Lab 2 – LCD display and external memory interfacing

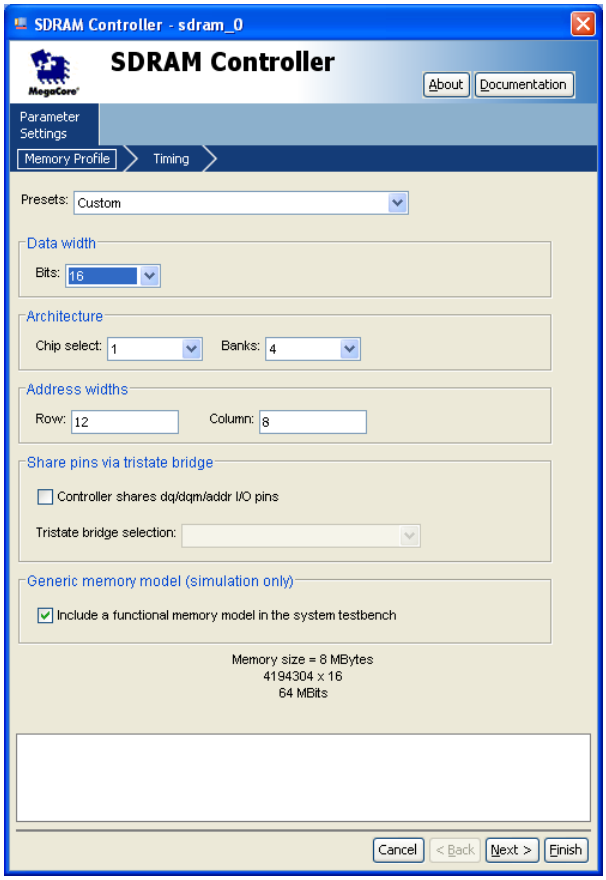
Objective: Create an interface to the LCD display, internal timer functions, and interface with the SDRAM memory as well as on chip memory.

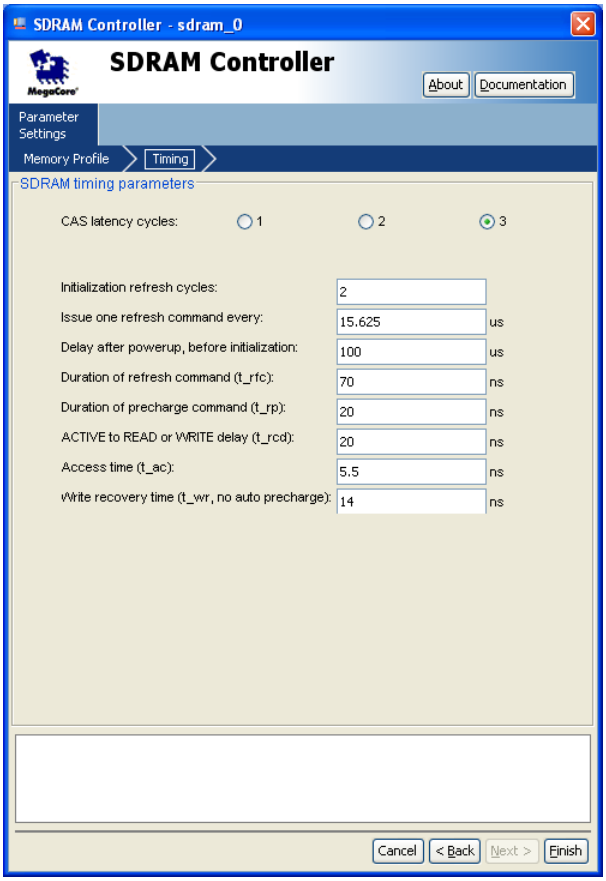
Topics Covered:

- Design Entry with Quartus II
- Implementation of soft core with SOPC builder
- Interface requirements for LCD display
- Interface requirements for external memory
- Configuration of internal timers
- Programming with the NIOS II IDE

Procedure:

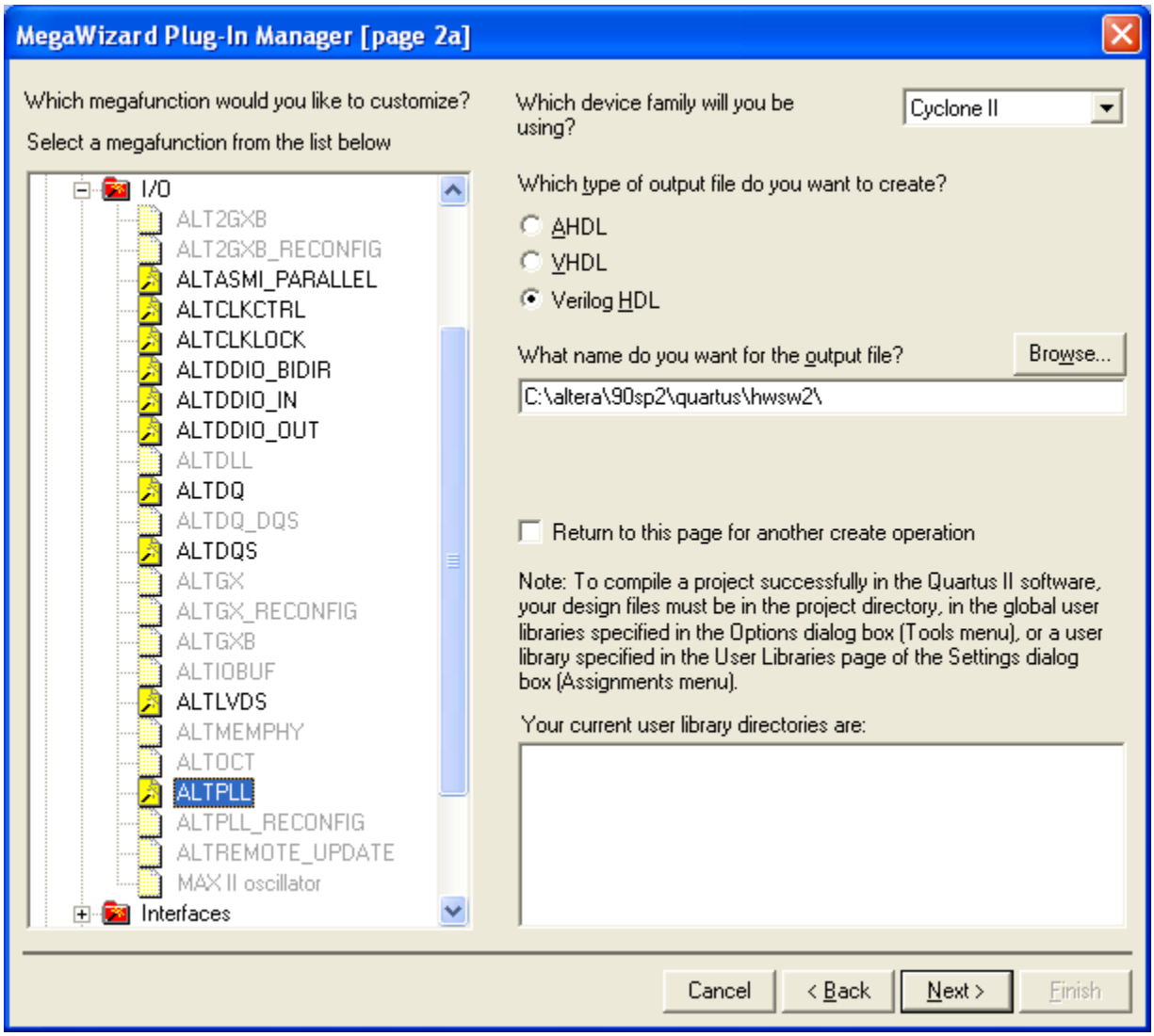
Create a new project in Quartus II and implement a small NIOS II core with JTAG level 1, LCD interface, timer, PIO for switches and lights with on chip memory at 32k for the program and an SDRAM configured as shown below.





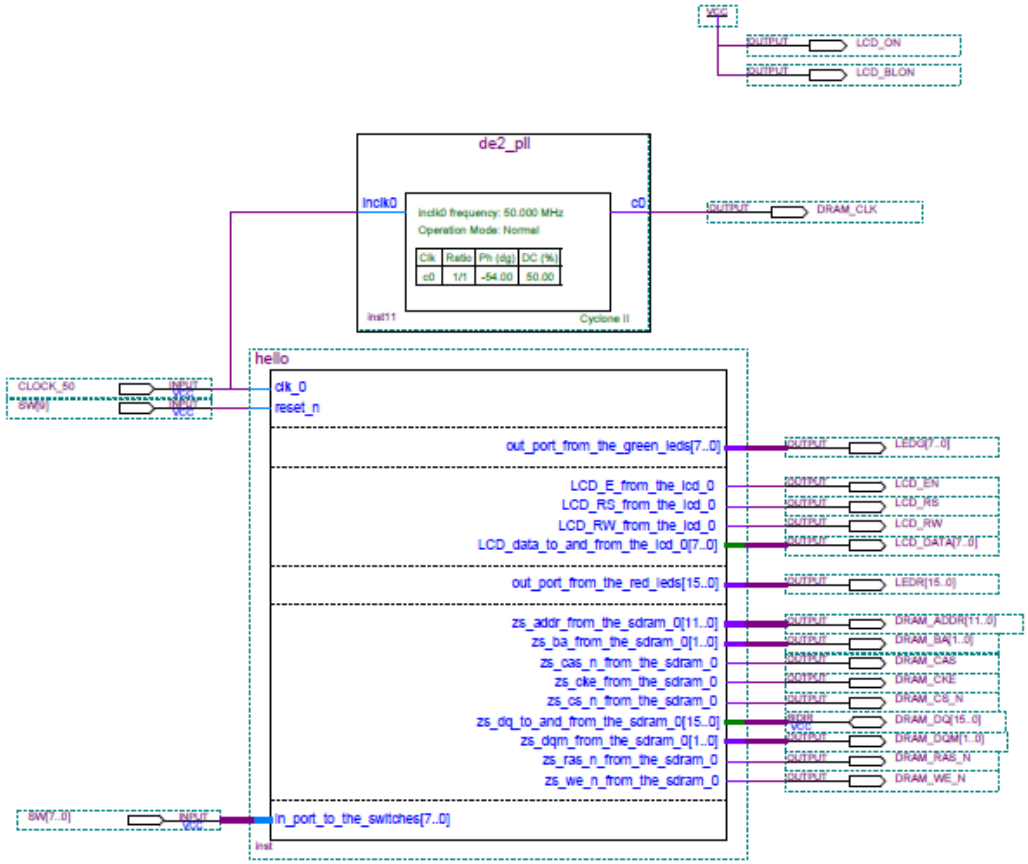
Remember to configure the reset and exception vectors to the on chip ram after using the SYSTEM menu to auto-assign the base addresses.

A Phase-Locked Loop component is required for interfacing the SDRAM since the NIOS II processor core operates on different clock edges. The SDRAM needs a clock signal that is phase shifted by 180 degrees. An inverter can do this but the phase shift also needs to be adjusted a bit to correct for the internal FPGA delays and the distance between the SDRAM and the FPGA on the DE board.



Enter a name for the file and on the third page select 50.00 MHz as the frequency of the inclock0 input. The rest of the settings should remain as default. On page 4 of the MegaWizard manager, un-select all checkmarks. Click NEXT twice to get to page 6 and enter a Clock phase shift of -54 deg (-3ns). Leave the other options as the default. Click Finish.

Connect the core in Quartus to the external input and output pins and use the naming conventions in the “DE2_pin_assignments.csv” file so that you can import the FPGA pin locations.



Be sure to double check the names of the labels correspond to the SDRAM pins to make sure the assignments are made properly. If they are not assigned properly the FPGA will not route the signals to the proper location and the memory will not be properly controlled.

Scan the output during compilation and also open the pin assignment editor to ensure the I/O pins are properly assigned. You should also be able to find the DE2 data sheet online to verify the proper pin assignments for all of the peripheral devices. This will be useful for future reference.

The data sheet for the LCD is “CFAH1602BTMCJP.pdf” if you can’t find it in your directory you should search the Altera website to get it since the timing for the initialization and control of the LCD is outlined.

Compile this file and load it into the DE2 board for programming in NIOS II. Build the syslib for the new processor core then create a C file in the main folder with the following code.

Your new NIOS core uses the LCD display on the DE2 board for display. There are two functions defined for this new peripheral. The first initializes the device and follows the steps outlined in the data sheet for the LCD device on the DE2 board. The second function tests the operation and control of the device and displays a simple HELLO WORLD output.

```c
#include <stdio.h>
#include <unistd.h>
#include "system.h"
#include "alt_types.h"
#include "altera_avalon_timer_regs.h"
#include "altera_avalon_pio_regs.h"
#include "altera_avalon_lcd_16207_regs.h"

#define LCD_WR_COMMAND_REG 0
#define LCD_RD_STATUS_REG 1
#define LCD_WR_DATA_REG 2
#define LCD_RD_DATA_REG 3

void lcd_init( void) {

    usleep(15000); /* Wait for more than 15 ms before init */

    /* Set function code four times -- 8-bit, 2 line, 5x7 mode */
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x38);
    usleep(4100);  /* Wait for more than 4.1 ms */
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x38);
    usleep(100);   /* Wait for more than 100 us */
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x38);
    usleep(5000);  /* Wait for more than 100 us */
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x38);
    usleep(100);   /* Wait for more than 100 us */

    /* Set Display to OFF*/
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x08);
    usleep(100);

    /* Set Display to ON */
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x0C);
    usleep(100);

    /* Set Entry Mode -- Cursor increment, display doesn't shift */
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x06);
    usleep(100);

    /* Set the Cursor to the home position */
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x02);
    usleep(2000);

    /* Display clear */
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x01);
    usleep(2000);
}

alt_u32 test_lcd( void) {
    int i;
    char message[17] = "Hello World...   ";
    char done[15] = "Done!          ";

     /* Write a simple message on the first line. */
        for(i = 0; i < 16; i++) {
            IOWR(LCD_0_BASE, LCD_WR_DATA_REG,message[i]);
            usleep(100);
```

```
        }
        /* Count along the bottom row */
        /* Set Address */
        IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG,0xC0);
        usleep(1000);

        /* Display Count */
        for (i = 0; i < 10; i++) {
            IOWR(LCD_0_BASE, LCD_WR_DATA_REG, (char)(i+0x30) );
            usleep(500000); /* Wait 0.5 sec */
        }

        /* Write "Done!" message on first line. */
        /* Set Address */
        IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG,0x80);
        usleep(1000);

        /* Write data */
        for(i = 0; i < 14; i++) {
            IOWR(LCD_0_BASE, LCD_WR_DATA_REG, done[i]);
            usleep(100);
        }
        return(0);
}

alt_u32 wait_onesec( void) {
    IOWR_ALTERA_AVALON_TIMER_PERIODL( TIMER_0_BASE,(48000000 & 0xFFFF));
    IOWR_ALTERA_AVALON_TIMER_PERIODH( TIMER_0_BASE,((48000000>>16) &
0xFFFF));
    IOWR_ALTERA_AVALON_TIMER_STATUS ( TIMER_0_BASE, 0);
    IOWR_ALTERA_AVALON_TIMER_CONTROL ( TIMER_0_BASE, 0x4);
     while((IORD_ALTERA_AVALON_TIMER_STATUS(TIMER_0_BASE) &
                ALTERA_AVALON_TIMER_STATUS_TO_MSK) == 0) {}
    return(0);
}

int main( void) {
    int count;

    lcd_init();
    test_lcd();
    count = 0;
    while (1) {
        count = count & 0xff;
        IOWR_ALTERA_AVALON_PIO_DATA( RED_LEDS_BASE, count);
        wait_onesec();
        IOWR_ALTERA_AVALON_PIO_DATA( RED_LEDS_BASE, count++);
    }
    return (0);
}
```

When the program runs you should see the LCD reset and show “HELLO WORLD” then count and display DONE.

After the LCD finishes the RED leds should flash off and on with a one second frequency determined by the internal timer. This frequency is defined by the function **wait_onesec**

Verification: _____ part 1

The next step is to work with the SDRAM and write a test program to verify the operation of the memory as well as program new capabilities into the LCD display.

Use switches SW[3..0] to change between the LCD test, SDRAM test, timer and count functions.

```
#include <stdio.h>
#include <unistd.h>
#include "system.h"
#include "alt_types.h"
#include "sys/alt_irq.h"
#include "altera_avalon_timer_regs.h"
#include "altera_avalon_pio_regs.h"
#include "altera_avalon_lcd_16207_regs.h"

#define LCD_WR_COMMAND_REG 0
#define LCD_RD_STATUS_REG 1
#define LCD_WR_DATA_REG 2
#define LCD_RD_DATA_REG 3

#define SDRAM_MAX_WORDS 100

void lcd_init( void) {

    usleep(15000); /* Wait for more than 15 ms before init */

    /* Set function code four times -- 8-bit, 2 line, 5x7 mode */
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x38);
    usleep(4100);  /* Wait for more than 4.1 ms */
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x38);
    usleep(100);   /* Wait for more than 100 us */
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x38);
    usleep(5000);  /* Wait for more than 100 us */
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x38);
    usleep(100);   /* Wait for more than 100 us */

    /* Set Display to OFF*/
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x08);
    usleep(100);

    /* Set Display to ON */
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x0C);
    usleep(100);

    /* Set Entry Mode -- Cursor increment, display doesn't shift */
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x06);
    usleep(100);

    /* Set the Cursor to the home position */
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x02);
    usleep(2000);
```

```
    /* Display clear */
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x01);
    usleep(2000);
}

alt_u32 test_lcd( void) {
    int i;
    char message[17] = "Hello World...   ";
    char done[15] = "Done!          ";

      /* Set the Cursor to the home position */
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x02);
    usleep(2000);

    /* Display clear */
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x01);
    usleep(2000);

    /* Write a simple message on the first line. */
        for(i = 0; i < 16; i++) {
            IOWR(LCD_0_BASE, LCD_WR_DATA_REG,message[i]);
            usleep(100);
        }
        /* Count along the bottom row */
        /* Set Address */
        IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG,0xC0);
        usleep(1000);

        /* Display Count */
        for (i = 0; i < 10; i++) {
            IOWR(LCD_0_BASE, LCD_WR_DATA_REG, (char)(i+0x30) );
            usleep(500000); /* Wait 0.5 sec */
        }

        /* Write "Done!" message on first line. */
        /* Set Address */
        IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG,0x80);
        usleep(1000);

        /* Write data */
        for(i = 0; i < 14; i++) {
            IOWR(LCD_0_BASE, LCD_WR_DATA_REG, done[i]);
            usleep(100);
        }
        return(0);
}

alt_u32 lcd_sdram_message(void) {
    int i;
    char message[13] = "SDRAM Testing";

     /* Set the Cursor to the home position */
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x02);
    usleep(2000);

    /* Display clear */
```

```
    IOWR(LCD_0_BASE, LCD_WR_COMMAND_REG, 0x01);
    usleep(2000);

     /* Write the testing message on the first line. */
        for(i = 0; i < 10; i++) {
            IOWR(LCD_0_BASE, LCD_WR_DATA_REG,message[i]);
            usleep(100);
        }
    return(0);
}

alt_u32 lcd_sdram_message2(void) {
    int i;
    char message[10] = "Errors =  ";

    /* Set Address to bottom row*/

// Write CODE TO perform this function
// Be sure to wait after the function for the necessary delay time


     /* Write the result message. */

// Write the CODE to perform this function
// Be sure to wait after each character for the necessary delay time

    return(0);
}

alt_u32 clear_lcd(void) {

    /* Set the Cursor to the home position */

// Write the CODE to perform this function
// Be sure to wait after each character for the necessary delay time

    /* Display clear */

// Write the CODE to perform this function
// Be sure to wait after each character for the necessary delay time

    return(0);
}

alt_u32 wait_onesec( void) {
    IOWR_ALTERA_AVALON_TIMER_PERIODL( TIMER_0_BASE,(48000000 & 0xFFFF));
    IOWR_ALTERA_AVALON_TIMER_PERIODH( TIMER_0_BASE,((48000000>>16) &
0xFFFF));
    IOWR_ALTERA_AVALON_TIMER_STATUS ( TIMER_0_BASE, 0);
    IOWR_ALTERA_AVALON_TIMER_CONTROL ( TIMER_0_BASE, 0x4);
     while((IORD_ALTERA_AVALON_TIMER_STATUS(TIMER_0_BASE) &
                ALTERA_AVALON_TIMER_STATUS_TO_MSK) == 0) {}
    return(0);
}

alt_u32 test_sdram( void){
    alt_u32 i;
```

```
    alt_u32 errors=0;
    alt_u32 *buffer = (alt_u32 *)SDRAM_0_BASE;

    /* Write data to SDRAM */
    for (i=0;i<SDRAM_MAX_WORDS;i++) {
        buffer[i] = (i + 1000000);
        usleep(5000);
        IOWR_ALTERA_AVALON_PIO_DATA( RED_LEDS_BASE,i);
    }

    lcd_sdram_message2();

    /* Check output from SDRAM */
    for (i=0;i<SDRAM_MAX_WORDS;i++) {
        if(buffer[i] != (i+1000000))
            errors++;
    }
    return(errors);
}

int main( void) {
   unsigned int number;
   int i;
   alt_u32 ret_val;

    lcd_init();

    while(1) {

      number = IORD_ALTERA_AVALON_PIO_DATA(SWITCHES_BASE);
      IOWR_ALTERA_AVALON_PIO_DATA(GREEN_LEDS_BASE,(number & 0x0F));

        switch(number) {
            case 0x1:   /* test the LCD */
                ret_val = test_lcd();
                usleep(500000);
                clear_lcd();
                break;
            case 0x2:   /* test the SDRAM */
                lcd_sdram_message();
                ret_val=test_sdram();
                IOWR(LCD_0_BASE, LCD_WR_DATA_REG, (char)(ret_val+0x30) );
                usleep(500000);
                IOWR_ALTERA_AVALON_PIO_DATA(RED_LEDS_BASE,0x0000);
                clear_lcd();
                break;
            case 0x4:   /* test timer */
                IOWR_ALTERA_AVALON_PIO_DATA(RED_LEDS_BASE,0xFFFF);
                wait_onesec();
                IOWR_ALTERA_AVALON_PIO_DATA(RED_LEDS_BASE,0x0000);
                break;
            case 0x8:  /* test LEDs */

               // Write the CODE to have the LEDS count from 0 to 255
               // Add enough delay so that you can see the lights change
```

```
                /* turn them off when done */
                IOWR_ALTERA_AVALON_PIO_DATA(RED_LEDS_BASE,0);
                break;
            default:    /* Do nothing */
                break;
        }
        number = 0;
        usleep(500000);
    }
    return(0);
}
```

Demonstrate the features to test each of the four functions:

Verification 2: _____