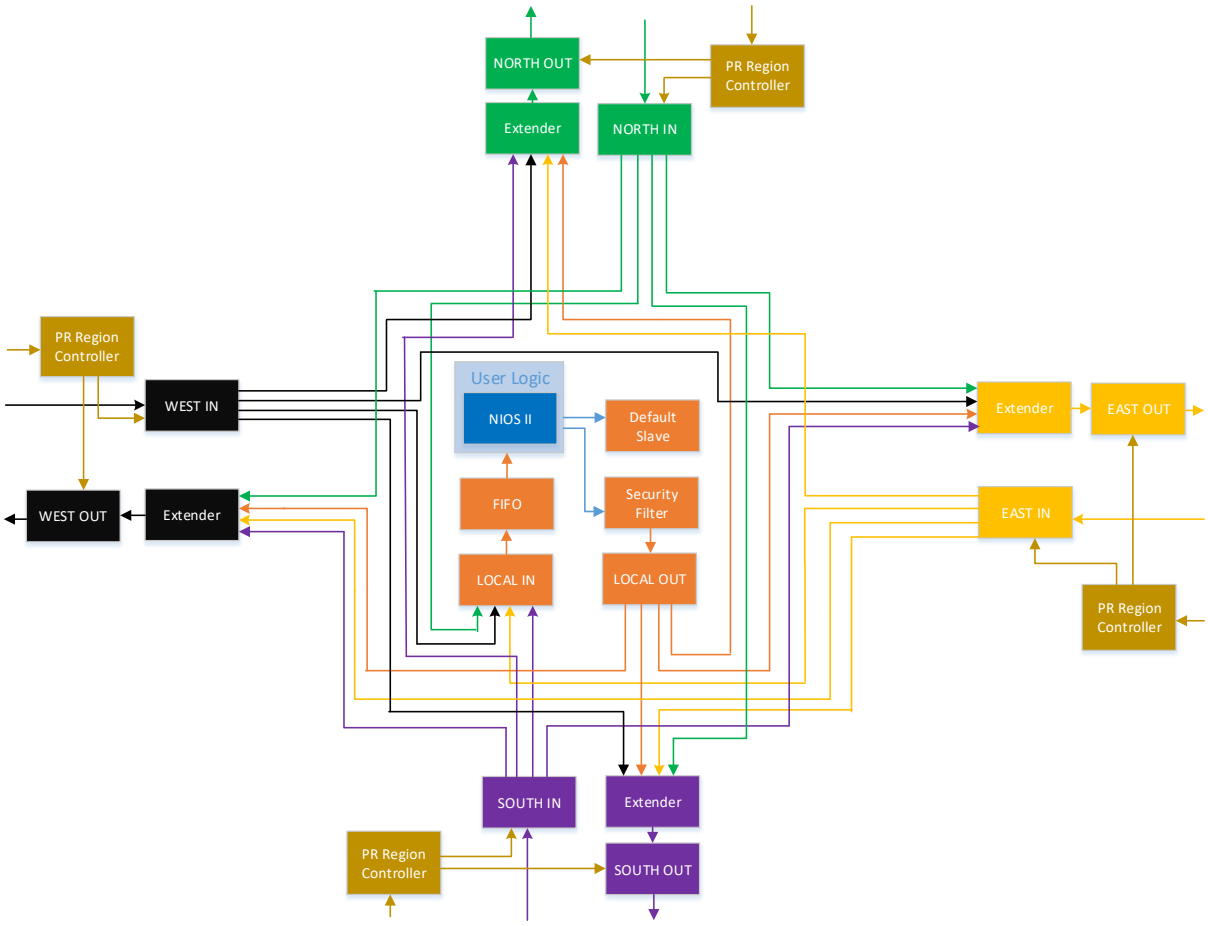
Target Device: Stratix 10 H-tile ES1 1SG280HU2F50E2VGS1

Development kit: Stratix 10 GX FPGA Development Kit

Source Code (access required):

https://github.com/IntelLabs/Sector-Oriented-Partial-Reconfiguration-Demonstrator

## Sector NoC Infrastructure




*Sector NoC Infrastructure*

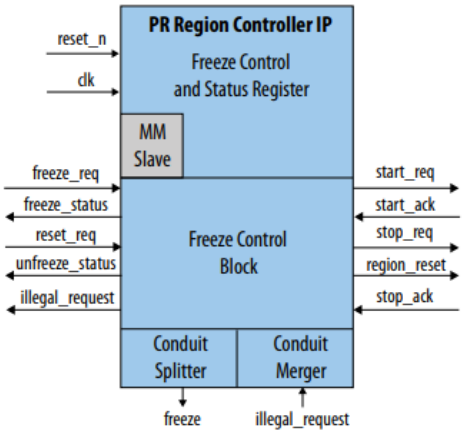
Each sector is implemented as illustrated in figure above. There will be incoming and outgoing ports for data transfer for each direction. Sectors that are placed at the boundary will have less ports. For example, sectors at the top row will not have north ports as there is no adjacent sector at the top side.

Incoming data from any port can be forwarded to another port (to adjacent sector); or directed to local_in. This data will be stored in FIFO and can be read by user logic (AVMM data master). User logic can also send data to any outgoing data ports via local_out.

The design is implemented based on hierarchical partial reconfiguration, where the entire sector is parent partition; while the user logic is child partition. User logic can be reconfigurable with different persona but standardized interface signals. Before partial reconfiguration take place, master (sector 0) will send freeze command to PR Region Controllers to freeze the impacted IN and OUT ports. The master will then unfreeze after PR is completed.

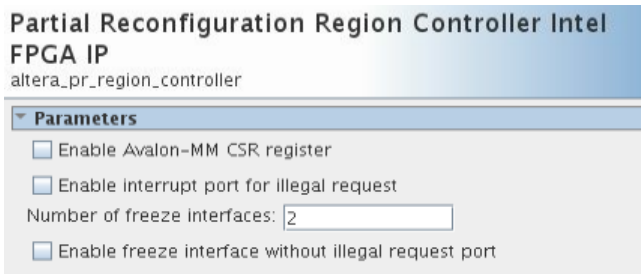
PR Region Controller

Each IN/OUT pair is connected to a PR Region Controller via freeze conduit signals. The controllers are controlled by out-of-bound conduit signals (conduit_control) from Sector 0. The reason conduit interface is chosen instead of AVMM CSR register is because the unfreeze command will be part of NoC data transfer which may end up in lock out situation (command cannot get pass frozen sector).




*PR Region Controller IP Core*

Each controller will have 2 freeze interfaces: one for IN port and one for OUT port. The handshake signals will be connected to a module where they will be loopback. The region_reset output to PR region is not used and replaced by reset from Sector 0 master.

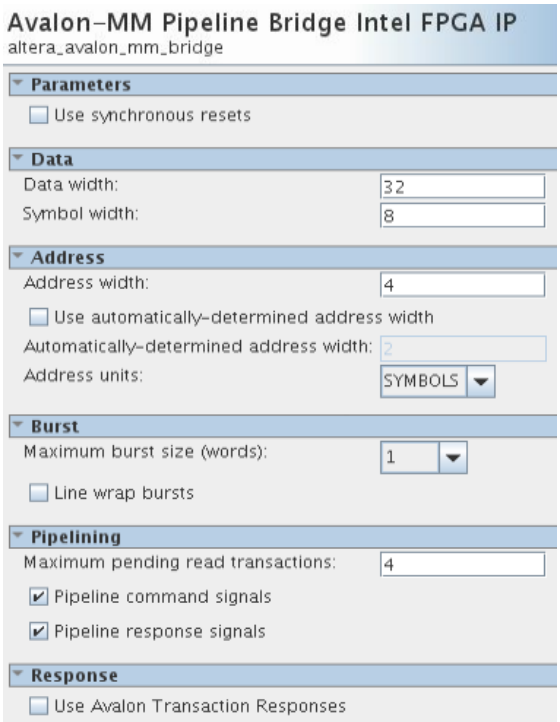



*PR Region Controller IP Configuration*

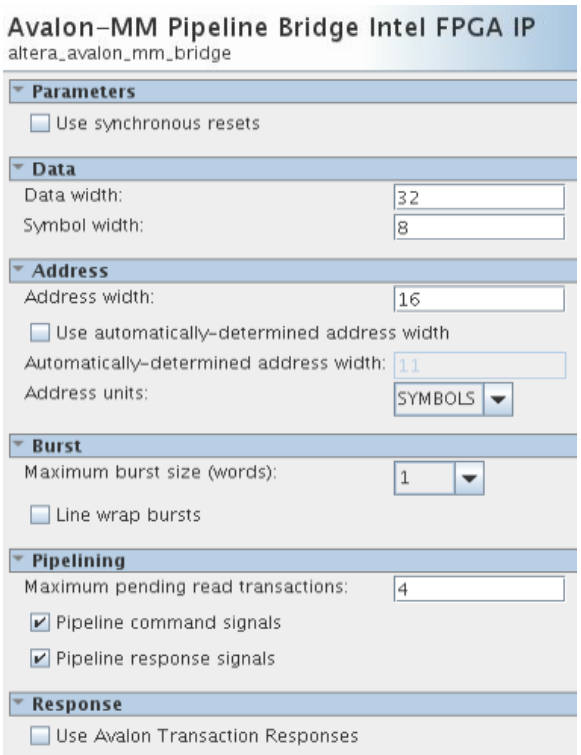
Reference:
https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug-qpp-pr.pdf (Section 2.5)

Local_in and _out ports

Local_in module is used to direct data from other sector (via north/east/south/west_in) to local FIFO. On the other hand, local_out is used to send local data to other sector. Both modules are implemented using pipeline bridge IP.



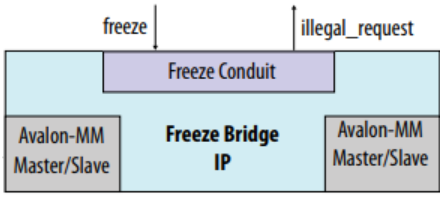

*Local_in IP configuration*




*Local_out IP configuration*

North/ East/ South/ West_in and _out ports

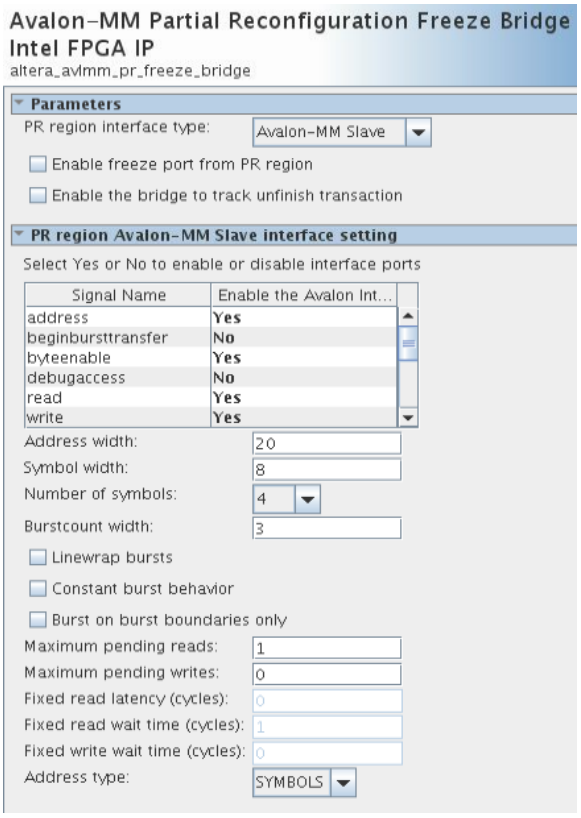
In and out ports are used to transfer data from sector to sector. All in and out ports are implemented using AVMM PR Freeze Bridge. This IP is similar to pipeline bridge but with additional freeze interface. The freeze port will be connected to PR region controller block.



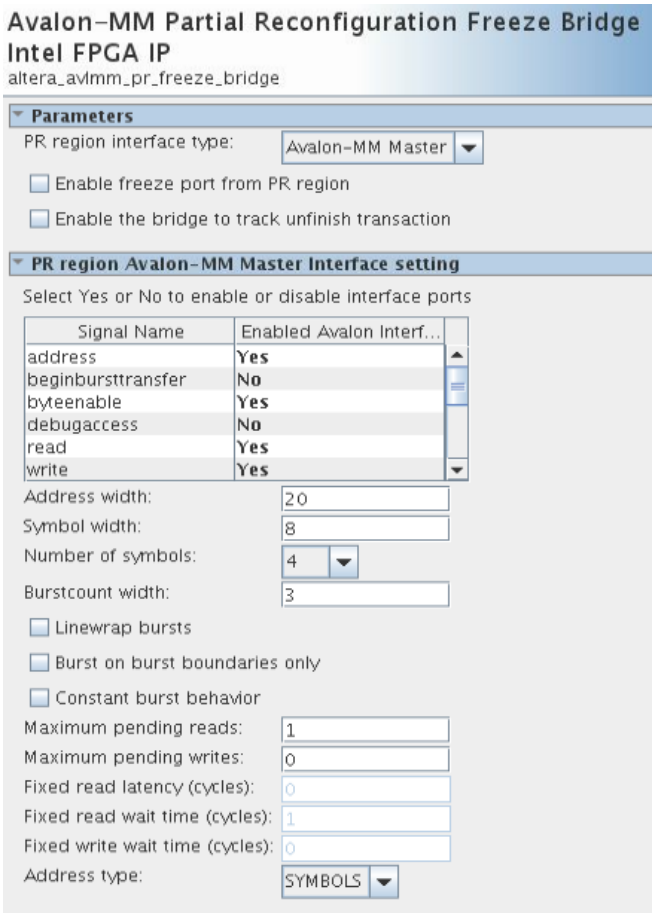

*Avalon-MM Partial Reconfiguration Freeze Bridge*

The out ports are connected to in ports of adjacent sector, likewise the in ports are connected to out ports of adjacent sector. For example, south_out of sector1 connects to north_in of sector4 and south_in of sector1 connects north_out of sector4.

Freeze bridge IP of in ports will have AVMM Slave interface while out ports will have AVMM Master interface.




*IN port Freeze Bridge IP Configuration*




*OUT port Freeze Bridge IP Configuration*

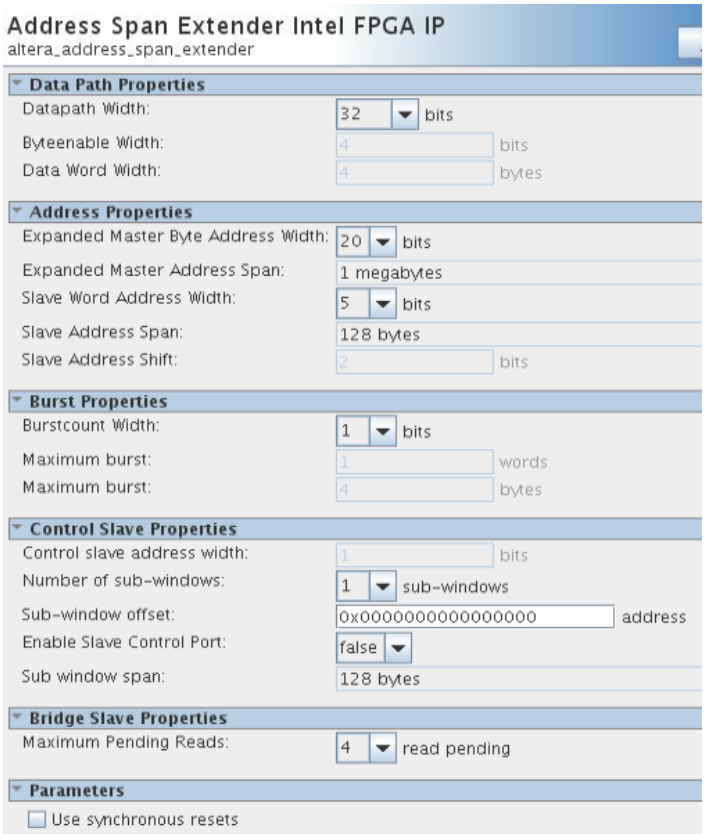
Reference:

[https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug-qpp-pr.pdf](https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug-qpp-pr.pdf) (Section 2.6)

Extenders

Extenders are needed to expand and fix the address range between AVMM masters and slaves (in and out ports). Each outgoing port in a sector will be connected to an extender.

As the intention is that this module can work without manual control, its slave control port is disabled. For this reason, data path from sector to sector is pre-determined as the extender could not expand the range to its fullest without control. There will be no alternative route to reach another sector. For example, the path from sector 0 to sector 4 is sector 0 (go south) -> sector 3 (go east) -> sector 4. Alternative path such as sector0 (go east) -> sector1 (go south) -> sector 4 is not supported.




*Example of Address Span Extender IP Configuration*

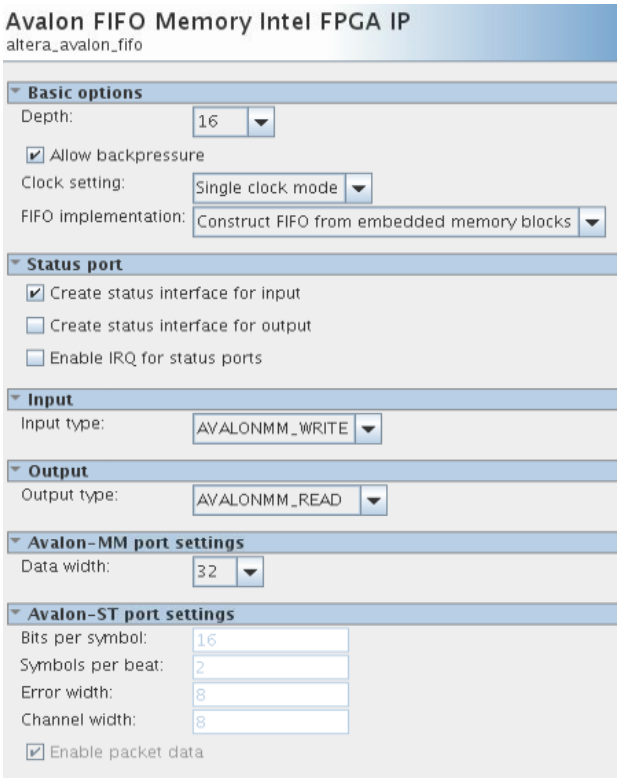
The expanded master address width is fixed for all extenders; the slave address width varies based on how far the data need to be sent. The longer the path the bigger the address width. In 3x3 design, the longest data path is to cross 4 sectors e.g. sector8 to sector 0.

| Data path (sectors) | Slave word address width (bits) |
|---|---|
| 1 | 5 |
| 2 | 6 |
| 3 | 7 |
| 4 | 8 |

Reference: Quartus Handbook Volume 1-> Platform Designer -> Bridges -> Address Span Extender

FIFO

FIFO is used to store data that was sent from another sector. The data is only writable by another sector, and only readable by local master from user logic. To avoid underflow or overflow issue, the local master can read FIFO data count via in_csr AVMM interface before accessing FIFO data.

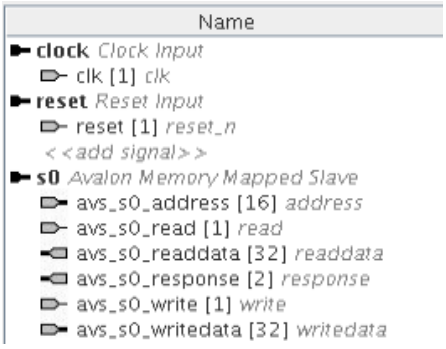



*FIFO Configuration*

Default_slave

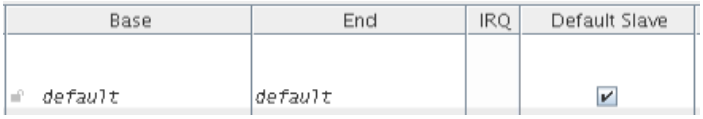
Any invalid command that was issued from local master will be routed to this module. Examples of invalid command: AVMM transactions not to a specified address, master writes to local FIFO, master reads from security filter. The commands that are routed to this module will be terminated.

A new component is created in Platform Designer based on default_slave.sv.




*Default_slave block signal interfaces*

In platform designer, this module must be specified as Default Slave for local master from user logic block.




*Default_slave in System Contents tab*

Security Filter

There are two purposes for this module: to map sector address to AVMM address and to filter data from going to certain sector. All valid transactions from local master will go through this module before they go to the next port.

Sector addresses (used by user_logic to write to other sector) are the same for all sectors, e.g. the sector address to sector0 is 0x0 for all sector. However, the AVMM address could vary from sector to sector. The AVMM address can be calculated by adding all the offsets of modules along the data path. Refer to section NoC Data Path for complete address map.

For example, from sector1 to sector 0:

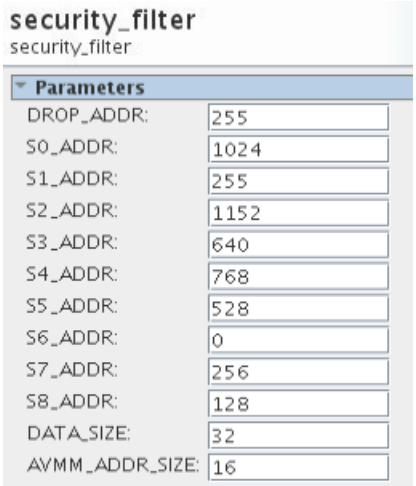
Sector 1 (from user logic Avalon Master)

| | |
|---|---|
| Security_filter | 0x0 |
| Local_out | 0x0 |
| West_out_extender | 0x400 |
| West_out_freeze_bridge | 0x0 |

Sector 0 (from East_in)

| | |
|---|---|
| Local_in | 0x0 |
| FIFO | 0x0 |

AVMM Address from Sector1 to Sector0 = 0x400 or 1024d




*Address map of Sector1*

All AVMM address must be populated (in decimal) in Platform Designer as shown above. The address map will be different from sector to sector. DROP_ADDR (255d) will be assigned for loopback address as this is unwanted, e.g. sector1 to sector1. As observe from the address map, a sector can only send data to another 8 sectors for 3x3 design. Any sector address beyond that is not accessible.

The CSR interface for filter enabling is controlled by Sector0 via NoC (not out-of-bound). The master can choose to enable/disable filter to each sector individually. By default, all filters are disabled.

| Bit | Field | Access | Default value | Descriptions |
|---|---|---|---|---|
| 0 | Sector 0 access | R/W | 0 | Write 1 to this bit to block access to sector0.<br>Write 0 to this bit to allow access to sector0. |
| 1 | Sector 1 access | R/W | 0 | Write 1 to this bit to block access to sector1.<br>Write 0 to this bit to allow access to sector1. |
| 2 | Sector 2 access | R/W | 0 | Write 1 to this bit to block access to sector2.<br>Write 0 to this bit to allow access to sector2. |
| 3 | Sector 3 access | R/W | 0 | Write 1 to this bit to block access to sector3.<br>Write 0 to this bit to allow access to sector3. |
| 4 | Sector 4 access | R/W | 0 | Write 1 to this bit to block access to sector4.<br>Write 0 to this bit to allow access to sector4. |
| 5 | Sector 5 access | R/W | 0 | Write 1 to this bit to block access to sector5.<br>Write 0 to this bit to allow access to sector5. |
| 6 | Sector 6 access | R/W | 0 | Write 1 to this bit to block access to sector6.<br>Write 0 to this bit to allow access to sector6. |
| 7 | Sector 7 access | R/W | 0 | Write 1 to this bit to block access to sector7.<br>Write 0 to this bit to allow access to sector7. |
| 8 | Sector 8 access | R/W | 0 | Write 1 to this bit to block access to sector8.<br>Write 0 to this bit to allow access to sector8. |

*Security filter avl_csr register map*

This module is created based on security_filter.sv in Platform Designer.

*Signals & interfaces of security_filter module*

PR handshake

This module is used to loopback handshake signals from each PR region controllers. It can be connected to 4 controllers. Ideally, the handshake signals should be implemented in PR region involved. But in this design, the master needs to be aware of the sequence of event and will trigger PR event accordingly. Therefore the handshake signals will be just simple loopback to fulfil the requirement of PR region controller.

This module is created based on pr_handshake.sv in Platform Designer.

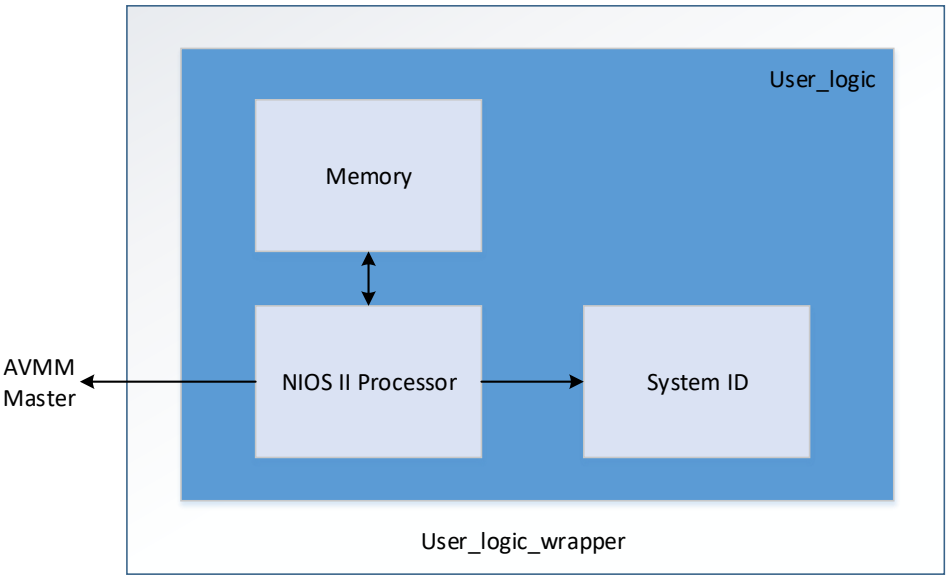
*Signals & interfaces of pr_handshake module*

**User logic**

User logic is where user/tenant design is implemented. The module is reconfigurable with different persona by partial reconfiguration. The module interface signals must follow the same format as defined in each wrapper file:

*Signals & Interface of user logic module*

In current design, user logic module in each sector is implemented with s1_user_logic_wrapper.sv, s2_user_logic_wrapper.sv, and so on so forth. Similar user logic is implemented in each sector as below, with different system ID. User logic (userX.qsys) is implemented as a Platform Designer system.




*User logic architecture block diagram*

The master (Nios II Processor) will periodically read its FIFO count via AVMM Master bus. If it detects that the FIFO is not empty, it will send the data + system ID to sector0. It is a simple user logic design use to validate the functionality of the NoC.

To create a different persona for user logic, the interface of the module/ wrapper must be the same as above with a pair of AVMM master bus that connects to NoC. A new project revision can be created to rebind the new module with current user logic. Please refer to section partial reconfiguration for more information on this.

**Sector 0**

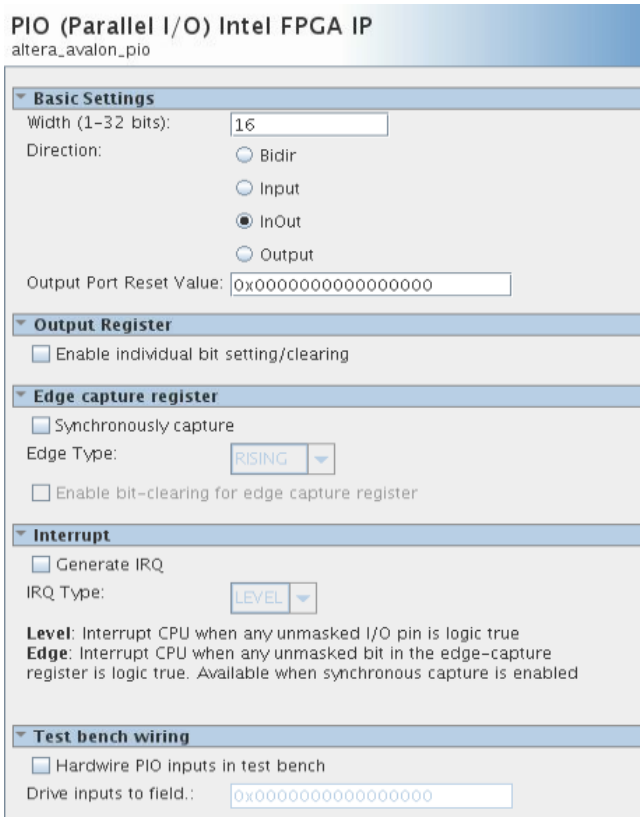
Sector0 is the master/ interface for debugging and validation. It offers a way to write, reset, freeze another sector on the fly. The infrastructure of sector 0 is similar to other sectors with additional modules to support the extra control features.

User0

In order to enable JTAG interface for validation, JTAG debug interface of Nios II Processor in user logic is enabled. In that way, data can be sent to any sector from sector 0 using System Console tool. As the AVMM master (of sector 0) is connected to more slave modules than just north/ east/ south/ west out, it is not connected to security filter to allow full access of address range.

PR PIO and PR Control

PR PIO modules are connected to sector0 AVMM Master to control PR requests. There is one PR PIO module for each sector PR request control. They are implemented using PIO IP with 16-bit InOut signals (16-bit input & 16-bit output signals).




*PR PIO Configuration*

| Bit | Field | Access | Default Values | Descriptions |
|---|---|---|---|---|
| 0 | Port0 freeze request | W | 0 | Write 1 to this bit to freeze port0 interface.<br>Write 0 to this bit to clear freeze request. |
| 1 | Port1 freeze request | W | 0 | Write 1 to this bit to freeze port1 interface.<br>Write 0 to this bit to clear freeze request. |
| 2 | Port2 freeze request | W | 0 | Write 1 to this bit to freeze port2 interface.<br>Write 0 to this bit to clear freeze request. |
| 3 | Port3 freeze request | W | 0 | Write 1 to this bit to freeze port3 interface.<br>Write 0 to this bit to clear freeze request. |
| 4 | Port0 reset **(reserved)** | W | 0 | Write 1 to start resetting the PR persona.<br>Write 0 to stop resetting the PR persona. |
| 5 | Port1 reset **(reserved)** | W | 0 | Write 1 to start resetting the PR persona.<br>Write 0 to stop resetting the PR persona. |
| 6 | Port2 reset **(reserved)** | W | 0 | Write 1 to start resetting the PR persona.<br>Write 0 to stop resetting the PR persona. |
| 7 | Port3 reset **(reserved)** | W | 0 | Write 1 to start resetting the PR persona.<br>Write 0 to stop resetting the PR persona. |
| 8 | Port0 unfreeze request | W | 0 | Write 1 to this bit to unfreeze port0 interface.<br>Write 0 to this bit to clear unfreeze request. |
| 9 | Port1 unfreeze request | W | 0 | Write 1 to this bit to unfreeze port1 interface.<br>Write 0 to this bit to clear unfreeze request. |
| 10 | Port2 unfreeze request | W | 0 | Write 1 to this bit to unfreeze port2 interface.<br>Write 0 to this bit to clear unfreeze request. |
| 11 | Port3 unfreeze request | W | 0 | Write 1 to this bit to unfreeze port3 interface.<br>Write 0 to this bit to clear unfreeze request. |
| 12 | N/A | | 0 | |
| 13 | N/A | | 0 | |
| 14 | N/A | | 0 | |
| 15 | N/A | | 0 | |

*PR PIO Write Command Register Map*

| Bit | Field | Access | Default Values | Descriptions |
|---|---|---|---|---|
| 0 | Port0 unfreeze status | R | 0 | 0 = Port0 unfreeze request bit is low.<br>1 = Port0 interface released from freeze. |
| 1 | Port1 unfreeze status | R | 0 | 0 = Port1unfreeze request bit is low.<br>1 = Port1 interface released from freeze. |
| 2 | Port2 unfreeze status | R | 0 | 0 = Port2 unfreeze request bit is low.<br>1 = Port2 interface released from freeze. |
| 3 | Port3 unfreeze status | R | 0 | 0 = Port3 unfreeze request bit is low.<br>1 = Port3 interface released from freeze. |
| 4 | Port0 freeze status | R | 0 | 0 = Port0 freeze request bit is low.<br>1 = Port0 ready to enter freeze state. |
| 5 | Port1 freeze status | R | 0 | 0 = Port1 freeze request bit is low.<br>1 = Port1 ready to enter freeze state. |
| 6 | Port2 freeze status | R | 0 | 0 = Port2 freeze request bit is low.<br>1 = Port2 ready to enter freeze state. |

| | | | | |
|---|---|---|---|---|
| 7 | Port3 freeze status | R | 0 | 0 = Port3 freeze request bit is low.<br>1 = Port3 ready to enter freeze state. |
| 8 | Port0 illegal request [0] | R | 0 | High indicates a read or write issue when Port0_in freeze bridge is in freeze state. |
| 9 | Port0 illegal request [1] | R | 0 | High indicates a read or write issue when Port0_out freeze bridge is in freeze state. |
| 10 | Port1 illegal request [0] | R | 0 | High indicates a read or write issue when Port1_in freeze bridge is in freeze state. |
| 11 | Port1 illegal request [1] | R | 0 | High indicates a read or write issue when Port1_out freeze bridge is in freeze state. |
| 12 | Port2 illegal request [0] | R | 0 | High indicates a read or write issue when Port2_in freeze bridge is in freeze state. |
| 13 | Port2 illegal request [1] | R | 0 | High indicates a read or write issue when Port2_out freeze bridge is in freeze state. |
| 14 | Port3 illegal request [0] | R | 0 | High indicates a read or write issue when Port3_in freeze bridge is in freeze state. |
| 15 | Port3 illegal request [1] | R | 0 | High indicates a read or write issue when Port3_out freeze bridge is in freeze state. |

*PR PIO Read Command Register Map*

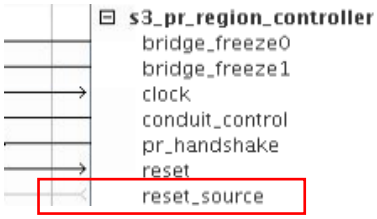
The conduit signals of the PIO modules are connected to PR Control module (RTL-based) to translate the conduit signals to correct interface format, so that they can be connected to PR Region Controllers.

*Signals & Interface of PR Control module*

All these signals will be connected out-of-bound (not part of NoC signals) directly to PR region controllers from sector0.

PR Reset Source and Conduit

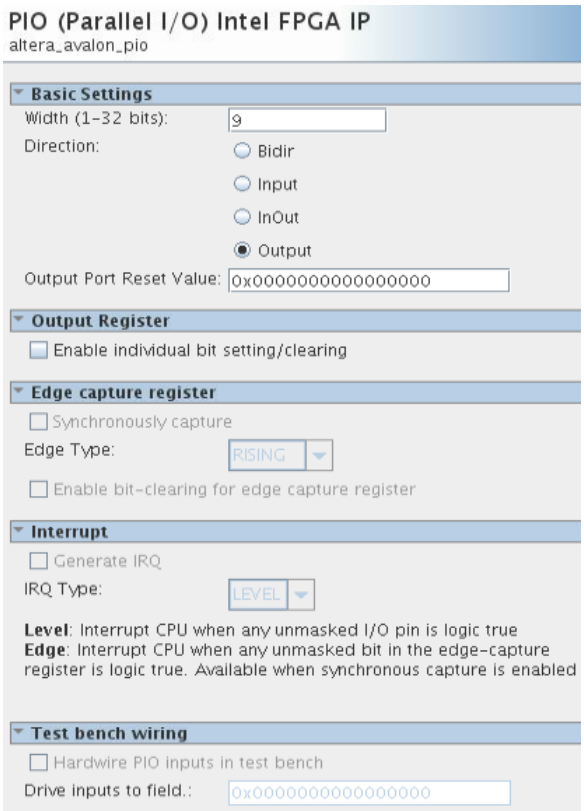
During hardware validation, it is found that reset output from PR region controllers should not be connected to PR region. The reason being when the PR region is under reset, it will also assert reset to its neighbouring sector, which will cause reset loop and the entire system to go into reset state.




*Reset_source output disconnected from PR Region Controller*

As the reset signal is part of PR region controller interface requirement, the signal is reserved at the interface of PR Control modules in sector0 (see previous page).

Out-of-bound reset is added in sector0 so that the master can assert reset directly to PR region. The module is also implemented using PIO IP with 9-bit output signals.



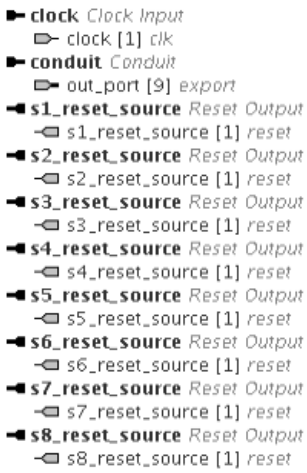

*PR Reset Source IP Configuration*

| Bit | Field | Access | Default Values | Descriptions |
|---|---|---|---|---|
| 0 | N/A | W | 0 | |
| 1 | Sector 1 reset | W | 0 | Write 1 to start resetting Sector1.<br>Write 0 to stop resetting Sector1. |

| | | | | |
|---|---|---|---|---|
| 2 | Sector 2 reset | W | 0 | Write 1 to start resetting Sector2.<br>Write 0 to stop resetting Sector2. |
| 3 | Sector 3 reset | W | 0 | Write 1 to start resetting Sector3.<br>Write 0 to stop resetting Sector3. |
| 4 | Sector 4 reset | W | 0 | Write 1 to start resetting Sector4.<br>Write 0 to stop resetting Sector4. |
| 5 | Sector 5 reset | W | 0 | Write 1 to start resetting Sector5.<br>Write 0 to stop resetting Sector5. |
| 6 | Sector 6 reset | W | 0 | Write 1 to start resetting Sector6.<br>Write 0 to stop resetting Sector6. |
| 7 | Sector 7 reset | W | 0 | Write 1 to start resetting Sector7.<br>Write 0 to stop resetting Sector7. |
| 8 | Sector 8 reset | W | 0 | Write 1 to start resetting Sector8.<br>Write 0 to stop resetting Sector8. |

*PR Reset Source Register Map*

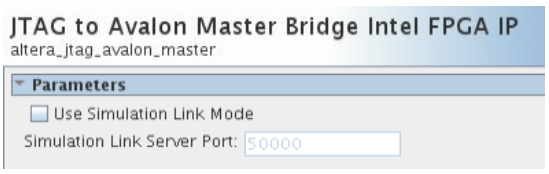
PR reset source conduit module (RTL-based) will translate conduit signals from PR reset source module to reset output format.




*Signals & Interface of PR reset source conduit module*

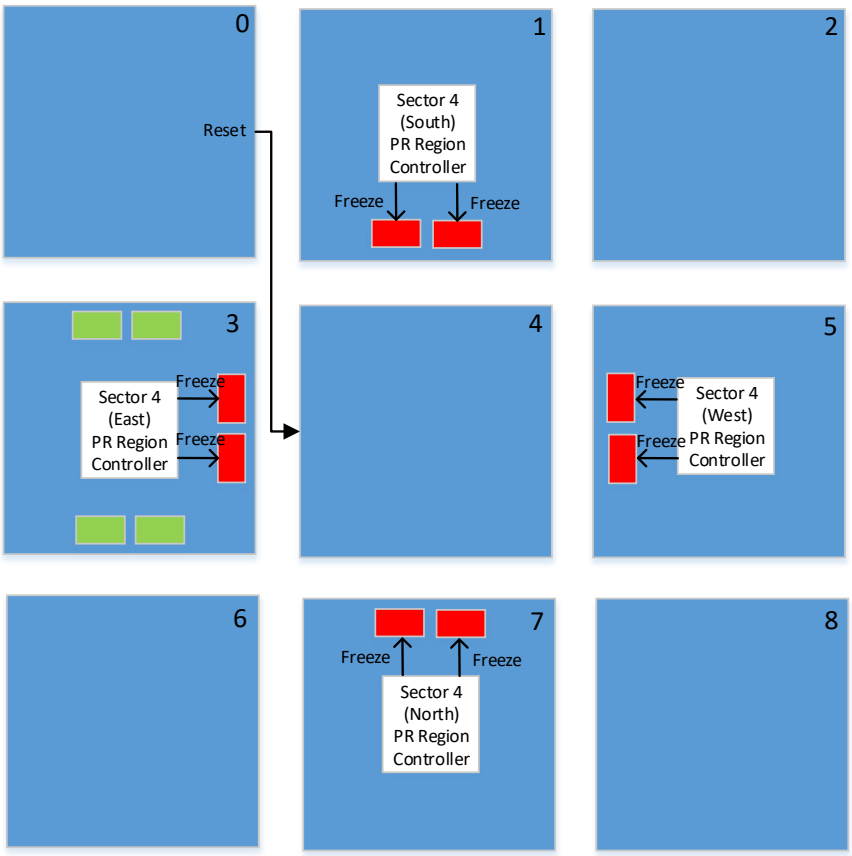
**System**

This system uses a differential 100MHz system clock (pin J20 & J19) and two resets, one pushbutton reset (pinA20 – S4 on LED daughter board) and one JTAG-triggered reset. The latter is a reset output from Master module, implemented with JTAG to Avalon Master Bridge IP. By having this, it is possible to trigger remote system reset. The Avalon master is not connected to any module in current design.




*Master IP Configuration*

Conventionally, one PR region controller will trigger freeze request and reset to one PR region. In this design, each PR region (a sector) is part of NoC infrastructure. During PR the interface signals will be floating and might cause accidental write or read. To prevent that from happening, each neighbouring port (both in and out ports) will be frozen and each pair will be controlled by one controller. Reset for all PR regions will be coming from sector0.




*PR Region Controllers for Sector4*

As illustrated in the block diagram, the red ports are the ones needed to be frozen before reconfiguration of sector4. In every sector there will be one PR region controller for each direction. The controllers are controlled via conduit signals from sector0.

NoC Data Path

The data path for each sector is pre-determined in current design, as the addressing of the bridges is not flexible. This will also impact future design improvement to make this design scalable to include more sectors. North/south bound path will take precedence than east/west bound path. Do note that incoming and outgoing path may not be the same, e.g. paths of sector0 to sector8 and sector8 to sector0 are not the same.

| From Sector0 to | Data Path | AVMM Address |
|---|---|---|
| Sector1 | 0 -> 1 | 0x480 |
| Sector2 | 0 -> 1 -> 2 | 0x400 |
| Sector3 | 0 -> 3 | 0x300 |
| Sector4 | 0 -> 3 -> 4 | 0x2a0 |
| Sector5 | 0 -> 3 -> 4 -> 5 | 0x210 |
| Sector6 | 0 -> 3 -> 6 | 0x110 |
| Sector7 | 0 -> 3 -> 6 -> 7 | 0x90 |
| Sector8 | 0 -> 3 -> 6 -> 7 -> 8 | 0x0 |

| From Sector1 to | Data Path | AVMM Address |
|---|---|---|
| Sector0 | 1 -> 0 | 0x400 |
| Sector2 | 1 -> 2 | 0x480 |
| Sector3 | 1 -> 4 -> 3 | 0x280 |
| Sector4 | 1 -> 4 | 0x300 |
| Sector5 | 1 -> 4 -> 5 | 0x210 |
| Sector6 | 1 -> 4 -> 7 -> 6 | 0x0 |
| Sector7 | 1 -> 4 -> 7 | 0x100 |
| Sector8 | 1 -> 4 -> 7 -> 8 | 0x80 |

| From Sector2 to | Data Path | AVMM Address |
|---|---|---|
| Sector0 | 2 -> 1 -> 0 | 0x400 |
| Sector1 | 2 -> 1 | 0x480 |
| Sector3 | 2 -> 5 -> 4 -> 3 | 0x200 |
| Sector4 | 2 -> 5 -> 4 | 0x280 |
| Sector5 | 2 -> 5 | 0x300 |
| Sector6 | 2 -> 5 -> 8 -> 7 -> 6 | 0x0 |
| Sector7 | 2 -> 5 -> 8 -> 7 | 0x80 |
| Sector8 | 2 -> 5 -> 8 | 0x100 |

| From Sector3 to | Data Path | AVMM Address |
|---|---|---|
| Sector0 | 3 -> 0 | 0x300 |
| Sector1 | 3 -> 0 -> 1 | 0x280 |
| Sector2 | 3 -> 0 -> 1-> 2 | 0x200 |
| Sector4 | 3 -> 4 | 0x4a0 |
| Sector5 | 3 -> 4 -> 5 | 0x410 |

| | | |
|---|---|---|
| Sector6 | 3 -> 6 | 0x110 |
| Sector7 | 3 -> 6 -> 7 | 0x90 |
| Sector8 | 3 -> 6 -> 7 -> 8 | 0x0 |

| From Sector4 to | Data Path | AVMM Address |
|---|---|---|
| Sector0 | 4 -> 1 -> 0 | 0x0 |
| Sector1 | 4 -> 1 | 0x100 |
| Sector2 | 4 -> 1-> 2 | 0x80 |
| Sector3 | 4 -> 3 | 0x400 |
| Sector5 | 4 -> 5 | 0x490 |
| Sector6 | 4 -> 7 -> 6 | 0x200 |
| Sector7 | 4 -> 7 | 0x300 |
| Sector8 | 4 -> 7 -> 8 | 0x280 |

| From Sector5 to | Data Path | AVMM Address |
|---|---|---|
| Sector0 | 5 -> 2 -> 1 -> 0 | 0x0 |
| Sector1 | 5 -> 2 -> 1 | 0x80 |
| Sector2 | 5 -> 2 | 0x100 |
| Sector3 | 5 -> 4 -> 3 | 0x400 |
| Sector4 | 5 -> 4 | 0x480 |
| Sector6 | 5 -> 8 -> 7 -> 6 | 0x200 |
| Sector7 | 5 -> 8 -> 7 | 0x280 |
| Sector8 | 5 -> 8 | 0x300 |

| From Sector6 to | Data Path | AVMM Address |
|---|---|---|
| Sector0 | 6 -> 3 -> 0 | 0x100 |
| Sector1 | 6 -> 3 -> 0 -> 1 | 0x80 |
| Sector2 | 6 -> 3 -> 0 -> 1-> 2 | 0x0 |
| Sector3 | 6 -> 3 | 0x300 |
| Sector4 | 6 -> 3 -> 4 | 0x2a0 |
| Sector5 | 6 -> 3 -> 4 -> 5 | 0x210 |
| Sector7 | 6 -> 7 | 0x490 |
| Sector8 | 6 -> 7 -> 8 | 0x400 |

| From Sector7 to | Data Path | AVMM Address |
|---|---|---|
| Sector0 | 7 -> 4 -> 1 -> 0 | 0x0 |
| Sector1 | 7 -> 4 -> 1 | 0x100 |
| Sector2 | 7 -> 4 -> 1-> 2 | 0x80 |
| Sector3 | 7 -> 4-> 3 | 0x200 |
| Sector4 | 7 -> 4 | 0x300 |
| Sector5 | 7 -> 4 -> 5 | 0x290 |
| Sector6 | 7 -> 6 | 0x480 |
| Sector8 | 7 -> 8 | 0x400 |

| From Sector8 to | Data Path | AVMM Address |
|---|---|---|
| Sector0 | 8 -> 5 -> 2 -> 1 -> 0 | 0x0 |
| Sector1 | 8 -> 5 -> 2 -> 1 | 0x80 |
| Sector2 | 8 -> 5 -> 2 | 0x100 |
| Sector3 | 8 -> 5 -> 4-> 3 | 0x200 |
| Sector4 | 8 -> 5 -> 4 | 0x280 |
| Sector5 | 8 -> 5 | 0x300 |
| Sector6 | 8 -> 7 -> 6 | 0x400 |
| Sector7 | 8 -> 7 | 0x480 |

**Design Implementation**

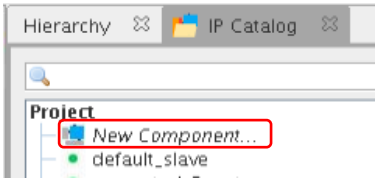
Platform Designer

The entire design is implemented in Platform Designer. Example of hierarchy of sector8:

```
System [system.qsys]
        |- Sector8 [sector8.qsys]
                |- default_slave
                |- fifo
                |- pr_handshake
                |- s5_pr_region_controller
                |- s7_pr_region_controller
                |- s8_local_in
                |- s8_local_out
                |- s8_north_in
                |- s8_north_out
                |- s8_north_out_extender
                |- s8_west_in
                |- s8_west_out
                |- s8_west_out_extender
                |- security filter
                |- user_logic_wrapper
                        |- user8 [user8.qsys]
```

The top-level entity is system.qsys, where sector0 to sector8 are instantiated. In each sector.qsys, NoC-related blocks are instantiated as well as user logic wrapper. The lowest hierarchy would be user.qsys.

Most of the IPs used are off-the-shelf IPs from IP catalog. For the RTL-based IP, new component is created.




*Create new component in Platform Designer*

In Component Editor window, source file(s) can be added and interface signals must be defined. After this, the module will be added in IP catalog and can be instantiated just as other IPs.

Partial Reconfiguration

This design is implemented in such a way to support hierarchical partial reconfiguration (HPR). Each sector is a parent partition while user logic is child partition.

In Quartus, create reconfigurable design partition for each sector and user logic. As sector0 acts a system master, it will be part of static region and is not reconfigurable.

Assignments View | Compilation View

Design Partitions Window

| Partition Name | Hierarchy Path | Type | Preservation Level | Empty | Partition Database File | Entity Re-binding | Color |
|---|---|---|---|---|---|---|---|
| <<new>> | | | | | | | |
| root_partition | \| | | | | | | |
| s4_parent_partition | sector4 | Reconfigurable | Not Set | No | | | |
| s1_parent_partition | sector1 | Reconfigurable | Not Set | No | | | |
| s2_parent_partition | sector2 | Reconfigurable | Not Set | No | | | |
| s3_parent_partition | sector3 | Reconfigurable | Not Set | No | | | |
| s5_parent_partition | sector5 | Reconfigurable | Not Set | No | | | |
| s6_parent_partition | sector6 | Reconfigurable | Not Set | No | | | |
| s7_parent_partition | sector7 | Reconfigurable | Not Set | No | | | |
| s8_parent_partition | sector8 | Reconfigurable | Not Set | No | | | |
| s1_user_partition | sector1\|user_logic | Reconfigurable | Not Set | No | | | |
| s2_user_partition | sector2\|user_logic | Reconfigurable | Not Set | No | | | |
| s3_user_partition | sector3\|user_logic | Reconfigurable | Not Set | No | | | |
| s4_user_partition | sector4\|user_logic | Reconfigurable | Not Set | No | | | |
| s5_user_partition | sector5\|user_logic | Reconfigurable | Not Set | No | | | |
| s6_user_partition | sector6\|user_logic | Reconfigurable | Not Set | No | | | |
| s7_user_partition | sector7\|user_logic | Reconfigurable | Not Set | No | | | |
| s8_user_partition | sector8\|user_logic | Reconfigurable | Not Set | No | | | |

*HPR Design Partitions*

A parent partition will take up the entire sector (sector-aligned) while a child partition will be placed in the center of parent partition. Each sector will be placed side-by-side in 3 rows to form a 3x3 floorplan.

Logic Lock Regions Window

| Region Name | Members | Width | Height | Origin | Reserved | Core-Only | Size/State | Routing Region |
|---|---|---|---|---|---|---|---|---|
| Logic Lock Regions | | | | | | | | |
| s0 | sector0 | 34 | 36 | X53_Y109 | On | On | Fixed/Locked | Fixed with expansion 0 |
| s1 | sector1 | 34 | 36 | X87_Y109 | On | On | Fixed/Locked | Fixed with expansion 0 |
| s1_user | sector1\|user_logic | 28 | 30 | X90_Y112 | On | On | Fixed/Locked | Fixed with expansion 0 |
| s2 | sector2 | 34 | 36 | X121_Y109 | On | On | Fixed/Locked | Fixed with expansion 0 |
| s2_user | sector2\|user_logic | 28 | 30 | X124_Y112 | On | On | Fixed/Locked | Fixed with expansion 0 |
| s3 | sector3 | 34 | 36 | X53_Y73 | On | On | Fixed/Locked | Fixed with expansion 0 |
| s3_user | sector3\|user_logic | 28 | 30 | X56_Y76 | On | On | Fixed/Locked | Fixed with expansion 0 |
| s4 | sector4 | 34 | 36 | X87_Y73 | On | On | Fixed/Locked | Fixed with expansion 0 |
| s4_user | sector4\|user_logic | 28 | 30 | X90_Y76 | On | On | Fixed/Locked | Fixed with expansion 0 |
| s5 | sector5 | 34 | 36 | X121_Y73 | On | On | Fixed/Locked | Fixed with expansion 0 |
| s5_user | sector5\|user_logic | 28 | 30 | X124_Y76 | On | On | Fixed/Locked | Fixed with expansion 0 |
| s6 | sector6 | 34 | 36 | X53_Y37 | On | On | Fixed/Locked | Fixed with expansion 0 |
| s6_user | sector6\|user_logic | 28 | 30 | X56_Y40 | On | On | Fixed/Locked | Fixed with expansion 0 |
| s7 | sector7 | 34 | 36 | X87_Y37 | On | On | Fixed/Locked | Fixed with expansion 0 |
| s7_user | sector7\|user_logic | 28 | 30 | X90_Y40 | On | On | Fixed/Locked | Fixed with expansion 0 |
| s8 | sector8 | 34 | 36 | X121_Y37 | On | On | Fixed/Locked | Fixed with expansion 0 |
| s8_user | sector8\|user_logic | 28 | 30 | X124_Y40 | On | On | Fixed/Locked | Fixed with expansion 0 |

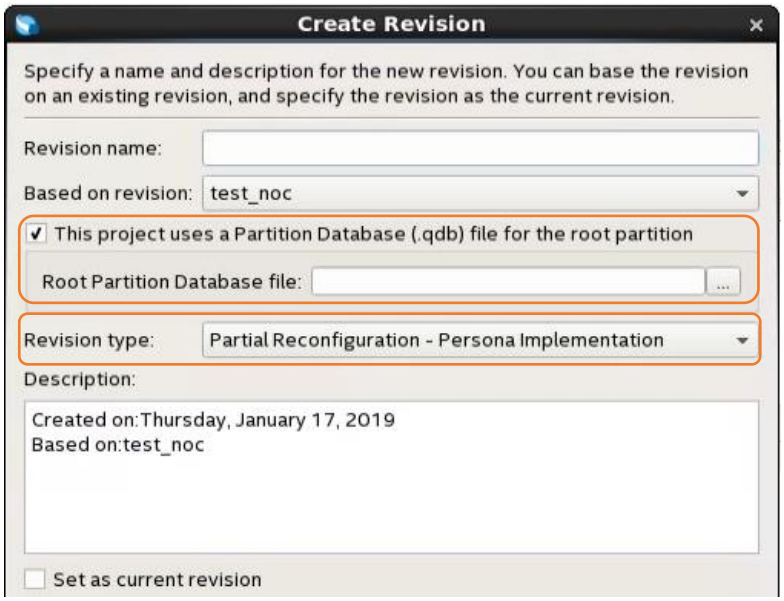
*Logic Lock Regions of Design Partitions*

In order to create multiple personas of user logic, multiple Quartus project revisions have to be created. The initial design will be the base revision.

Revisions:

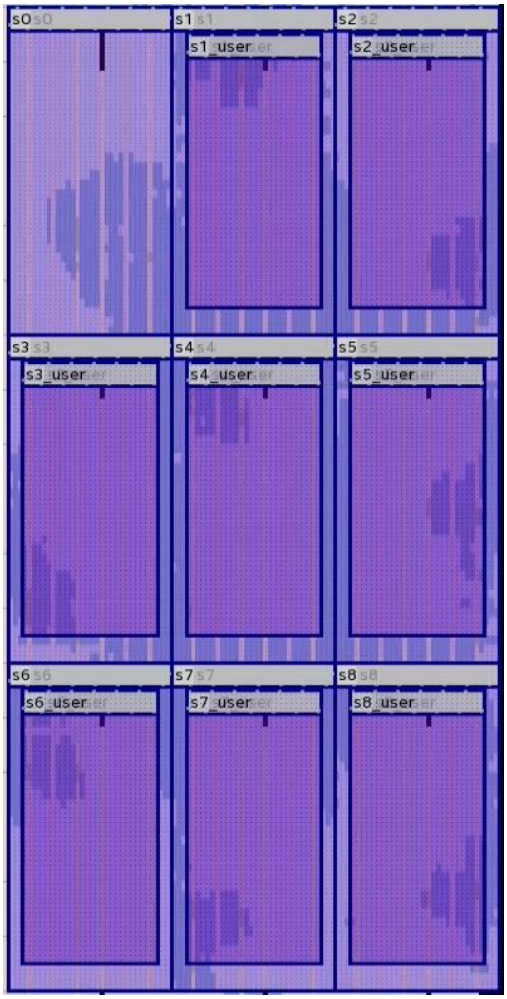
| Revision Name | Revision Type | Top-level Entity | Family | Device |
|---|---|---|---|---|
| test_noc | Partial Reconfiguration - Base | system | Stratix 10 | 1SG280HU2F50E2VGS1 |
| test_noc_v2 | Partial Reconfiguration - Persona Implementation | system | Stratix 10 | 1SG280HU2F50E2VGS1 |
| test_noc_bad | Partial Reconfiguration - Persona Implementation | system | Stratix 10 | 1SG280HU2F50E2VGS1 |
| test_noc_v1 | Partial Reconfiguration - Persona Implementation | system | Stratix 10 | 1SG280HU2F50E2VGS1 |

*Project revisions*



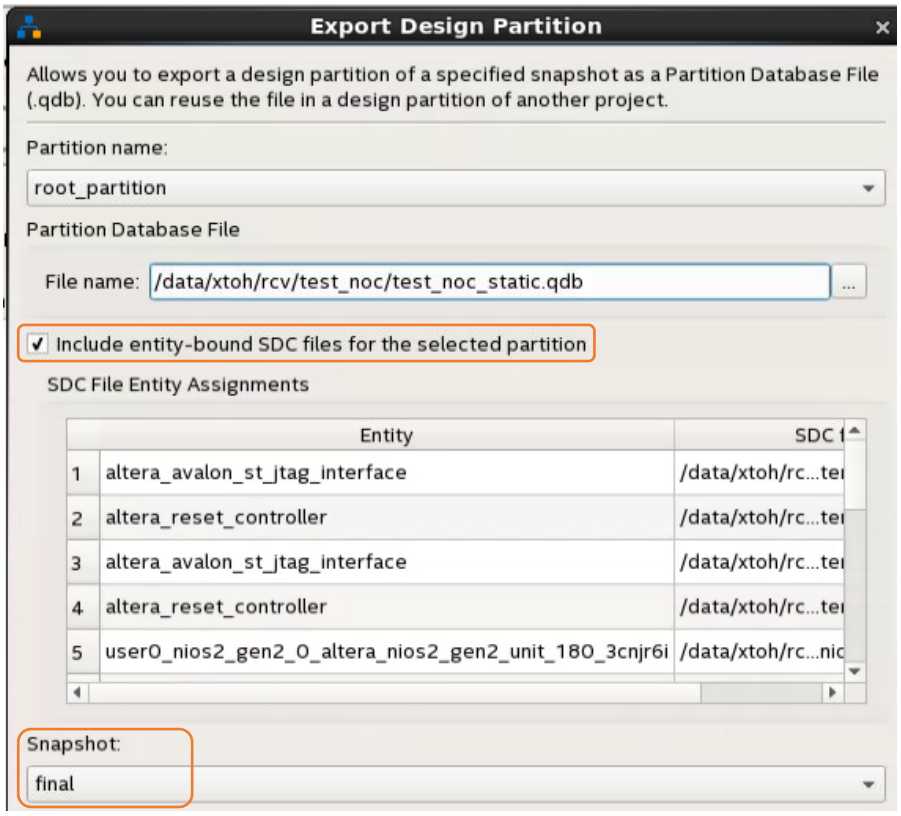

*Create New Revision*

Note: Root Partition Database File can be named at later stage.




*Design Floorplan*

After compiling the base revision, final snapshot of parent design partitions must be exported to be reused in another project revisions.




*Export Design Partition of Base Revision*

| Partition Name | Partition Database File |
|---|---|
| root_partition | test_noc_static.qdb |
| s1_parent_partition | s1_parent_partition_default_final.qdb |
| s2_parent_partition | s2_parent_partition_default_final.qdb |
| s3_parent_partition | s3_parent_partition_default_final.qdb |
| s4_parent_partition | s4_parent_partition_default_final.qdb |
| s5_parent_partition | s5_parent_partition_default_final.qdb |
| s6_parent_partition | s6_parent_partition_default_final.qdb |
| s7_parent_partition | s7_parent_partition_default_final.qdb |
| s8_parent_partition | s8_parent_partition_default_final.qdb |

<u>test_noc_v1, test_noc_v2 and test_noc_bad Project Revisions</u>

In both revisions, the parent partition will be the same as base revision by specifying the exported .qdb from base revision in Design Partition Window. On the other hand, user logic will be re-bind with different persona. In Entity Re-binding column, specify the entity name for the new persona. For test_noc_v1 revision, the existing user logic modules of base revision are shuffled around (shuffled sector ID). For test_noc_v2 revision, user logic partition will be re-bind with module with sector ID 0x10. For test_noc_bad revision, all user logic modules will be re-bind with malicious user logic (s9_user_logic). The Logic Lock

regions for all partitions remain the same. Compilation of these revisions will result in having different user logic persona for each sector.

Assignments View | Compilation View

| Partition Name | Hierarchy Path | Type | servation Le | Empty | Partition Database File | Entity Re-binding | Color |
|---|---|---|---|---|---|---|---|
| <<new>> | | | | | | | |
| root_partition | \| | | | | test_noc_static.qdb | | |
| s4_parent_partition | sector4 | Reconfigurable | Not Set | No | s4_parent_partition_default_final.qdb | | |
| s1_parent_partition | sector1 | Reconfigurable | Not Set | No | s1_parent_partition_default_final.qdb | | |
| s2_parent_partition | sector2 | Reconfigurable | Not Set | No | s2_parent_partition_default_final.qdb | | |
| s3_parent_partition | sector3 | Reconfigurable | Not Set | No | s3_parent_partition_default_final.qdb | | |
| s5_parent_partition | sector5 | Reconfigurable | Not Set | No | s5_parent_partition_default_final.qdb | | |
| s6_parent_partition | sector6 | Reconfigurable | Not Set | No | s6_parent_partition_default_final.qdb | | |
| s7_parent_partition | sector7 | Reconfigurable | Not Set | No | s7_parent_partition_default_final.qdb | | |
| s8_parent_partition | sector8 | Reconfigurable | Not Set | No | s8_parent_partition_default_final.qdb | | |
| s1_user_partition | sector1\|user_logic | Reconfigurable | Not Set | No | | s2_user_logic_wrapper | |
| s2_user_partition | sector2\|user_logic | Reconfigurable | Not Set | No | | s3_user_logic_wrapper | |
| s3_user_partition | sector3\|user_logic | Reconfigurable | Not Set | No | | s4_user_logic_wrapper | |
| s4_user_partition | sector4\|user_logic | Reconfigurable | Not Set | No | | s5_user_logic_wrapper | |
| s5_user_partition | sector5\|user_logic | Reconfigurable | Not Set | No | | s6_user_logic_wrapper | |
| s6_user_partition | sector6\|user_logic | Reconfigurable | Not Set | No | | s7_user_logic_wrapper | |
| s7_user_partition | sector7\|user_logic | Reconfigurable | Not Set | No | | s8_user_logic_wrapper | |
| s8_user_partition | sector8\|user_logic | Reconfigurable | Not Set | No | | s1_user_logic_wrapper | |

*Re-bind child partitions for test_noc_v1*

Assignments View | Compilation View

| Partition Name | Hierarchy Path | Type | servation Le | Empty | Partition Database File | Entity Re-binding | Color |
|---|---|---|---|---|---|---|---|
| <<new>> | | | | | | | |
| root_partition | \| | | | | test_noc_static.qdb | | |
| s4_parent_partition | sector4 | Reconfigurable | Not Set | No | s4_parent_partition_default_final.qdb | | |
| s1_parent_partition | sector1 | Reconfigurable | Not Set | No | s1_parent_partition_default_final.qdb | | |
| s2_parent_partition | sector2 | Reconfigurable | Not Set | No | s2_parent_partition_default_final.qdb | | |
| s3_parent_partition | sector3 | Reconfigurable | Not Set | No | s3_parent_partition_default_final.qdb | | |
| s5_parent_partition | sector5 | Reconfigurable | Not Set | No | s5_parent_partition_default_final.qdb | | |
| s6_parent_partition | sector6 | Reconfigurable | Not Set | No | s6_parent_partition_default_final.qdb | | |
| s7_parent_partition | sector7 | Reconfigurable | Not Set | No | s7_parent_partition_default_final.qdb | | |
| s8_parent_partition | sector8 | Reconfigurable | Not Set | No | s8_parent_partition_default_final.qdb | | |
| s1_user_partition | sector1\|user_logic | Reconfigurable | Not Set | No | | s9_user_logic_wrapper | |
| s2_user_partition | sector2\|user_logic | Reconfigurable | Not Set | No | | s9_user_logic_wrapper | |
| s3_user_partition | sector3\|user_logic | Reconfigurable | Not Set | No | | s9_user_logic_wrapper | |
| s4_user_partition | sector4\|user_logic | Reconfigurable | Not Set | No | | s9_user_logic_wrapper | |
| s5_user_partition | sector5\|user_logic | Reconfigurable | Not Set | No | | s9_user_logic_wrapper | |
| s6_user_partition | sector6\|user_logic | Reconfigurable | Not Set | No | | s9_user_logic_wrapper | |
| s7_user_partition | sector7\|user_logic | Reconfigurable | Not Set | No | | s9_user_logic_wrapper | |
| s8_user_partition | sector8\|user_logic | Reconfigurable | Not Set | No | | s9_user_logic_wrapper | |

*Re-bind child partitions for test_noc_bad*

Assignments View | Compilation View

| Partition Name | Hierarchy Path | Type | eservation Lev | Empty | Partition Database File | Entity Re-binding | Color |
|---|---|---|---|---|---|---|---|
| <<new>> | | | | | | | |
| root_partition | \| | | | | test_noc_static.qdb | | |
| s4_parent_partition | sector4 | Reconfigurable | Not Set | No | s4_parent_partition_default_final.qdb | | |
| s1_parent_partition | sector1 | Reconfigurable | Not Set | No | s1_parent_partition_default_final.qdb | | |
| s2_parent_partition | sector2 | Reconfigurable | Not Set | No | s2_parent_partition_default_final.qdb | | |
| s3_parent_partition | sector3 | Reconfigurable | Not Set | No | s3_parent_partition_default_final.qdb | | |
| s5_parent_partition | sector5 | Reconfigurable | Not Set | No | s5_parent_partition_default_final.qdb | | |
| s6_parent_partition | sector6 | Reconfigurable | Not Set | No | s6_parent_partition_default_final.qdb | | |
| s7_parent_partition | sector7 | Reconfigurable | Not Set | No | s7_parent_partition_default_final.qdb | | |
| s8_parent_partition | sector8 | Reconfigurable | Not Set | No | s8_parent_partition_default_final.qdb | | |
| s1_user_partition | sector1\|user_logic | Reconfigurable | Not Set | No | | s10_user_logic_wrapper | |
| s2_user_partition | sector2\|user_logic | Reconfigurable | Not Set | No | | s10_user_logic_wrapper | |
| s3_user_partition | sector3\|user_logic | Reconfigurable | Not Set | No | | s10_user_logic_wrapper | |
| s4_user_partition | sector4\|user_logic | Reconfigurable | Not Set | No | | s10_user_logic_wrapper | |
| s5_user_partition | sector5\|user_logic | Reconfigurable | Not Set | No | | s10_user_logic_wrapper | |
| s6_user_partition | sector6\|user_logic | Reconfigurable | Not Set | No | | s10_user_logic_wrapper | |
| s7_user_partition | sector7\|user_logic | Reconfigurable | Not Set | No | | s10_user_logic_wrapper | |
| s8_user_partition | sector8\|user_logic | Reconfigurable | Not Set | No | | s10_user_logic_wrapper | |

*Re-bind child partitions for test_noc_v2*

Adding a new persona to the design

After compiling the base revision, new persona of user logic can still be created. The interface signals of the persona module must be the same as user_logic_wrapper module. The new module can be implemented by re-binding the entity of interest in Design Partitions Window in new project revision. Please ensure that the resources needed for the user logic can be fit into a single sector.

After compilation, each revision will generate one .sof, eight parent partition .rbf for each sector and eight child partition .rbf for each sector.

HPR Reference Design: https://github.com/intel/fpga-partial-reconfig/blob/master/tutorials/s10_pcie_devkit_blinking_led_hpr/AN-826.pdf

**Design Limitations and Areas of Improvement**

1. Static Infrastructure

   The NoC infrastructure of each sector is static. The sector AVMM addresses are hard-coded in security_filter module, and the number of in/out ports is customized based on the location of the sector. Therefore, the parent partition should remain the same as base revision as it cannot be relocated to another sector. Only the child partition can be recompiled to run in a different sector than base revision.

2. Partial reconfiguration process will impact data transfer of other sectors

   For PR of a sector to take place, the neighboring ports will be frozen, and the sector will go into reset state. Throughout this process the data transfer of another sector will not be completed if the data path will cross any of the frozen ports. As the data paths are fixed, the only workaround is to have a master that is aware of the sequence of events and will ensure that the PR and data transfer are performed sequentially.

3. Origin of incoming data is unknown

   In current design, all 9 sectors can write to any other 8 sectors. The data will be placed in FIFO of destination sector. However, receiver sector will not be able to know the origin of the data. For future improvement, a unique label/ID can be embedded with the data so that the receiver sector can identify the sender. This can be implemented with either same AVMM data bus (< 32-bit data + ID) or have a wider AVMM data bus (32-bit data + ID).

4. Performance

   Current sector-to-sector bandwidth is about 44 MB/s. The performance could be improved by supporting burst data transfer or reduce pipeline stage of the interconnect (current 4 stages).

**Hardware Validation**

The purpose of hardware validation is to validate the functionality of the NoC, bitstream relocatability and security filter feature.

There are 4 revisions of images generated for test:

1. Base revision – each user logic will periodically pool the status of local FIFO and will return the data + system ID (same as sector number) to sector0 upon detecting a data.
2. Test_noc_v1 revision – similar as base revision but with each sector will have different system ID.
3. Test_noc_v2 revision – similar as base revision but with each sector will have system ID of 0x10.
4. Test_noc_bad revision – user logic repetitively writes 0xFFFFFFFF to sector0

All commands will be issued from master (sector 0) via System Console.

<u>Board bring-up</u>

1. Program test_noc.sof using Programmer
2. Execute system console script, where all commands are written. The script should perform system reset via jtag_debug_reset_system command and open service path for sector0 JTAG master.

<u>Sanity Check</u>

1. Master writes to all sectors to ensure NoC is functional and each sector should return the data + system ID to master.

FIFO Count increases with each write

```
% sector_write 0xaa
Sector 0 FIFO Status : 0x00000000
Written to sector1 FIFO.
Sector 0 FIFO Status : 0x00000001
Written to sector2 FIFO.
Sector 0 FIFO Status : 0x00000002
Written to sector3 FIFO.
Sector 0 FIFO Status : 0x00000003
Written to sector4 FIFO.
Sector 0 FIFO Status : 0x00000004
Written to sector5 FIFO.
Sector 0 FIFO Status : 0x00000005
Written to sector6 FIFO.
Sector 0 FIFO Status : 0x00000006
Written to sector7 FIFO.
Sector 0 FIFO Status : 0x00000007
Written to sector8 FIFO.
Sector 0 FIFO Status : 0x00000008
```

Read 8 Master's FIFO data

```
% fiford 8
FIFO data 0: 0x000100aa          <- From sector1
Sector 0 FIFO Status : 0x00000007
FIFO data 1: 0x000200aa          <- From sector2
Sector 0 FIFO Status : 0x00000006
FIFO data 2: 0x000300aa          <- From sector3
Sector 0 FIFO Status : 0x00000005
FIFO data 3: 0x000400aa          <- From sector4
Sector 0 FIFO Status : 0x00000004
FIFO data 4: 0x000500aa          <- From sector5
Sector 0 FIFO Status : 0x00000003
FIFO data 5: 0x000600aa          <- From sector6
Sector 0 FIFO Status : 0x00000002
FIFO data 6: 0x000700aa          <- From sector7
Sector 0 FIFO Status : 0x00000001
FIFO data 7: 0x000800aa          <- From sector8
Sector 0 FIFO Status : 0x00000000
```

<u>Sector Partial Reconfiguration</u>

1. Master asserts freeze request to PR region controllers and reset to PR sector

```
% s1freeze
Sent freeze requests to Sector1 PR Region Controllers.
Reset Sector1.
```

2. Program test_noc_v2.s1_parent_partition.rbf, rbf of parent partition from test_noc_v2 (sector-aligned)
3. Master sends unfreeze requests and de-assert reset

```
% s1unfreeze
Sent unfreeze requests to Sector1 PR Region Controllers.
De-assert reset to Sector1.
```

4. Repeat sanity check and the result will show that reconfigured sector will return data with different system ID.

```
% sector_write 0x12
Sector 0 FIFO Status : 0x00000000
Written to sector1 FIFO.
Sector 0 FIFO Status : 0x00000001
Written to sector2 FIFO.
Sector 0 FIFO Status : 0x00000002
Written to sector3 FIFO.
Sector 0 FIFO Status : 0x00000003
Written to sector4 FIFO.
Sector 0 FIFO Status : 0x00000004
Written to sector5 FIFO.
Sector 0 FIFO Status : 0x00000005
Written to sector6 FIFO.
Sector 0 FIFO Status : 0x00000006
Written to sector7 FIFO.
Sector 0 FIFO Status : 0x00000007
Written to sector8 FIFO.
Sector 0 FIFO Status : 0x00000008
```

```
% fiford 8
FIFO data 0: 0x00100012
Sector 0 FIFO Status : 0x00000007
FIFO data 1: 0x00020012
Sector 0 FIFO Status : 0x00000006
FIFO data 2: 0x00030012
Sector 0 FIFO Status : 0x00000005
FIFO data 3: 0x00040012
Sector 0 FIFO Status : 0x00000004
FIFO data 4: 0x00050012
Sector 0 FIFO Status : 0x00000003
FIFO data 5: 0x00060012
Sector 0 FIFO Status : 0x00000002
FIFO data 6: 0x00070012
Sector 0 FIFO Status : 0x00000001
FIFO data 7: 0x00080012
Sector 0 FIFO Status : 0x00000000
```

ID changed from 1 to 10

5. This proves that the sector has been successfully reconfigured.

<u>Security Filter</u>

1. Perform PR using .rbf of parent partition from test_noc_bad.
2. Without the master sending any data to the sector, it will still repetitively write 0xFFFFFFFF to the master causing the FIFO count to increase.

```
% regrd 0x40020
0x00000000
% s2unfreeze
Sent unfreeze requests to Sector2 PR Region Controllers.
De-assert reset to Sector2.
% regrd 0x40020
0x00000003
% regrd 0x40020
0x00000004
% fiford 4
FIFO data 0: 0xffffffff
FIFO data 1: 0xffffffff
FIFO data 2: 0xffffffff
FIFO data 3: 0xffffffff
```

Read sector0 FIFO count

FIFO count increasing without any write

3. After enabling security filter from that sector to master, master no longer receive data from that sector.

```
% s2filter 0x1
Written 0x1 to sector2 security filter.

% regrd 0x40020
0x00000006
% regrd 0x40020
0x00000006
```

Enable filter to block data transfer from sector2 to sector0

FIFO count no longer increasing