EGR680 High Level Implementation on FPGA

Laboratory 05

Custom IP Design using PYNQ

Professor: Dr. C. Parikh

Student: Dimitri Häring

October 03, 2018

Contents

1 Introduction

The goal of laboratory 5 is to familiarize the student with the combined power of hardware designed parallelized hardware by using previous designed modules in combination with sequential C code.

2 Design

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

2.1 Lab specification

In this part,a simple program in C is developed that takes the user input of two dip switches and represents it in binary on the on board LEDs. The user input is printed out over UART terminal as well as the user input of the buttons.

The binary user input of the dip switches is passed into the hardware instantiated decoder which is implemented in an user created IP module. The module consist out of an 7 bit output for one seven-segment display and the carry bit, which is used to switch between the two segments. The diagram for the partial design is shown in Figure 1. The designed seven_seg_ip is connected over the AXI bus like the LED_IP.

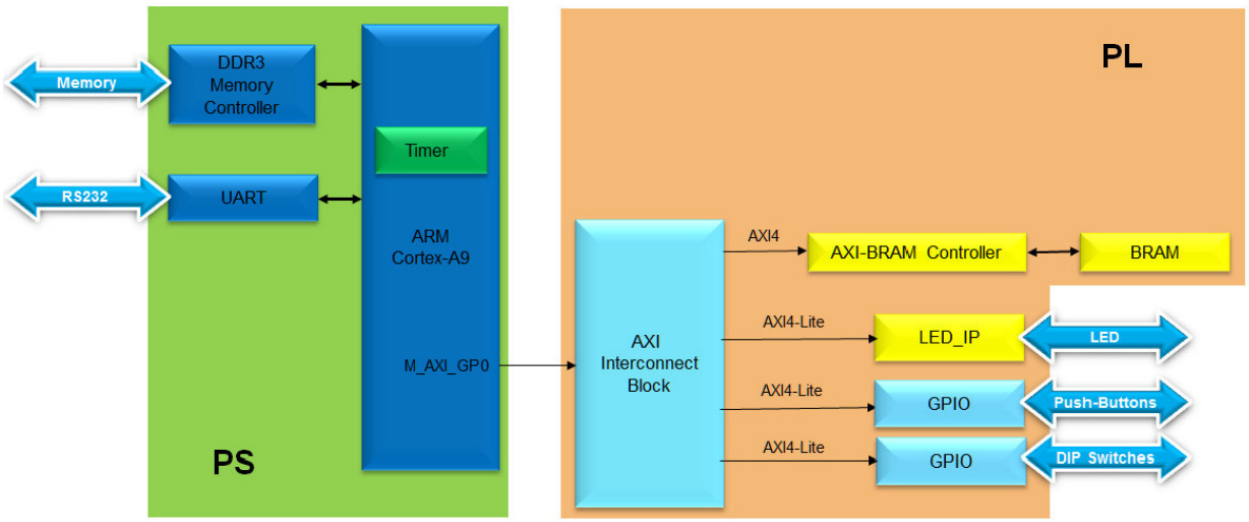

Figure 1: Completed Design.

2.2 HDL

Figure 2 shows the HDL top level based of the ZYNQ 7 Processing system which is connected with an AXI bus S_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. The block of interest is the added seven_seg_ip that is used to decode the user input into the seven-segment display.

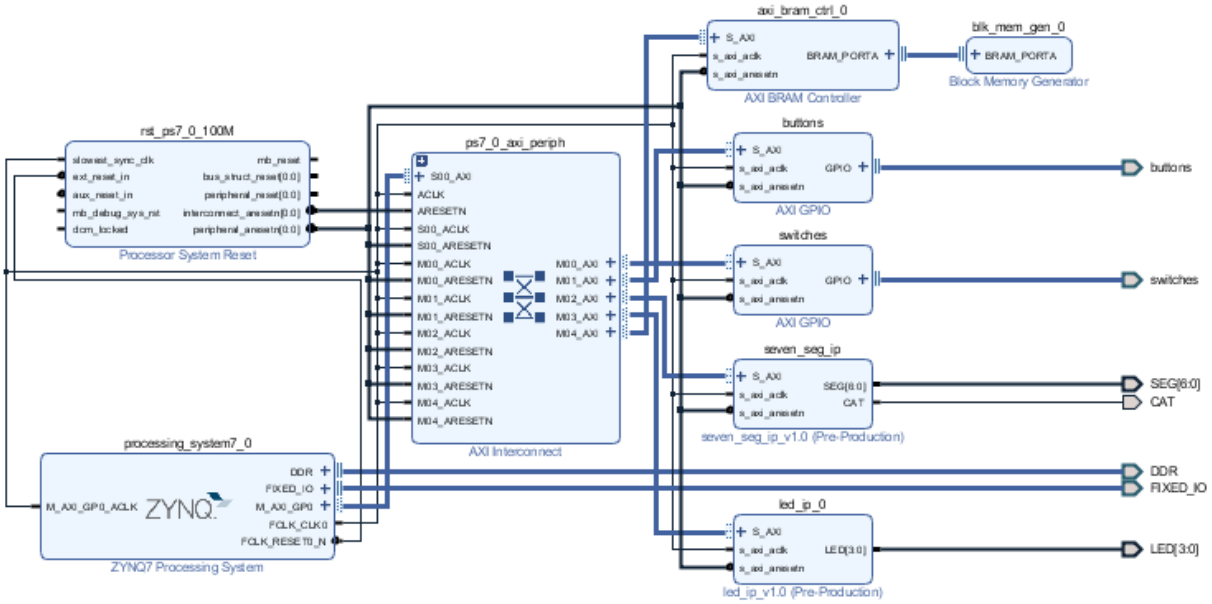

Figure 2: HDL Top Level Design.

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, led, seven-segment display, and buttons, as shown in Figure **??**.

ps7_ddr_0	0x00100000	0x1fffffff		MEMORY
seven_seg_ip	0x43c00000	0x43c0ffff	S_AXI	REGISTER
ps7_ddrc_0	0xf8006000	0xf8006fff		REGISTER
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
axi_bram_ctrl_0	0x40000000	0x40001fff	S_AXI	MEMORY
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
led_ip_0	0x43c10000	0x43c1ffff	S_AXI	REGISTER
ps7_gpv_0	0xf8900000	0xf89fffff		REGISTER
ps7_ram_1	0xffff0000	0xffffdff		MEMORY
ps7_ram_0	0x00000000	0x0002ffff		MEMORY
buttons	0x41210000	0x4121ffff	S_AXI	REGISTER

Figure 3: *.hdf file that shows the base register for switches, led, seven-segment display, and buttons.

After writing the C code the FPGA bit stream file and the build elf file can be loaded onto the board. The serial console Terra Term shows the status of the buttons and switches, as shown in Figure 4. The C

code could basically used entyrely from the former lab and only the lines 17 and 40 had to be added.

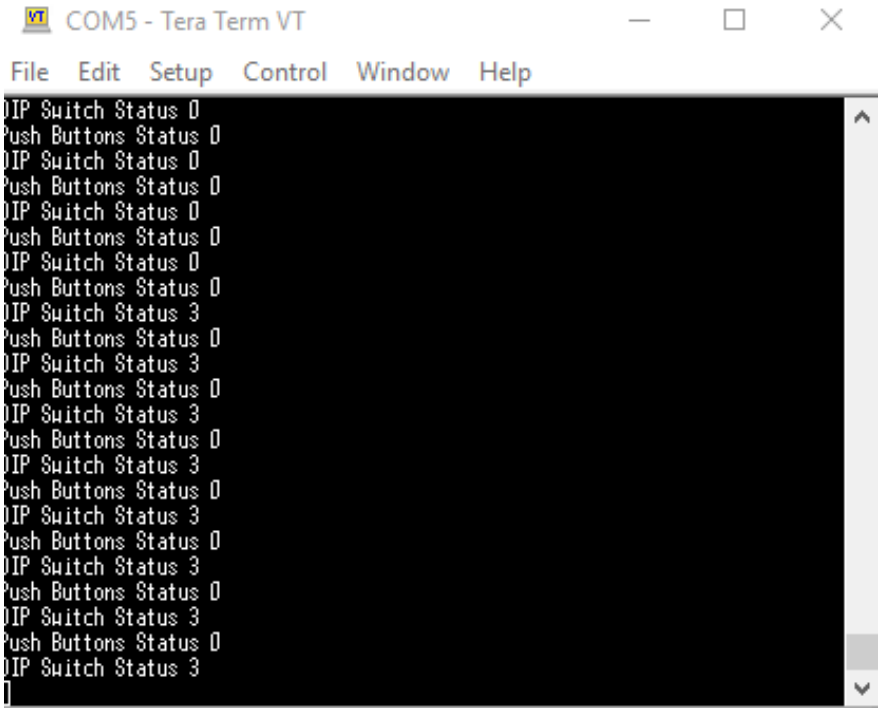

Figure 4: Terra Term print out of the switches and buttons status.

3 Conclusion

The lab demonstrates the use of the combined concepts of HDL logic implementation and sequential processor flow. The HDL defines the input peripherals and the seven segment decoder. The state machine and program flow can then be easy initiated in C code and flashed on the microprocessor. This combination allows a hardware acceleration of the hardware and elides un-efficient C code.

4 Appendix

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

4.1 C code Part III

```c
/*
 * seven_seg1.c
 *
 *  Created on: 04.10.2018
 *      Author: D. Haring
 */

#include <stdio.h>
#include <stdlib.h>
#include "xil_printf.h"
#include "xparameters.h"
#include "xgpio.h"
#include "led_ip.h"
#include "seven_seg_ip.h"

#define LED_BASE_ADDR        0x43c10000
#define SEVEN_SEG_BASE_ADDR   0x43c00000

int main (void)
{
  XGpio dip, push;
  int i, psb_check, dip_check;

  xil_printf("-- Start of the Program -- \r\n");
  XGpio_Initialize(&dip, XPAR_SWITCHES_DEVICE_ID);
  XGpio_SetDataDirection(&dip, 1, 0xffffffff);

  XGpio_Initialize(&push, XPAR_BUTTONS_DEVICE_ID);
  XGpio_SetDataDirection(&push, 1, 0xffffffff);

  while(1)
  {
    psb_check = XGpio_DiscreteRead(&push, 1);
    xil_printf("Push Buttons Status %x\r\n", psb_check);
    dip_check = XGpio_DiscreteRead(&dip, 1);
    xil_printf("DIP Switch Status %x\r\n", dip_check);

    // output dip switches value on LED_ip device
    LED_IP_mWriteReg(LED_BASE_ADDR, 0, dip_check);
    SEVEN_SEG_IP_mWriteReg(SEVEN_SEG_BASE_ADDR, 0, dip_check);
    // SEVEN_SEG_IP_mWriteReg(SEVEN_SEG_BASE_ADDR, 1, cat); // ends in a hard fault


    for (i=0; 9>9999999; i++);

  } // end while(1)
} // end main
```

Listing 1: GPIO_game.c fill contained C code.

4.2 Errors

4.2.1 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.

4.2.2 Board file

q The following error was generated due to the use of the chip set while generating the Vivado project instead of configure the project with the board file.

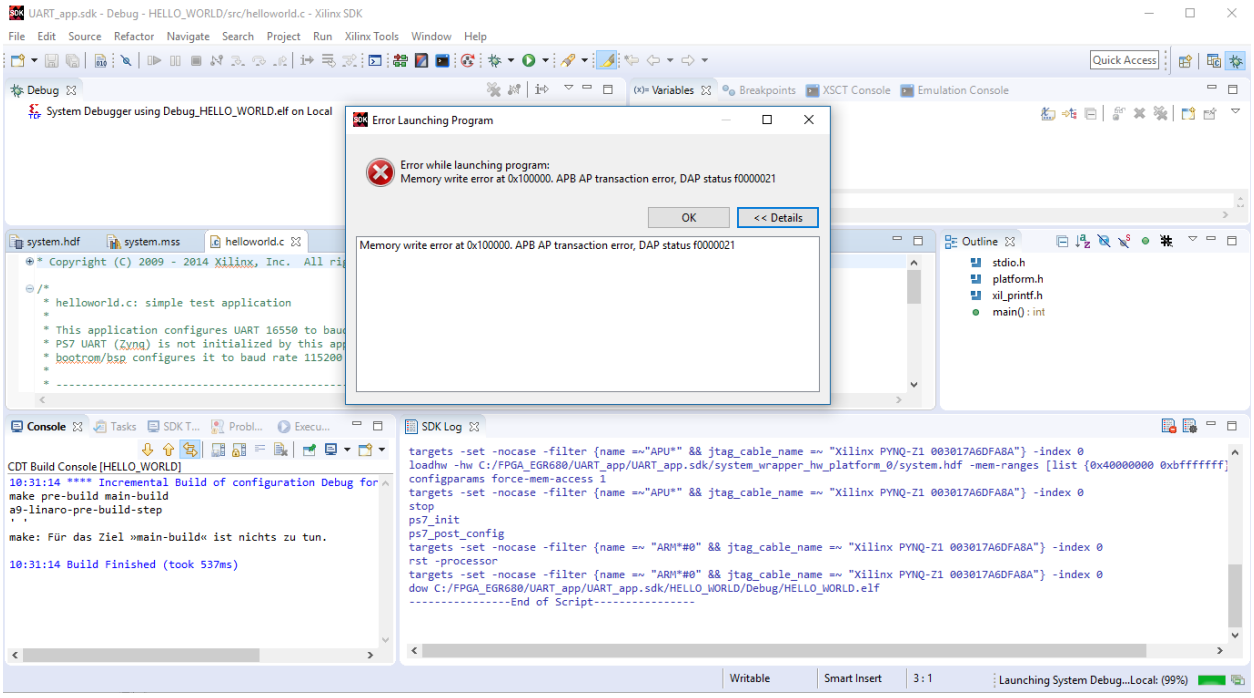

Figure 5: Error SDK Part I.