Lab #6 LED Ping Pong Game on Zybo with Polling

Name: Yixing Guo ID: U201714750 Date: 2020/12/10

Lab6 code:

```
//pingpong file for Lab #6
//Revised by Yixing Guo to add pressing early penalty
//10, 12, 2020
#include "xparameters.h"
#include "xgpio.h"
#include "led_ip.h"
// Include scutimer header file
#include "XScuTimer.h"
//====================================================
XScuTimer Timer; /* Cortex A9 SCU Private Timer Instance */
void delay(void);
void MoveBallRight(void);
void MoveBallLeft(void);
void Game(int);
void switchSpeed(void);

#define ONE_TENTH   32500000 // half of the CPU clock speed/10
#define START 1
#define STOP  0
#define LEFT  0
#define RIGHT 1
#define RESETBUTTON 0b0100
#define STARTBUTTON 0b0010
#define LEFTPADDLE  0b1000
#define RIGHTPADDLE 0b0001

int psb_check, dip_check, dip_check_prev, LedState, Status;
XGpio dip, push;
// PS Timer related definitions
XScuTimer_Config *ConfigPtr;
XScuTimer *TimerInstancePtr = &Timer;

int LED_PATTERNS[4] = {0b1000, 0b0100, 0b0010, 0b0001};
int scoreright, scoreleft;
char GameOver, StartDirection;

int main(void)
{
    unsigned int i;

    //initialize variables, timers, ports
    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);

    // Initailize the timer
    ConfigPtr = XScuTimer_LookupConfig(XPAR_PS7_SCUTIMER_0_DEVICE_ID);
    Status = XScuTimer_CfgInitialize(TimerInstancePtr, ConfigPtr, ConfigPtr-
>BaseAddr);
    if (Status != XST_SUCCESS)
    {
        xil_printf("Timer init() failed\r\n");
        return XST_FAILURE;
    }
    switchSpeed();
    // Set AutoLoad mode
    XScuTimer_EnableAutoReload(TimerInstancePtr);
    // Start the timer
    XScuTimer_Start(TimerInstancePtr);

    xil_printf("-- Start of the Ping Pong Program --\r\n");
    GameOver = STOP;
    scoreright = 0;
    scoreleft = 0;
    xil_printf("Score Left = %d   Score Right = %d\r\n", scoreleft, scoreright);

    int resetOneShot = 0, startPlayer = LEFT;
    while (1)
    {
        switchSpeed();
        // Read push buttons and reset score if Button 2 is pressed
        psb_check = XGpio_DiscreteRead(&push, 1);
        if (psb_check == RESETBUTTON && resetOneShot == 0) //reset game
        {
            resetOneShot = 1;
            GameOver = STOP;
            scoreright = 0;
            scoreleft = 0;
            LED_IP_mWriteReg(XPAR_LED_IP_0_S_AXI_BASEADDR, 0, 0b0000);
            xil_printf("\n\rNew Game - Scores Reset\r\n");
        }
        if (psb_check == STARTBUTTON)
        {
            GameOver = START; //start game
            xil_printf("\n\rGame Start\r\n");
            //start the game and follow StartDirection}
            startPlayer = !startPlayer;
            Game(startPlayer);
            // game end
            resetOneShot = 0;
            xil_printf("\n\rGame End\r\n");
            xil_printf("Score Left = %d   Score Right = %d\r\n", scoreleft,
scoreright);
        }
    } //while(1)
} //main()

void Game(int startPlayer)
{
```

```
    if (startPlayer == LEFT)
    {
        LedState = 0;
        StartDirection = RIGHT;
    }
    else
    {
        LedState = 3;
        StartDirection = LEFT;
    }

    // clear time counter
    delay();

    LED_IP_mWriteReg(XPAR_LED_IP_0_S_AXI_BASEADDR, 0, LED_PATTERNS[LedState]);
    delay();

    while (GameOver == START)
    {
        if (StartDirection == LEFT)
        {
            MoveBallLeft();
        }
        else
        {
            MoveBallRight();
        }
    }
}

void MoveBallRight(void)
{
    char i, EarlyPress;
    EarlyPress = 0;
    //move LED to the right
    LED_IP_mWriteReg(XPAR_LED_IP_0_S_AXI_BASEADDR, 0, LED_PATTERNS[++LedState]);
    delay();
    // check for button pushes
    //    psb_check = XGpio_DiscreteRead(&push, 1);
    if (psb_check == RIGHTPADDLE)
    {
        EarlyPress = 1;
    }
    //set StartDirection
    if (LedState == 3 && EarlyPress == 1)
    {
        StartDirection = LEFT;
    }
    //set GameOver; display scores
    else if (LedState == 3)
    {
        GameOver = STOP;
        scoreleft++;
    }
}

void MoveBallLeft(void)
{
```

```
    char i, EarlyPress;
    EarlyPress = 0;
    //move LED to the left
    LED_IP_mWriteReg(XPAR_LED_IP_0_S_AXI_BASEADDR, 0, LED_PATTERNS[--LedState]);
    delay();
    // check for button pushes
    //    psb_check = XGpio_DiscreteRead(&push, 1);
    if (psb_check == LEFTPADDLE)
    {
        EarlyPress = 1;
    }
    //set StartDirection
    if (LedState == 0 && EarlyPress == 1)
    {
        StartDirection = RIGHT;
    }
    //set GameOver; display scores
    else if (LedState == 0)
    {
        GameOver = STOP;
        scoreright++;
    }
}

void delay(void)
{
    // Load timer with delay in multiple of ONE_T
    int psb = 0;
    while (!XScuTimer_IsExpired(TimerInstancePtr))
    {
        psb_check = XGpio_DiscreteRead(&push, 1);
        if (psb == 0)
        {
            psb = psb_check;
        }
    }
    // clear status bit
    XScuTimer_ClearInterruptStatus(TimerInstancePtr);
    psb_check = psb;
}

void switchSpeed(void)
{
    dip_check = XGpio_DiscreteRead(&dip, 1);
    if (dip_check != dip_check_prev)
    {
        xil_printf("Switch Game Speed: %d\r\n", dip_check);
        dip_check_prev = dip_check;
        // load timer with the new switch settings
        XScuTimer_LoadTimer(TimerInstancePtr, ONE_TENTH * dip_check);
    }
}
```