EGR680 High Level Implementation on FPGA

Laboratory 04

Embedded System Design on PYNQ - Hardcore Processors

Professor: Dr. C. Parikh

Student: Dimitri Häring

September 25, 2018

Contents

1 Introduction

The goal of the lab 3 is to familiarize the student with a finite state machine implementation in verilog.

2 Design

In this section the design and decisions that where made to achieve the laboratory are discussed.

2.1 SDK Lab specification

In this part, you will create a simple hardcore ARM Cortex-A9 based embedded system on the PYNQ board. The embedded system design is broken up into three parts: Hardware design of the ARM Cortex-A9 hardcore processor, application software design using SDK, and finally hardware implementation of the software running on the hardcore processor. The application you will design in this part is a UART application that prints "HELLO WORLD" to a terminal emulator like Tera Term. Use the figure below as a flow guide for this lab. The diagram for the completed design is shown in Figure 1.

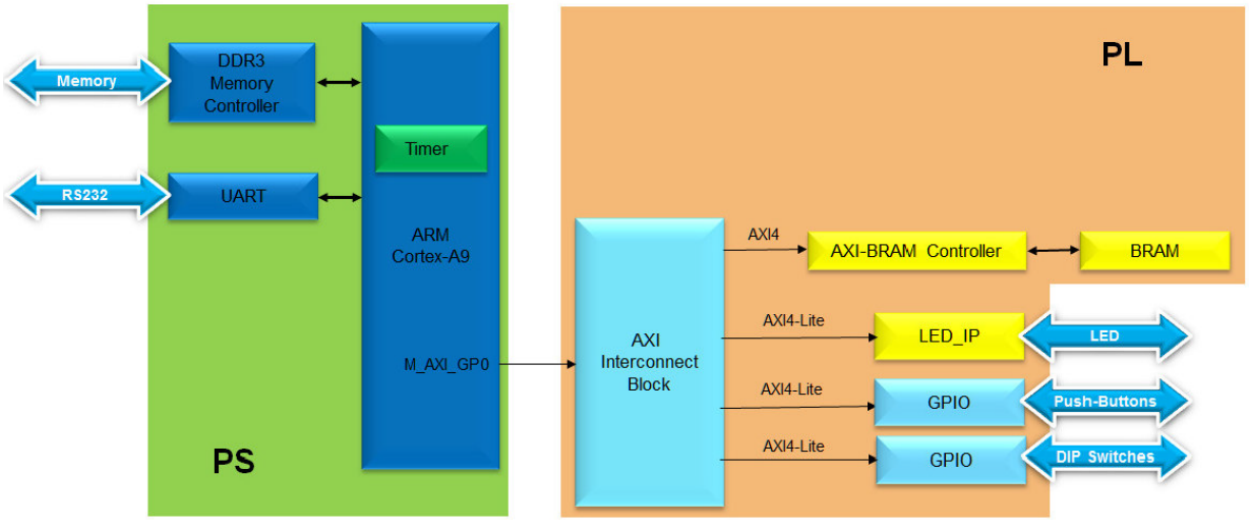

Figure 1: Completed Design.

2.2 HDL

Figure 2 shows the HDL top level based of the ZYNQ 7 Processing system which is connected with an AXI bus S00_AXI to a intellectual property (IP) block that manages peripherals. From there an AXI bus is used to connect two general purpose input output (GPIO) IP blocks, one for buttons and another one for switches. Furthermore, a Processor System Reset IP block is used that interconnects all resets.

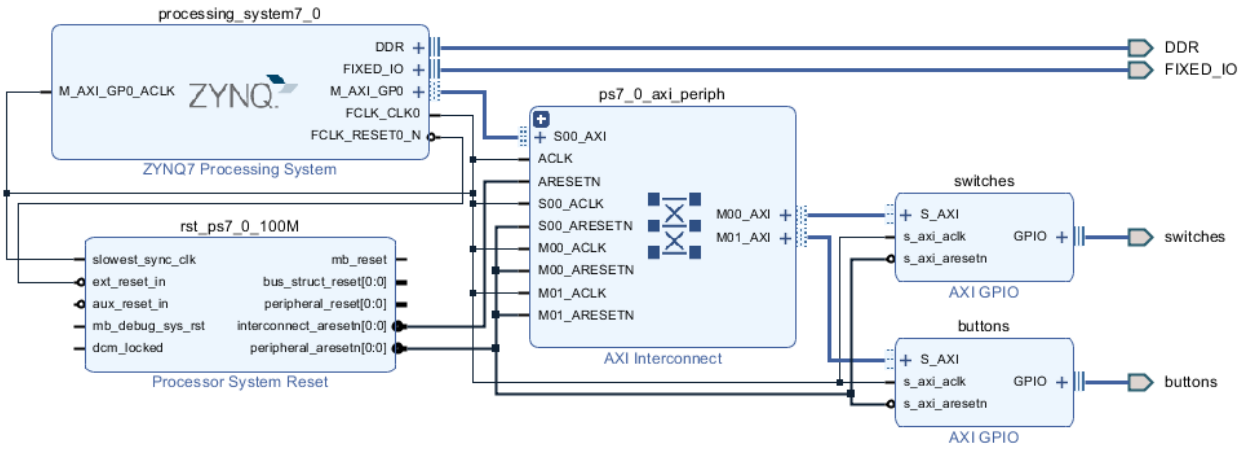

Figure 2: HDL Top Level Design.

2.3 SDK

After the HDL top level is defined the SDK can be launched with **File** → **Launch SDK**. The *.hdf file should be shown or can be opened that shows the base registers for the switches and buttons, as highlighted in Figure 3.

ps7_ocmc_0	0xf800c000	0xf800cfff		REGISTER
ps7_pl310_0	0xf8f02000	0xf8f02fff		REGISTER
ps7_coresight_comp_0	0xf8800000	0xf88fffff		REGISTER
ps7_uart_0	0xe0000000	0xe0000fff		REGISTER
ps7_scugic_0	0xf8f00100	0xf8f001ff		REGISTER
switches	0x41200000	0x4120ffff	S_AXI	REGISTER
ps7_dev_cfg_0	0xf8007000	0xf80070ff		REGISTER
ps7_dma_ns	0xf8004000	0xf8004fff		REGISTER
ps7_gpv_0	0xf8900000	0xf89fffff		REGISTER
ps7_ram_1	0xffff0000	0xfffffdff		MEMORY
ps7_ram_0	0x00000000	0x0002ffff		MEMORY
buttons	0x41210000	0x4121ffff	S_AXI	REGISTER

Figure 3: *.hdf file that shows the base register for switches and buttons.

After writing the C code goven in part two and loading the the bit stream file and the build elf file onto the board the seriel co nsol terra term shows the status of the buttons and sitches as shown in Figure 4

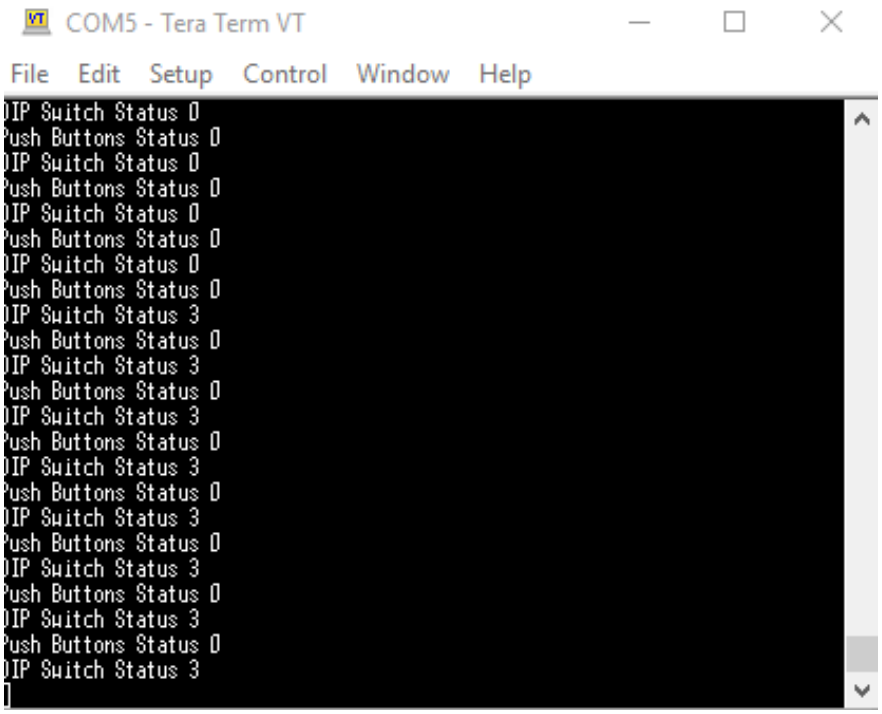

Figure 4: Terra Term print out of the switches and buttons status.

2.4 Errors

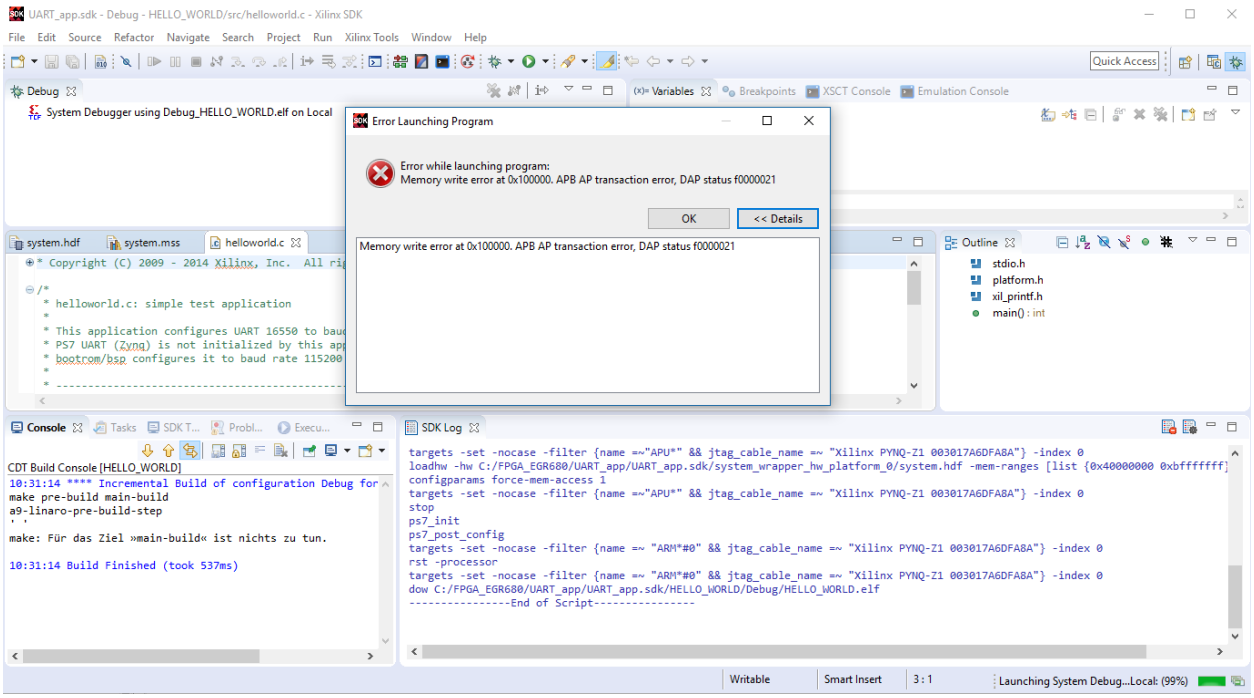

Figure 5: Error SDK Part I.

3 Simulation

4 Conclusion

5 Appendix

The appendix contains code listening and other large information parts that contain partial or complete relevance to the reports topic.

5.1 Lesson Learned

5.1.1 Vivado Language Templates

Figure 6 shows a handy tool build into Vivado software package that is called Language Templates and it seems to contain most of the verilog syntax.

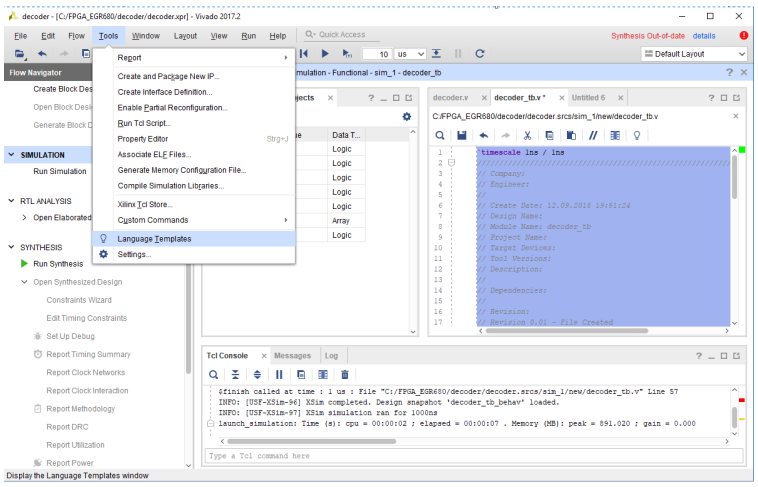

Figure 6: Vivado Language Templates.

5.1.2 Clock divider not working

Listening 1 shows a code snipe out of the clock divider that caused a lot of trouble by preventing the clock divider preventing dividing the clock. It seems that the last statement $clk_out <= clk_out$; overwrote the statement $clk_out <= clk_out$; which results either in an unknown state or an zero on the divided clock instead of the desired toggling clock behavior.

Listing 1: Clock divider issue.

```
else
begin

clk_temp <= clk_temp + 1;
if (clk_temp >= 500000)
//if (clk_temp >= 2) // Used for testbench
begin
clk_out <= ~clk_out; // no clue why this line does not toggle the clk_out
//    if (clk_out == 0)
//    begin
//    clk_out=1;
//    end
//    else
//    begin
//    clk_out=0;
//    end
```

```
clk_temp <= 0;
end
end // else rst
//clk_out <= clk_out; // could this lane of HDL be causong the problem of the not work
```

5.1.3 Implementation Error [Place 30-574] Poor placement for routing between an I/O pin and BUFG

Poor placement for routing between an I/O pin and BUFG is a state where a clk signal is routed to anon clock net or a non clock signal is routed to clock net like a posedge or negedge clk signal. Now it seems this clock placement error that occurs by implementing the project is somewhat related to the decoder. As the decoder module is commented out of the top level block the error is not there any more. Still the source or what pin shall causing the issue could not be located at first. The issue was that the in the decoder initialization the clk and rst pin where switched.