Xilinx Answer 71322
Reading AXI PCIe Gen3/XDMA internal registers using JTAG to AXI Master IP

Important Note: This downloadable PDF of an Answer Record is provided to enhance its usability and readability. It is important to note that Answer Records are Web-based content that are frequently updated as new information becomes available. You are reminded to visit the Xilinx Technical Support Website and review (Xilinx Answer 71322) for the latest version of this Answer.

Introduction

This document describes the use of JTAG to AXI Master IP to access the internal configuration registers through the AXI4-Lite interface of the AXI Bridge for PCI Express Gen3 (AXI PCIe Gen3) and the DMA Subsystem for PCI Express (XDMA). A block diagram of the design and Tcl commands have been provided to read and write to the internal registers and probing the resulting AXI transactions in Vivado ILA.

AXI Bridge for PCI Express Gen3 Architecture

Figure 1 shows AXI PCIe Gen3 architecture (Ref. PG194). The red arrow in the diagram shows access to the bridge block registers through the AXI4-Lite interface.

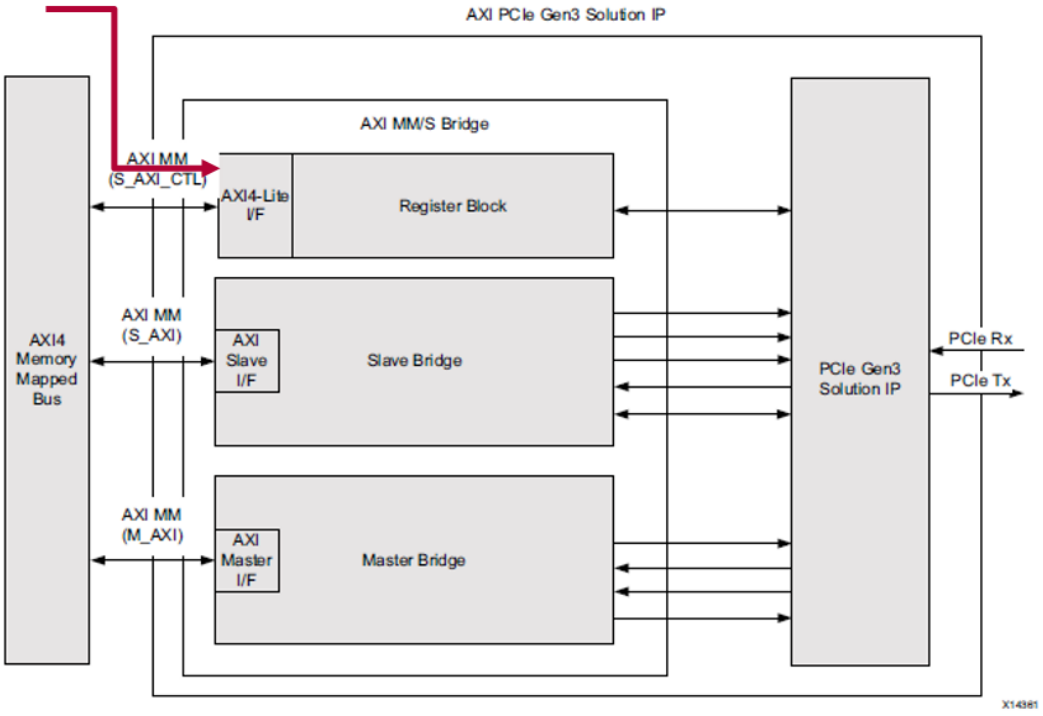

Figure 1 - AXI PCIe Gen3 Architecture

DMA Subsystem for PCI Express Architecture

Figure 2 shows XDMA Architecture (Ref. PG214). The red arrow in the diagram shows access to the fblock registers through the AXI4-Lite interface.

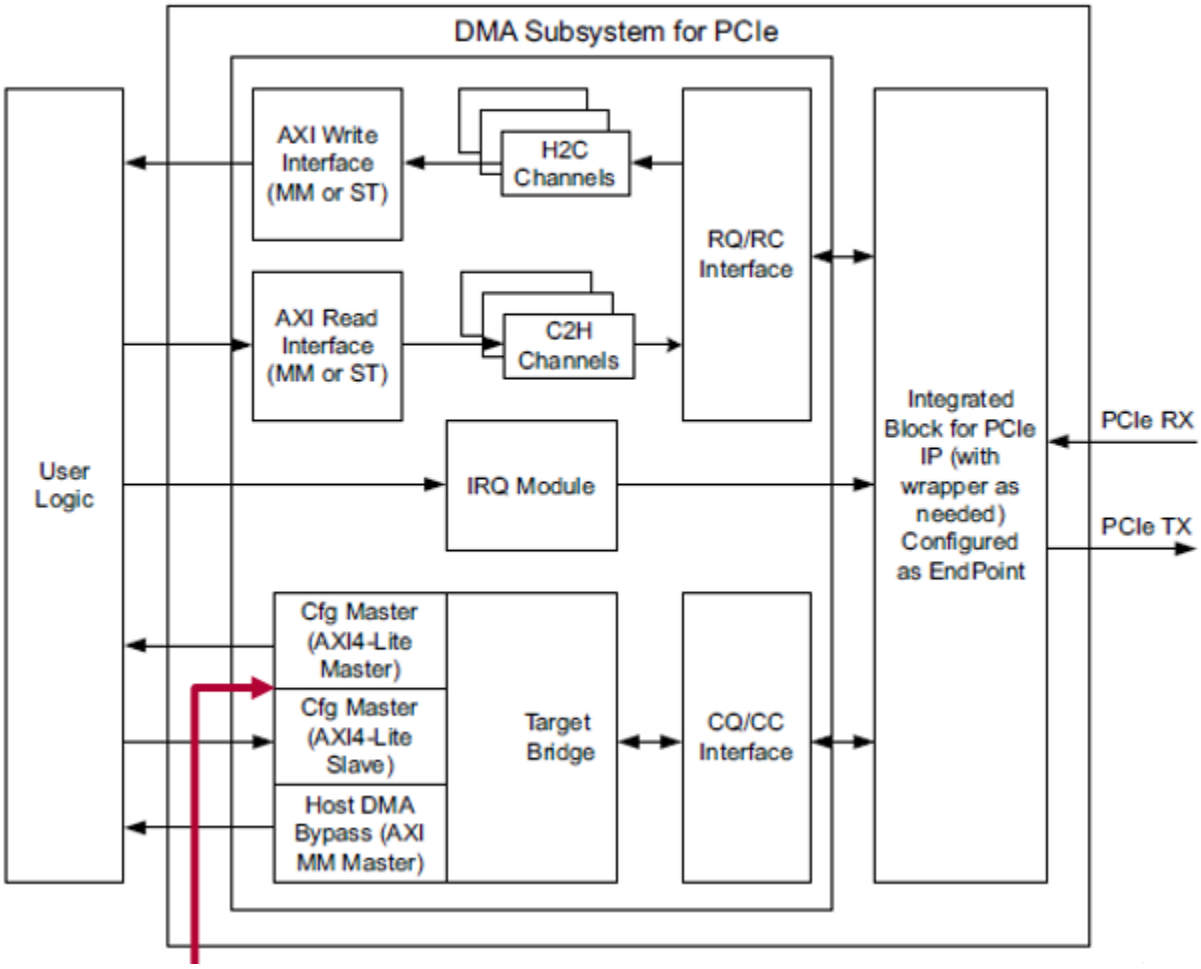

Figure 2 - XDMA Architecture

Register Memory Map

Accessibility	Offset	Contents	Location
RO - EP	0x000 - 0x12F	PCIe Configuration Space Header	Part of integrated PCIe configuration space.
RO	0x130	Bridge Info	AXI bridge defined memory-mapped register space.
RO - EP	0x134	Bridge Status and Control	
R/W	0x138	Interrupt Decode	
R/W	0x13C	Interrupt Mask	
RO - EP	0x140	Bus Location	
RO	0x144	Physical-Side Interface (PHY) Status/Control	
RO - EP, R/W - RC	0x148	Root Port Status/Control	
RO - EP, R/W - RC	0x14C	Root Port MSI Base 1	
RO - EP, R/W - RC	0x150	Root Port MSI Base 2	
RO - EP, R/W - RC	0x154	Root Port Error FIFO Read	
RO - EP, R/W - RC	0x158	Root Port Interrupt FIFO Read 1	
RO - EP, R/W - RC	0x15C	Root Port Interrupt FIFO Read 2	
RO	0x160 - 0x164	Reserved (zeros returned on read)	
R/W	0x168	Configuration Control	AXI bridge defined memory-mapped register space.
RO	0x16C - 0x1FF	Reserved (zeros returned on read)	
RO	0x200	VSEC Capability 2	
RO	0x204	VSEC Header 2	
R/W	0x208 - 0x234	AXI Base Address Translation Configuration Registers	AXI bridge defined memory-mapped space.
RO	0x238 - 0xFFF	Reserved (zeros returned on read)	

Table 1 - Register Memory Map (Ref: PG194)

Table 1 lists the internal registers of AXI PCIe Gen3. These registers can be accessed through the AXI4-Lite Control Interface and are offset from the base address which can be retrieved in the address editor tab of the IP Integrator block design.

AXI PCIe Gen3/XDMA (Bridge Mode) Config AXI4-Lite Memory Mapped Interface Signals

AXI4-Lite Control Interface **Note**: For all signals in this interface, **s_axi_ctl*** is used for the AXI Bridge for PCIe Gen3 core, and **s_axil_*** is used for the DMA/Bridge Subsystem for PCIe in AXI Bridge mode.		
Endpoint configuration: s_axi_ctl_awaddr[11:0] | s_axil_awaddr[31:0] Root Port configuration: s_axi_ctl_awaddr[27:0] | s_axil_awaddr[31:0]	I	Slave write address. For DMA/Bridge Subsystem for PCIe in AXI Bridge mode, address must always be 512 MB aligned. • s_axil_awaddr[28] = 1'b0 selects Bridge Register Memory Map and Enhanced Configuration Access Memory Map. • s_axil_awaddr[28] = 1'b1 selects DMA/Bridge Subsystem for PCIe Register Memory Map. For AXI Bridge for PCIe Gen3 core, address must be aligned to 4 KB in Endpoint configuration and 256 MB in Root Port configuration.
s_axi_ctl_awvalid | s_axil_awvalid	I	Slave write address valid
s_axi_ctl_awready | s_axil_awready	O	Slave write address ready
s_axi_ctl_wdata[31:0] | s_axil_wdata[31:0]	I	Slave write data
s_axi_ctl_wstrb[3:0] | s_axil_wstrb[3:0]	I	Slave write strobe
s_axi_ctl_wvalid | s_axil_wvalid	I	Slave write valid
s_axi_ctl_wready | s_axil_wready	O	Slave write ready
s_axi_ctl_bresp[1:0] | s_axil_bresp[1:0]	O	Slave write response
s_axi_ctl_bvalid | s_axil_bvalid	O	Slave write response valid
s_axi_ctl_bready | s_axil_bready	I	Slave response ready
Endpoint configuration: s_axi_ctl_araddr[11:0] | s_axil_araddr[31:0] Root Port configuration: s_axi_ctl_araddr[27:0] | s_axil_araddr[31:0]	I	Slave read address. For DMA/Bridge Subsystem for PCIe in AXI Bridge mode, address must always be 512 MB aligned. • s_axil_araddr[28] = 1'b0 selects Bridge Register Memory Map and Enhanced Configuration Access Memory Map • s_axil_araddr[28] = 1'b1 selects DMA/Bridge Subsystem for PCIe Register Memory Map For AXI Bridge for PCIe Gen3 core, address must be aligned to 4 KB in Endpoint configuration and 256 MB in Root Port configuration.
s_axi_ctl_arvalid | s_axil_arvalid	I	Slave read address valid
s_axi_ctl_arready | s_axil_arvalid	O	Slave read address ready
s_axi_ctl_rdata[31:0] | s_axil_arvalid	O	Slave read data
s_axi_ctl_rresp[1:0] | s_axil_arvalid	O	Slave read response
s_axi_ctl_rvalid | s_axil_arvalid	O	Slave read valid
s_axi_ctl_rready | s_axil_arvalid	I	Slave read ready

Table 2 - AXI4-Lite Control Interface (Ref: PG194)

AXI Bridge for PCI Express IP Integrator Block Design

Figure 3 shows AXI PCIe Gen3 IP Integrator design with JTAG to AXI IP and System ILA. The JTAG to AXI is used to access the register space through the AXI4-Lite Interface. The System ILA is used to capture the waveform showing the transactions.

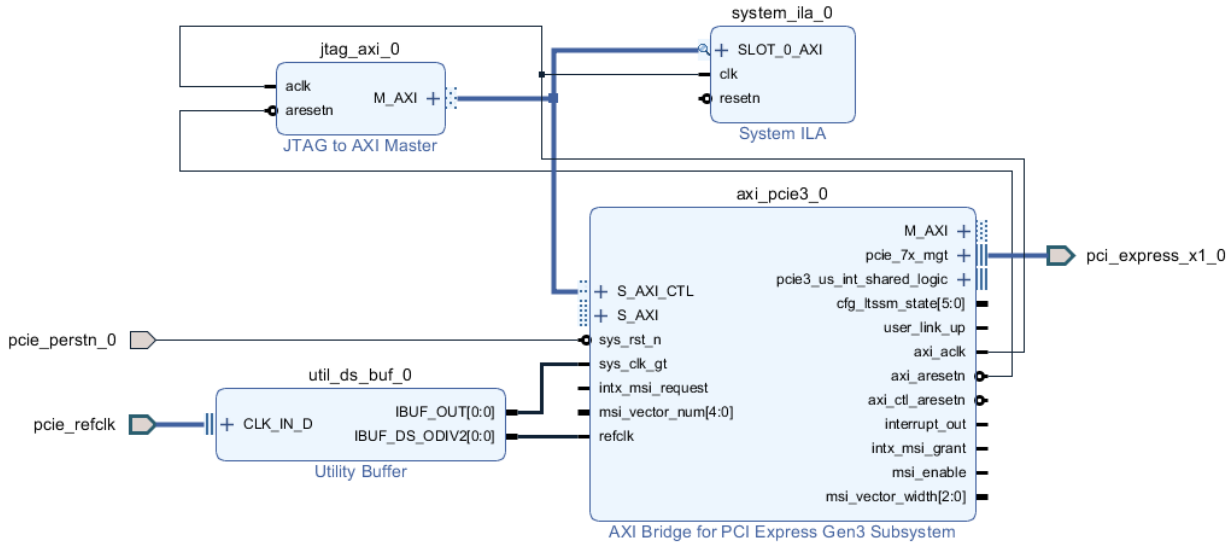

Figure 3 - AXI PCIe Gen3 IPI Block Design

DMA Subsystem for PCI Express IP Integrator Block Design

Figure 4 shows XDMA IP Integrator design with JTAG to AXI IP and System ILA. The JTAG to AXI is used to access the register space through the AXI4-Lite Interface. The System ILA is used to capture the waveform showing the transactions.

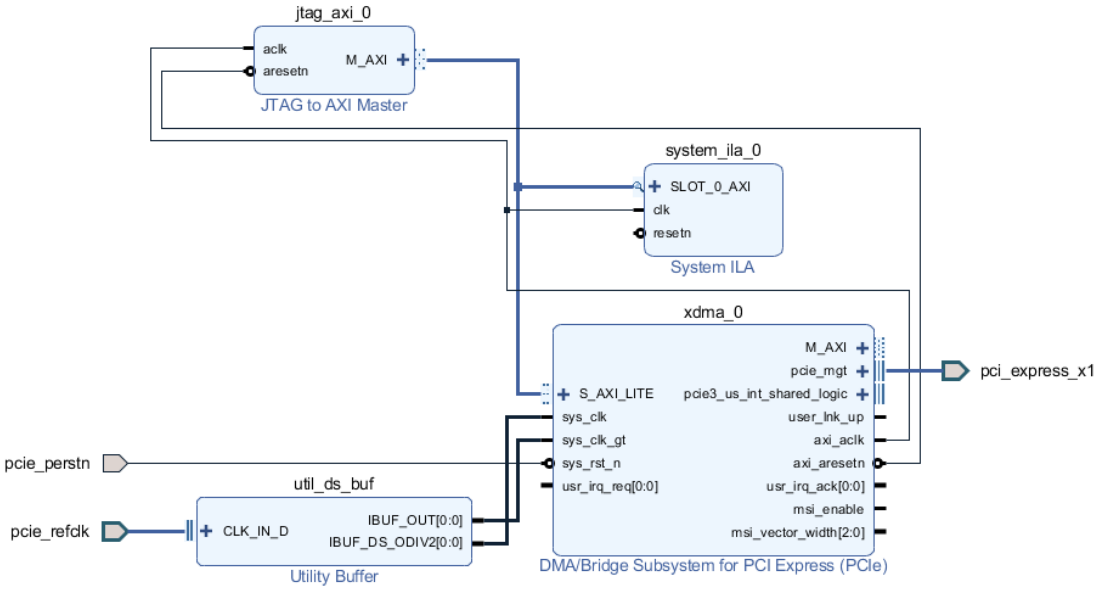

Figure 4 - XDMA IPI Block Design

JTAG to AXI Master IP

JTAG to AXI Master IP is used here to generate AXI Transactions on the AXI4-Lite interface. Through this IP, Tcl commands are used to read and write to internal registers of the AXI PCIe Gen3 and XDMA. For more information on JTAG to AXI Master IP, see (PG174).

Tcl procedures for read/write AXI transactions

The following AXI read and AXI write procedures were created to simplify the commands used to make the accesses. Once these procedures are run in the Tcl console, simplified accesses are made using the simple *igd_axi_rd/wr* commands and base address and the offset address of the register space that is to be accessed.

The Tcl procedures can be copied into the Vivado Hardware Manager Tcl console and then the Tcl commands for read/write AXI burst transactions can be used.

- ***igd_axi_rd***
  - *igd_axi_rd* procedure creates an AXI read burst transaction offset from the base address
- ***igd_axi_wr***
  - *igd_axi_wr* procedure creates an AXI write burst transaction offset from the base address with a data value

```
**************************************************************************************************************************************************
proc igd_axi_rd {base addr} {
    create_hw_axi_txn igd_hw_read [get_hw_axis hw_axi_1] -quiet -type READ -address [format
%08x [expr $addr + $base]]
    run_hw_axi -quiet [get_hw_axi_txns -quiet igd_hw_read]
    set rdata 0x[get_property DATA [get_hw_axi_txns -quiet igd_hw_read]]
    delete_hw_axi_txn -quiet igd_hw_read
    return $rdata
}
proc igd_axi_wr {base addr data} {
    upvar #0 _igd_gpio_wrv d
```

```
    create_hw_axi_txn  igd_hw_write  [get_hw_axis  hw_axi_1]  -quiet  -type  WRITE  -address
[format %08x [expr $addr + $base]] -data [format %08x $data]
    run_hw_axi -quiet [get_hw_axi_txns -quiet igd_hw_write]
    delete_hw_axi_txn -quiet igd_hw_write
    dict set d [format %08x [expr $addr + $base]] $data
    return $data
}
****************************************************************************************************************************************************
```

Test steps

Step 1: Open the Address Editor in Vivado block design to get the base address for the AXI PCIe address space. For the example shown in Figure 5, the base address is 0x00000000. This will be used as the base address for the AXI read and write commands.

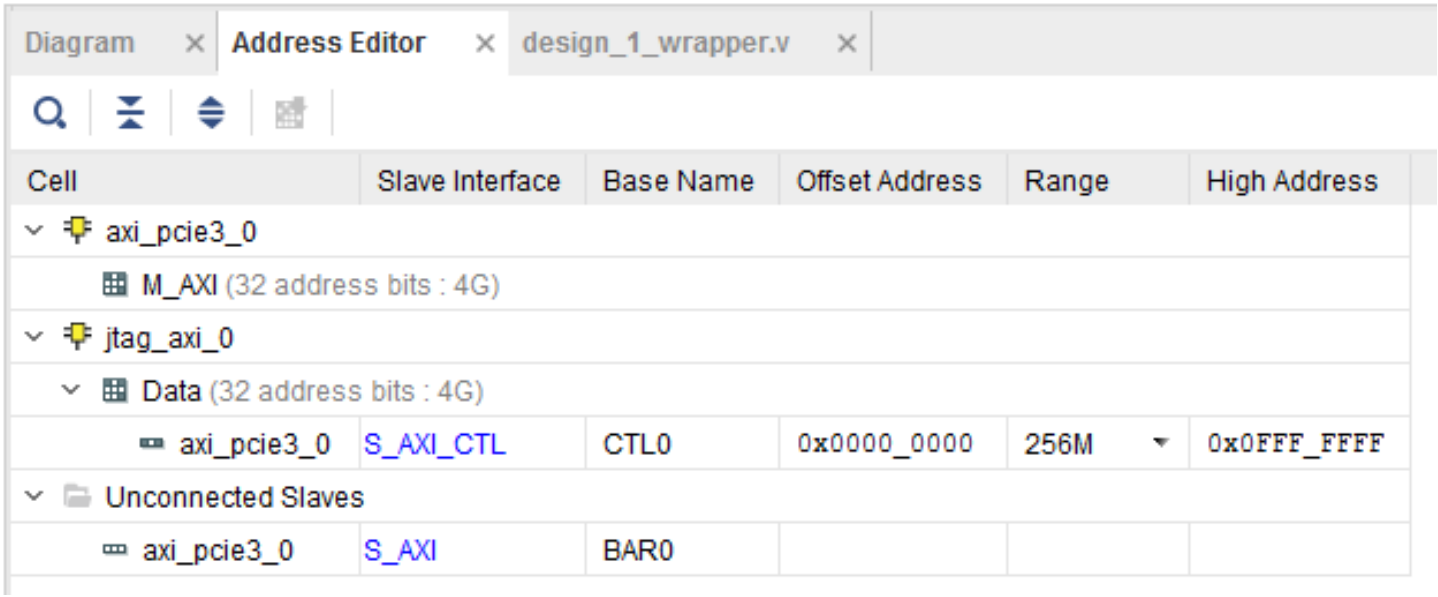

Figure 5 – IP Integrator Address Editor

Step 2: Open Hardware manager and set up the trigger as per the screen capture shown in Figure 6.

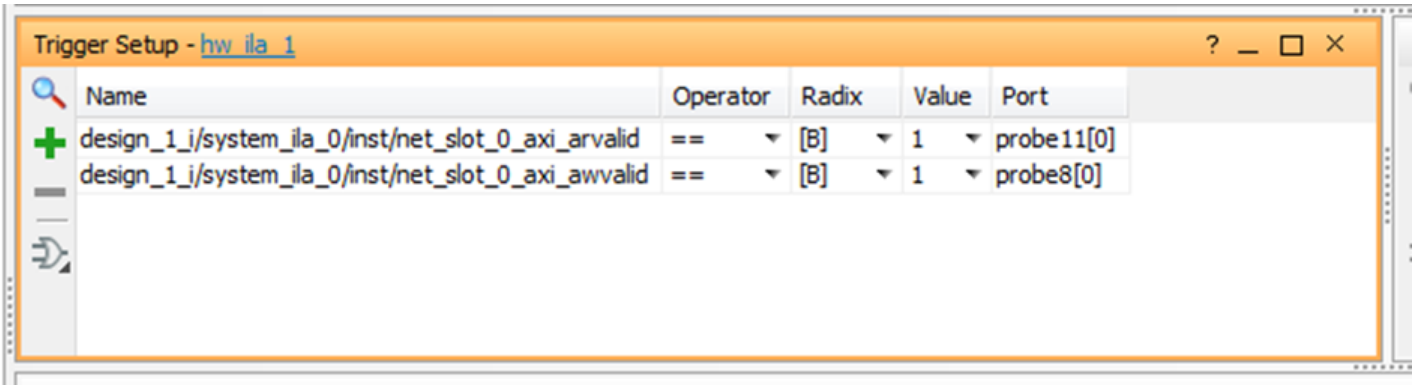

Figure 6 - Trigger Setup

Step 3: Copy the procedures for reads and writes in the Tcl console and run these.

Figure 7 - Running Tcl Procs

Step 4: Set up the trigger for ILA as mentioned in Step-2 and run the *igd_axi_wr* command as shown in Figure 8.

```
igd_axi_wr 0x00000000 0x208 0xdeadbeef
```
Result : `0xdeadbeef`

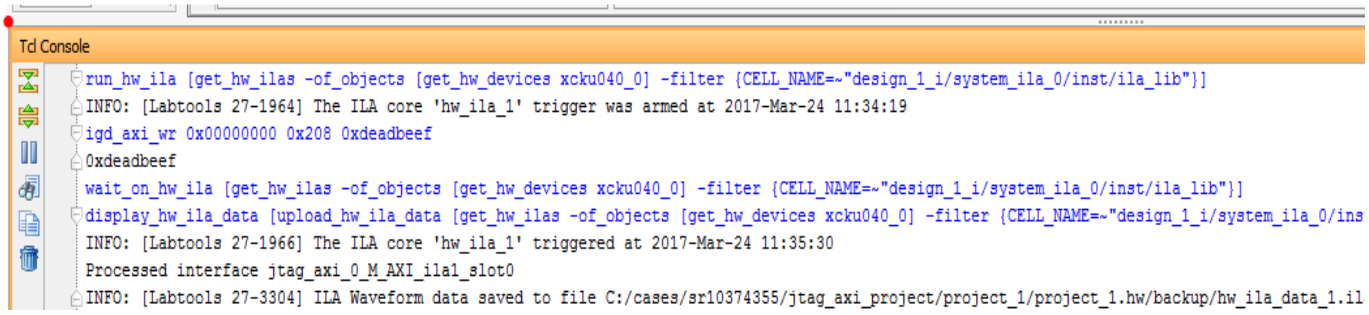

Figure 8 - Writing to AXI PCIe Gen3 Internal Register

Figure 9 shows that a write to offset address 0x208 and 0xdeadbeef is seen with correct toggling of the *awvalid* and *awready* signals.

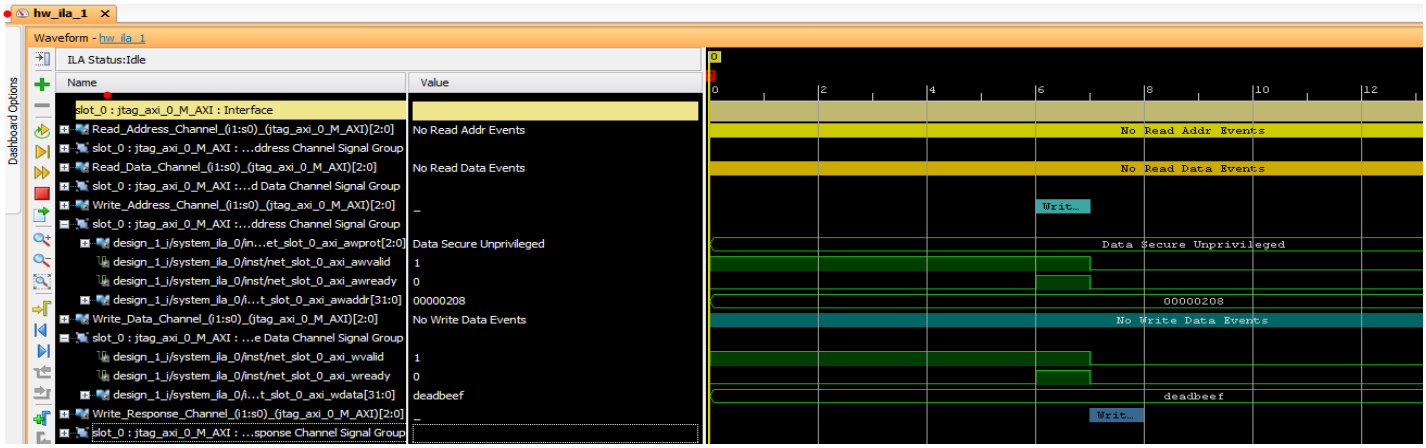

Figure 9 - Register Write ILA Waveform

Step 5: Arm the ILA and run the *igd_axi_rd* command as per below screen capture.

```
igd_axi_rd 0x00000000 0x208
```
Result : `0xdeadbeef`

Figure 10 - Reading from AXI PCIe Gen3 Internal Register

Figure 11 shows that a read from offset address 0x208 and 0xdeadbeef is seen with correct toggling of the *arvalid* and *arready* signals.

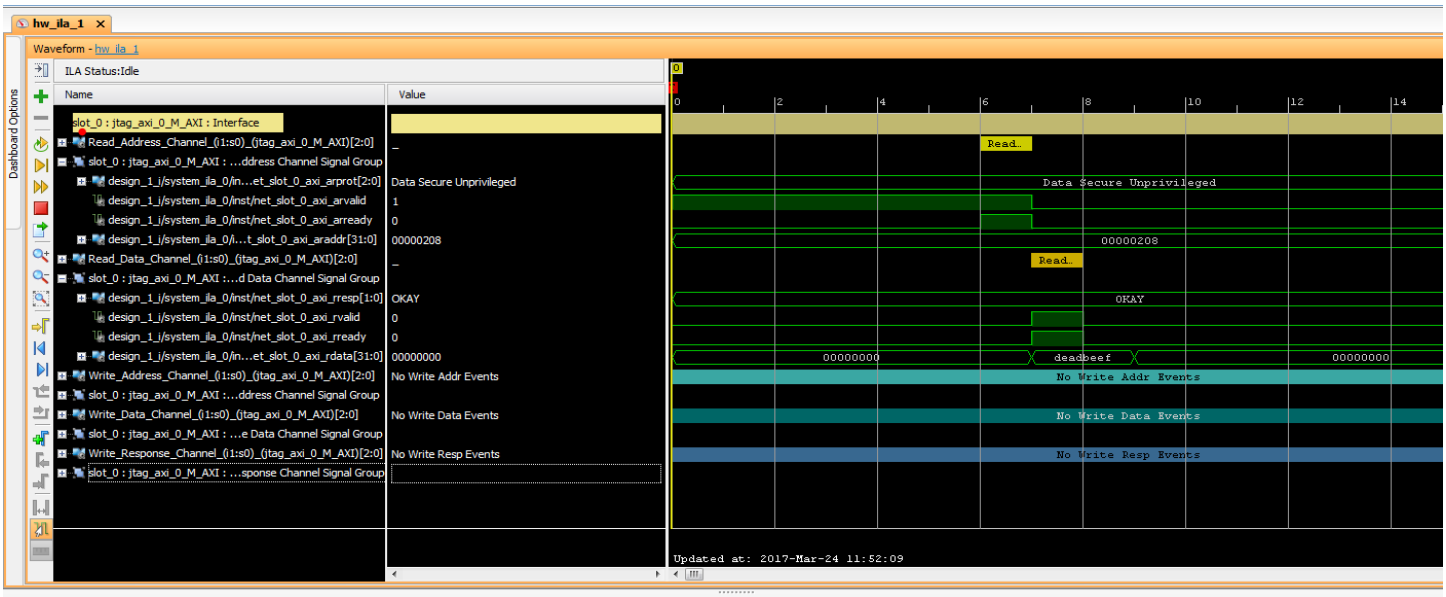

Figure 11 - Register Read ILA Waveform

Conclusion

This document presented how JTAG to AXI Master IP can be used to access the internal configuration registers through the AXI4-Lite interface of the AXI Bridge for PCI Express Gen3 (AXI PCIE Gen3) and the DMA Subsystem for PCI Express (XDMA). The illustrated steps would be helpful for debugging by reading registers listed in Table 1. The method described would be used particularly when the host processor hangs and the registers could not be read from the software.

Reference

PG194: AXI Bridge for PCI Express Gen3 Subsystem v3.0
PG195: DMA/Bridge Subsystem for PCI Express v4.1

Revision History

07/30/2018 - Initial release