AXI PR Controller v1.0 User Guide

Daniel Ly-Ma

May 16, 2018

Overview

The AXI PR Controller is designed to efficiently program partial bitstreams that are stored in offchip memory and to readback partial reconfigurable regions on the FPGA back to offchip memory. It is to be used with an embedded processing system using the AXI interface such as the MicroBlaze and ARM processor. It supports the 7series, Ultrascale and Ultrascale+ families of chips.

Design Information

These are some basic steps to use the controller in your design.

1. Make sure that you select the correct family in the IP configuration window for the controller.
2. Connect the m_axi bus to your offchip memory where the partial bitstream will be stored.
3. Connect the s_axi_ctrl bus to the MicroBlaze or ARM.
4. If using interrupts, connect the done_interrupt pin to the MicroBlaze or ARM.

Partial Bitstream Format

WARNING: The *.bit partial bitstreams generated by Vivado cannot be used directly with the controller. They must be converted to *.bin file to strip out the header. To do this, use the follow command in the Vivado tcl console on your partial bitstream:

```
write_cfgmem -format BIN -interface SMAPx32 -disablebitswap
-loadbit "up 0x0 <partial_bitfile>"
```

CTRL (Control Register - Offset 00h)

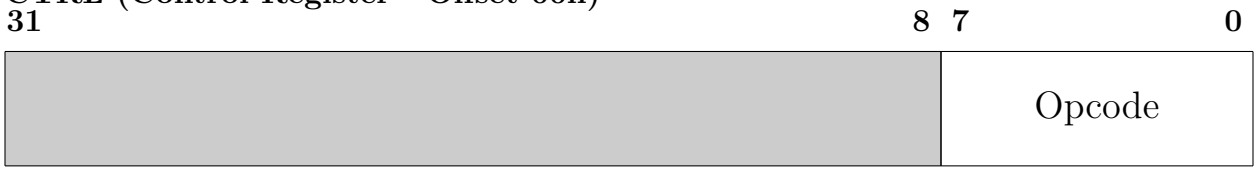

Table 1: CTRL Register Description

Bits	Field Name	Default Value	Access Type	Description
31 to 8	Reserved	0	RO	Writing to these bits has no effect.
7 to 0	Opcode	0	R/W	Controls whether it is a write or read operation. Set to 0 to write a bitstream to the ICAP and 1 to read bits from the ICAP. • 0 = Write to ICAP • 1 = Read from ICAP

BA_MSB (Base Address (MSB) Register - Offset 04h)

Table 2: BASE_MSB Register Description

Bits	Field Name	Default Value	Access Type	Description
31 to 8	Reserved	0	RO	Writing to these bits has no effect.
7 to 0	Base Address (MSB)	0	R/W	If write operation,indicates the MSB 8 bits of the source address of the partial bitstream. If read operation, indicates the MSB 8 bits of the destination address to store readback bitstream

BA (Base Address Register - Offset 08h)

Table 3: BA Register Description

Bits	Field Name	Default Value	Access Type	Description
31 to 0	Base Address	0	R/W	If write operation,indicates the lower 32 bits of source address of the partial bitstream. If read operation, indicates the lower 32 bits of the destination address to store readback bitstream

BIT_SIZE (Bitstream Size Register - Offset 0Ch)

Table 4: BIT_SIZE Register Description

Bits	Field Name	Default Value	Access Type	Description
31 to 0	Base Address	0	R/W	If write operation, indicates the size (bytes) of the partial bitstream to be written. If read operation, indicates the size (bytes) of the readback to save from the ICAP. The CTRL, and BA registers should be written before writing to the BIT_SIZE register since writing to this register will create a task and put it on the task queue. Writes to this register will be ignored while the controller is busy.

NUM_BIT (Bitstream Count Register - Offset 10h)

Table 5: NUM_BIT Register Description

Bits	Field Name	Default Value	Access Type	Description
31 to 0	Base Address	0	R/W	Indicates how many operations that were queued. Must match the number of times BIT_SIZE was written to. Writing to this register will cause the controller to begin processing the tasks on the queue. Writes to this register will be ignored while the controller is busy.

STATUS (Status Register - Offset 14h)

Table 6: STATUS Register Description

Bits	Field Name	Default Value	Access Type	Description
31 to 4	Reserved	0	RO	Writing to these bits has no effect.
3	DMErr	0	RO	Indicates an error with the datamover
2	PRErr	0	RO	Indicates an error with the ICAP
1	Busy	0	RO	Indicates that the controller is busy
0	Done	0	R/W	Indicates that the controller has finished processing the all the tasks on the queue. Must be cleared before new tasks can be queued. Writing a 1 to this bit will clear it.

Programming Sequence

Writing a Partial Bitstream

These steps will show you how to use the controller to write a partial bitstream to the FPGA.

1. Set the task to be in write mode by writing 0 to the Opcode field in CTRL (CTRL.Opcode = 0).
2. Point the task to the base address of the partial bitstream in DDR memory by writing the appropriate address to BA and BA_MSB.
3. Write the size of the partial bitstream in bytes to the BIT_SIZE register. This will add the task to the queue.
4. Write 1 to the NUM_BIT register (NUM_BIT.Count = 1). This will tell the controller to start processing tasks on the queue.
5. Either poll the STATUS.Done bit or wait for the controller to generate an interrupt.
6. Clear the interrupt and STATUS.Done bit by writing a 1 to STATUS.Done (STATUS.Done = 1).
7. Ready for another transfer. Go back to step 1.

Writing Multiple Partial Bitstreams

These steps will show you how to use the controller to write multiple partial bitstream to the FPGA.

1. Set the task to be in write mode by writing 0 to the Opcode field in CTRL (CTRL.Opcode = 0).
2. Point the task to the base address of the partial bitstream in DDR memory by writing the appropriate address to BA and BA_MSB.
3. Write the size of the partial bitstream in bytes to the BIT_SIZE register. This will add the task to the queue.
4. Repeat steps 1 - 3 for all of the partial bitstreams you want to program.
5. Write the number of tasks that were enqueued to the NUM_BIT register. This will tell the controller to start processing tasks on the queue.
6. Either poll the STATUS.Done bit or wait for the controller to generate an interrupt.
7. Clear the interrupt and STATUS.Done bit by writing a 1 to STATUS.Done (STATUS.Done = 1).
8. Ready for another transfer. Go back to step 1.

Readback

These steps will show you how to readback the bits in a partial reconfigurable region on the FPGA.

1. Put the ICAP into readback state by enqueuing a write task for your first stage readback partial bitstream using the steps outlined above.
2. Set a new task to be in read mode by writing 1 to the Opcode field in CTRL (CTRL.Opcode = 1).
3. Point the task to the base address of where you want to store the readback data in DDR memory by writing the appropriate address to BA and BA_MSB.
4. Write the size of the readback data in bytes to the BIT_SIZE register. This will add the task to the queue.
5. Take the ICAP out of readback state by enqueuing a write task for your second stage readback partial bitstream using the steps outlined above.
6. Write 3 to the NUM_BIT register (NUM_BIT.Count = 1). This will tell the controller to start processing tasks on the queue.
7. Either poll the STATUS.Done bit or wait for the controller to generate an interrupt.
8. Clear the interrupt and STATUS.Done bit by writing a 1 to STATUS.Done (STATUS.Done = 1).
9. Ready for another transfer. Go back to step 1.