Description: Description: Description: Description: Description: Description: Description: Description: Description: Description: Texas A&M University Logo - aTm

Lab #2: SDK

Khanh Nguyen

UIN# 525000335

ECEN 449– 503

Date: February 2, 2017

**INTRODUCTION:**

The purpose of the lab is to get us the first touch to get used with Software Development Kit (SDK). We learned how to create a Microblaze processor system and design a software to control the LEDs and switches.

**PROCEDURE:**

1/ Create a new block design and add the MicroBlaze processor to the design. Run block automation.

2/ Configure the system clock (single ended clock capable) of 100MHz.

3/ Add GPIO block and configure it to be the output of LEDs.

4/ Connect the components automatically by running connection automation.

5/ Connect High and Low signal to reset pins of the processor and the clock.

6/ Add the contraints file for the LEDs.

7/ Generate bitstream and create a HDL wrapper and switch to SDK

8/ Create and import the provided C code to the SDK

9/ Program the FPGA through SDK and configure to run it.

10/ Add a new 8-bit GPIO IP block as inputs and use it to create a “count” program that can increment, decrement and keep track of the count value.

**RESULT:**

I created a new 8-it GPIO blocks for the inputs of the counter program. I went through the header file xparameters.h to obtain the address of the block and use it in the SetDataDirection function. The logic is almost the same as lab 1, however, this time I used C.

**CONCLUSION:**

I learned how to use the SDK to create a MicroBlaze system and use GPIO blocks to create inputs and output for the system. I also learned to how to implement the FPGA using C.

The process of reading through the header files and the C code helps me understand the syntax and the use of the new functions: SetDataDirection, DiscreteWrite, DiscreteRead, etc in SDK.

**VERILOG CODES:**

**PART1:**

**C file**

#include <xparameters.h>

#include <xgpio.h>

#include <xstatus.h>

#include <xil\_printf.h>

/\* Definitions \*/

#define GPIO\_DEVICE\_ID XPAR\_LED\_DEVICE\_ID /\* GPIO device that LEDs are connected to \*/

#define WAIT\_VAL 1000000000

int delay (void);

int main()

{

int count;

int count\_masked;

XGpio leds;

int status;

status=XGpio\_Initialize(&leds, GPIO\_DEVICE\_ID);

XGpio\_SetDataDirection(&leds,1,0x00);//Assign Gpio block as output in channel 1

if (status !=XST\_SUCCESS){//check if the initialization success

xil\_printf("Intialization failed");

}

count=0;

while (1) //infinity loop to keep the program running

{

count\_masked=count & 0xF; //0xF = 1111 in binary which are the 4 bits of LEDs

XGpio\_DiscreteWrite(&leds,1,count\_masked);//write the value of count\_masked to LEDs

xil\_printf("Value of LEDs = 0x%x \n \r",count\_masked);// print value of LEDs

delay(); // call delay function

count++; //increase count

}

return(0);

}

int delay(void)

{

volatile int delay\_count=0; //initialize delay

while (delay\_count< WAIT\_VAL) // increase delay time ulti it reaches the wait time

delay\_count++;

return (0);

}

**XDC file:**

#Initialize 100MHz clock

#clock\_rtl

set\_property PACKAGE\_PIN L16 [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]

#Initialize LEDs

#led\_tri\_o

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]}]

**PART 2:**

**C file:**

#include <xparameters.h>

#include <xgpio.h>

#include <xstatus.h>

#include <xil\_printf.h>

#define LED XPAR\_LED\_DEVICE\_ID

#define SWITCH XPAR\_SWITCH\_DEVICE\_ID

// Search the xparameters.h file for the GPIO block addresses

#define WAIT\_VAL 1000000000

int delay (void) ;

int main()

{

int count;

int count\_masked;

XGpio leds;

XGpio switches;

int status1, status2;

int val,val1,val2;

status1 =XGpio\_Initialize(&leds, LED);

XGpio\_SetDataDirection(&leds,1,0x00); // Initialize GPIO block 1 as output

if(status1 != XST\_SUCCESS) {//check for errors

xil\_printf("Initialization failed");

}

status2 =XGpio\_Initialize(&switches, SWITCH);

XGpio\_SetDataDirection(&switches,1,0xFF);//GPIO block 2 as input (switches)

if(status2 != XST\_SUCCESS) {//check for errors

xil\_printf("Initialization failed");

}

count =0;

while (1)

{

val = XGpio\_DiscreteRead(&switches,1); // read values from block 2

val1 = val & 0x0F;//lower 4 bit value

if(val1 == 0x1) { //if switch is 1, increment count

count++;

count\_masked = count & 0xF;

xil\_printf("Value of COUNT = 0x%x \n\r", count\_masked);

}

else if(val1 == 0x2) { // if switch is 2, decrement count

count--;

count\_masked = count & 0xF;

xil\_printf("Value of COUNT = 0x%x \n\r", count\_masked);

}

else if(val1 == 0x8) { // if switch is 8, display count

count\_masked = count & 0xF;

XGpio\_DiscreteWrite(&leds,1,count\_masked); //Write to LEDs

xil\_printf("Value of COUNT = 0x%x \n\r", count\_masked);

}

else if(val1 == 0x4) { //if switch is 4, display the status of switches

val2 = XGpio\_DiscreteRead(&switches,1) & 0xF0;// 0xF0 is 11110000 (upper 4 bits)

val2 = val2/16;

XGpio\_DiscreteWrite(&leds,1,val2);//write to LEDs

xil\_printf("Value of SWITCHES = 0x%x \n\r",val2);

}

delay();

}

return(0);

}

int delay(void)

{

volatile int delay\_count = 0;

while(delay\_count < WAIT\_VAL)

delay\_count++;

return(0);

}

**XDC File:**

#clock\_rt1

set\_property PACKAGE\_PIN L16 [ 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\_o

set\_property PACKAGE\_PIN M14 [ get\_ports {led\_tri\_o[0]}]

//Use the variable name generated in the design tab

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]}]

#switch\_tri\_i

set\_property PACKAGE\_PIN R18 [ get\_ports {switch\_tri\_i[0]}]

set\_property IOSTANDARD LVCMOS33 [ get\_ports {switch\_tri\_i[0]}]

set\_property PACKAGE\_PIN P16 [ get\_ports {switch\_tri\_i[1]}]

set\_property IOSTANDARD LVCMOS33 [ get\_ports {switch\_tri\_i[1]}]

set\_property PACKAGE\_PIN V16 [ get\_ports {switch\_tri\_i[2]}]

set\_property IOSTANDARD LVCMOS33 [ get\_ports {switch\_tri\_i[2]}]

set\_property PACKAGE\_PIN Y16 [ get\_ports {switch\_tri\_i[3]}]

set\_property IOSTANDARD LVCMOS33 [ get\_ports {switch\_tri\_i[3]}]

set\_property PACKAGE\_PIN G15 [ get\_ports {switch\_tri\_i[4]}]

set\_property IOSTANDARD LVCMOS33 [ get\_ports {switch\_tri\_i[4]}]

set\_property PACKAGE\_PIN P15 [ get\_ports {switch\_tri\_i[5]}]

set\_property IOSTANDARD LVCMOS33 [ get\_ports {switch\_tri\_i[5]}]

set\_property PACKAGE\_PIN W13 [ get\_ports {switch\_tri\_i[6]}]

set\_property IOSTANDARD LVCMOS33 [ get\_ports {switch\_tri\_i[6]}]

set\_property PACKAGE\_PIN T16 [ get\_ports {switch\_tri\_i[7]}]

set\_property IOSTANDARD LVCMOS33 [ get\_ports {switch\_tri\_i[7]}]

**QUESTIONS:**

**(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?**

In previous lab the clock rate was 125MHz, and since I have to reverse the clock every 0.5s, so 40,000,000 clock cycles will be required for 0.5s.

In this lab, I don’t have to reverse the clock so 100,000,000 clock cycles will be required for 1s

100MHz=10E-8s => (10E-8s)\*(10E8)=1s

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

It is declared like that because the delay can be changed and the timescale per clock cycle depends on the processor.

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

It is the infinity loop, the loop will run until user aborts the program.

**(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?**

It was my first time implement the FPGA board using SDK and C. Therefore, I still feel Verilog is a little bit easier. Software implementation is little bit harder (due to some driver problems and unknown errors, I had to recreate the whole design three times). However, using SDK and C, I was able to check the output from both software and FPGA board. Pure software implementation can reduce the cost and the risk of hardware failure, while hardware implementation rely a lot on the quality of hardware used.