Adding GPIO to the Rocket Core on the Zedboard

Ashton Johnson & Ian Swepston

May 1, 2017

1 Purpose

This document describes how to add a GPIO peripheral to the RISC-V Rocket core that can be deployed to a Zedboard. For information about the rocket chip, see Rocket-Chip on Github.

2 Assumptions

This document assumes the following items are installed on your computer.

- Java JDK
- Scala Build Tool (SBT) (available via apt)
- Xilinx Vivado 2016.2
- fpga-zynq repository

3 Prerequisites

- Download the fpga-zynq repository by performing the following from the command line:

```
git clone https://github.com/ucb-bar/fpga-zynq
cd fpga-zynq
export TOP=$(pwd)
```

- Initialize the sub-repositories:

```
make init-submodules
```

4 Generate default core (optional)

Generating the default configuration is simple. If everything as been configured correctly, we only require two steps:

- Move to the target board directory

```
cd $TOP/zedboard
```

- Generate the Vivado project

```
make rocket
```

- This process will take less than 10 minutes. At the end of this, you should see a new folder in the folder named `zedboard_rockethchip_ZyncConfig`.

5 Adding AXI Peripherals

+In the next section we add a new AXI peripheral to a the Rocket-Chip.

5.1 Adding Adding GPIO.

Adding a GPIO peripheral will allow us to utilize the buttons, switches and LEDs on the zedboard.

5.1.1 Rocket-chip Configuration

- We begin by creating opening up the generated Xilinx Vivado project that was created earlier. To open the vivado project, perform the following:

```
vivado $TOP/zedboard/zedboard_rocketchip_ZynqConfig.xpr
```
If you cannot open the project, see the Appendix.

- Open the Block design. In the Flow Nagivator on the left, click the "Open Block Design" icon.

Once in the Block Design, we can drive from the TCL Console at the bottom of the screen.

- Add the GPIO to the design:

```
create_bd_cell -type ip -vlnv xilinx.com:ip:axi_gpio:2.0 axi_gpio_0
```

- Configure the GPIO to interface to the LEDs

```
apply_bd_automation -rule xilinx.com:bd_rule:board -config {Board_Interface "leds_8
```

- Update the AXI Interconnect to have two Master ports. This will allow us to connect to the GPIO:

```
set_property -dict [list CONFIG.NUM_MI {2}] [get_bd_cells axi_interconnect_1]
```

- Connect the GPIO to the AXI Interconnect:

```
connect_bd_intf_net [get_bd_intf_pins axi_gpio_0/S_AXI] -boundary_type upper [get_b
```

- Connect the clock to the GPIO:

```
connect_bd_net [get_bd_pins axi_gpio_0/s_axi_aclk] [get_bd_pins axi_interconnect_1/
```

- Connect the reset to the GPIO:

```
connect_bd_net [get_bd_pins axi_gpio_0/s_axi_aresetn] [get_bd_pins proc_sys_reset_0
```

- Connect the clock to the new Master AXI Port:

```
connect_bd_net [get_bd_ports ext_clk_in] [get_bd_pins axi_interconnect_1/M01_ACLK]
```

- Connect the reset to the new Master AXI Port:

```
connect_bd_net [get_bd_pins axi_interconnect_1/M01_ARESETN] [get_bd_pins proc_sys_r
```

- Assign an memory address to the GPIO:

```
assign_bd_address
```

- Clean up the diagram (optional):

```
regenerate_bd_layout
```

- Validate the updated desig. This step should indicate Validation Sucessful:

```
validate_bd_design
```

- Save the new design:

```
save_bd_design
```

- Close out of the Block Diagram:

```
close_bd_design [get_bd_designs system]
```

- Close out of Vivado. We will build the bitstream the traditional way for this project.

5.1.2 Build Bitstream

- Now that the GPIO has been added, we need to build the bitstream that is loaded into the Programmable Logic of the Zynq on the Zedboard. Ensure that you are in the zedboard folder:

```
cd $TOP/zedboard
```

- Build the bitstream

```
vivado -mode tcl
```

- Run the build script. This will take some time.

```
source ./src/tcl/make_bitstream_ZynqConfig.tcl
```

- When the build finishes, exit out of Vivado:

```
exit
```

5.1.3 Update Boot Image

- To load in the new configuration, a new `boot.bin` file will need to be created. Here we also need to specify out new configuration:

```
make fpga-images-zedboard/boot.bin
```

5.1.4 Upload Boot Image

- Now you just need to upload the boot image to the SD card:

```
make load-sd SD=path_to_mounted_sdcard
```

6 Conclusion

You should have successfully added a GPIO peripheral to the design, and loaded the new design onto the FPGA with this tutorial. To access this GPIO, read and write to address location We encourage you to look around the .scala files both in the fpga-zynq folders, and down int eh rocket-chip folders. Specifically, these are located in these two locations:
`/fpga-zynq/common/src/main/scala/`
`/fpga-zynq/rocket-chip/src/main/scala/`

7 Appendix

7.1 Add Vivado to PATH

If Vivado is not in your path, you will need to perform the following:

- Locate the Vivado installation. A good place to look is `/opt/Xillinx/`
- Add the following line to your `@~/.bashrc` file

```
PATH=$PATH:/opt/Xilinx/Vivado/2016.2/bin
```