Homework 5 -I2C EEPROM

Zackary McClamma[*†]

November 18, 2019

1 Introduction

This homework assignment was focused on resource sharing and consisted of running the μC/OS-II^{TM}operating system with two tasks, a mutex, and an EEPROM read and write function. The first task consists of displaying a menu in the terminal with four options, Write EEPROM, Read EEPROM, Red LED on, and Red LED off. The second task toggles a green LED on and off, changing the state of the LED every 500ms.

2 System Design

This system uses i2c communications to read and write to an EEPROM device on the board. The options for telling the OS what to do are selected by a user by using the menu displayed in the terminal. The system includes the Nios II processor, an I2C controller, on-chip memory (RAM), and two peripherals a red LED and a green LED. In contrast to the last assignment where we created our own print function, in this assignment we will be using printf and scanf to read from and write to the terminal.

3 Theory of Operation

In this system the majority of the work is done in software since the only peripherals I2C controller, the LEDs, and the clock. The Nios-II processor is first loaded onto the FPGA along with the memory. The operating system is then loaded onto the processor and is used to create two tasks. The first task prints menu where a user can select from four options. The first option is "Write EEPROM if the user selects this option they will be prompted for an address to write to and the data they want to write. The second option is "Read EEPROM" when this is selected the user will be prompted for an address and the program will return the data at that location. Options 3 and 4 are "Turn on Red LED" and "Turn off Red LED" respectively, as the menu choices indicate these will turn the Red LED on or off. A diagram of the functioning system can be seen in Figure 1

[*] University of Dayton

[†] Dept. of Electrical and Computer Engineering, University of Dayton, 300 College Park, Dayton, OH 45469-0226, U.S.A. E-mail: mcclammaz1@udayton.edu

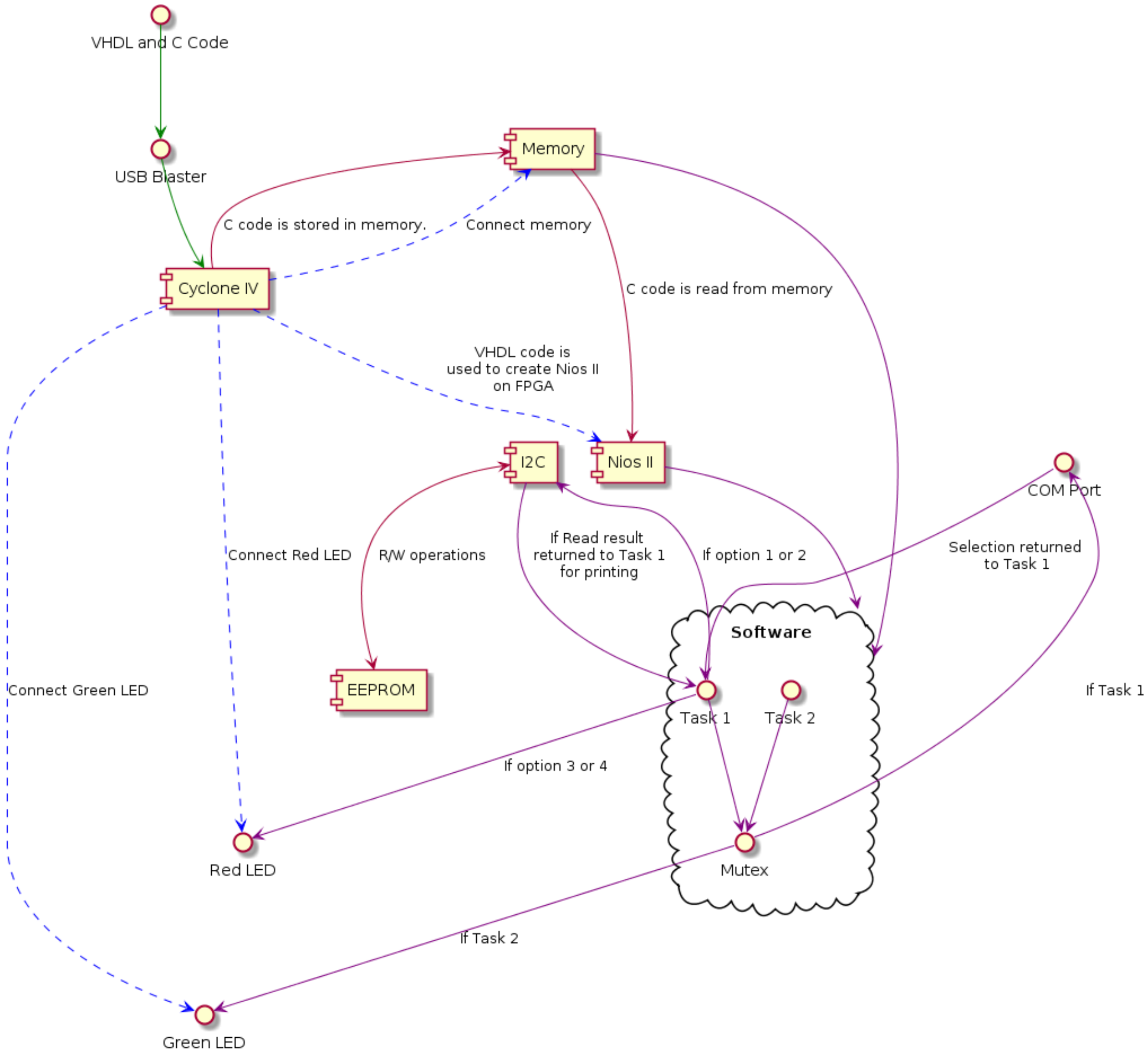
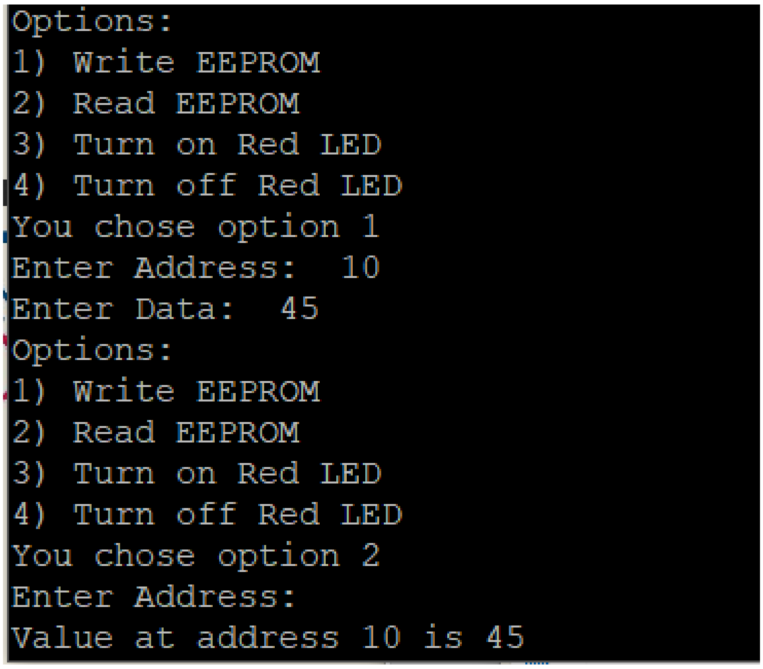


Figure (1) Embedded System Block Diagram

4 Results

I was able to get everything to work correctly except the tasking, for some reason every time that I generated the hello_uosii program there was an error in "hw5_bsp/HAL/src/os_cpu_c.c" in the OSTaskHook function. I attempted to regenerate multiple times and every time it produced the same file and the file would not show an error when building but as soon as I would run the program the error would be there even though I verified the file was the same as before I ran my program. For this reason both tasks wouldn't run at the same time, the program would only run whichever program I put at a higher priority. I am not sure why it happened with this assignment and no the last one. Below is an image of the terminal output for the EEPROM. I can demonstrate the LED functionality in class if needed.



```
Options:
1) Write EEPROM
2) Read EEPROM
3) Turn on Red LED
4) Turn off Red LED
You chose option 1
Enter Address:  10
Enter Data:  45
Options:
1) Write EEPROM
2) Read EEPROM
3) Turn on Red LED
4) Turn off Red LED
You chose option 2
Enter Address:
Value at address 10 is 45
```

Figure (2) Terminal output of EEPROM Read and Write

5 Conclusion

This homework assignment took me to the last minute to complete because of an oversight on my part. I neglected to assign pins to the i2c clock and data and it took me until a few hours before the assignment was due to realize it. Other than that the i2c was a bit confusing to figure out but once I was able to tackle that and actually able to troubleshoot it, the rest of the assignment was pretty easy.

A VHDL Code

```
-- ECE532 HW5
-- Zackary McClamma
-- 18-Nov-2019

library ieee;
  use ieee.std_logic_1164.all;

entity homework5 is
  port
  (
                        i_clk          : in  std_logic := 'X'; -- clk
                        i_reset_n      : in  std_logic := 'X'; -- reset_n
      i_uart_rxd        : in  std_logic := 'X'; -- rxd
      o_uart_txd        : out std_logic;        -- txd
      b_i2c_scl         : inout std_logic;
      b_i2c_sda         : inout std_logic;
      green_led         : out std_logic;
      red_led           : out std_logic

  );
end homework5;

architecture sch of homework5 is

  component hw5_cpu is
        port (
        clk_clk              : in  std_logic := 'X'; -- clk
        reset_reset_n        : in  std_logic := 'X'; -- reset_n
                                uart_rxd        : in  std_logic := 'X';
                                uart_txd     : out std_logic;
        i2c_sda_in       : in std_logic;
        i2c_scl_in       : in std_logic;
        i2c_sda_oe       : out std_logic;
        i2c_scl_oe       : out std_logic;
        green_led_export : out std_logic;
        red_led_export   : out std_logic

        );
  end component hw5_cpu;

  signal w_i2c_sda_in      : std_logic;
  signal w_i2c_scl_in      : std_logic;
  signal w_i2c_sda_oe      : std_logic;
  signal w_i2c_scl_oe      : std_logic;

begin
```

```
    b_i2c_scl <= '0' when w_i2c_scl_oe = '1' else 'Z';
    b_i2c_sda <= '0' when w_i2c_sda_oe = '1' else 'Z';
    w_i2c_scl_in <= b_i2c_scl;
    w_i2c_sda_in <= b_i2c_sda;

    u0 : component hw5_cpu
        port map (
        clk_clk             => i_clk,             --           clk.clk
        reset_reset_n       => i_reset_n,       --         reset.reset_n
                                uart_rxd      => i_uart_rxd,
                                uart_txd     => o_uart_txd,
        green_led_export => green_led,
        red_led_export => red_led,
        i2c_sda_in => w_i2c_sda_in,
        i2c_scl_in => w_i2c_scl_in,
        i2c_sda_oe => w_i2c_sda_oe,
        i2c_scl_oe => w_i2c_scl_oe

        );
end sch;
```

B C Code

B.1 Headers

```
/*
 * Name: Zackary McClamma
 * Course: ECE 532
 * Assignment: Homework 5
 * Date: 18 NOV 2019
 * File: hw5.h
 *
 * */

#ifndef HW5_H_
#define HW5_H_

#include <sys/alt_irq.h>
#include <sys/alt_timestamp.h>

#define I2C_BASE 0x80000
#define UART_BASE 0x60000
#define RED_LED_BASE 0x90000
#define GREEN_LED_BASE 0xA0000

#define _500_MS_TICKS 23000000
```

```
#define TIMER_IRQ 2

typedef struct str_timer_regs{
        unsigned int stats;
        unsigned int control;
        unsigned int periodl;
        unsigned int periodh;
        unsigned int snapl;
        unsigned int snaph;
}timer_regs;

typedef volatile struct{
        unsigned int uart_rxdata;
        unsigned int uart_txdata;
        unsigned int uart_status;
        unsigned int uart_control;
        unsigned int uart_divisor;
        unsigned int uart_eop;
}uart_reg;

typedef volatile struct{
        unsigned int i2c_tfr_cmd;
        unsigned int i2c_rxdata;
        unsigned int i2c_ctrl;
        unsigned int i2c_iser;
        unsigned int i2c_isr;
        unsigned int i2c_status;
        unsigned int i2c_tfr_cmd_fifo_lvl;
        unsigned int i2c_rx_data_fifo_lvl;
        unsigned int i2c_scl_low;
        unsigned int i2c_scl_high;
        unsigned int i2c_sda_hold;
}i2c_reg;

typedef volatile struct{
        unsigned int data;
        unsigned int dir;
        unsigned int intmask;
        unsigned int edge;
        unsigned int outset;
        unsigned int outclear;
}gpio_regs;

void i2c_init(void);
void eep_write(unsigned short addr, unsigned char data);
unsigned char eep_read(unsigned short addr);
void printMenu(void);

void led_on(int* base);
void led_off(int* base);
void timer_isr(void);


#endif /* HW5_H_ */
```

B.2 Source

```
/*
 * Name: Zackary McClamma
 * Course: ECE 532
 * Assignment: Homework 5
 * Date: 18 NOV 2019
 * File: main.c
 *
 * */
#include <stdio.h>
#include <system.h>
#include "hw5.h"
#include "includes.h"

OS_EVENT *tex;

/* Definition of Task Stacks */
#define   TASK_STACKSIZE       2048
OS_STK    task1_stk[TASK_STACKSIZE];
OS_STK    task2_stk[TASK_STACKSIZE];

/* Definition of Task Priorities */

#define TASK1_PRIORITY      2
#define TASK2_PRIORITY      1

/* Prints "Hello World" and sleeps for three seconds */
void task1(void* pdata)
{
  INT8U err;
  i2c_init();
  while (1)
  {
        int opt, addr, data, addrSet, dataSet = 0;
        unsigned char read;
        //display menu
        printf("Options:\r\n");
        printf("1)␣Write␣EEPROM\r\n");
        printf("2)␣Read␣EEPROM\r\n");
        printf("3)␣Turn␣on␣Red␣LED\r\n");
        printf("4)␣Turn␣off␣Red␣LED\r\n");
        scanf("%d", &opt);
        fflush(stdin);
```

```
printf("You␣chose␣option␣%d\r\n",opt);
switch(opt){

case 1:
        addrSet = 0;
        dataSet = 0;
        while(addrSet == 0)
        {
                printf("Enter␣Address:␣");
                scanf("%d", &addr);
                fflush(stdin);
                printf("␣%d\r\n", addr);

                if((addr < 0) || (addr > 127))
                {
                        printf("Invalid␣input␣try␣again\r\n");
                }
                else
                {
                        addrSet = 1;
                }
        }
        while(dataSet == 0)
        {
                printf("Enter␣Data:␣");
                scanf("%d", &data);
                fflush(stdin);
                printf("␣%d\r\n", data);

                if (data > 255)
                {
                        printf("Invalid␣input␣try␣again\r\n");
                }
                else
                {
                        dataSet = 1;
                }
        }
        eep_write(addr, data);
        OSTimeDlyHMSM(0, 0, 1, 0);
        break;
case 2:
        addrSet = 0;
        while(addrSet == 0)
        {
                printf("Enter␣Address:␣");
                scanf("%d", &addr);
                fflush(stdin);
                printf("\r\n");
                if((addr < 0) || (addr > 127))
```

```
                        {
                                printf("Invalid␣input␣try␣again\r\n");
                        }
                        else
                        {
                                addrSet = 1;
                        }
                }
                read = eep_read(addr);
                printf("Value␣at␣address␣%d␣is␣%d\r\n", addr, (int)read);
                OSTimeDlyHMSM(0, 0, 1, 0);
                break;
        case 3:
                led_on(RED_LED_BASE);
                OSTimeDlyHMSM(0, 0, 1, 0);
                break;
        case 4:
                led_off(RED_LED_BASE);
                OSTimeDlyHMSM(0, 0, 1, 0);
                break;
        default:
                printf("Invalid␣option␣please␣try␣again.");
                break;

        }
   // OSTimeDlyHMSM(0, 0, 1, 0);
        //OSMutexPost(tex);
        //OSTimeDlyHMSM(0, 0, 0, 500);
  }
}
/* Toggles the Green LED every 500ms */
void task2(void* pdata)
{
        while(1){
          INT8U err;
          OSMutexPend(tex,0,&err);
          unsigned int* gled = GREEN_LED_BASE;

                  if(*gled == 0x01){
                          led_off(GREEN_LED_BASE);
                  }
                  else{
                        led_on(GREEN_LED_BASE);
                  }
          OSMutexPost(tex);
          OSTimeDlyHMSM(0, 0, 0, 500);
        }
}
/* The main function creates two task and starts multi-tasking */
int main(void)
```

```
{
  INT8U err;
  tex = OSMutexCreate(1,&err);
  OSTaskCreateExt(task1,
                  NULL,
                  (void *)&task1_stk[TASK_STACKSIZE-1],
                  TASK1_PRIORITY,
                  TASK1_PRIORITY,
                  task1_stk,
                  TASK_STACKSIZE,
                  NULL,
                  0);


  OSTaskCreateExt(task2,
                  NULL,
                  (void *)&task2_stk[TASK_STACKSIZE-1],
                  TASK2_PRIORITY,
                  TASK2_PRIORITY,
                  task2_stk,
                  TASK_STACKSIZE,
                  NULL,
                  0);
  OSStart();
  return 0;
}

void led_on(int* base){
        INT8U err;
        OSMutexPend(tex,0,&err);
        *base = 0x01;
        OSMutexPost(tex);
        return;
}

void led_off(int* base){
        INT8U err;
        OSMutexPend(tex,0,&err);
        *base = 0x00;
        return;
}

void i2c_init(void){
        i2c_reg *reg = I2C_0_BASE;

        reg->i2c_scl_low = 1000;
        reg->i2c_scl_high = 1000;
        reg->i2c_sda_hold = 500;
        reg->i2c_tfr_cmd_fifo_lvl = 8;
        reg->i2c_rx_data_fifo_lvl = 8;
}

void eep_write(unsigned short addr, unsigned char data){
        i2c_reg* reg = I2C_0_BASE;
        INT8U err;
        unsigned char addr_low = addr >> 8;
        unsigned char addr_high = addr & 0x00FF;
        OSMutexPend(tex,0,&err);
        reg->i2c_ctrl = 0x1;
        reg->i2c_tfr_cmd = 0x2A0;
        reg->i2c_tfr_cmd = addr_high;
        reg->i2c_tfr_cmd = addr_low;
        reg->i2c_tfr_cmd = data | 0x100;

        while(reg->i2c_status != 0);
        reg->i2c_ctrl = 0x0;
        OSMutexPost(tex);
        OSTimeDlyHMSM(0, 0, 0, 5);
}

unsigned char eep_read(unsigned short addr){
        i2c_reg* reg = I2C_0_BASE;
        INT8U err;
        unsigned char addr_low = addr >> 8;
        unsigned char addr_high = addr & 0x00FF;
        unsigned char data;
        unsigned char tempData = 0;
        OSMutexPend(tex,0,&err);

        reg->i2c_ctrl = 0x1;
        reg->i2c_tfr_cmd = 0x2A0;
        reg->i2c_tfr_cmd = addr_high;
        reg->i2c_tfr_cmd = addr_low;
        reg->i2c_tfr_cmd = 0x2A1;
        reg->i2c_tfr_cmd = tempData | 0x100;


        while(reg->i2c_status != 0);
        data = reg->i2c_rxdata;
        reg->i2c_ctrl = 0x0;
        OSMutexPost(tex);
        return data;
}
```