Vivado SDK Basic IP Integrator

Robert Bara
Robert.Bara@temple.edu

Summary

This laboratory serves as an introduction to embedded system development by using Vivado and SDK to modify the procedure from the *Zynq Book Tutorials* and create a basic IP integrator for the Zybo Board. A hardware design using Vivado is modified to have two GPIOs for buttons and LEDs, which is then programmed by modifying the C program *LED_test_tut_1C.c* in SDK, to wire the buttons as inputs and vary the LEDs as outputs depending on which button is enabled.

Introduction

To create an IP Integrator, this lab stems from following the first tutorial in the *Zynq Book Tutorials* as a basic foundation. Two GPIOs are added to the hardware design to yield the buttons on the Zybo Board as the input, and LEDs will act as an output based on the following tasks:

The rightmost Button 0 is a system RESET and when released the LED count will start at *signed integer* "-8" and count up to *signed integer* "7", while displaying their 4-bit *signed binary* equivalents with roll over, on the LEDs.

Button 1 suspends all further LED count operations when pressed. Once released the LED count will begin at 7 and counts down to -8, while displaying their 4-bit *signed binary* equivalents with roll over, on the LEDs.

Button 2 will suspend all further LED count operations, similarly to Button 1, but once released, the LEDs will start at 0 and use binary to represent an increasing/decreasing, repeating bar graph. The sequence will loop as follows: 0000, 0001, 0011, 0111, 1111, 0111, 0011, 0001, 0000.

The leftmost Button 3 suspends all further LED count operations when pushed, but when released the LED count repeats a pattern: 001, 0110, 1010, 0101, 1100, 0011, 1001, 0110. Starting with 1001.

Discussion

Hardware Design

The hardware design within this lab begins with adding the ZYNQ 7 Processing System which will wire up all of our additional components by bringing in the reset block (rst_ps7_0_100M) and the processor (ps7_axi_periph). Next, two GPIOs should be added with GPIO_0 wiring to the ZYBO's push buttons as inputs, and GPIO_1 wiring the LEDs to display an output. Figure 1 demonstrates the completed block diagram below:

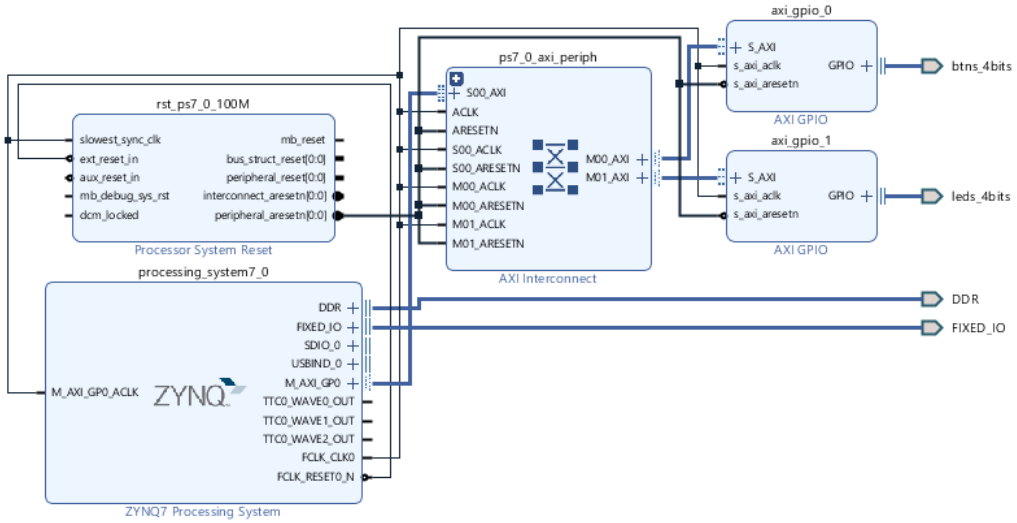

Figure 1. Vivado Block Diagram for IP Integrator

Upon completing the design, the design must be validated, then a wrapper should be created in order to generate a bitstream. Finally, when the bitstream is written, the hardware is exported, and SDK can be launched to program the board.

C Program

The C program used within this lab, stems as a modified code of *LED_test_tut_1C.c* and runs as follows.

Initialization

The top of the C program consists of defining header files to be included. Global variables to make the program easier to follow are written underneath and can be found in the appendix. Finally, the LED and Button IDs are declared from the AXI GPIOs in the hardware design:

```
//Definitions
#define BTNS_DEVICE_ID XPAR_AXI_GPIO_0_DEVICE_ID
#define LEDS_DEVICE_ID XPAR_AXI_GPIO_1_DEVICE_ID
```

Figure 2. AXI GPIO definitions. Buttons=GPIO_0, LEDs=GPIO_1

The program jumps to the main function, which initializes the LEDs as outputs and push buttons, while ensuring that the status is true, if not the program will fail:

```
// LED initialization
status = XGpio_Initialize(&LEDInst, LEDS_DEVICE_ID);
if (status != XST_SUCCESS) return XST_FAILURE;
// Button initialization
status = XGpio_Initialize(&BTNInst, BTNS_DEVICE_ID);
if (status != XST_SUCCESS) return XST_FAILURE;
//Setting LEDs as outputs
XGpio_SetDataDirection(&LEDInst, LED_CHANNEL, 0x00);
//Setting Buttons as Inputs
XGpio_SetDataDirection(&BTNInst, BTN_CHANNEL, 0xFF);
```

Figure 3. LED/Button Initialization, Status check

Subroutines

Main then calls the function LEDOutputExample, which continuously runs unless a failure occurs, as seen in the appendix. LEDOutputExample is ran by using the buttons as a volatile integer that gets stored to a temporary value known as hold. The function starts by setting the LED's so they are off, and awaits an input from the buttons. The button input is then assigned to the temporary value hold and hold is checked if the value is even or odd. It is checked because each button has a number value associated with it:btn[0]=1, btn[1]=2, btn[2]=4, btn[3]=8, since btn[0] is also a system reset, if the any of these buttons are pressed while holding down btn[0] or only btn[0] is pressed, their sum will be ODD, which then runs the task described in the lab manual, which starts the LEDs at signed binary -8 and counts up to signed binary 8, then rolls over. Notice the delay is written into the code in order for the LED's to properly update so they do not blink too fast.

```
if(hold%2!=0){
    while(1){
    btn=XGpio_DiscreteRead(&BTNInst, BTN_CHANNEL);
    printf("SYSTEM RESET\n");
    led=-8; //initializing LED=-8 in binary
    XGpio_DiscreteWrite(&LEDInst, LED_CHANNEL, led);
                        for (led = -8; led < 9; led++){ //counting up
                            for (Delay = 0; Delay < LED_DELAY; Delay++);
                            if (led==9){ //rolls over
                                led=-8;
                            }
                            //outputs to LEDs
                            XGpio_DiscreteWrite(&LEDInst, LED_CHANNEL, led);
```

Figure 4. Checking if ODD, performing System Reset

The system reset is done as a sort of priority logic, which is why this case is done as an "if" statement. The second half of this case will be seen within all the remaining cases. The idea of this is after updating an LED, the program checks for a button input and if the button input is not the same as it was when it started, it will suspend the LEDs and hold until a button is released. The program then exits the for loop and while loop and either

restarts the case for the same button or goes through the entire subroutine again to determine whether the value is even or odd, then deciding which case to run.

```
        btn=XGpio_DiscreteRead(&BTNInst, BTN_CHANNEL);
        if (btn!=0) break;
        else continue;
    }
    if (btn!=0) break;
    else continue;
```

Figure 5. Recurring method to exit conditionals

The remainder of the function uses a switch statement to capture the remaining tasks. Case 2 represent the binary input 4b0010, which is the case of btn[1]. Btn[1] will count down using signed binary from 7 to signed binary -8, then roll over back to 7. After this it has the option to break out of the loop similar to figure 5.

```
case 2: //count down from 7 to -8
    while(1){
        printf("BUTTON 1\n");
        XGpio_DiscreteWrite(&LEDInst, LED_CHANNEL, led);//Suspends LEDs
        led=7; //initializing LED=7 in binary
        for (led = 7; led >-9; led--){ //counting down to -8
            for (Delay = 0; Delay < LED_DELAY; Delay++);
            if (led==-9){
                led=-7; //restarts the loop after -8
            }
            //outputs to LEDs
            XGpio_DiscreteWrite(&LEDInst, LED_CHANNEL, led);
            printf("Button 1-Led:%d\n",led);
```

Figure 6. Case 3 which uses while and for loops to count down from 7 to -8

Case 3 will correspond to btn[2], which will display a bar graph across the LEDs by increasing, decreasing, and repeating the following pattern 0000, 0001, 0011, 0111, 1111, 0111, 0011, 0001. Although I could have made the bar graph pattern an array and iterate across the array, as I do for Case 4's pattern, I initially realized there is a mathematical relationship, where each number increases by the current value assigned to the LED multiplied by 2, and then adding 1 to that value. I decided to use this for the bar graph method of counting up, so I initialized the LEDs to be 0, and then used a for loop to count up by using this equation. To break out of this for loop and know when to count down, when the LEDs=0 again, the conditional if statement is modified to recognize it is time to break out and go into the for loop to count back down, aside from also checking the button input.

```
case 4: //Simulates a Bar Graph Pattern across the LEDs
    while(1){
        printf("BUTTON 2\n");
        XGpio_DiscreteWrite(&LEDInst, LED_CHANNEL, led);//Suspends LEDs
        led=0; //initialize graph to 0
        //Going up lighting up each LED until it reaches  4b1111
        for (led = 1; led<16; (led=led*2+1)){
            for (Delay = 0; Delay < LED_DELAY; Delay++);
            //outputs to LEDs
            XGpio_DiscreteWrite(&LEDInst, LED_CHANNEL, led);
            printf("Button 2-Led:%d\n",led);

            //If another input button is pressed, the loop is broken and changes to another conditional
            btn=XGpio_DiscreteRead(&BTNInst, BTN_CHANNEL);
            if (led==0||btn!=0) break;
            else continue;
        }
```

Figure 7. Case 4, Bar graph simulated across the LEDs

The code then breaks out of this loop and repeats the inverse of this block to count back down before checking to exit upon an LED output.

```
  //Going down lighting up each LED until it reaches 4b000
      for (led = 7; led > -1; (led=(led-1)/2)){
          for (Delay = 0; Delay < LED_DELAY; Delay++);
          printf("Led:%d\n",led);
          //outputs to LEDs
          XGpio_DiscreteWrite(&LEDInst, LED_CHANNEL, led);
          printf("Button 2-Led:%d\n",led);

          //If another input button is pressed, the loop is broken and changes to another conditional
          btn=XGpio_DiscreteRead(&BTNInst, BTN_CHANNEL);
          if (led==0||btn!=0) break;
          else continue;
  }
  if (btn!=0) break;
  else continue;
  }
  break;
```

Figure 8. Counting down the bar graph, checking to exit the loop or not

The last case is for btn[3], which repeats a pattern by declaring the binary pattern as decimals within and array and using the variable “i” to iterate throughout a for loop and update the LEDs. All of this is written within a while loop to continuously run, and the remainder of the case is the same as figure 5, which checks if the case should be exited.

```
case 8:
    /* The repeating pattern 1001, 0110, 1010, 0101, 1100, 0011
     * in Decimal is 9,6,10,5,12 and these values are held within the array pat
     * And is displayed on the LEDs if Button 3 is pressed
    */
    while(1){
    printf("BUTTON 2\n");
    XGpio_DiscreteWrite(&LEDInst, LED_CHANNEL, led);//Suspends LEDs
    for (i=0;i<5;i++){
        led=pat[i]; //LED equals the iteration across the pattern array
        XGpio_DiscreteWrite(&LEDInst, LED_CHANNEL, led);
        printf("Button 3-Led:%d\n",led);
        for (Delay = 0; Delay < LED_DELAY; Delay++);
```

Figure 9. Btn[3], displaying a repeating pattern

The remainder of the switch statement includes a default statement that sets btn and hold equal to 1, so if the program somehow fails, a system reset is activated.

Verification

The hardware demo may be examined using the following link:

https://www.youtube.com/watch?v=Cc5cwt90gec&ab_channel=RobertBara

Upon determining if all of the tasks were achieved, there is one error I did realize within my code which is in case 4 which is for btn[2], when I hold the button, it does not suspend one LED value, it alternates between the initial values of each for loop and I think this is because of the way that I am exiting the loop. I am leaving this method in though, so I can learn from my mistakes in the future, to fix this error, I would simply rewrite case 4 to be exactly like case 8, except I would create a new array of the bar graph increasing/decreasing and just loop the iteration across the array. This error can be seen below in the following screenshots:

Figure 10. Holding btn[2], alternates between suspending 1

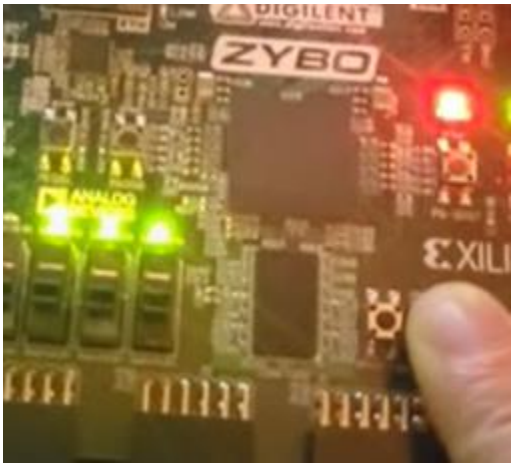

Figure 11. Holding btn[2], alternates between suspending 7

The only other possible error I may have is a misinterpretation of how the LED suspension should be. When I hold down a button on my board, it freezes the LED to what is initialized and then upon release it restarts that case. Its possible that I could fix this if the LED suspension is actually supposed to hold the current value and keep going if the same button is being held, rather than restart. To do that, I figure I would just have to create another variable to hold the led value, and it would just be an additional if statement to add.

The following screenshots ensure that everything else is coded correctly upon holding buttons

This first screen shot proves that if btn[0] and any other btn will be held, the LEDs will hold signed binary -8, and then upon release will perform a system reset by counting from -8 to 8 with roll over.

Figure 12. Holding btn[0] and btn[1], suspending -8

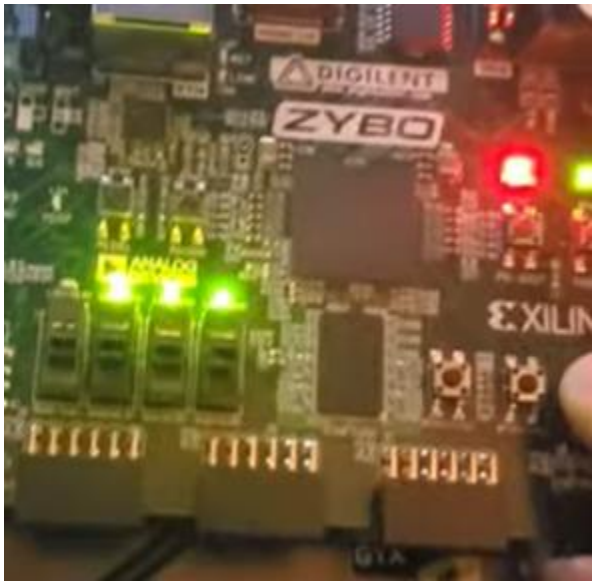

Figure 13. Holding btn[2], suspending 7

Figure 14. Holding btn[3], suspending 4b1001

Conclusion

This lab serves as a refreshment to programing hardware using C. The lab introduced Vivado for hardware design, as well SDK which uses eclipse as an IDE for programming. Putting the two programs together and using the Zynq Book Tutorials 1A, 1B, 1C, 2A, and 2B as well as the C sample program *LED_test_tut_1C.c* as a starting point, led to the creation of a basic IP Integrator, which uses push buttons for inputs and 4 LEDs as outputs. I would say I am confident about 90% of my design functions as it is properly stated in the lab manual, and any possible errors involving suspending the LEDs when being held, were discussed, and talked about how to fix these potential logic errors, in the verification section under results.

Appendix

C Code

```
/*
 * ECE 2623: Embedded Systems Laboratory 1                Spring 2021
 * Vivado Basic SDK IP Integrator By Robert Bara
 */

/* Include Files */
#include "xparameters.h"
#include "xgpio.h"
#include "xstatus.h"
#include "xil_printf.h"

//Definitions
#define BTNS_DEVICE_ID XPAR_AXI_GPIO_0_DEVICE_ID
#define LEDS_DEVICE_ID XPAR_AXI_GPIO_1_DEVICE_ID
#define LED -8
#define LED_DELAY 50000000
#define LED_CHANNEL 1
#define BTN_CHANNEL 1
#define printf xil_printf

XGpio LEDInst, BTNInst;

//Outputting to LED function
int LEDOutputExample(void)
{
	//declaring variables
	volatile int Delay;
	volatile int btn;
	static int pat[]={9,6,10,5,12}; //pattern of binary numbers converted to
decimal for case 4
	volatile int i=0; //iterates over array

	int led = LED; // Hold current LED value. Initialize to LED definition
	volatile int hold; // Holds the Button Input, so btn's value can be
changed

	while(1){ //loops the function
			XGpio_DiscreteWrite(&LEDInst, LED_CHANNEL,
0);//Initializes LEDs as off
			//First the Input btn is read
			btn=XGpio_DiscreteRead(&BTNInst,
BTN_CHANNEL);printf("Button Input: %d\n", btn);
			//The Button's input is assigned to a temporary value Hold
			hold=btn;

			/*//Checking if Hold is ODD, then run Task for Button 0,
the System Reset
			 * If more than 1 button is held down their input values
are added and creates
			 * an ODD number, so the System Reset is Ran
			 * btn[0] activates the System Reset which counts up from
signed -8 to 8
```

```
                     */
                    if(hold%2!=0){
                          while(1){
                          btn=XGpio_DiscreteRead(&BTNInst, BTN_CHANNEL);
                          printf("SYSTEM RESET\n");
                          led=-8; //initializing LED=-8 in binary
                          XGpio_DiscreteWrite(&LEDInst, LED_CHANNEL, led);
                                                              for (led = -8; led
< 9; led++){ //counting up
                                                                    for (Delay =
0; Delay < LED_DELAY; Delay++);
                                                                    if (led==9){
//rolls over
                                                                          led=-
8;
                                                                    }
                                                                    //outputs to
LEDs
          XGpio_DiscreteWrite(&LEDInst, LED_CHANNEL, led);

          printf("Button 0-Led:%d\n",led);

                                              /*If another input button is pressed
and is not an ODD sum,
                                               *The loop is broken and changes to
another conditional
                                               */

          btn=XGpio_DiscreteRead(&BTNInst, BTN_CHANNEL);
                                                                    if (btn!=0)
break;
                                                                    else
continue;
                                                              }
                                                              if (btn!=0) break;
                                                              else continue;
                    }
                    }

                    /*
                     * If the button input is an even value,
                     * Hold is assigned the value and activates a case to the
corresponding buttons
                     * btn[1]=2               btn[2]=4           btn[3]=8
                     */
                    else{

                    switch(hold){
                    case 2: //count down from 7 to -8
                          while(1){
                                printf("BUTTON 1\n");
                                XGpio_DiscreteWrite(&LEDInst, LED_CHANNEL,
led);//Suspends LEDs
                                led=7; //initializing LED=7 in binary
```

```
                                for (led = 7; led >-9; led--){ //counting
down to -8
                                      for (Delay = 0; Delay < LED_DELAY;
Delay++);
                                      if (led==-9){
                                            led=-7; //restarts the loop
after -8
                                      }
                                      //outputs to LEDs
                                      XGpio_DiscreteWrite(&LEDInst,
LED_CHANNEL, led);
                                      printf("Button 1-Led:%d\n",led);

                                      //If another input button is pressed,
the loop is broken and changes to another conditional
                                      btn=XGpio_DiscreteRead(&BTNInst,
BTN_CHANNEL);
                                      if (btn!=0) break;
                                      else continue;
                                }
                                if (btn!=0) break;
                                else continue;
                                }

                          break;

                    case 4: //Simulates a Bar Graph Pattern across the LEDs
                          while(1){
                                printf("BUTTON 2\n");
                                XGpio_DiscreteWrite(&LEDInst, LED_CHANNEL,
led);//Suspends LEDs
                                led=0; //initialize graph to 0
                                //Going up lighting up each LED until it
reaches  4b1111
                                for (led = 1; led<16; (led=led*2+1)){
                                      for (Delay = 0; Delay < LED_DELAY;
Delay++);
                                      //outputs to LEDs
                                      XGpio_DiscreteWrite(&LEDInst,
LED_CHANNEL, led);
                                      printf("Button 2-Led:%d\n",led);

                                      //If another input button is pressed,
the loop is broken and changes to another conditional
                                      btn=XGpio_DiscreteRead(&BTNInst,
BTN_CHANNEL);
                                      if (led==0||btn!=0) break;
                                      else continue;
                                }
                          //Going down lighting up each LED until it reaches
4b000
                                for (led = 7; led > -1; (led=(led-1)/2)){
                                      for (Delay = 0; Delay < LED_DELAY;
Delay++);
                                      printf("Led:%d\n",led);
```

```
                                        //outputs to LEDs
                                        XGpio_DiscreteWrite(&LEDInst,
LED_CHANNEL, led);
                                        printf("Button 2-Led:%d\n",led);

                                        //If another input button is pressed,
the loop is broken and changes to another conditional
                                        btn=XGpio_DiscreteRead(&BTNInst,
BTN_CHANNEL);
                                        if (led==0||btn!=0) break;
                                        else continue;
                                }
                                if (btn!=0) break;
                                else continue;
                                }
                                break;

                        case 8:
                                /* The repeating pattern 1001, 0110, 1010, 0101,
1100, 0011
                                 * in Decimal is 9,6,10,5,12 and these values are
held within the array pat
                                 * And is displayed on the LEDs if Button 3 is
pressed
                                */
                                while(1){
                                printf("BUTTON 2\n");
                                XGpio_DiscreteWrite(&LEDInst, LED_CHANNEL,
led);//Suspends LEDs
                                for (i=0;i<5;i++){
                                        led=pat[i]; //LED equals the iteration across
the pattern array
                                        XGpio_DiscreteWrite(&LEDInst, LED_CHANNEL,
led);
                                        printf("Button 3-Led:%d\n",led);
                                        for (Delay = 0; Delay < LED_DELAY; Delay++);

                                        //If another input button is pressed, the
loop is broken and changes to another conditional
                                        btn=XGpio_DiscreteRead(&BTNInst,
BTN_CHANNEL);
                                        if (led==0||btn!=0) break;
                                        else continue;
                                }
                                if (btn!=0) break;
                                else continue;
                                }

                                break;
                        default:
                                //For Safety, if program finds another input, the
System Reset will be ran
                                hold=1; btn=1;
                        break;
                                }
```

```
        }
        }

        return XST_SUCCESS; /* Should be unreachable */
}

//MAIN
int main(void){

        int Status;
        int status;
        // LED initialization
        status = XGpio_Initialize(&LEDInst, LEDS_DEVICE_ID);
        if (status != XST_SUCCESS) return XST_FAILURE;
        // Button initialization
        status = XGpio_Initialize(&BTNInst, BTNS_DEVICE_ID);
        if (status != XST_SUCCESS) return XST_FAILURE;
        //Setting LEDs as outputs
        XGpio_SetDataDirection(&LEDInst, LED_CHANNEL, 0x00);
        //Setting Buttons as Inputs
        XGpio_SetDataDirection(&BTNInst, BTN_CHANNEL, 0xFF);
        //Go to LED Output function, Buttons will determine what outputs to LEDs
        Status = LEDOutputExample();
        if (Status != XST_SUCCESS) {
                xil_printf("GPIO output to the LEDs failed!\r\n");
        }

        return 0;
}
```