ECEN 749 LAB REPORT

Excercise #2: Using Software Development Kit(SDK)

Tanmay Verma

09.21.2017

INTRODUCTION

The purpose of this lab exercise was to get acquainted with the Xilinx SDK. The IDE was used to create and program Software on a bare metal Xilinx Microblaze Soft-core processor. The FPGA was used to implement a Microblaze based System-on-chip and interact with the peripherals.

PROCEDURE

The following steps were carried out:

1. A new block design was created in the project.
2. Microblaze IP, GPIO unit and two constant producing IPs were added in the design.
3. The resets pin were permanently disabled in the block design.
4. GPIO was configured with 4 width all output bus.
5. The constraint file was added.
6. The H/W was wrapped under HDL.
7. The bitstream was generated and exported to SDK.
8. The C program was imported to the project in SDK.
9. Program FPGA was initiated.
10. Run configuration was modified to include the elf application and change to JTAG UART connection and then launched.
11. Device operation was verified.
12. A second project was opened and this time another GPIO unit was included in the block design. This block had 8-bits wide all input bus for four switches and four push buttons.
13. Steps 6-11 were repeated for this new block design with an updated C program.

RESULTS

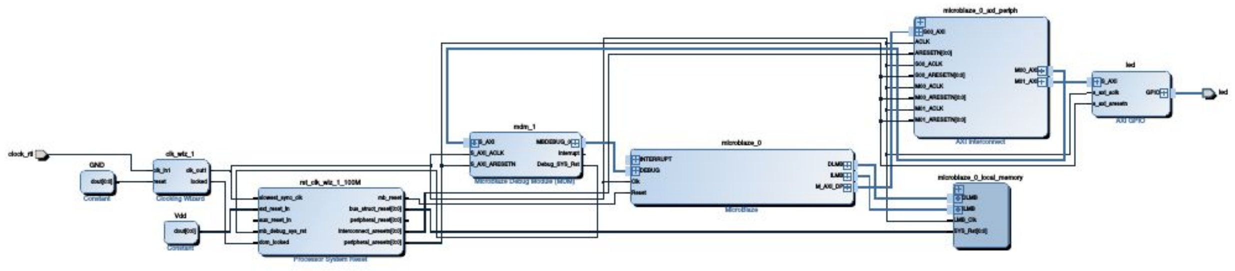
The two parts of this exercise were highly linked as the latter one only required an additional all input 8-bit GPIO unit over the former. The program which was run on SoC comprised of simple conditional statements controlling the response as per the given requirements.

CONCLUSION

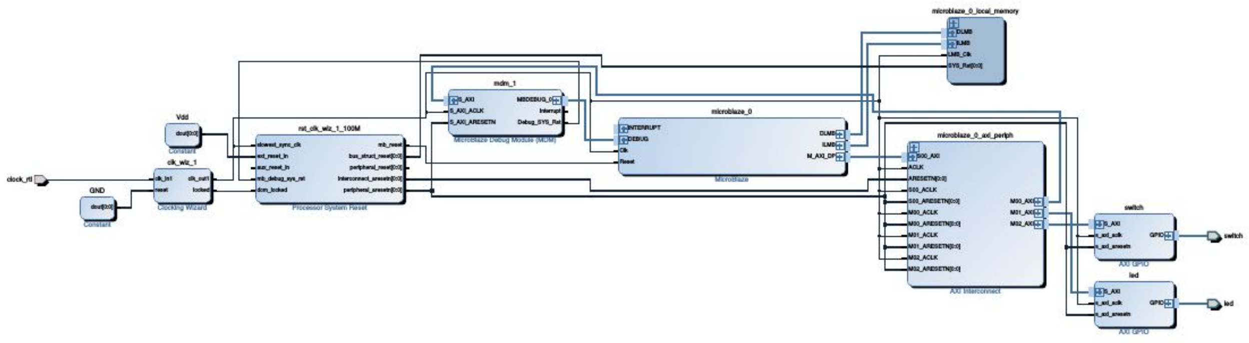
The exercise explored how FPGA can be programmed to behave as a System-on-Chip with Microblaze IP cores and then develop and run a C program on the SoC using Software Development Kit.

Design

Part 1:



Part 2:



C codes

Part 1.

```
#include <xparameters.h>

#include <xgpio.h>

#include <xstatus.h>

#include <xil_printf.h>


/* Definitions */


#define GPIO_DEVICE_ID  XPAR_LED_DEVICE_ID // present in header value

#define WAIT_VAL                10000000


int delay(void);


int main()

{

        int count;

        int count_masked;

        XGpio leds;

        int status;


        status=XGpio_Initialize(&leds,GPIO_DEVICE_ID); //initializes GPIO module to output value

        XGpio_SetDataDirection(&leds,1,0x00); //sets direction to output for the GPIO module

        if (status != XST_SUCCESS) {

        xil_printf("Initialization Failed |n" );

        }


        count = 0;


        while(1) {

        count_masked = count & 0xF; //extracts 4 bits for the 4 LED values
        XGpio_DiscreteWrite(&leds,1,count_masked); //used to write value on the GPIO port
        delay();
        count++;
        }
        return(0);
}
// delay() is a function used to add wait time after every update so the changes are visible to human eye.
int delay(void){
        volatile int delay_count = 0;
        while(delay_count < WAIT_VAL)
        delay_count++;
        return(0);
}
```

Part 2.

```
#include <xparameters.h>
#include <xgpio.h>
#include <xstatus.h>
#include <xil_printf.h>

/* Definitions */

#define GPIO_LED_DEVICE_ID    XPAR_LED_DEVICE_ID            // stores the GPIO id for led output
#define GPIO_SWITCH_DEVICE_ID XPAR_SWITCH_DEVICE_ID //stores the GPIO id for switch and button input.
#define WAIT_VAL    10000000

int delay(void);

int main()
{
  int count;
  int count_masked;
  int led_status;


  int sw_in;

  int switches_masked;


  XGpio leds;

  XGpio switches;

  int status;


  status=XGpio_Initialize(&leds,GPIO_LED_DEVICE_ID); // initializes led GPIO module

  XGpio_SetDataDirection(&leds,1,0x00); // sets the direction to output

  if (status != XST_SUCCESS) {

          xil_printf("LED Initialization Failed \n" );

  }


  status=XGpio_Initialize(&switches,GPIO_SWITCH_DEVICE_ID); // initializes second GPIO for switch and button

          XGpio_SetDataDirection(&switches,1,0xFF); // sets the direction to input

          if (status != XST_SUCCESS) {

                  xil_printf("Input Switches Initialization Failed \n" );

  }


  count = 0;

  count_masked = 0;


  while(1) {

          sw_in = XGpio_DiscreteRead(&switches,1); //reads the 8-bit input from switches and buttons

         //The input format is  B3, B2, B1, B0, Sw3, Sw2, Sw1, Sw0

          if (sw_in & (1 << 4)) {  // if B0 is pressed

                  count++;             //the count value is incremented

                  count_masked = count & 0xF;

                  xil_printf("The count value is incremented to %d \t LED Value %d\n", count_masked, led_status);

          } else if (sw_in & (1 << 5)) { //if B1 is pressed

                  count--;               //the count is decremented
```

```
            count_masked = count & 0xF;
            xil_printf("The count value is decremented to %d \t LED Value %d|n", count_masked, led_status);
        } else if (sw_in & (1 << 6)) { //if B2 is pressed
            switches_masked = sw_in & 0xF; //the switch pattern is extracted to be written on led
            led_status = switches_masked;
            xil_printf("The Switches status is %d and is displayed on LED |n", switches_masked, led_status);
            XGpio_DiscreteWrite(&leds,1,switches_masked);
        } else if (sw_in & (1 << 7)) { //if B3 is pressed
            led_status = count_masked; // the count is displayed on led
            xil_printf("The COUNT value displayed on LED %d |n", led_status);
            XGpio_DiscreteWrite(&leds,1,led_status);
        } else {
            //Do Nothing
        }
        delay();
  }
  return(0);
}

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

QUESTIONS

6.(a) In the first part of the lab, we created a delay function by implementing a counter. The goal was to update the LEDs approximately every second as we did in the previous lab. Compare the count value in this lab to the count value you used as a delay in the previous lab. If they are different, explain why? Can you determine approximately how many clock cycles are required to execute one iteration of the delay for-loop? If so, how many?

The sys_clk on ZYBO board is of frequency 125MHz. The COUNT value in previous experiment was 62500000 and is 10000000 in this experiment. They are different because in lab 1 we were directly counting the positive edges in the clock. Whereas in this case the count represents the number of loop iterations which is a software metric.

Yes. As per our C code, 10000000 iterations takes 1 sec. And in 1 sec we have 125000000 clock cycles. The number of clock cycles per iteration comes out to be **12.5 cycles.**

(b) Why is the count variable in our software delay declared as volatile?

Declaring a variable volatile keeps the variable in Register file so that it is readily present to operate on. In order to have a consistent delay time across multiple executions of delay loops it is imperative to have it in cpu register always.

(c) What does the while(1) expression in our code do?

It keeps the program running. In the first part after every loop iteration the count was modified to display on LED and then wait for a second. It will keep our Zybo board engaged displaying the updated value continuously. Similarly, in the second part it served the same purpose of keeping the execution of code continuous. After every second, the output value and internal state was modified as per the fed input

(d) Compare and contrast this lab with the previous lab. Which implementation do you feel is easier? What are the advantages and disadvantages associated with a purely software implementation such as this when compared to a purely hardware implementation such as the previous lab?

This lab develops a Counter using a software code running on a Microblaze which is soft-core processor. The former lab implemented a counter on Hardware itself. I personally feel the software implementation was easier because the hardware IPs were already available and I just had to write a code to execute as per my requirements. The advantage of software implementation is its flexibility. The different add-on requirements could be easily implemented by small changes in software. Whereas in hardware implementation many changes were involved.

The disadvantage of software implementation is the design requires a lot of extra hardware resources.