COEN 317 Lab 3 (UJ-X)

by

Mamadou Diao Kaba

27070179

Performed on

March 5th, 2024

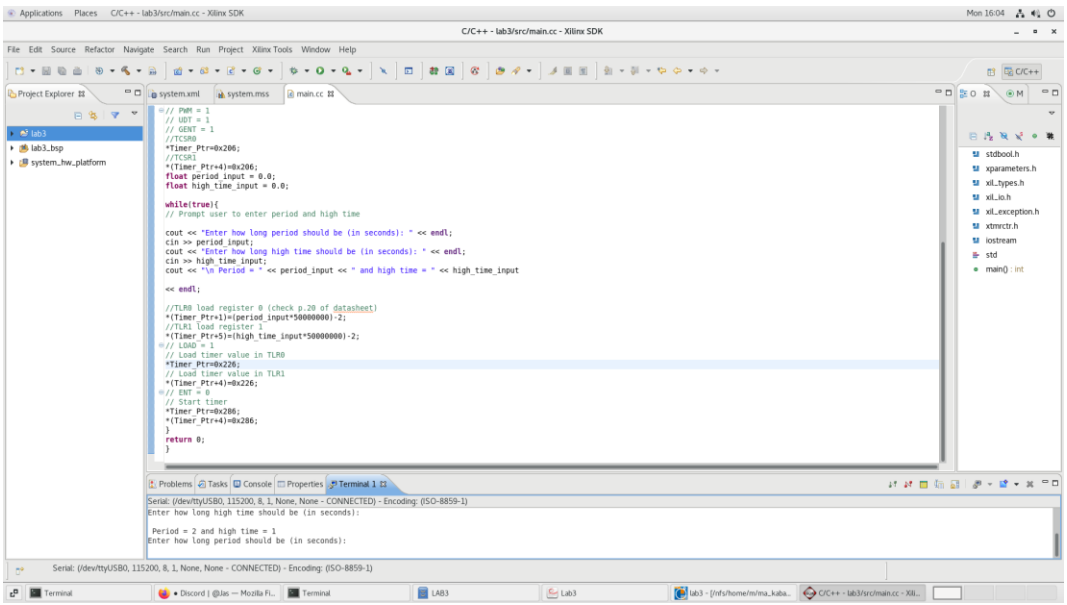
"I certify that this submission is my original work and meets the Faculty's Expectations of Originality."

Introduction

In this laboratory session, we explore the LogiCORE IP AXI Timer, a component featuring two built-in hardware counters. The lab exercises involve utilizing the timer across three distinct operational modes: as a basic timer/counter, in capture mode, and as a PWM (Pulse Width Modulation) device.

Results

In this part, we are asked to implement the PWM mode for the AXI Timer. Following the small changes mentioned in the lab, I made it to the SDK where I wrote the program for the main. To write the program, I referred to the pseudo code given in part 2 for the order of the process, the equations to calculate the period and the high time given in the lab manual, and the pseudo code given in part 3 for what the procedure should be. Additionally, I referred to the AXI Timer DS764 manual to find which bits I need to set and which registers to use. In the end, I managed to get the second light to dim thus, validating the code.



Conclusion

To sum up, the lab's goals were successfully achieved. Although I encountered some technical hurdles, such as the SDK's limitations on inputting values and IMPACT hindering the completion of the Capture mode, I was nonetheless able to gain a comprehensive understanding of the AXI Timer's operation, familiarizing myself with its registers and various control/status options.

APPENDIX

```
#include "stdbool.h"
#include "xparameters.h"
#include "xil_types.h"
#include "xgpio.h"
#include "xil_io.h"
#include "xil_exception.h"
#include "xtmrctr.h"
#include <iostream>
using namespace std;

int main()

{
        //modified from the code completed in the prelab

   static XGpio GPIOInstance_Ptr;
   u32* timerPtr = (u32*) XPAR_TMRCTR_0_BASEADDR; //creating a u32 variable that points to the address

   int xStatus;

   cout << "####  Counter Application Starts ####" << endl;


   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   //Step-1: AXI GPIO Initialization
   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
	xStatus = XGpio_Initialize(&GPIOInstance_Ptr, XPAR_AXI_GPIO_FOR_OUTPUT_DEVICE_ID);
	if(xStatus != XST_SUCCESS)
	{
		cout << "GPIO A Initialization FAILED" << endl;
		return 1;
	}

	//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
	//Step-2: AXI GPIO Set the Direction
	//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
	//XGpio_SetDataDirection(XGpio *InstancePtr, unsigned Channel, u32 DirectionMask);
	//we use only channel 1, and 0 is the the parameter for output

	XGpio_SetDataDirection(&GPIOInstance_Ptr, 1, 0);

	//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
	//Step-3: PMW Timer Initialization and Setting
	//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

	*timerPtr = 0x206; //setting the control/status of reg0's PWM's timer to enable pulse width, enabling generate mode and, enable count down
	*(timerPtr + 4) = 0x206; //setting the second timeer's control/status

	 //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
	 //Step-4: Setting the Timer Option
	 //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

	//the following are variables to control the timer
	float cycleTime; //variable for the duty cycle
	float periodTimer; //variable for the period

	cout << "Enter duty cycle time: " << endl; cin >> cycleTime;
	cout << "Enter period (s): " << endl; cin >> periodTimer;

	//----------perform the PWM-------------------------

	while(true){
       //50000000Hz  (given in lab)
	  *(timerPtr + 1) = (periodTimer * 50000000) -2;  // put results in loads
	   *(timerPtr + 5) = (periodTimer*(cycleTime/100)*50000000)-2; //putting percentage of duty in the timer 2's load
```

```
    //change controls to enable load the values from above into the timer
    *timerPtr = 0x226;
    *(timerPtr + 4) = 0x226;

    //change control to stop load and enable the counter
    *timerPtr = 0x286;
    *(timerPtr + 4) = 0x286;
    }
    return 0;
}
```