AXI VIP Tutorial 1

Introduction

The goal of this tutorial is to demonstrate how to use the Xilinx AXI Verification IP (VIP) to test and verify your custom AXI lite peripherals using the Xilinx Vivado Simulator. We will show you how to connect the VIP to your design and use it to write to and read from registers of the AXI GPIO IP.

Step 1: Create a new project and block diagram

- 1. Create a new project for the **Nexys 4** board.
- 2. Create a new block design and name it **design_1**.
- 3. Add and wire the **AXI Verification IP**, **AXI GPIO**, and **Constant** IPs to the diagram as shown in Figure ?? below. Make sure that the port names match the figure.

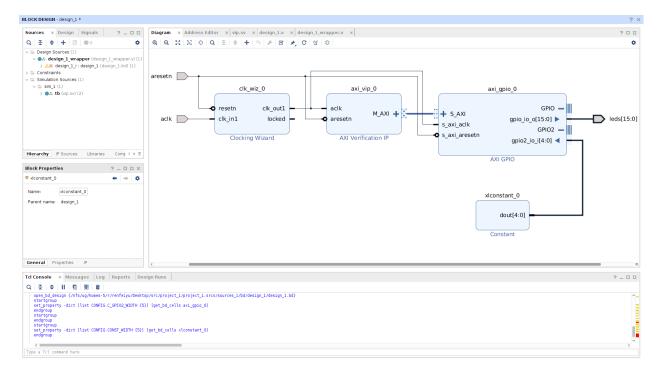


Figure 1: Block design.

4. Configure the **basic settings** tab of the **AXI VIP IP** as shown in the following figure. Leave the advance settings at their default values. This will set the VIP into AXI lite mode.

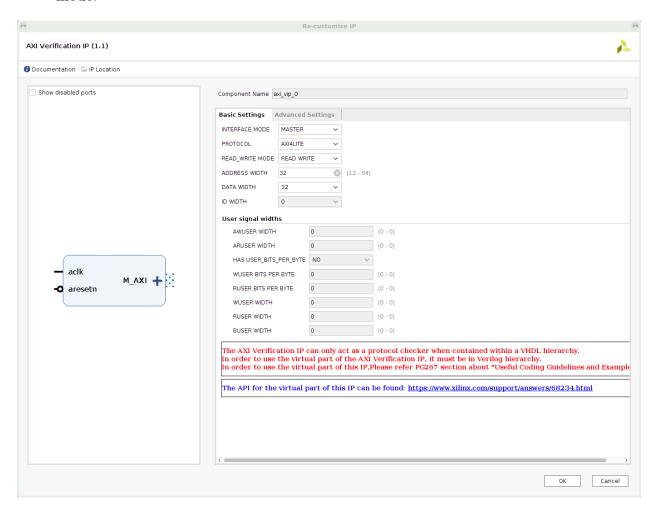


Figure 2: AXI VIP configuration window.

5. Configure the **AXI GPIO IP** as shown in the following figure. We will be using the VIP to set the LEDs and read from the push buttons.

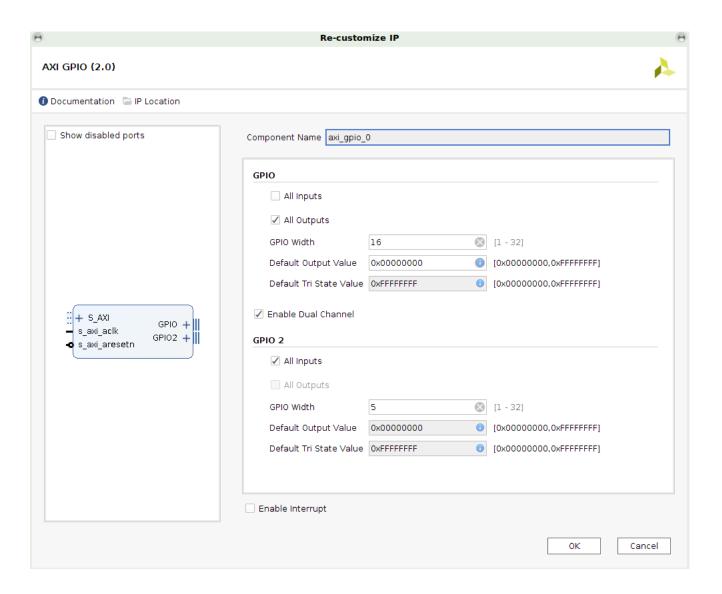


Figure 3: AXI GPIO configuration window.

6. Configure the **Constant** IP as shown in the following figure. The constant value of 5 will simulate pushing buttons 0 and 2.

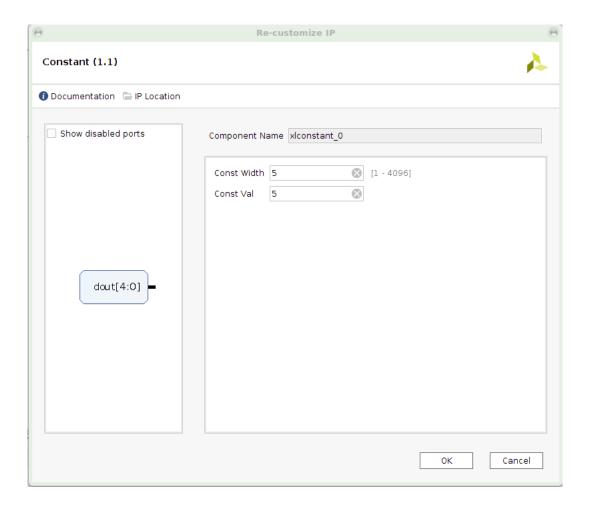


Figure 4: Constant configuration window.

7. Go the address editor tab and set the offset address to 0x40000000.

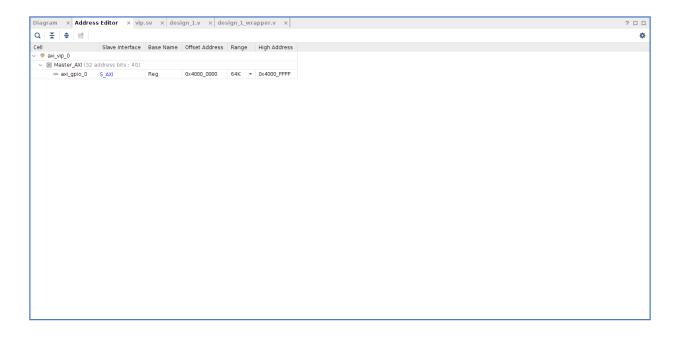


Figure 5: Address Editor

8. Validate the block design, create the HDL wrapper, and generate the block design.

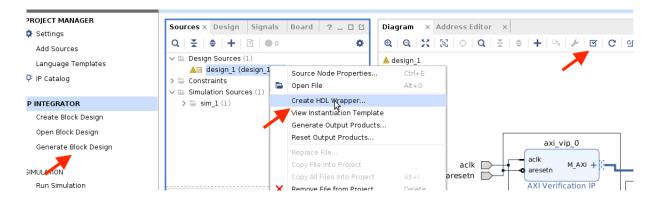


Figure 6: Generating the block design files.

Step 2: Creating the testbench

1. Import **tb.sv** as a **simulation source**.

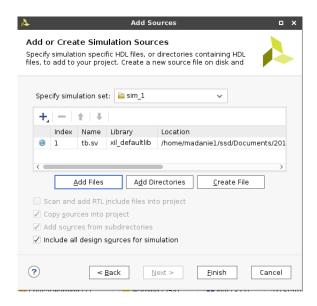


Figure 7: Importing testbench files.

2. Open the tb.sv file in Vivado text editor. Replace the **<vip_component_name>** in line 4 and 15 with the vip component name. Refer to Figure 10 to determine what the component name is for your design.

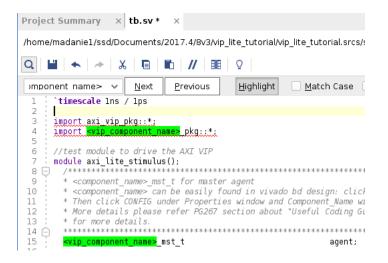


Figure 8: Replace lines 4 and 15 with the appropriate component name for your VIP. **Note:** Syntex checker errors can be ignored, refer to simulation log for errors instead.

```
Project Summary × tb.sv
/home/madanie1/ssd/Documents/2017.4/8v3/vip lite tutorial/vip lite tutorial.srcs/sources 1/imports/new/tb.sv
timescale lns / lps
    import axi vip pkg::*;
    import design l axi vip 0 0 pkg::*;
    //test module to drive the AXI VIP
    module axi_lite_stimulus();
                            <component name> mst t for master agent
      * <component_name> can be easily found in vivado bd design: click on the instance,
     * Then click CONFIG under Properties window and Component_Name will be shown
      * More details please refer PG267 section about "Useful Coding Guidelines and Examples"
      * for more details.
      design_l_axi_vip_0_0_mst_t
                                                 agent:
```

Figure 9: Example of what lines 4 and 15 should look like.

```
vip

design_1_wrapper (design_1_wrapper.v) (1)

design_1_i: design_1_(design_1.bd) (1)

design_1_i: design_1_(design_1.bd) (1)

design_1_i: design_1_(design_1.bd) (1)

design_1_i: desig
```

Figure 10: How to find the component and instance name.

3. Go to line 60, and replace **<bd_instance_name>** and **<vip_instance_name>** with the bd instance name and vip instance name. Refer to Figure 10 to determine what the instance names are for your design.

Figure 11: Replace line 60 with the appropriate instance names for your BD and VIP.

```
Project Summary × tb.sv
 /home/madanie1/ssd/Documents/2017.4/8v3/vip_lite_tutorial/vip_lite_tutorial.srcs/sources
 46 🖯 //Constants
         //AXI GPIO base address and register offsets
        localparam GPIO_BASE_ADDRESS = 'h40000000;
localparam GPIO_DATA_OFFSET = 'h0;
localparam GPIO_TRI_OFFSET = 'h4;
localparam GPIO2_DATA_OFFSET = 'h8;
  48
  49
  51
         localparam GPIO2_TRI_OFFSET = 'hC;
           initial begin
 54
55
                Before agent is newed, user has to run simulation with an empty tes
path of the AXI VIP's instance. Message like
 56
              * "Xilinx AXI VIP Found at Path: my_ip_exdes_tb.DUT.ex_design.axi_vip

* out. Pass this path to the new function.
              agent = new("master vip agent",DUT.design_l_i.axi_vip_0.inst.IF);
 61
              agent.start_master();
```

Figure 12: Example of what line 60 should look like.

4. Set the instance under Simulation Sources as the top instance if it isn't already. The simulation source hierarchy should look like the figure below.

Figure 13: Simulation source hierarchy.

Step 3: Simulate the testbench

1. Before simulation, go to tb.sv and analyze line 66 to 75. This is where our test vectors are located. We will be writing to the output register of the GPIO bank that is connected to the LEDs. Then we'll set the push button GPIOs to be inputs and then wait for all writes to finish. We will then read the GPIO data register for the push buttons, and print the value of that register to the tcl console.

```
//Turn on LED 0,1,3,5,7
writeRegister(GPIO_BASE_ADDRESS + GPIO_DATA_OFFSET, 'hAB);
//Set pushbutton GPIOs to input
writeRegister(GPIO_BASE_ADDRESS + GPIO2_TRI_OFFSET, 'hFFFFFFFF);
//Wait for all writes to complete
agent.wait_drivers_idle();
//Read push button values
readRegister(GPIO2_DATA_OFFSET, Rdatabeat);
//Print read data to tcl console (similar to printf)
swrite('GPIO2_Data_register_value: 0x%08x\n'', Rdatabeat[0]);
```

Figure 14: TB test vectors.

2. Start the simulation by clicking "Run Simulation" on the navigation pane. Add the AXI bus signals to the simulation window and click relaunch simulation. Make sure that you select the VIP instance in the scope window.

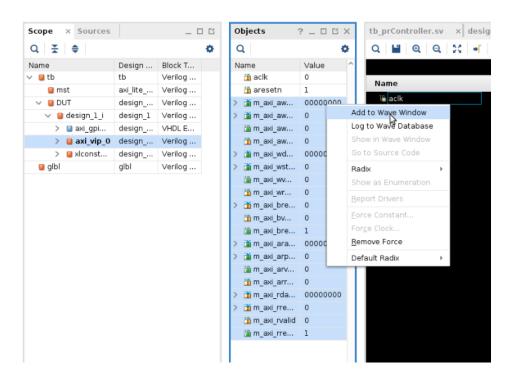


Figure 15: Adding AXI signals to the simulation.



Figure 16: Relaunching the simulation.

3. Analyze the tb results...

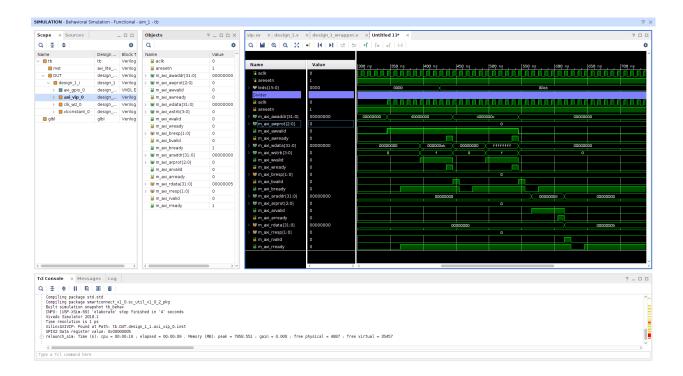


Figure 17: Simulation waveform.

```
Tcl Console × Messages Log

Q | X | $ | II | B | B | II |

run_program: Time (s): cpu = 00:00:13; elapsed = 00:00:06. Memory (MB): peak = 6629.590; gain = 0.000; free physical = 52300; free virtual = 122034

INFO: [USF-XSim-69] 'elaborate' step finished in '7' seconds
launch_simulation: Time (s): cpu = 00:00:13; elapsed = 00:00:06. Memory (MB): peak = 6629.590; gain = 0.000; free physical = 52300; free virtual = 122034

Vivado Simulator 2017.4
Time resolution is 1 ps

XilinxAXIVIP: Found at Path: tb.DUT.design_li.axi_vip_0.inst

EFIO2 Data register value: 0x00000005

Prelaunch_sim: Time (s): cpu = 00:00:20; elapsed = 00:00:13. Memory (MB): peak = 6629.590; gain = 0.000; free physical = 52288; free virtual = 122022

Type a Tcl command here
```

Figure 18: Tcl console print out of register.

Troubleshooting

1. If the Vivado simulator bugs out and starts throwing errors that are not related to your source files, you can try to fix it by reseting your behaviour simulation.

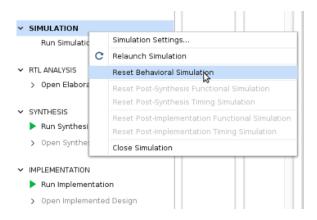


Figure 19: Resetting the simulation.

2. If you want to create more advanced AXI testbenches with the Vivado AXI VIP refer to their example design.

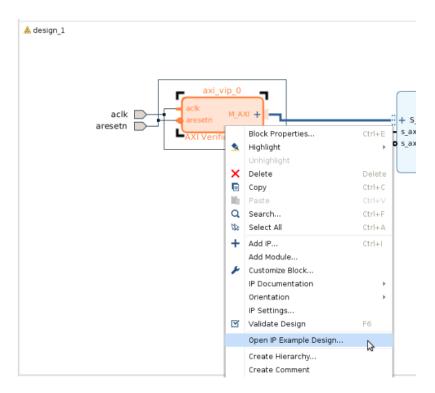


Figure 20: Advanced example design for the AXI VIP.