XAPP1214 (v1.0) June 24, 2014

Bridging an AXI4-Lite Interface to DRP Interfaces

Author: Luis Bielich

Summary

This application note provides a reference design with a custom IP created in Vivado® IP Packager for bridging an AXI4-Lite interface to one or more Dynamic Reconfiguration Port (DRP) interfaces. The DRP is a common port used to reconfigure clock management blocks, serial transceivers, the Xilinx Analog Digital Converter (XADC), or the PCI Express® blocks without requiring a new bitstream. Although these macros are configured with attributes in the source code, you can reprogram them while the design is up and running through the DRP interface.

Introduction

The DRP interface is a memory mapped interface to registers within specific integrated macros in the FPGA. Allowing access to these registers enables more flexibility for the macros by preventing the need to reprogram the FPGA.

Although the DRP interface is a simple interface, neither MicroBlaze™ nor ARM® processors include a DRP interface. Therefore, a bridge is generally required from AXI4 or AXI4-Lite interfaces to DRP for the processor to interact with the DRP interface of the macro.

It is also common for applications to require access to multiple DRP interfaces. This is especially true with multi-lane transceiver applications where each transceiver has its own DRP interface. The DRP interface can be used to modify transceiver phase-locked loop (PLL) rates or perform an in-system eye scan. These applications require access to multiple DRP interfaces and it is preferable to have only one slave port to the AXI interconnect to keep the interconnect size as small as possible. Adding more master interfaces to the interconnect can quickly increase the size of the AXI interconnect. The IP packaged with this application note allows for multiple DRP interfaces with only one AXI slave port, shown in Figure 1.

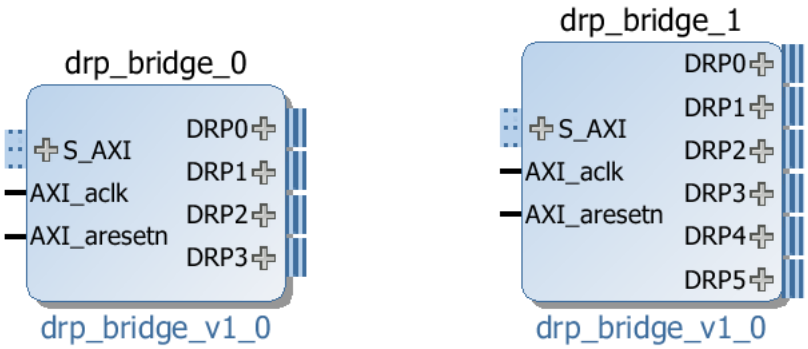

Figure 1: **Multiple DRP Interfaces to One AXI Interface**

Hardware Description

The AXI4-Lite to DRPs IP is compliant with the AXI4-Lite specification and is functional with Xilinx® DRP interfaces. The IP can drive up to 32 different DRP interfaces with one AXI4-Lite slave interface. One AXI request is accepted and quickly translated to DRP. After the DRP interface responds with a ready pulse, the next AXI request is accepted. Although the IP accepts one AXI request at a time, the IP continues to function when back-to-back incoming AXI transactions are presented.

With multiple DRP interfaces, the addressing is contiguous between each DRP interface. Each DRP interface occupies $2^{(\#\ of\ DRP\ address\ bits+2)}$. The "+2" is part of the equation because the bridge does not look at the lower 2 bits of the AXI address. See Address Offset for more information.

The DRP data width maps directly to the AXI data and the upper AXI data bits are ignored when the widths do not match. Generally, the DRP data width is 16 bits and the AXI data width is fixed to 32 bits. This means the upper 16 bits on the AXI interface are ignored for writes, and the upper 16 bits are kept at 0 for reads.

The packaged IP allows for three different options within the IP customization window, shown in Figure 2:

- DRP Count
- DRP Address Width
- DRP Data Width

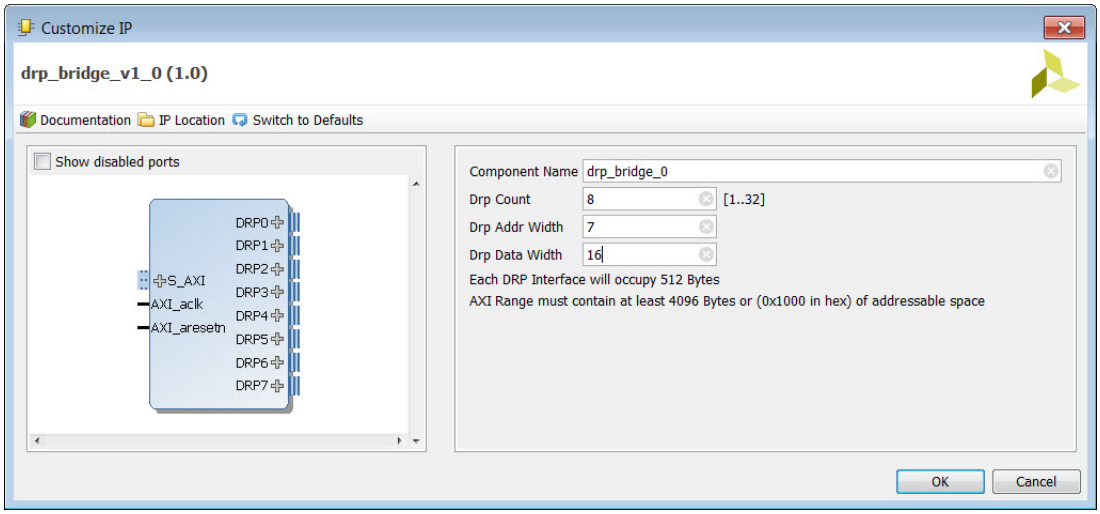

Figure 2: **IP Options**

DRP Count

The DRP Count enables the addition of more DRP interfaces. As the count increases, the necessary AXI addressable space increases, and the count is shown in the customization window. In Figure 2, the IP requires 4KB of addressable space to allow for eight DRP interfaces each containing 7-bit DRP address widths.

DRP Address Width

The DRP Address Width depend on the DRP interface of the slave. For example, the MMCM in 7 series devices has seven bits of DRP address, whereas the GTH transceiver has nine bits of DRP address. Each of the DRP interfaces of the IP have the same address width. If the IP must interface with DRP interfaces of different widths, choose the widest width as the option for IP core. Each slave interface is contiguous relative to the DRP Address option chosen in the IP.

DRP Data Width

The Data Width for the DRP interface is generally 16 bits. This will remain an option if a future DRP interface increases the width of the data.

Required AXI Address Range

The Required AXI Address Range is calculated at the bottom of the Customize IP window as shown in Figure 3, and is calculated by using Equation 1.

$$(\#\ \text{of DRP interfaces}) * 2^{(\#\ \text{of DRP address bits}+2)} \qquad \textit{Equation 1}$$

The calculation of the addressable space must match Address Range in the Address Editor as shown in Figure 3.

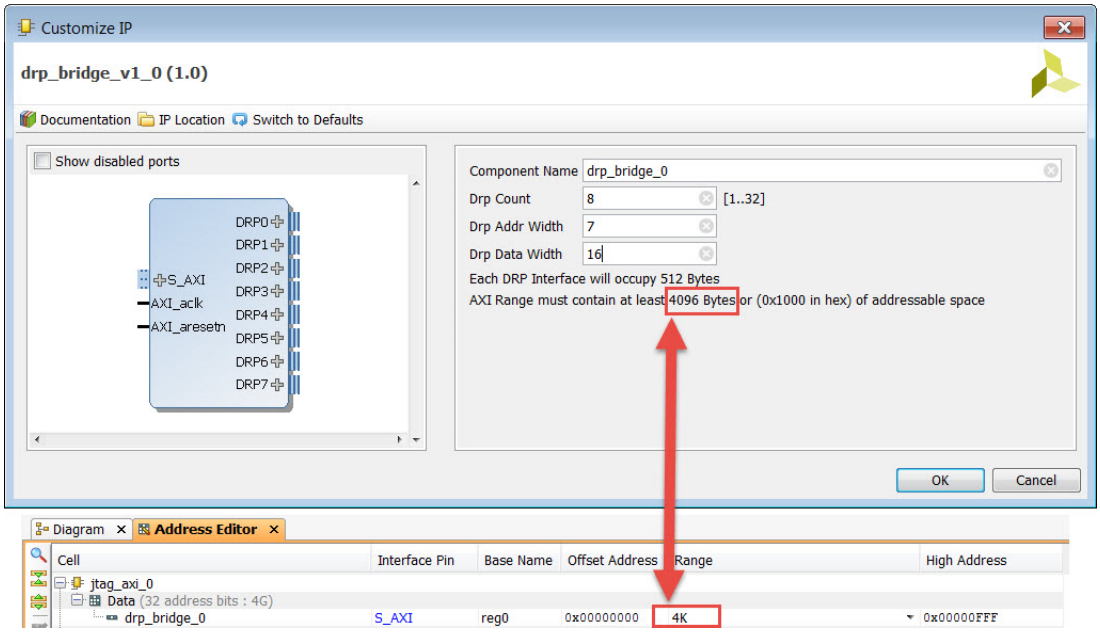

Figure 3: **Correlation of Required AXI Address Range and the Address Editor**

The offset for every DRP interface is also calculated in the IP window, shown in Figure 4. The equation to calculate the DRP offset is described in Equation 2:

$$2^{(\#\ \text{of DRP address bits}+2)} \qquad \textit{Equation 2}$$

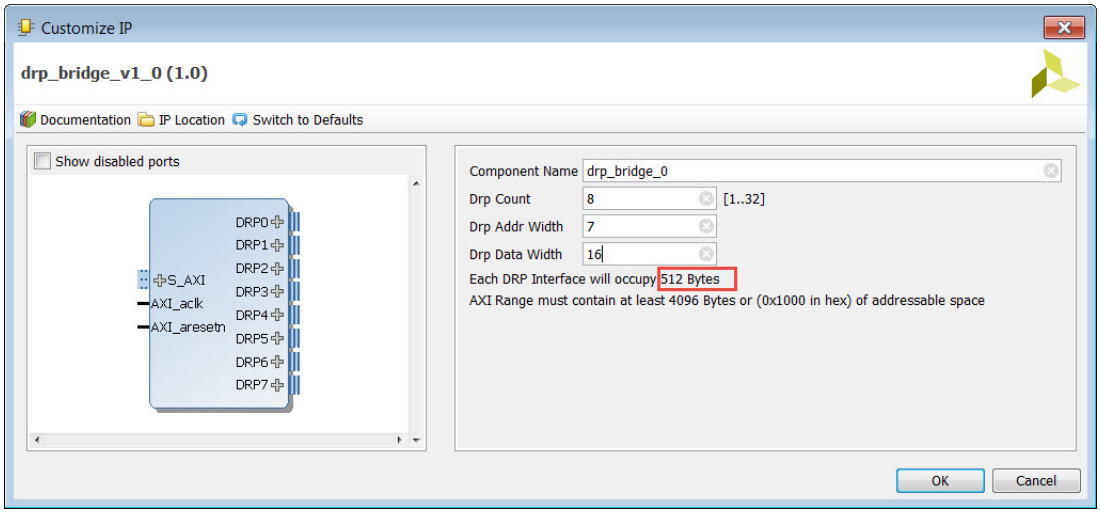

Figure 4: **DRP Offset in the Customization Window**

Address Offset

The address mapping from the AXI interface is offset by two bits to the DRP address. The bridge is implemented this way to keep it lightweight by not supporting unaligned transfers. This results in the lower two bits of the AXI address being ignored. For example, if the base address for the AXI4-Lite to DRPs Bridge is 0xC0000000, then an access to address 0xC0000010 results in an access to address 0x004 to the DRP interface. If a request to address 0xC0000011 is requested, this similarly accesses 0x004 to the DRP interface because the lower two address bits are ignored. Software must account for the address mapping. Table 1 describes several examples of how the bridge functions.

Table 1: **Example of Address Correlation between AXI and DRP**

AXI Address Offset	AXI Data	Resulting DRP Address	Resulting DRP Data
x0	x00001234	x0	x1234
x4	x00005678	x1	x5678
x8	x00009ABC	x2	x9ABC
xC	x0000DEF0	X3	xDEF0
…	…	…	…

As a result of the of the two-bit offset, the address to the next subsequent DRP interface is offset with respect to the DRP Address width plus two. Table 2 shows two different examples of where the next subsequent DRP interface is accessed:

Table 2: **Example of DRP Subsequent Base Address**

DRP Address Width	DRP 0 Base Address	DRP 1 Base Address	DRP 2 Base Address	DRP3 Base Address
7	0	x200	x400	…
8	0	x400	x800	…
9	0	x800	x1000	…
10	0	x1000	x2000	…
…	…	…	…	…

Decode Error Response

The amount of addressable space allocated in Vivado IP integrator is a power of two. If the DRP count is not equal to a power of two, an unmapped address space from the DRP interface to the AXI space occurs. When a request to this unmapped area occurs, the bridge responds with a decode error. For example, when the DRP address width is seven and there are three DRP interfaces, the AXI Address space from `x600-x7FF` is not mapped to anything and results in a decode error on the `rresp` or `bresp` signals.

Latency

Overall latency consists of two portions - fixed latency from the bridge design, and a variable latency from the DRP ready pulse response. The variable portion depends on the DRP slave and can also depend on the register being accessed within the macro. The fixed portion of the latency for writes requests are four cycles of latency. The fixed portion of latency for reads requests are three cycles of latency.

Limitations

The use of Write Strobes (WSTRB) is not supported from the AXI interface. The bridge assumes WSTRB is always High. Read modify writes are recommended instead of using WSTRB. Read-modify-writes are generally used for the DRP interface because the DRP registers are usually not segmented on byte boundaries. For this reason, it is necessary to read-modify-write in software rather than use the WSTRB capabilities.

Unaligned requests are not supported. Keep requests aligned to the 32-bit data width of the AXI4-Lite interface.

Implementing the Reference Design

Click here to download the design files associated with this application note.

The example design was created using Vivado Design Suite 2014.2 and allows you to customize the AXI traffic in either simulation or in hardware. For simulation, the test bench contains an AXI model where the traffic can be modified in the `AXI_model.sv` file. In hardware, the JTAG to AXI IP allows for custom AXI traffic through Tcl in the hardware manager for a KC705 board. Figure 5 shows a block diagram of the design.

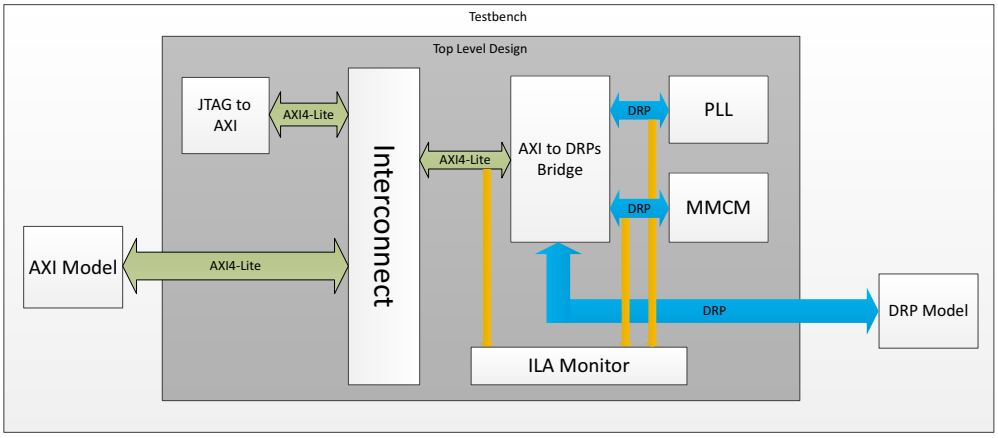

Figure 5: **Block Diagram of Example Design**

To open the example design, source the `runme.tcl` in within the `project` directory, shown in Figure 6.

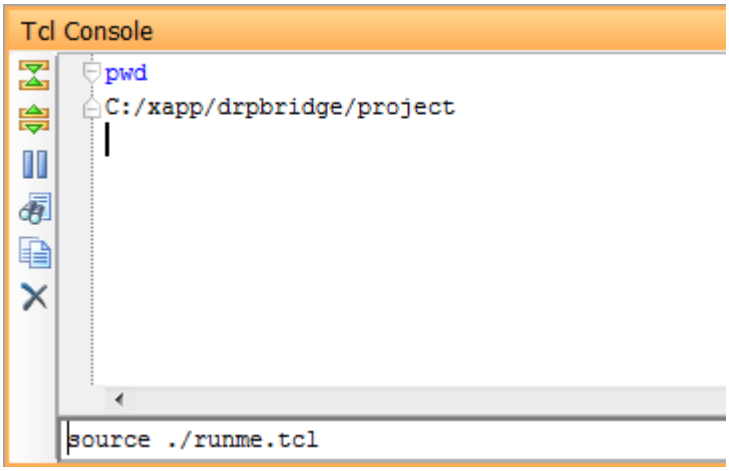

Figure 6: **Sourcing Script to Start Example Design Project**

After sourcing the `runme.tcl` script, a project opens with all of the appropriate sources added to the design. To run the simulation, a simulator supporting SystemVerilog is necessary because the AXI model and the DRP model both use SystemVerilog constructs. Due to the need for SystemVerilog constructs, Vivado Simulator is unable to simulate the design. Instead,

ModelSim or Questa SIM can be used to simulate the design. If a custom test pattern is desired, the traffic from `AXI_model.sv` can be modified in the following area:

```
173 //      memory_write( 32'h00000000, 32'h00001234 );
174 //      memory_write( 32'h00000200, 32'h00005678 );
175       memory_write( 32'h00000400, 32'h00009ABC );
176       memory_write( 32'h00000404, 32'h00001234 );
177       //#(PERIOD*5);
178       memory_read ( 32'h00000000 );
179       memory_read ( 32'h00000200 );
180       memory_read ( 32'h00000400 );
181       memory_read ( 32'h00000404 );
```

Figure 7: **Stimulus Calls for Simulation**

The resulting simulation performs a read to the DRP on PLL and mixed-mode clock manager (MMCM) and performs a write and read on the DRP memory model. Figure 8 show a write transaction.

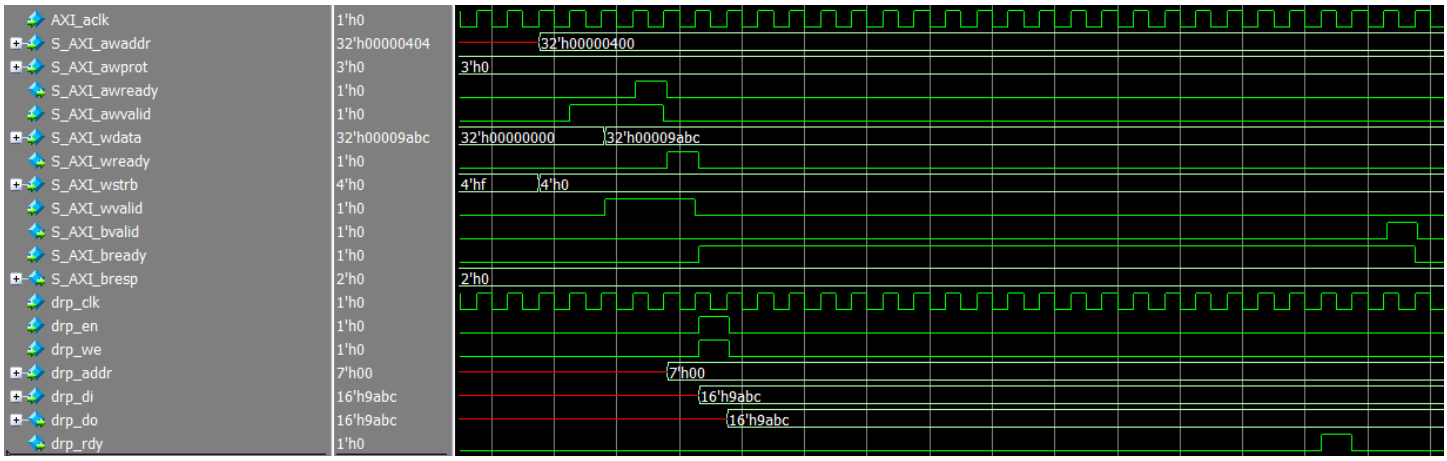

Figure 8: **AXI4 Write Transaction from Address 0x400 to Address 0x0 of the DRP Memory Model**

Figure 9 show a read transaction.

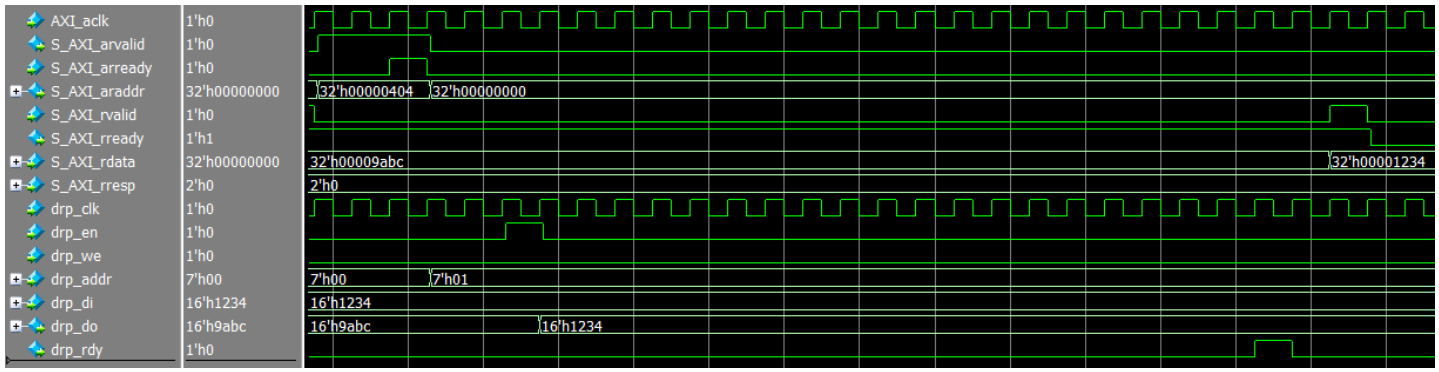

Figure 9: **AXI4 Read Transaction from Address 0x404 to Address 0x1 of the DRP Memory Model**

If a simulator supporting SystemVerilog is not available, proceed to hardware by generating a bit file. After generating the bit file, program the FPGA with the hardware manager and source the `drp_run.tcl` script within the hardware manager Tcl Console. The `drp_run.tcl` must be sourced while in the hardware manager and after the FPGA is programmed. The Tcl script enables the Integrated Logic Analyzer (ILA) core to trigger on a DRP transaction and also

initiate AXI traffic into the system from the JTAG to AXI IP. After the ILA core is triggered, the waveform shows an AXI transaction and the corresponding DRP transaction.

Adding IP to a Custom Design

The AXI4-Lite to DRP bridge is packaged in the `source/packaged_ip` directory. To include this IP into a custom design:

1. Ensure that the packaged IP is in the IP repository.
2. In the **Project Settings** menu, select the packaged_ip directory as shown in Figure 10.
3. Select **OK**.

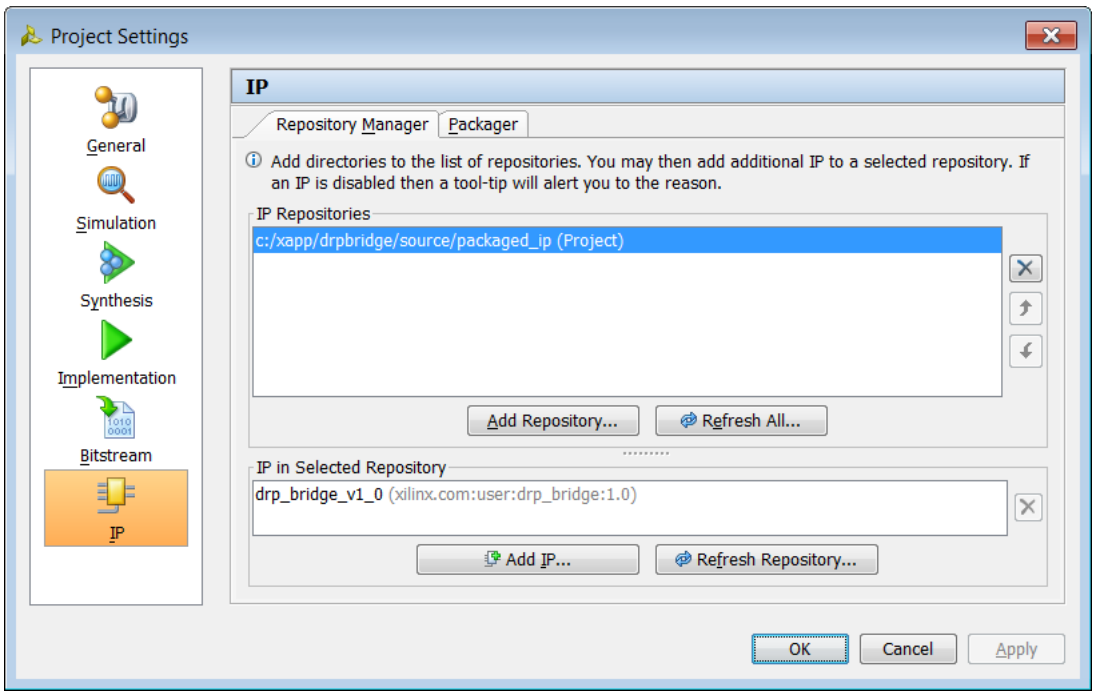

Figure 10: **Adding IP to Repository**

After mapping the repository to the packaged IP, the bridge appears in the Vivado IP catalog, shown in Figure 11.

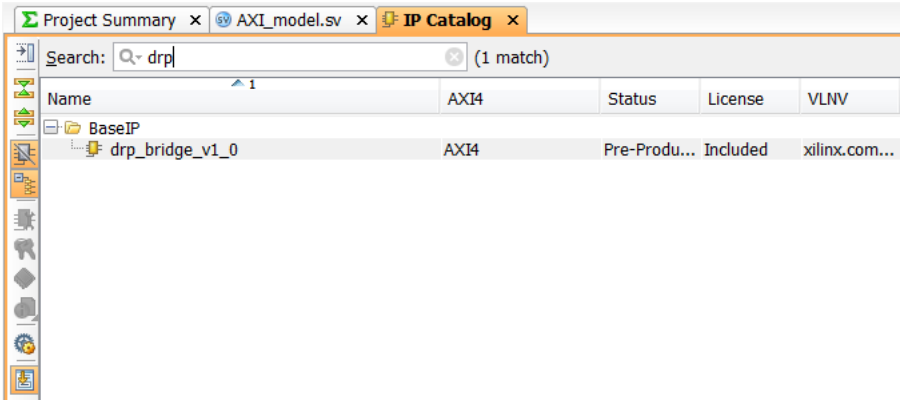

Figure 11: **Viewing the IP within the IP Catalog**

List of Macros with a DRP Interface

The DRP interface is available in the following Xilinx macros:

- MMCM
- PLL
- All Transceivers (GTP, GTH, GTX)
- PCI Express integrated block
- DCM
- XADC

See the corresponding user guide for a register map of the DRP registers.

Resource Utilization

Table 3 describes the resource utilization of the IP under several configurations.

Table 3: **Resource Utilization**

Number of DRP Interfaces	LUTs	FFs
1	42	62
8	90	78
16	142	95
32	268	130

File Structure

Figure 12 shows the directory structure of the reference design:

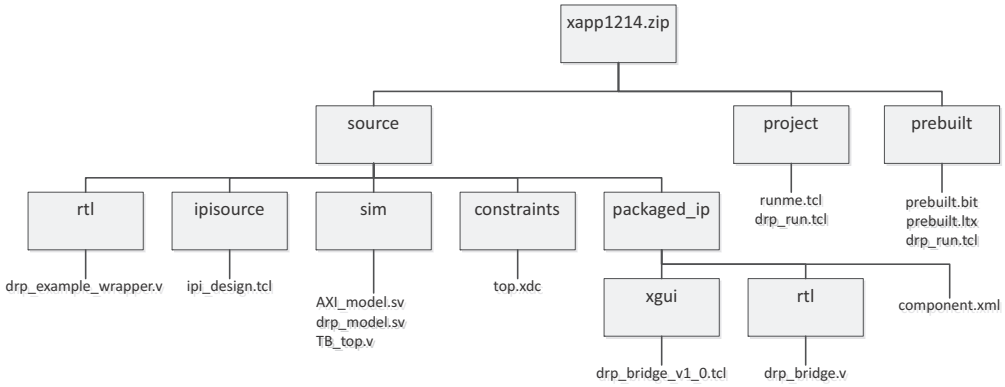

Figure 12: **Design File Hierarchy**

Conclusion

The DRP interface is a commonly used interface to allow reprogramming of internal registers within certain macros. Having an AXI bridge to DRP enables the DRP interface to extend to the plug-and-play infrastructure from AXI peripherals.

Revision History

The following table shows the revision history for this document.

Date	Version	Description of Revisions
06/24/2014	1.0	Initial Xilinx release.

Notice of Disclaimer