ECE 354: Lab 1 Report

Group 14C: J. Lagasse, A. Sjogren, O. Onyenokwe, L. Wu

1. Introduction

There are many goals of this lab, such as understanding design flow in embedded system design, familiarizing ourselves with the Altera CAD tools including Quartus Prime and NIOS II IDE, learning the differences between traditional microcontrollers and soft core processors, learning how to use on chip memory and SDRAM in hardware designs, and gaining a deeper understanding of hardware and software development flow. The lab is split into two parts: the first part consists of a simple hello world tutorial, while the second part requires further manipulating the hardware and software by creating a counting program that counts up to a specific number in about 1 minute. This number is calculated by adding all individual digits of each member's student ID number and taking the modulo, which is the input to the counting program. The counter is required to display the values in decimal on the seven segment display and in binary on the LEDs. We were able to accomplish all these details in our project.

2. Detailed Procedure

Part 1:

We started out by creating a new project in Quartus Prime and added in necessary components using Qsys. After hardware components, such as NIOS II, JTAG UART and interval timer were added, we connected these components in Qsys. We connected clock and reset outputs to the clock and reset inputs of each component, and we also connected on-chip memory with the processor. We were able to generate a Verilog file from Qsys and analyzed it using the "Start Analysis and Synthesis" button. Once the Verilog file was free of errors, we used assignment editor to assign reset and clock pins, then full compilation was performed. After we made sure the hardware design was error-free, we then programmed the FPGA and moved onto software. In NIOS II Software Build Tools for Eclipse, we programmed the Hello World small template onto the chip and ran it as NIOS II hardware with our reset switch at logic HIGH.

Part 2:

The procedures for part 2 were extremely similar to part 1 producers, except for minor changes in hardware design, the use of a different software template, and the need to rewrite the template's code to suit the needs of this project. In Qsys, we used an external off chip SDRAM for both instruction and data memory instead of an on-chip memory. We also used parallel input output hardware for the LEDs, Hexadecimal Seven Segment Displays and keys for the reset buttons. Instead of using Assignment editor to assign pins by hand, we imported a provided pin assignment file. As for software, we modified a provided count binary program so that it tooks in digits, summed them up, counted up to the calculated sum under one minute, and displayed the count in binary on the Hexadecimal Seven Segment Displays.

3. Hardware Changes

Part 1:

In the first part of the lab, we did not need to make many changes to the hardware. Our hardware consisted of the following components: the NIOS II processor, a clock source, 20480 bytes of on-chip memory, a JTAG UART, an interval timer and a clock source. We simply connected all components together as detailed in the lab tutorial, making sure to connect the clock, reset, and memory pins properly.

Part 2:

For part two, we went through all the same general processes as in part one except we used different components. We first had to add the NIOS II processor, JTAG UART, and Interval Timer such as in part one. We also had to add a PLL because the SDRAM requires a different clock frequency than the NIOS II processor, as well as an SDRAM controller to regulate this memory chip following the specifications in the lab instructions. We also included three PIO parallel I/O devices to handle our toggle switch inputs, LED outputs, and seven segment display outputs. Some components such as the I/O pins needed to be renamed in order to work properly with the pin assignment file and program file provided to us. We then had to make the connections between reset and clock inputs and outputs, as well as the instruction and data master from the processor to the SDRAM. We imported a file that contained the correct pin assignments, and set the pin assignment HDL module as the top level module.

4. Software Changes

Part 1:

The software component of the first part of this lab consisted only of using the BSP template Hello World Small and running it using our hardware configuration. No changes to the software were necessary.

Part 2:

In the second part, many changes were made to the code. We started off using the BSP template Count Binary but found we needed to alter it to fulfil the objectives of this project. We first changed the **initial_message** method to print out our names, out group number, project number, and the date to the console for neatness when running our program. We then went to the main method and decided to create a new method **student_id** to prompt the user to enter in the student ID numbers of each group member, add all digits together, and return a number max_count to count up to. The number **max_count** that is returned is the sum modulo 100 to allow it to fit on only two segments of the seven segment display. We then altered the **main** method to have the program stop counting at **max_count**, and altered the timing between each number (using **max_count** as a parameter) such that the count would always equal **max_count** at about 1 minute, regardless of the size of **max_count**. Lastly we needed to change what was printed on the seven segment display from hex values to decimal. We achieved this by altering the method **sevenseg_set_hex** such that the program only used digits from 0 and 9 and never used a through f. To implement this, we took the modulo 10 of the input and saved it as **hex**, and then divided the input by 10 and saved it as **div**. Then line 163 was changed using **hex** and **div** as inputs to make sure it always printed out decimal characters. After making these changes, our code was successful. Screenshots of our code are below:

```
#ifdef SEVEN_SEG_PIO_BASE
static void sevenseg_set_hex(int hex)
{
    static alt_u8 segments[10] = {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7C, 0x07,
            0x7F, 0x67, /* 0-9 */ 0x77, 0x7C, 0x39, 0x5E, 0x79, 0x71}; /* a-f */

    int div = hex/10; // compute division of hex value divided by 10
    hex = hex%10;     // compute modulo of hex value modulo 10

    unsigned int data = segments[hex & 15] | (segments[div & 15] << 8); // alter segments to print out correct data

    IOWR_ALTERA_AVALON_PIO_DATA(SEVEN_SEG_PIO_BASE, data);
}
#endif
```

```
static void initial_message() // changed initial message to print out group members names
{
    /* Prints out standard welcome message to console. */
    printf("\n\n**************************************************************\n");
    printf("* ECE 354: Computer Systems Lab II - Project 1 Part 2 - 02/14/17 *\n");
    printf("* Group 14C: J. Lagasse, A. Sjogren, O. Onyenokwe, L. Wu          *\n");
    printf("**************************************************************\n");
}
```

```
static int student_id() // new method for computing value to count to from student ID numbers
{
    /* Create 4 char array place holders to store student ID from console. */
    char user_input_1 [8];
    char user_input_2 [8];
    char user_input_3 [8];
    char user_input_4 [8];

    /* Prints commands to user to console. */
    printf("Please enter a Student ID number when prompted. \n");
    printf("If there are less than 4 group members, you can enter 0 or press the enter key to advance. \n");

    /* Reads each student ID individually and stores it in char array place holders. */
    printf("Enter Student ID: \n");
    gets(user_input_1);
    printf("Enter Student ID: \n");
    gets(user_input_2);
    printf("Enter Student ID: \n");
    gets(user_input_3);
    printf("Enter Student ID: \n");
    gets(user_input_4);

    /* Processing student ID's to obtain individual sums. */
    int p;
    int sum1 = 0;
    for(p = 0; p < 8; ++p) {
        int temp1 = user_input_1[p] - '0'; /* subtracting '0' converts the char to an int */
        if(temp1==-48){ break; }
        sum1 = sum1+temp1;
    }
    int sum2 = 0;
    for(p = 0; p < 8; ++p) {
        int temp2 = user_input_2[p] - '0'; /* subtracting '0' converts the char to an int */
        if(temp2==-48){ break; }
        sum2 = sum2+temp2;
    }
    int sum3 = 0;
    for(p = 0; p < 8; ++p) {
        int temp3 = user_input_3[p] - '0'; /* subtracting '0' converts the char to an int */
        if(temp3==-48){ break; }
        sum3 = sum3+temp3;
    }
```

```c
    int sum4 = 0;
    for(p = 0; p < 8; ++p) {
        int temp4 = user_input_4[p] - '0'; /* subtracting '0' converts the char to an int */
        if(temp4==-48){ break; }
        sum4 = sum4+temp4;
    }

    /* Print statements for checking student ID sums*/
/*
    printf("Sum 1: %i\n",sum1);
    printf("Sum 2: %i\n",sum2);
    printf("Sum 3: %i\n",sum3);
    printf("Sum 4: %i\n",sum4);
*/

    // calculate modulo
    int result = (sum1+sum2+sum3+sum4)%100;

    // print up to modulo
    printf("Counting up to: %i\n",result);

    // return value to count to
    return result;

}
```

```c
int main(void) // main method of program
{
    int i;
    int  __attribute__ ((unused))  wait_time;  /* Attribute suppresses "var set but not used" warning. */
    FILE * lcd;

    count = 0;

    /* Initialize the LCD, if there is one.
     */
    lcd = LCD_OPEN();
    if(lcd != NULL) {lcd_init( lcd );}

    /* Initialize the button pio. */

#ifdef BUTTON_PIO_BASE
    init_button_pio();
#endif

/* Initial message to output. */

    initial_message();               // print initial message
    int max_count = student_id();    // ask users for ID numbers, compute modulo
    int count_step = 0;              // initialize the count step to zero

/* Continue 0-ff counting loop. */

    while( 1 )
    {
        if (edge_capture != 0)
        {
            /* Handle button presses while counting... */
            handle_button_press('c', lcd);
        }
        /* If no button presses, try to output counting to all. */
        else
        {
            count_all( lcd );
        }
        /*
         * If done counting, wait about 7 seconds...
         * detect button presses while waiting.
         */
```

```
        if( count == max_count )
        {
            count=0;    // reset count to zero to reset counter after hitting value

            LCD_PRINTF(lcd, "%c%s %c%s %c%s Waiting...\n", ESC, ESC_TOP_LEFT,
                        ESC, ESC_CLEAR, ESC, ESC_COL1_INDENT5);
            printf("\nWaiting...");
            edge_capture = 0; /* Reset to 0 during wait/pause period. */

            /* Clear the 2nd. line of the LCD screen. */
            LCD_PRINTF(lcd, "%c%s, %c%s", ESC, ESC_COL2_INDENT5, ESC,
                        ESC_CLEAR);
            wait_time = 0;
            for (i = 0; i<70; ++i)
            {
                printf(".");
                wait_time = i/10;
                LCD_PRINTF(lcd, "%c%s %ds\n", ESC, ESC_COL2_INDENT5,
                    wait_time+1);

                if (edge_capture != 0)
                {
                    printf( "\nYou pushed:  " );
                    handle_button_press('w', lcd);
                }
                usleep(100000); /* Sleep for 0.1s. */
            }
            /*  Output the "loop start" messages before looping, again.
             */
            initial_message();  // reprint the initial message
            lcd_init( lcd );
        }

        int minute = 60000000;          // value of 1 minute in microseconds
        usleep((minute/3.8)/max_count); // sets timing of counter so that it counts up to modulo in 1 minute
        count++;                        // increase count variable
    }
    count_step++;                       // increase count step variable

    LCD_CLOSE(lcd);
    return 0;
}
```

5. Problems & Solutions

Part 1:

The first main problem that we encountered came when trying to run and print "Hello World!" to the console. When we programmed the board, we had initially left the reset switch on logic low, and switched it to logic high expecting the statement to be printed. However, the board needed to be executed with reset switch logic high, because the reset button is active when the switch is on logic low. This is why executing the board on eclipse on logic low did not work. When we initially put the switch on high, and switched the logic low then back to high, the statement was printed successfully.

Part 2:

One main problem that we encountered was a make error at the end of Part 2 when attempting to create a new BSP file in Eclipse. The TA Sachin helped us realize our error; we were missing a connection between the data and instructor master to the SDRAM that was not caught by the compiler. The data and instructor master needed to be connected to the SDRAM because both data and instructions are partitioned in the SDRAM memory so that the processor can easily access both sets of information later when it needs it. When we didn't have the connection, the memory was not able to be allocated, but after we made the connection the BSP file was created without error. We did not have many issues with the software, however it took us a while to visualize what we needed to change for the seven segment display method to print decimal values instead of hexadecimal. The program also would properly restart the count from zero after the first count and would count up to the right value with the correct timing every time.

After we finished this project, we practiced it again the day before the demo to prepare. Surprisingly to us, when we tried to run part 2 the code would run the first iteration of the count fine, but then would not restart the count properly and would not keep the correct timing for future iterations. We reprogrammed the board, and the error went away immediately, so we thought that the error was just a one time flaw. However, to our dismay, this error occurred during the demo as well, notifying us that something must've been wrongly declared in our code. We are going to include a copy of our code in the moodle submission so that this can be verified.

6. Conclusion

In this lab, we were able to understand the design flow in embedded system design. Using Altera CAD tools including Quartus Prime and NIOS II IDE, we were able to learn the differences between traditional microcontrollers and soft core processors. We also learned how to use on chip memory and SDRAM in hardware designs, and gaining a deeper understanding of hardware and software development flow. By the end of part one, we got a good understanding of how we could use the Altera tools and how they communicate with each other. By the end of part two, we solidified these learning outcomes and also manipulated code to achieve our specific goal by the end of the project.