Ryerson University

Department of Electrical, Computer, & Biomedical Engineering
Faculty of Engineering & Architectural Science

Course Title:	Systems-on-Chip Design
Course Number:	COE 838
Semester/Year (e.g.F2016)	W2022

Instructor:	Dr. Gul Khan

Assignment/Lab Number:	3
Assignment/Lab Title:	DE1-SoC Tutorial

Submission Date:	Monday, February 28th, 2022
Due Date:	Wednesday, March 2nd, 2022

Student LAST Name	Student FIRST Name	Student Number	Section	Signature*
Pano	Toni	500822828	01	T. P.

*By signing above you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a “0” on the work, an “F” in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at: http://www.ryerson.ca/senate/current/pol60.pdf

Contents

Appendix A: C++ Code for ARM DS-5 in Eclipse

A.i. Code Modified by Student

main.c

```
/*
 * main.c
 *
 *  Created on: 2014-11-15
 */

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <time.h>
#include <sys/mman.h>
#include <stdbool.h>
#include <pthread.h>
#include "hwlib.h"
#include "socal/socal.h"
#include "socal/hps.h"
#include "socal/alt_gpio.h"
#include "hps_0.h"
#include "led.h"
#include "seg7.h"
#include "sw.h"


#define LW_SIZE 0x00200000
#define LWHPS2FPGA_BASE 0xff200000

volatile uint32_t *h2p_lw_led_addr = NULL;
volatile uint32_t *h2p_lw_hex_addr = NULL;
volatile uint32_t *h2p_lw_sw_addr = NULL;

void led_blink(void){
       int i=0;
       int j;
       for(j = 0; j < 4; j++){
              printf("LED ON \r\n");
              for(i=0;i<=10;i++){
                     LEDR_LightCount(i);
                     usleep(100*1000);
              }

              printf("LED OFF \r\n");

              for(i=0;i<=10;i++){
                     LEDR_OffCount(i);
                     usleep(100*1000);
              }
       }
}

/*//from Cyclone V DE1 manual, page 26: https://www.ee.ryerson.ca/~courses/coe838/Data-
Sheets/DE1-SoC_User_Manual.pdf
 *   -0-
 *  |   |
 *  5   1
 *  |   |
 *   -6-
 *  |   |
 *  4   2
 *  |   |
 *   -3-
 * hex num = 6543210
 * */
static char op_symbol[4] = {'-', '+', '*', 'r'};
static uint8_t op_sseg[4] = { 0b1000000,     //minus : -
```

```
                                                            0b1110011,     //plus : p
                                                            0b1100011,     //multiply : * :
square
                                                            0b0100001}; //remainder : r
static uint8_t equals_sign = 0b1001000; //equals sign : =
static unsigned char szMap[] = {
        63, 6, 91, 79, 102, 109, 125, 7,
        127, 111, 119, 124, 57, 94, 121, 113
    };  // 0,1,2, ... 9, a, b, c, d, e, f

void alu_loop(void){
        uint32_t switches;
        uint8_t a, b, op, out;

        printf("in alu_loop()\n");
        while(1){
                //read switches
                switches = readSwitches();
                //printf("read switches = %x\n", switches);
                //printf("write to LEDs\n");
                //turn on LEDs for each switch that is on
                alt_write_word(h2p_lw_led_addr, switches);
                //switches = 0b1001000011;
                //printf("process switch states as inputs\n");
                //convert switches into variables they represent
                //0x02C0 = 0000 0011 1100 0000
                a = (uint8_t)((switches & 0x03C0) >> 6);
                //0x0030 = 0000 0000 0011 0000
                op = (uint8_t)((switches & 0x0030) >> 4);
                //0x000F = 0000 0000 0000 1111
                b = (uint8_t)(switches & 0x000F);

                //calculate result
                switch(op){
                        case 0b00: //subtract
                                out = a - b;
                                break;

                        case 0b01: //add
                                out = a + b;
                                break;

                        case 0b10: //multiply
                                out = a * b;
                                break;

                        case 0b11: //find remainder
                                if(b == 0){
                                        out = 0xFF;
                                }
                                else{
                                        out = a % b;
                                }
                                break;

                        default:
                                out = 0xAA;
                                break;
                }
                //printf("display result to 7segs\n");
                //show input number A on 7-segment display
                alt_write_word(h2p_lw_hex_addr+5, szMap[a]);
                //show opcode symbol on 7-segment display
                alt_write_word(h2p_lw_hex_addr+4, op_sseg[op]);
                //show input number B on 7-segment display
                alt_write_word(h2p_lw_hex_addr+3, szMap[b]);
                //show equals sign on 7-segment display
                alt_write_word(h2p_lw_hex_addr+2, equals_sign);

                //display output on last two ssegs of board
                if((op == 0b00) && (a < b)){ //subtract with negative result
```

```
                        //display minus in left sseg
                        alt_write_word(h2p_lw_hex_addr+1, op_sseg[0]);
                        //display magnitude of result (positive number) in right sseg
                        alt_write_word(h2p_lw_hex_addr, szMap[-out]);
                }
                else if((op == 0b11) && (b == 0)){ //remainder of modulus 0
                        //display minus in both ssegs
                        alt_write_word(h2p_lw_hex_addr+1, op_sseg[0]);
                        alt_write_word(h2p_lw_hex_addr, op_sseg[0]);
                }
                else{ //all other cases
                        //show upper half of operation output on 7-segment display
                        alt_write_word(h2p_lw_hex_addr+1, szMap[(out & 0xF0) >> 4]);
                        //show lower half of operation output on 7-segment display
                        alt_write_word(h2p_lw_hex_addr, szMap[out & 0x0F]);
                }

                //print ALU state to serial terminal
                printf("A op B = out -> %d %c %d = %d\n", a, op_symbol[op], b, out);
                usleep(100*1000);
        }
}


int main(int argc, char **argv){

        pthread_t id;
        int ret;
        void *virtual_base;
        int fd;


        //open the /dev/mem to access the FPGA space for reading and writing
        if( ( fd = open( "/dev/mem", ( O_RDWR | O_SYNC ) ) ) == -1 ) {
                printf( "ERROR: could not open \"/dev/mem\"...\n" );
                return( 1 );
        }

        //map the virtual memory space to virtual_base, that is 2MB in size (0x00200000), at
address LWHPS2FPGA_BASE
        virtual_base =  mmap( NULL, LW_SIZE, ( PROT_READ | PROT_WRITE ), MAP_SHARED, fd,
LWHPS2FPGA_BASE);

        //check that the mapping was successful
        if( virtual_base == MAP_FAILED ) {
                printf( "ERROR: mmap() failed...\n" );
                close( fd );
                return(1);
        }

        // map the address space for the LED and HEX registers into user space so we can interact
with them.
        // i.e. the address exists at virtual_base + the offset of your IP component
        h2p_lw_led_addr= virtual_base + ((uint32_t)(LED_PIO_BASE));
        h2p_lw_hex_addr= virtual_base + ( (uint32_t)(SEG7_IF_0_BASE));
        h2p_lw_sw_addr = virtual_base + ((uint32_t)(SW_PIO_BASE));

        //create and run the thread for the LED
        printf("LEDs\n");
        ret=pthread_create(&id,NULL,(void *)led_blink,NULL);

        if(ret!=0){
                printf("Creat pthread error!\n");
                exit(1);
        }

        //and run the SEVEN SEG process
        printf("7SEG\n");
        SEG7_All_Number();

        //once the SEG7 show is complete, display a message
        //display_msg();

        pthread_join(id,NULL);
          printf("main joined LED thread\n");

        //only do ALU functions after both LED light show and SEVEN SEG process have ended
        printf("alu\n");
        alu_loop();

        if( munmap( virtual_base, LW_SIZE) != 0 ) {
                printf( "ERROR: munmap() failed...\n" );
                close( fd );
                return( 1 );
        }

        close( fd );

        return 0;

}
```

sw.h

```
/*
 * sw.h
 *
 *  Created on: 2022-02-10
 *      Adapted from the led.h file for COE 838 Lab 3
 */

#ifndef SW_H_
#define SW_H_

#include "hwlib.h" //needed for bool type

uint32_t readSwitches(void);

#endif
```

sw.c

```
/*
 * sw.c
 *
 *  Created on: 2022-02-10
 *      Adapted from the led.c file for COE 838 Lab 3
 */

#include "sw.h"
#include "hwlib.h"
#include "socal/socal.h"
#include "socal/hps.h"
#include "socal/alt_gpio.h"

extern volatile unsigned long* h2p_lw_sw_addr;

uint32_t readSwitches(void){
        uint32_t sw = alt_read_word(h2p_lw_sw_addr);

        //bits 0 to 9 are switch states, bits 10 to 15 are not switches and should be set to 0
        uint16_t bitMask = 0x000003ff;
        sw = sw & bitMask;

        return sw;
}
```

A.ii. Header Files generated by QSys

The files “soc_system.h”, “hps_0_arm_a9_0.h”, “hps_0_arm_a9_1.h”, and “hps_0_bridges.h” generated by QSys are each longer than 500 lines. In order to keep this report short, the shortest file generated by QSys, “hps_0.h”, has been included in this report. The rest of the files shall be submitted in a .zip folder with this report.

hps_0.h

```
#ifndef _ALTERA_HPS_0_H_
#define _ALTERA_HPS_0_H_

/*
 * This file was automatically generated by the swinfo2header utility.
 *
 * Created from SOPC Builder system 'soc_system' in
 * file './soc_system.sopcinfo'.
 */

/*
 * This file contains macros for module 'hps_0' and devices
 * connected to the following masters:
 *   h2f_axi_master
 *   h2f_lw_axi_master
 *
 * Do not include this header file and another header file created for a
 * different module or master group at the same time.
 * Doing so may result in duplicate macro names.
 * Instead, use the system header file which has macros with unique names.
 */

/*
 * Macros for device 'SEG7_IF_0', class 'SEG7_IF'
 * The macros are prefixed with 'SEG7_IF_0_'.
 * The prefix is the slave descriptor.
 */
#define SEG7_IF_0_COMPONENT_TYPE SEG7_IF
#define SEG7_IF_0_COMPONENT_NAME SEG7_IF_0
#define SEG7_IF_0_BASE 0x0
#define SEG7_IF_0_SPAN 64
#define SEG7_IF_0_END 0x3f

/*
 * Macros for device 'sw_pio', class 'altera_avalon_pio'
 * The macros are prefixed with 'SW_PIO_'.
 * The prefix is the slave descriptor.
 */
#define SW_PIO_COMPONENT_TYPE altera_avalon_pio
#define SW_PIO_COMPONENT_NAME sw_pio
#define SW_PIO_BASE 0x20
#define SW_PIO_SPAN 32
#define SW_PIO_END 0x3f
#define SW_PIO_BIT_CLEARING_EDGE_REGISTER 0
#define SW_PIO_BIT_MODIFYING_OUTPUT_REGISTER 0
#define SW_PIO_CAPTURE 0
#define SW_PIO_DATA_WIDTH 10
#define SW_PIO_DO_TEST_BENCH_WIRING 0
#define SW_PIO_DRIVEN_SIM_VALUE 0
#define SW_PIO_EDGE_TYPE NONE
#define SW_PIO_FREQ 50000000
#define SW_PIO_HAS_IN 1
#define SW_PIO_HAS_OUT 0
#define SW_PIO_HAS_TRI 0
#define SW_PIO_IRQ_TYPE NONE
#define SW_PIO_RESET_VALUE 0

/*
```

```
 * Macros for device 'led_pio', class 'altera_avalon_pio'
 * The macros are prefixed with 'LED_PIO_'.
 * The prefix is the slave descriptor.
 */
#define LED_PIO_COMPONENT_TYPE altera_avalon_pio
#define LED_PIO_COMPONENT_NAME led_pio
#define LED_PIO_BASE 0x30
#define LED_PIO_SPAN 32
#define LED_PIO_END 0x4f
#define LED_PIO_BIT_CLEARING_EDGE_REGISTER 0
#define LED_PIO_BIT_MODIFYING_OUTPUT_REGISTER 0
#define LED_PIO_CAPTURE 0
#define LED_PIO_DATA_WIDTH 10
#define LED_PIO_DO_TEST_BENCH_WIRING 0
#define LED_PIO_DRIVEN_SIM_VALUE 0
#define LED_PIO_EDGE_TYPE NONE
#define LED_PIO_FREQ 50000000
#define LED_PIO_HAS_IN 0
#define LED_PIO_HAS_OUT 1
#define LED_PIO_HAS_TRI 0
#define LED_PIO_IRQ_TYPE NONE
#define LED_PIO_RESET_VALUE 0


#endif /* _ALTERA_HPS_0_H_ */
```

Appendix B: VHDL Code for Quartus II 14.0

a) Code Modified by Student

LED_HEX_FPGA.vhd

```
-----------------------------
-- HED_LED_FPGA Tutorial
-- Lab 3
-- COE838 Systems-on-Chip Design
-----------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

ENTITY LED_HEX_FPGA IS
	PORT( CLOCK_50, HPS_DDR3_RZQ,HPS_ENET_RX_CLK, HPS_ENET_RX_DV		: IN
STD_LOGIC;
		HPS_DDR3_ADDR						: OUT
STD_LOGIC_VECTOR(14 DOWNTO 0);
		HPS_DDR3_BA						: OUT
STD_LOGIC_VECTOR(2 DOWNTO 0);
		HPS_DDR3_CS_N							: OUT
STD_LOGIC;
		HPS_DDR3_CK_P, HPS_DDR3_CK_N, HPS_DDR3_CKE		: OUT STD_LOGIC;
		HPS_USB_DIR, HPS_USB_NXT, HPS_USB_CLKOUT		: IN STD_LOGIC;
		HPS_ENET_RX_DATA					: IN
STD_LOGIC_VECTOR(3 DOWNTO 0);
		HPS_SD_DATA, HPS_DDR3_DQS_N, HPS_DDR3_DQS_P			: INOUT
STD_LOGIC_VECTOR(3 DOWNTO 0);
		HPS_ENET_MDIO							: INOUT
STD_LOGIC;
		HPS_USB_DATA						: INOUT
STD_LOGIC_VECTOR(7 DOWNTO 0);
		HPS_DDR3_DQ						: INOUT
STD_LOGIC_VECTOR(31 DOWNTO 0);
		HPS_SD_CMD						: INOUT STD_LOGIC;
		HPS_ENET_TX_DATA, HPS_DDR3_DM					: OUT
STD_LOGIC_VECTOR(3 DOWNTO 0);
		HPS_DDR3_ODT, HPS_DDR3_RAS_N, HPS_DDR3_RESET_N		: OUT STD_LOGIC;
		HPS_DDR3_CAS_N, HPS_DDR3_WE_N					: OUT
STD_LOGIC;
		HPS_ENET_MDC, HPS_ENET_TX_EN					: OUT
STD_LOGIC;
		LEDR							: OUT
STD_LOGIC_VECTOR(9 DOWNTO 0);
		HEX0, HEX1, HEX2, HEX3, HEX4, HEX5			: BUFFER
STD_LOGIC_VECTOR(6 DOWNTO 0);
		HPS_USB_STP, HPS_SD_CLK, HPS_ENET_GTX_CLK		: OUT STD_LOGIC;
		SW								: IN
STD_LOGIC_VECTOR(9 DOWNTO 0));
END LED_HEX_FPGA;

ARCHITECTURE Behaviour OF LED_HEX_FPGA IS

	--instantiate the soc_systtem component here

    component soc_system is
        port (
            clk_clk                          : in    std_logic                     := 'X';
-- clk
            hps_0_h2f_reset_reset_n          : out   std_logic;
-- reset_n
            hps_io_hps_io_emac1_inst_TX_CLK  : out   std_logic;
-- hps_io_emac1_inst_TX_CLK
            hps_io_hps_io_emac1_inst_TXD0    : out   std_logic;
-- hps_io_emac1_inst_TXD0
            hps_io_hps_io_emac1_inst_TXD1    : out   std_logic;
-- hps_io_emac1_inst_TXD1
            hps_io_hps_io_emac1_inst_TXD2    : out   std_logic;
-- hps_io_emac1_inst_TXD2
```

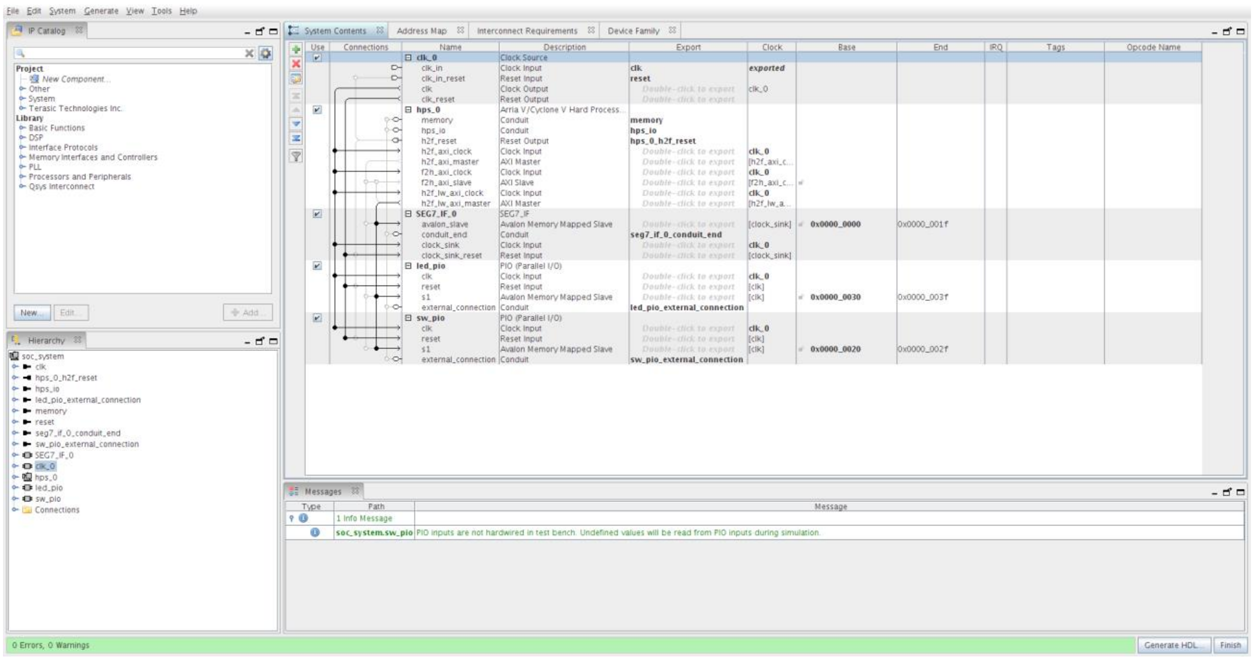
```
            hps_io_hps_io_emac1_inst_TXD3      : out   std_logic;
-- hps_io_emac1_inst_TXD3
            hps_io_hps_io_emac1_inst_RXD0      : in    std_logic                     := 'X';
-- hps_io_emac1_inst_RXD0
            hps_io_hps_io_emac1_inst_MDIO      : inout std_logic                     := 'X';
-- hps_io_emac1_inst_MDIO
            hps_io_hps_io_emac1_inst_MDC       : out   std_logic;
-- hps_io_emac1_inst_MDC
            hps_io_hps_io_emac1_inst_RX_CTL    : in    std_logic                     := 'X';
-- hps_io_emac1_inst_RX_CTL
            hps_io_hps_io_emac1_inst_TX_CTL    : out   std_logic;
-- hps_io_emac1_inst_TX_CTL
            hps_io_hps_io_emac1_inst_RX_CLK    : in    std_logic                     := 'X';
-- hps_io_emac1_inst_RX_CLK
            hps_io_hps_io_emac1_inst_RXD1      : in    std_logic                     := 'X';
-- hps_io_emac1_inst_RXD1
            hps_io_hps_io_emac1_inst_RXD2      : in    std_logic                     := 'X';
-- hps_io_emac1_inst_RXD2
            hps_io_hps_io_emac1_inst_RXD3      : in    std_logic                     := 'X';
-- hps_io_emac1_inst_RXD3
            hps_io_hps_io_sdio_inst_CMD        : inout std_logic                     := 'X';
-- hps_io_sdio_inst_CMD
            hps_io_hps_io_sdio_inst_D0         : inout std_logic                     := 'X';
-- hps_io_sdio_inst_D0
            hps_io_hps_io_sdio_inst_D1         : inout std_logic                     := 'X';
-- hps_io_sdio_inst_D1
            hps_io_hps_io_sdio_inst_CLK        : out   std_logic;
-- hps_io_sdio_inst_CLK
            hps_io_hps_io_sdio_inst_D2         : inout std_logic                     := 'X';
-- hps_io_sdio_inst_D2
            hps_io_hps_io_sdio_inst_D3         : inout std_logic                     := 'X';
-- hps_io_sdio_inst_D3
            hps_io_hps_io_usb1_inst_D0         : inout std_logic                     := 'X';
-- hps_io_usb1_inst_D0
            hps_io_hps_io_usb1_inst_D1         : inout std_logic                     := 'X';
-- hps_io_usb1_inst_D1
            hps_io_hps_io_usb1_inst_D2         : inout std_logic                     := 'X';
-- hps_io_usb1_inst_D2
            hps_io_hps_io_usb1_inst_D3         : inout std_logic                     := 'X';
-- hps_io_usb1_inst_D3
            hps_io_hps_io_usb1_inst_D4         : inout std_logic                     := 'X';
-- hps_io_usb1_inst_D4
            hps_io_hps_io_usb1_inst_D5         : inout std_logic                     := 'X';
-- hps_io_usb1_inst_D5
            hps_io_hps_io_usb1_inst_D6         : inout std_logic                     := 'X';
-- hps_io_usb1_inst_D6
            hps_io_hps_io_usb1_inst_D7         : inout std_logic                     := 'X';
-- hps_io_usb1_inst_D7
            hps_io_hps_io_usb1_inst_CLK        : in    std_logic                     := 'X';
-- hps_io_usb1_inst_CLK
            hps_io_hps_io_usb1_inst_STP        : out   std_logic;
-- hps_io_usb1_inst_STP
            hps_io_hps_io_usb1_inst_DIR        : in    std_logic                     := 'X';
-- hps_io_usb1_inst_DIR
            hps_io_hps_io_usb1_inst_NXT        : in    std_logic                     := 'X';
-- hps_io_usb1_inst_NXT
            memory_mem_a                       : out   std_logic_vector(14 downto 0);
-- mem_a
            memory_mem_ba                      : out   std_logic_vector(2 downto 0);
-- mem_ba
            memory_mem_ck                      : out   std_logic;
-- mem_ck
            memory_mem_ck_n                    : out   std_logic;
-- mem_ck_n
            memory_mem_cke                     : out   std_logic;
-- mem_cke
            memory_mem_cs_n                    : out   std_logic;
-- mem_cs_n
            memory_mem_ras_n                   : out   std_logic;
-- mem_ras_n
```

```
            memory_mem_cas_n                   : out   std_logic;
-- mem_cas_n
            memory_mem_we_n                    : out   std_logic;
-- mem_we_n
            memory_mem_reset_n                 : out   std_logic;
-- mem_reset_n
            memory_mem_dq                      : inout std_logic_vector(31 downto 0) := (others
=> 'X'); -- mem_dq
            memory_mem_dqs                     : inout std_logic_vector(3 downto 0)  := (others
=> 'X'); -- mem_dqs
            memory_mem_dqs_n                   : inout std_logic_vector(3 downto 0)  := (others
=> 'X'); -- mem_dqs_n
            memory_mem_odt                     : out   std_logic;
-- mem_odt
            memory_mem_dm                      : out   std_logic_vector(3 downto 0);
-- mem_dm
            memory_oct_rzqin                   : in    std_logic                     := 'X';
-- oct_rzqin
            reset_reset_n                      : in    std_logic                     := 'X';
-- reset_n
            led_pio_external_connection_export : out   std_logic_vector(9 downto 0);
-- export
            seg7_if_0_conduit_end_export       : out   std_logic_vector(47 downto 0);
-- export
                              sw_pio_external_connection_export  : in    std_logic_vector(9
downto 0)  := (others => 'X') -- export
        );
    end component soc_system;

       --SIGNALS instantiated here
       SIGNAL hex5_tmp, hex4_tmp, hex3_tmp, hex2_tmp, hex1_tmp, hex0_tmp,
hps_0_h2f_reset_reset_n :
       STD_LOGIC;

       BEGIN

       --port map soc_system here
       u0 : component soc_system
                      port map (
                                     clk_clk => CLOCK_50,
                                     reset_reset_n => '1',memory_mem_a => HPS_DDR3_ADDR,
                                     memory_mem_ba => HPS_DDR3_BA,
                                     memory_mem_ck => HPS_DDR3_CK_P,
                                     memory_mem_ck_n => HPS_DDR3_CK_N,
                                     memory_mem_cke => HPS_DDR3_CKE,
                                     memory_mem_cs_n => HPS_DDR3_CS_N,
                                     memory_mem_ras_n => HPS_DDR3_RAS_N,
                                     memory_mem_cas_n => HPS_DDR3_CAS_N,
                                     memory_mem_we_n => HPS_DDR3_WE_N,
                                     memory_mem_reset_n => HPS_DDR3_RESET_N,
                                     memory_mem_dq => HPS_DDR3_DQ,
                                     memory_mem_dqs => HPS_DDR3_DQS_P,
                                     memory_mem_dqs_n => HPS_DDR3_DQS_N,
                                     memory_mem_odt => HPS_DDR3_ODT,
                                     memory_mem_dm => HPS_DDR3_DM,
                                     memory_oct_rzqin => HPS_DDR3_RZQ,
                                     hps_io_hps_io_emac1_inst_TX_CLK => HPS_ENET_GTX_CLK,
                                     hps_io_hps_io_emac1_inst_TXD0 => HPS_ENET_TX_DATA(0),
                                     hps_io_hps_io_emac1_inst_TXD1 => HPS_ENET_TX_DATA(1),
                                     hps_io_hps_io_emac1_inst_TXD2 => HPS_ENET_TX_DATA(2),
                                     hps_io_hps_io_emac1_inst_TXD3 => HPS_ENET_TX_DATA(3),
                                     hps_io_hps_io_emac1_inst_RXD0 => HPS_ENET_RX_DATA(0),
                                     hps_io_hps_io_emac1_inst_MDIO => HPS_ENET_MDIO,
                                     hps_io_hps_io_emac1_inst_MDC => HPS_ENET_MDC,
                                     hps_io_hps_io_emac1_inst_RX_CTL => HPS_ENET_RX_DV,
                                     hps_io_hps_io_emac1_inst_TX_CTL => HPS_ENET_TX_EN,
                                     hps_io_hps_io_emac1_inst_RX_CLK => HPS_ENET_RX_CLK,
                                     hps_io_hps_io_emac1_inst_RXD1 => HPS_ENET_RX_DATA(1),
                                     hps_io_hps_io_emac1_inst_RXD2 => HPS_ENET_RX_DATA(2),
                                     hps_io_hps_io_emac1_inst_RXD3 => HPS_ENET_RX_DATA(3),
                                     hps_io_hps_io_sdio_inst_CMD => HPS_SD_CMD,
                                     hps_io_hps_io_sdio_inst_D0 => HPS_SD_DATA(0),
                                     hps_io_hps_io_sdio_inst_D1 => HPS_SD_DATA(1),
                                     hps_io_hps_io_sdio_inst_CLK => HPS_SD_CLK,
                                     hps_io_hps_io_sdio_inst_D2 => HPS_SD_DATA(2),
                                     hps_io_hps_io_sdio_inst_D3 => HPS_SD_DATA(3),
                                     hps_io_hps_io_usb1_inst_D0 => HPS_USB_DATA(0),
                                     hps_io_hps_io_usb1_inst_D1 => HPS_USB_DATA(1),
                                     hps_io_hps_io_usb1_inst_D2 => HPS_USB_DATA(2),
                                     hps_io_hps_io_usb1_inst_D3 => HPS_USB_DATA(3),
                                     hps_io_hps_io_usb1_inst_D4 => HPS_USB_DATA(4),
                                     hps_io_hps_io_usb1_inst_D5 => HPS_USB_DATA(5),
                                     hps_io_hps_io_usb1_inst_D6 => HPS_USB_DATA(6),
                                     hps_io_hps_io_usb1_inst_D7 => HPS_USB_DATA(7),
                                     hps_io_hps_io_usb1_inst_CLK => HPS_USB_CLKOUT,
                                     hps_io_hps_io_usb1_inst_STP => HPS_USB_STP,
                                     hps_io_hps_io_usb1_inst_DIR =>
HPS_USB_DIR,hps_io_hps_io_usb1_inst_NXT => HPS_USB_NXT,
                                     hps_0_h2f_reset_reset_n => hps_0_h2f_reset_reset_n,
                                     led_pio_external_connection_export => LEDR,
                                     seg7_if_0_conduit_end_export(47) => hex5_tmp,
                                     seg7_if_0_conduit_end_export(46 DOWNTO 40)=>HEX5,
                                     seg7_if_0_conduit_end_export(39) =>hex4_tmp,
                                     seg7_if_0_conduit_end_export(38 DOWNTO 32)=>HEX4,
                                     seg7_if_0_conduit_end_export(31) =>hex3_tmp,
                                     seg7_if_0_conduit_end_export(30 DOWNTO 24)=>HEX3,
                                     seg7_if_0_conduit_end_export(23) =>hex2_tmp,
                                     seg7_if_0_conduit_end_export(22 DOWNTO 16)=> HEX2,
                                     seg7_if_0_conduit_end_export(15) => hex1_tmp,
                                     seg7_if_0_conduit_end_export(14 DOWNTO 8) => HEX1,
                                     seg7_if_0_conduit_end_export(7) => hex0_tmp,
                                     seg7_if_0_conduit_end_export(6 DOWNTO 0) =>HEX0,
                                     sw_pio_external_connection_export  => SW
                        );
End Behaviour;
```

b) Unmodified Code given for Lab

The rest of the code given for the lab contains many files, and would make this report even longer. To keep this report short, the rest of the code shall be submitted in a .zip file along with this report.

Appendix C: QSys Setup



Use	Connections	Name	Description	Export	Clock	Base	End	IRQ	Tags	Opcode Name
✓		⊟ **clk_0**	Clock Source							
		clk_in	Clock Input	**clk**	***exported***					
		clk_in_reset	Reset Input	**reset**						
		clk	Clock Output	*Double-click to export*	clk_0					
		clk_reset	Reset Output	*Double-click to export*						
✓		⊟ **hps_0**	Arria V/Cyclone V Hard Process...							
		memory	Conduit	**memory**						
		hps_io	Conduit	**hps_io**						
		h2f_reset	Reset Output	**hps_0_h2f_reset**						
		h2f_axi_clock	Clock Input	*Double-click to export*	**clk_0**					
		h2f_axi_master	AXI Master	*Double-click to export*	[h2f_axi_c...					
		f2h_axi_clock	Clock Input	*Double-click to export*	**clk_0**					
		f2h_axi_slave	AXI Slave	*Double-click to export*	[f2h_axi_c...	✓				
		h2f_lw_axi_clock	Clock Input	*Double-click to export*	**clk_0**					
		h2f_lw_axi_master	AXI Master	*Double-click to export*	[h2f_lw_a...					
✓		⊟ **SEG7_IF_0**	SEG7_IF							
		avalon_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clock_sink]	✓ **0x0000_0000**	0x0000_001f			
		conduit_end	Conduit	**seg7_if_0_conduit_end**						
		clock_sink	Clock Input	*Double-click to export*	**clk_0**					
		clock_sink_reset	Reset Input	*Double-click to export*	[clock_sink]					
✓		⊟ **led_pio**	PIO (Parallel I/O)							
		clk	Clock Input	*Double-click to export*	**clk_0**					
		reset	Reset Input	*Double-click to export*	[clk]					
		s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	✓ **0x0000_0030**	0x0000_003f			
		external_connection	Conduit	**led_pio_external_connection**						
✓		⊟ **sw_pio**	PIO (Parallel I/O)							
		clk	Clock Input	*Double-click to export*	**clk_0**					
		reset	Reset Input	*Double-click to export*	[clk]					
		s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	✓ **0x0000_0020**	0x0000_002f			
		external_connection	Conduit	**sw_pio_external_connection**						

Type	Path	Message
●	1 Info Message	
●	**soc_system.sw_pio**	PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.

Figure 1. QSys system with connections to IP blocks for 7-segment displays, LEDs, and switches.