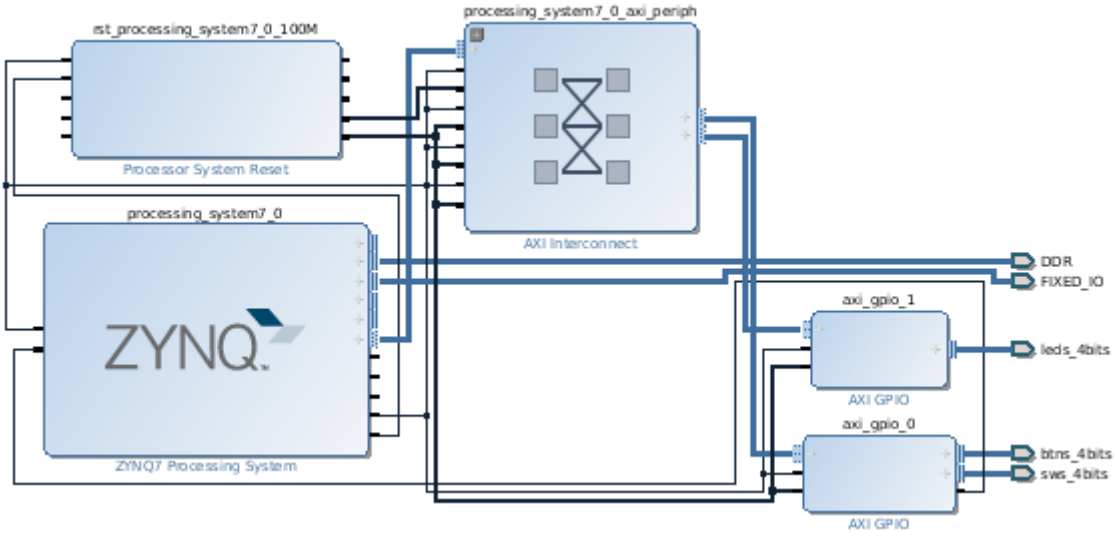
Digital Systems VLSI

Nanos Georgios

Initially, we created the requested Hardware / Software system at ZYBO. We also enabled the interrupts for the buttons. The resulting architecture is as follows:



Next, we programmed the system based on the following C source file, so that it implements the requested tennis game:

```
#include <stdio.h>
#include "platform.h"
#include "sleep.h"
#include "xtime_l.h"
```

```
#include "xscugic.h"
#include "xgpio.h"
#include "xil_exception.h"
#include "xil_printf.h"

#define COUNTS_PER_USECOND  (XPAR_CPU_CORTEXA9_CORE_CLOCK_FREQ_HZ / (2*1000000))
#define INTC_DEVICE_ID XPAR_PS7_SCUGIC_0_DEVICE_ID
#define BTNS_DEVICE_ID XPAR_AXI_GPIO_0_DEVICE_ID
#define LEDS_DEVICE_ID XPAR_AXI_GPIO_1_DEVICE_ID
#define INTC_GPIO_INTERRUPT_ID XPAR_FABRIC_AXI_GPIO_0_IP2INTC_IRPT_INTR
#define BTN_INT XGPIO_IR_CH1_MASK

typedef enum state {stc = 0, sts = 1, stm1 = 2, stm2 = 3, stg1 = 4, stg2 = 5} state;

state currentState = stc;
state nextState = stc;

int dir = 0;
int delay = 1000000;
int score[2] = {0,0};
int Ready_to_quit = 0;
int wake_up =0;

XGpio LEDInst, BTNInst;
XScuGic INTCInst;
int led_data;
int btn_value;

static void BTN_Intr_Handler(void *baseaddr_p);
static int InterruptSystemSetup(XScuGic *XScuGicInstancePtr);
static int IntcInitFunction(u16 DeviceId, XGpio *GpioInstancePtr);
void next();

int custom_sleep(unsigned int useconds)
{

    XTime tEnd, tCur;

    XTime_GetTime(&tCur);
    tEnd = tCur + ((XTime) useconds) * COUNTS_PER_USECOND;
```

```
    do
    {
        if (wake_up == 1){
            wake_up = 0;
            break;
        }

        XTime_GetTime(&tCur);
    } while (tCur < tEnd);

    return 0;
}

void BTN_Intr_Handler(void *InstancePtr) {
    XGpio_InterruptDisable(&BTNInst, BTN_INT);

    if((XGpio_InterruptGetStatus(&BTNInst) & BTN_INT) != BTN_INT) {
        return;
    }

    btn_value = XGpio_DiscreteRead(&BTNInst, 1);

    if((btn_value & 0b1000) == 0b1000 && (currentState == stg1 || currentState == sts)) {
        nextState = stm1;
        if (currentState == sts) xil_printf("In progress...\n\r");
        dir = 0;
        wake_up = 1;
    }

    else if ((btn_value & 0b0001) == 0b0001 && (currentState == stg2)) {
        nextState = stm2;
        dir = 1;
        wake_up = 1;
    }

    (void)XGpio_InterruptClear(&BTNInst, BTN_INT);
    XGpio_InterruptEnable(&BTNInst, BTN_INT);
}

int InterruptSystemSetup(XScuGic *XScuGicInstancePtr) {
```

```
    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, XGpio *GpioInstancePtr) {
    XScuGic_Config *IntcConfig;
    int status;
    IntcConfig = XScuGic_LookupConfig(DeviceId);
    status = XScuGic_CfgInitialize(&INTCInst, IntcConfig, IntcConfig->CpuBaseAddress);
    if(status != XST_SUCCESS) return XST_FAILURE;

    status = InterruptSystemSetup(&INTCInst);
    if(status != XST_SUCCESS) return XST_FAILURE;

    status = XScuGic_Connect(&INTCInst, INTC_GPIO_INTERRUPT_ID, (Xil_ExceptionHandler) BTN_Intr_Handler,  (void
*) GpioInstancePtr);
    if(status != XST_SUCCESS) return XST_FAILURE;

    XGpio_InterruptEnable(GpioInstancePtr, 1);
    XGpio_InterruptGlobalEnable(GpioInstancePtr);
    XScuGic_Enable(&INTCInst, INTC_GPIO_INTERRUPT_ID);
    return XST_SUCCESS;
}


void next() {
    switch(currentState) {
    case stc:
            nextState = stc;
            break;
    case sts:
        nextState = sts;
        break;
    case stm1:
        if(dir == 0) nextState = stm2;
```

```
        else nextState = stg1;
        break;
    case stm2:
        if(dir == 0) nextState = stg2;
        else nextState = stm1;
        break;
    case stg1:
    case stg2:
        nextState = stc;
        break;
    }
    return;
}

void axion() {
    switch(currentState) {
    case stc:
    {
        led_data = 0b0000;
        int choice = 0;
        xil_printf("\n1. Start\n\r");
        xil_printf("2. Set Delay\n\r");
        xil_printf("3. Show Score\n\r");
        xil_printf("4. Reset\n\r");
        xil_printf("5. Quit\n\r");

        fflush(stdin);
        scanf("%d", &choice);
        switch(choice) {
        case 1:
            xil_printf("Waiting for player 1...\n\r");
            currentState = sts;
            break;
        case 2:
            xil_printf("Set 500 <= delay <= 2000 ms\n\r");
            fflush(stdin);
            scanf("%d", &choice);
            if(choice >= 500 && choice <= 2000)
                delay = 1000*choice;
            else xil_printf("Errorn\r");
```

```
        if(choice == 500) xil_printf("Hard!\n\r");
        if(choice == 1000) xil_printf("Standard mode activated!\n\r");
        if(choice == 2000) xil_printf("Easy\n\r");

        break;
    case 3:
        xil_printf("The score is:\n\r");
        xil_printf("%d - %d\n\r", score[0], score[1]);
        if (score[0] > score [1]) xil_printf("Feeling lucky, Player 1?\n\r");
        if (score[0] < score [1]) xil_printf("Player 2 is winning! As expected...\n\r");
        if (score[0] == score [1]) xil_printf("A tie!\n\r");
        break;
    case 4:
        score[0] = 0;
        score[1] = 0;
        xil_printf("Score has been reset\n\r");
        break;
    case 5:
        xil_printf("Terminating application. \nAlways victories, my friends!\n\r");
        Ready_to_quit = 1;
        break;
    default:
        currentState = stc;
        break;
    }
    break;
}
case sts:
    led_data = 0b1000;
    break;
case stm1:
    led_data = 0b0100;
    break;
case stm2:
    led_data = 0b0010;
    break;
case stg1:
    led_data = 0b1000;
    break;
case stg2:
```

```
        led_data = 0b0001;
        break;
    }
}

int main()
{

    init_platform();

   int status;

   status = XGpio_Initialize(&LEDInst, LEDS_DEVICE_ID);
   if(status != XST_SUCCESS) return XST_FAILURE;

   status = XGpio_Initialize(&BTNInst, BTNS_DEVICE_ID);
   if(status != XST_SUCCESS) return XST_FAILURE;

   XGpio_SetDataDirection(&BTNInst, 1, 0xF); // set first channel tristate buffer to input
   XGpio_SetDataDirection(&LEDInst, 1, 0x0); // set first channel tristate buffer to output

   status = IntcInitFunction(INTC_DEVICE_ID, &BTNInst);
   if(status != XST_SUCCESS) return XST_FAILURE;



   print("Welcome\n\r");



   while(1)
   {
      //xil_printf("%d\n\r", currentState);
      if (Ready_to_quit == 1) break;
      axion();
      XGpio_DiscreteWrite(&LEDInst, 1, led_data);
      next();
      custom_sleep(delay);
      if (currentState == stg1 && nextState == stc) {
          score[1]++;
```

```
        xil_printf("haha?\n\r");
    }
    else if (currentState == stg2 && nextState == stc) {
        score[0]++;
        xil_printf("You can do better!\n\r");
    }
    currentState = nextState;
  }

  return 0;
}
```