Lab 2: Using the SDK

Abhinav Subramanian

ECEN 449-502

Due date: 2/4/2020

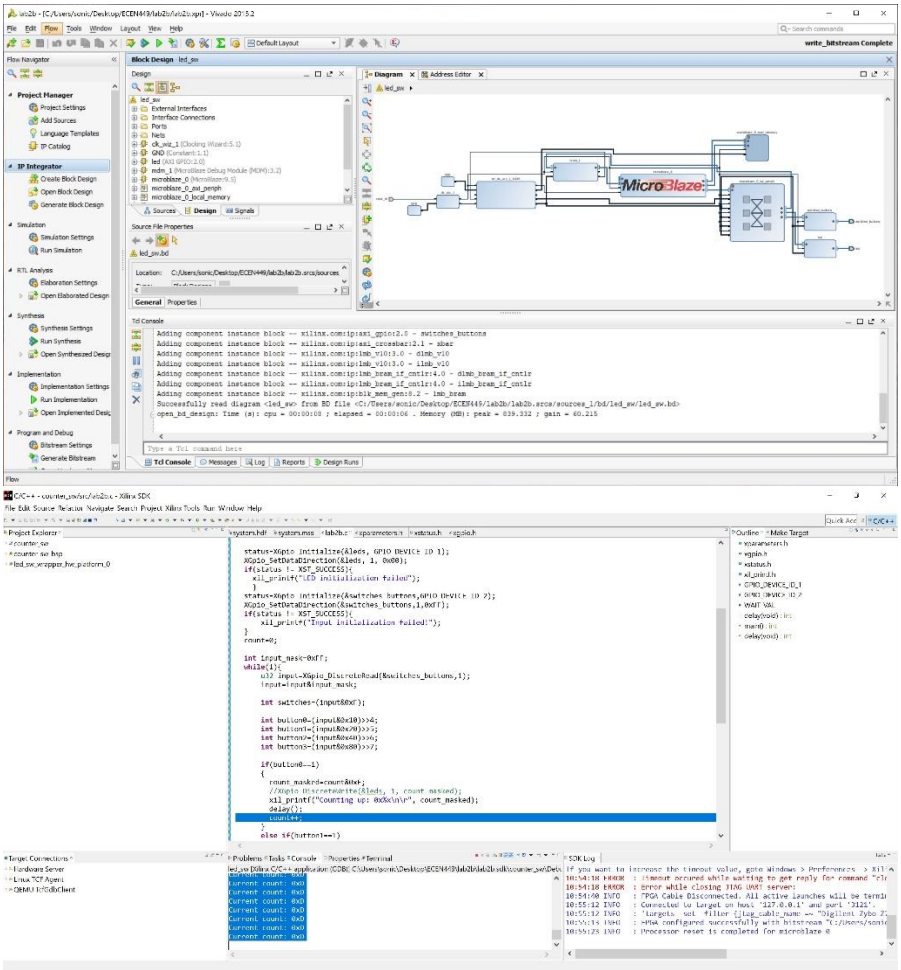
Introduction:

The aim of this lab was to learn how to configure the Zybo's hardware to run software than can control it in much the same way as pure hardware like Verilog can. We also compared its implementation with the latter and evaluated its advantages and disadvantages.

Procedure:

- Open Vivado and create a project new project, but don't add any new sources.
- Create a new block diagram and place the microblaze on the diagram. Automate the new design according to the appropriate configurations.
- Configure the clock appropriately and reautomate the connections.
- Now add the LED GPIO so we can use the LEDs.
- Replace the reset pins with VDD and GND.
- Create an XDC file to link the IO to the pins on the zybo.
- Generate the bitstream and export the hardware.
- Launch the SDK and create a new project. Copy the code from the lab manual into a file called lab2a.c and add it as a source file.
- Program the zybo and run the file.
- Repeat this process with the code in the appendix.

Results:



TCL Console:

Counting up: 0x0

Counting up: 0x1

Counting up: 0x2

Counting up: 0x3

Counting up: 0x4

Counting up: 0x5

Counting up: 0x6

Counting up: 0x7

Counting up: 0x8

Counting up: 0x9

Counting up: 0xA

Counting up: 0xB

Counting up: 0xC

Counting down: 0xD

Counting down: 0xC

Counting down: 0xB

Counting down: 0xA

Counting down: 0x9

Counting down: 0x8

Counting down: 0x7

Counting down: 0x6

Counting down: 0x5

Counting down: 0x4

Counting down: 0x3

Counting down: 0x2

Counting down: 0x1

Counting down: 0x0

Counting down: 0xF

Counting down: 0xE

Switches: 0x0

Switches: 0x0

Switches: 0x0

Switches: 0x0

Switches: 0x0

Switches: 0x0

Switches: 0x0

Switches: 0x0

Switches: 0x0

Switches: 0x1

Switches: 0x1

Switches: 0x1

Switches: 0x1

Switches: 0x1

Switches: 0x1

Switches: 0x1

Switches: 0x1

Switches: 0x1

Switches: 0x1

Switches: 0x1

Switches: 0x1

Switches: 0x3

Switches: 0x3

Switches: 0x3

Switches: 0x3

Switches: 0x3

Switches: 0x3

Switches: 0x3

Switches: 0x3

Switches: 0x3

Switches: 0x3

Switches: 0x3

Switches: 0x3

Switches: 0x3

Switches: 0x3

Switches: 0x3

Switches: 0x3

Switches: 0x3

Switches: 0x3

Switches: 0x3

Switches: 0x3

Switches: 0x3

Switches: 0x1

Switches: 0x0

Switches: 0x0

Switches: 0x0

Switches: 0x8

Switches: 0x8

Switches: 0x8

Switches: 0x8

Switches: 0x8

Switches: 0x8

Switches: 0x8

Switches: 0x8

Switches: 0x8

Switches: 0x8

Switches: 0x8

Switches: 0x8

Switches: 0x8

Switches: 0x8

Switches: 0x8

Switches: 0x8

Switches: 0x8

Switches: 0x0

Switches: 0x0

Switches: 0x0

Switches: 0x0

Switches: 0x0

Switches: 0x0

Switches: 0x0

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Current count: 0xD

Conclusion:

In this lab, I learned that it's possible to run software on the ZYBO microprocessor. Although it requires a little extra work setting it up, it's able to implement much the same functionality as Verilog. In fact, it's perhaps more efficient.

Postlab questions:

a) The count value I used in the previous lab (to divide the clock to 1Hz) was 62.5 million, whereas the one in this lab was about 10 million. It's possible that the Zybo's onboard clock is faster than the Microblaze's. By this logic, it should take approximately 6 clock cycles to execute 1 for loop.

b) The volatile designation indicates that the variable could change unexpectedly. This could happen to the count if, say, the program were reset.
c) The while(1) expression makes the program run infinitely.
d) I feel that for this specific case, Verilog is easier, as it's easier to think about the variables in terms of bits. However, for complicated programs, it might be easier to implement a software algorithm. Also, it's probably easier to debug the latter, as you can print output to the console. The downside of the software is it takes a long time to set up.

Appendix:

Lab02b.c

```
#include <xparameters.h>
#include <xgpio.h>
#include <xstatus.h>
#include <xil_printf.h>
/*definitions*/
#define GPIO_DEVICE_ID_1 XPAR_LED_DEVICE_ID //GPIO Device LEDs are connected to.
#define GPIO_DEVICE_ID_2 XPAR_SWITCHES_BUTTONS_DEVICE_ID
#define WAIT_VAL 10000000

int delay (void);

int main()
{
 int count;
 int count_masked;
 XGpio leds;
 XGpio switches_buttons;
 int status;

 status=XGpio_Initialize(&leds, GPIO_DEVICE_ID_1);
 XGpio_SetDataDirection(&leds, 1, 0x00);
 if(status != XST_SUCCESS){
  xil_printf("LED initialization failed");
        }
 status=XGpio_Initialize(&switches_buttons,GPIO_DEVICE_ID_2);
```

```
XGpio_SetDataDirection(&switches_buttons,1,0xFF);
if(status != XST_SUCCESS){
		xil_printf("Input initialization failed!");
}
count=0;

int input_mask=0xFF;
while(1){
		u32 input=XGpio_DiscreteRead(&switches_buttons,1);
		input=input&input_mask;

		int switches=(input&0xF);

		int button0=(input&0x10)>>4;
		int button1=(input&0x20)>>5;
		int button2=(input&0x40)>>6;
		int button3=(input&0x80)>>7;

		if(button0==1)
		{
			count_masked=count&0xF;
			//XGpio_DiscreteWrite(&leds, 1, count_masked);
			xil_printf("Counting up: 0x%x\n\r", count_masked);
			delay();
			count++;
		}
		else if(button1==1)
		{
			count_masked=count&0xF;
			//XGpio_DiscreteWrite(&leds, 1, count_masked);
			xil_printf("Counting down: 0x%x\n\r", count_masked);
			delay();
			count--;
		}
		else if(button2==1){
```

```
                xil_printf("Switches: 0x%x\n\r", switches);
                XGpio_DiscreteWrite(&leds, 1, switches);
        }
        if(button3==1){
                count_masked=count&0xF;
                xil_printf("Current count: 0x%x\n\r", count_masked);
                XGpio_DiscreteWrite(&leds,1,count_masked);
        }
        else{
                XGpio_DiscreteWrite(&leds,1,0x0);
        }

 }
 return(0);
}

int delay(void)
{
  volatile int delay_count=0;
  while(delay_count<WAIT_VAL)
    delay_count++;
  return (0);
}
```

XDC:

```
#clock_rtl
set_property PACKAGE_PIN K17 [get_ports clock_rtl]
set_property IOSTANDARD LVCMOS33 [get_ports clock_rtl]
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clock_rtl]

#led_tri_0
set_property PACKAGE_PIN M14 [get_ports {led_tri_o[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led_tri_o[0]}]

set_property PACKAGE_PIN M15 [get_ports {led_tri_o[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led_tri_o[1]}]
```

```
set_property PACKAGE_PIN G14 [get_ports {led_tri_o[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led_tri_o[2]}]

set_property PACKAGE_PIN D18 [get_ports {led_tri_o[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led_tri_o[3]}]

#switches_buttons_tri_i

set_property PACKAGE_PIN G15 [get_ports {switches_buttons_tri_i[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {switches_buttons_tri_i[0]}]

set_property PACKAGE_PIN P15 [get_ports {switches_buttons_tri_i[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {switches_buttons_tri_i[1]}]

set_property PACKAGE_PIN W13 [get_ports {switches_buttons_tri_i[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {switches_buttons_tri_i[2]}]

set_property PACKAGE_PIN T16 [get_ports {switches_buttons_tri_i[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {switches_buttons_tri_i[3]}]

set_property PACKAGE_PIN K18 [get_ports {switches_buttons_tri_i[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {switches_buttons_tri_i[4]}]

set_property PACKAGE_PIN P16 [get_ports {switches_buttons_tri_i[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {switches_buttons_tri_i[5]}]

set_property PACKAGE_PIN K19 [get_ports {switches_buttons_tri_i[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {switches_buttons_tri_i[6]}]

set_property PACKAGE_PIN Y16 [get_ports {switches_buttons_tri_i[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {switches_buttons_tri_i[7]}]
```