Lab #7 Interrupt-Driven Ping-Pong Game on Zybo

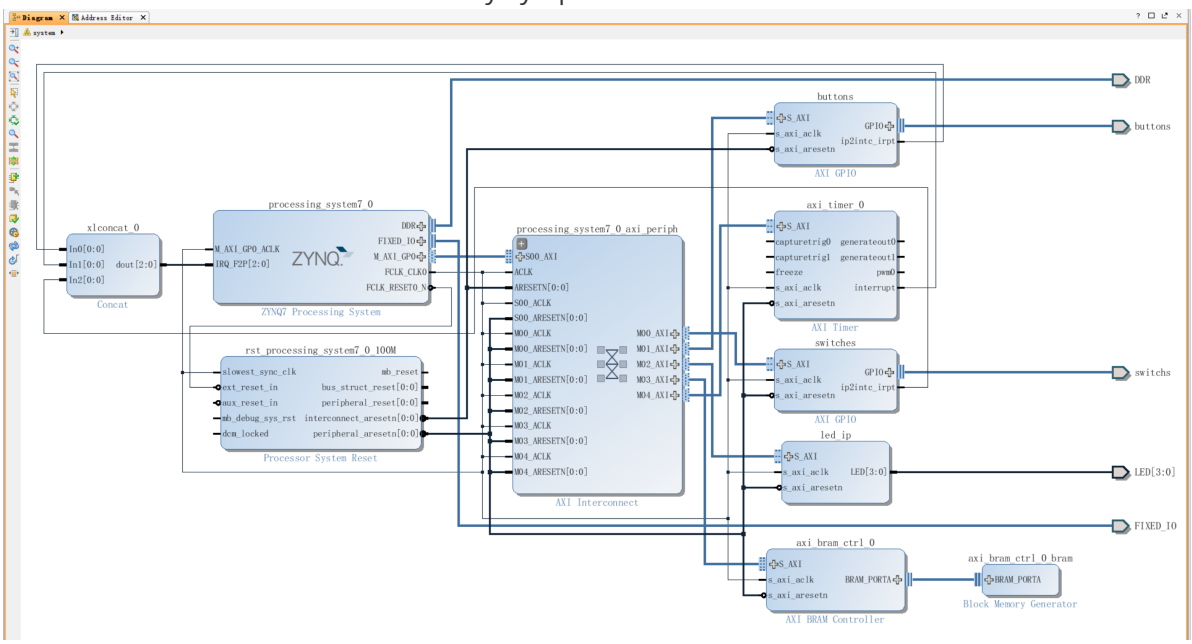
Name: Cunyang Liu ID:U201811446 Date:2021/08/19

In Lab7, I use interrupt trigger to complete ping-pong game. Before I write the game code, I first test the interrupt sample code of Zynq Book Tutorial 2. Referring to block design on PPT, I add AXI Timer module to Lab6, and use Concat module to connect interput interface. After building the Platform and running the program, it is found that the LED is not on. After checking, it is found that the LED IP is generated by us, rather than the conventional GPIO IP defined by the system.

Therefore, I will replace XGpio_DiscreteWrite(&LEDInst, 1, led_data); statements with LED_IP_mWriteReg(XPAR_LED_IP_0_S_AXI_BASEADDR, 0, led_data); to get the desired effect.

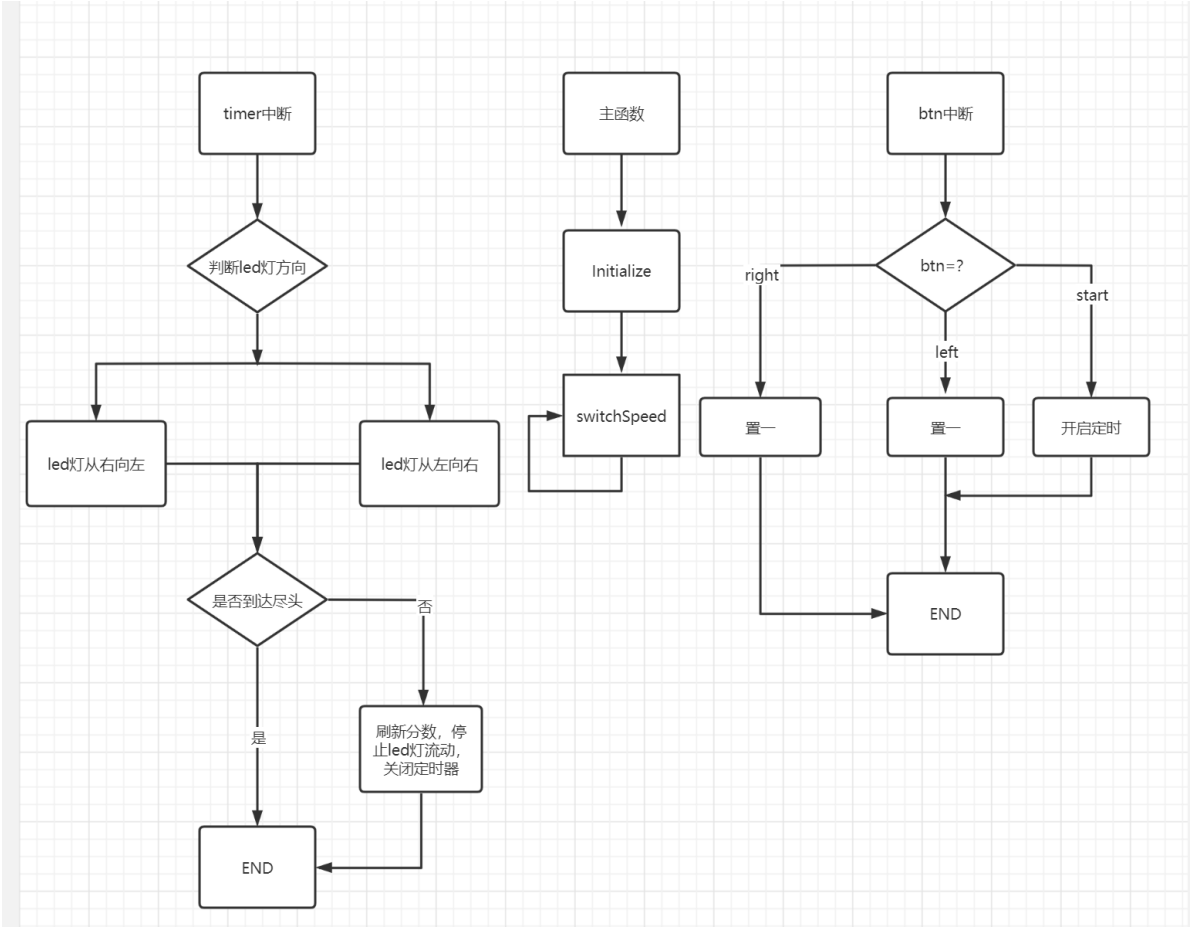
Then I started writing game code. Due to the use of interrupt program to control the game process, and Lab6 process is very different, so it has made great changes. After a period of debugging and modification, I finally completed the experiment.

Here are screen shots of Part2 of my Zynq Book tutorial




```
Serial: (COM4, 115200, 8, 1, None, None - CONNECTED) - Encoding: (ISO-8859-1)
-- Game Start --
Score Left = 0   Score Right = 1
-- Game End --
-- Left Player Win --
Score Left = 1   Score Right = 1
-- Game Start --
Score Left = 1   Score Right = 1
-- Game End --
```

Lab7 ASM Chart



Lab7 Code

```
//pingpong file for Lab #7
//Revised by Cunyang Liu to add pressing early penalty
//2021/8/19
#include "xparameters.h"
#include "xgpio.h"
#include "led_ip.h"
#include "xtmrctr.h"
#include "xscugic.h"
#include "xil_exception.h"
// Include scutimer header file
#include "XScuTimer.h"
//======================================================
// Parameter definitions
#define INTC_DEVICE_ID              XPAR_PS7_SCUGIC_0_DEVICE_ID
#define TMR_DEVICE_ID               XPAR_TMRCTR_0_DEVICE_ID
#define BTNS_DEVICE_ID              XPAR_BUTTONS_DEVICE_ID
#define SWITCHES_DEVICE_ID          XPAR_SWITCHES_DEVICE_ID
#define LEDS_DEVICE_ID              XPAR_LED_IP_0_DEVICE_ID
#define INTC_GPIO_INTERRUPT_ID      XPAR_FABRIC_BUTTONS_IP2INTC_IRPT_INTR
#define INTC_TMR_INTERRUPT_ID       XPAR_FABRIC_AXI_TIMER_0_INTERRUPT_INTR
#define LEDS_BASEADDR               XPAR_LED_IP_S_AXI_BASEADDR

#define BTN_INT                     XGPIO_IR_CH1_MASK
#define TMR_LOAD                    0xFA0A1EFD
#define TMR_LOAD_GEAR               0x22ADD0

// Game parameter
#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

#define LED_Display(data)       (LED_IP_mWriteReg(LEDS_BASEADDR, 0, (data)))
#define TMR_SetInterval(gear)   XTmrCtr_SetResetValue(&TMRInst, 0, (TMR_LOAD + TMR_LOAD_GEAR * (gear)))

XGpio SWInst, BTNInst;
XScuGic INTCInst;
// PS Timer related definitions
XScuTimer Timer; /* Cortex A9 SCU Private Timer Instance */
XScuTimer_Config *ConfigPtr;
XScuTimer *TimerInstancePtr = &Timer;
// TMR
XTmrCtr TMRInst;

int btn_value, tmr_count;
int psb_check, SWInst_check, SWInst_check_prev, LedState, Status;
int startPlayer = 0;
int scoreright, scoreleft;
char GameOver, StartDirection;
int LED_PATTERNS[4] = {0b1000, 0b0100, 0b0010, 0b0001};

void GameReset(void);
void GameStart(void);
void GameStop(void);
void MoveBallRight(void);
void MoveBallLeft(void);
void SwitchSpeed(void);
void InitializeSystem(void);
int InterruptSystemSetup(XScuGic *XScuGicInstancePtr);
int IntcInitFunction(u16 DeviceId, XTmrCtr *TmrInstancePtr, XGpio *GpioInstancePtr);

void GameReset(void)
{
    XTmrCtr_Stop(&TMRInst, 0);

    xil_printf("-- Game Reset --\r\n");
    GameOver = STOP;
    scoreright = 0;
    scoreleft = 0;
}

void GameStart(void)
{
    XTmrCtr_Stop(&TMRInst, 0);

    xil_printf("-- Game Start --\r\n");
    xil_printf("Score Left = %d   Score Right = %d\r\n", scoreleft, scoreright);
```

```
    GameOver = START;
    startPlayer = !startPlayer;
    if (startPlayer == 0) {
        LedState = 0;
        StartDirection = RIGHT;
    } else {
        LedState = 3;
        StartDirection = LEFT;
    }
    LED_Display(LED_PATTERNS[LedState]);

    // reset timer
    XTmrCtr_Reset(&TMRInst, 0);
    XTmrCtr_Start(&TMRInst, 0);
}

void GameStop(void) {
    XTmrCtr_Stop(&TMRInst, 0);

    GameOver = STOP;
}

void MoveBallRight(void)
{
    if (LedState == 3)
    {
        GameStop();
        ++scoreleft;
        xil_printf("-- Game End --\r\n");
        xil_printf("-- Left Player Win --\r\n");
        xil_printf("Score Left = %d   Score Right = %d\r\n", scoreleft,
scoreright);
        return;
    }
    //move LED to the right
    LED_Display(LED_PATTERNS[++LedState]);
}

void MoveBallLeft(void)
{
    if (LedState == 0)
    {
        GameStop();
        ++scoreright;
        xil_printf("-- Game End --\r\n");
        xil_printf("-- Right Player Win --\r\n");
        xil_printf("Score Left = %d   Score Right = %d\r\n", scoreleft,
scoreright);
        return;
    }
    //move LED to the right
    LED_Display(LED_PATTERNS[--LedState]);
}

void SwitchSpeed(void)
{
    SWInst_check = XGpio_DiscreteRead(&SWInst, 1);
    if (SWInst_check != SWInst_check_prev)
```

```
    {
        xil_printf("Switch Game Speed: %d\r\n", SWInst_check);
        SWInst_check_prev = SWInst_check;
        // load timer with the new switch settings
        TMR_SetInterval(SWInst_check);
    }
}

void BTN_Intr_Handler(void *InstancePtr)
{
    // Disable button interrupts
    XGpio_InterruptDisable(&BTNInst, BTN_INT);
    // Ignore additional button presses
    if ((XGpio_InterruptGetStatus(&BTNInst) & BTN_INT) != BTN_INT)
        return;
    btn_value = XGpio_DiscreteRead(&BTNInst, 1);

    switch (btn_value)
    {
    case RESETBUTTON:
        GameReset();
        break;
    case STARTBUTTON:
        if (GameOver == STOP)
        {
            GameStart();
        }
        break;
    case LEFTPADDLE:
        if (GameOver == START)
        {
            if (LedState == 0)
            {
                StartDirection = RIGHT;
            }
        }
        break;
    case RIGHTPADDLE:
        if (GameOver == START)
        {
            if (LedState == 3)
            {
                StartDirection = LEFT;
            }
        }
        break;
    }

    XGpio_InterruptClear(&BTNInst, BTN_INT);
    // Enable btn interrupts
    XGpio_InterruptEnable(&BTNInst, BTN_INT);
}

void TMR_Intr_Handler(void *data, u8 TmrCtrNumber)
{
    if (XTmrCtr_IsExpired(&TMRInst, 0))
    {
        // reset timer and start running again
```

```
        if (tmr_count == 3)
        {
            XTmrCtr_Stop(&TMRInst, 0);

            if (GameOver == STOP)
            {
                return;
            }

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

            tmr_count = 0;
            XTmrCtr_Reset(&TMRInst, 0);
            XTmrCtr_Start(&TMRInst, 0);
        }
        else
            tmr_count++;
    }
}

// Initialize system configuration
void InitializeSystem(void)
{
    xil_printf("-- Start Initialize System --\r\n");

    int status;

    status = XGpio_Initialize(&SWInst, SWITCHES_DEVICE_ID);
    if (status != XST_SUCCESS)
    {
        xil_printf("Initialize Switch Device Failed!\r\n");
        return;
    }
    XGpio_SetDataDirection(&SWInst, 1, 0xffffffff);

    status = XGpio_Initialize(&BTNInst, BTNS_DEVICE_ID);
    if (status != XST_SUCCESS)
    {
        xil_printf("Initialize Button Device Failed!\r\n");
        return;
    }
    XGpio_SetDataDirection(&BTNInst, 1, 0xffffffff);

    status = XTmrCtr_Initialize(&TMRInst, TMR_DEVICE_ID);
    if (status != XST_SUCCESS)
    {
        xil_printf("Initialize Timer Device Failed!\r\n");
        return;
    }
    XTmrCtr_SetHandler(&TMRInst, TMR_Intr_Handler, &TMRInst);
    TMR_SetInterval(0);
```

```
    XTmrCtr_SetOptions(&TMRInst, 0, XTC_INT_MODE_OPTION |
XTC_AUTO_RELOAD_OPTION);

    status = IntcInitFunction(INTC_DEVICE_ID, &TMRInst, &BTNInst);
    if (status != XST_SUCCESS)
    {
        xil_printf("Initialize Interrupt Failed!\r\n");
        return;
    }
    XTmrCtr_Start(&TMRInst, 0);

    xil_printf("-- Initialization Success --\r\n");
}

int InterruptSystemSetup(XScuGic *XScuGicInstancePtr)
{
    // Enable button interrupt
    XGpio_InterruptEnable(&BTNInst, BTN_INT);
    XGpio_InterruptGlobalEnable(&BTNInst);

    Xil_ExceptionRegisterHandler(XIL_EXCEPTION_ID_INT,

(Xil_ExceptionHandler)XScuGic_InterruptHandler,
                                 XScuGicInstancePtr);
    Xil_ExceptionEnable();

    return XST_SUCCESS;
}

int IntcInitFunction(u16 DeviceId, XTmrCtr *TmrInstancePtr, XGpio
*GpioInstancePtr)
{
    XScuGic_Config *IntcConfig;
    int status;

    // Interrupt controller initialization
    IntcConfig = XScuGic_LookupConfig(DeviceId);
    status = XScuGic_CfgInitialize(&INTCInst, IntcConfig, IntcConfig-
>CpuBaseAddress);
    if (status != XST_SUCCESS)
    {
        xil_printf("Initialize Interrupt Controller Failed!\r\n");
        return XST_FAILURE;
    }

    // interrupt setup
    status = InterruptSystemSetup(&INTCInst);
    if (status != XST_SUCCESS)
    {
        xil_printf("Enable Interrupt Failed!\r\n");
        return XST_FAILURE;
    }

    // Connect button interrupt to handler
    status = XScuGic_Connect(&INTCInst,
                             INTC_GPIO_INTERRUPT_ID,
                             (Xil_ExceptionHandler)BTN_Intr_Handler,
                             (void *)GpioInstancePtr);
```

```
    if (status != XST_SUCCESS)
    {
        xil_printf("Connect GPIO Interrupt Failed!\r\n");
        return XST_FAILURE;
    }

    // Connect timer interrupt to handler
    status = XScuGic_Connect(&INTCInst,
                             INTC_TMR_INTERRUPT_ID,
                             (Xil_ExceptionHandler)TMR_Intr_Handler,
                             (void *)TmrInstancePtr);
    if (status != XST_SUCCESS)
    {
        xil_printf("Connect Timer Interrupt Failed!\r\n");
        return XST_FAILURE;
    }

    // Enable gpio interrupts interrupt
    XGpio_InterruptEnable(GpioInstancePtr, 1);
    XGpio_InterruptGlobalEnable(GpioInstancePtr);

    // Enable GPIO and timer interrupts in the controller
    XScuGic_Enable(&INTCInst, INTC_GPIO_INTERRUPT_ID);
    XScuGic_Enable(&INTCInst, INTC_TMR_INTERRUPT_ID);

    return XST_SUCCESS;
}

int main(void)
{
    xil_printf("-- Start of the Program --\r\n");

    InitializeSystem();

    xil_printf("-- Start of the Ping Pong Program --\r\n");

    GameReset();

    while (1)
    {
        SwitchSpeed();
    }
}
```