Name: Yongjiang Chen

Part 1:Introduction

This Project aims to design and implement three traffic lights on a BASYS-3 board. The simulated Traffic Lights follow the British Traffic Standards. I use LEDs and VGA display to display the traffic light signals and pedestrian crossing light as well as the pedestrian waiting light.

Part 2: User's Guide

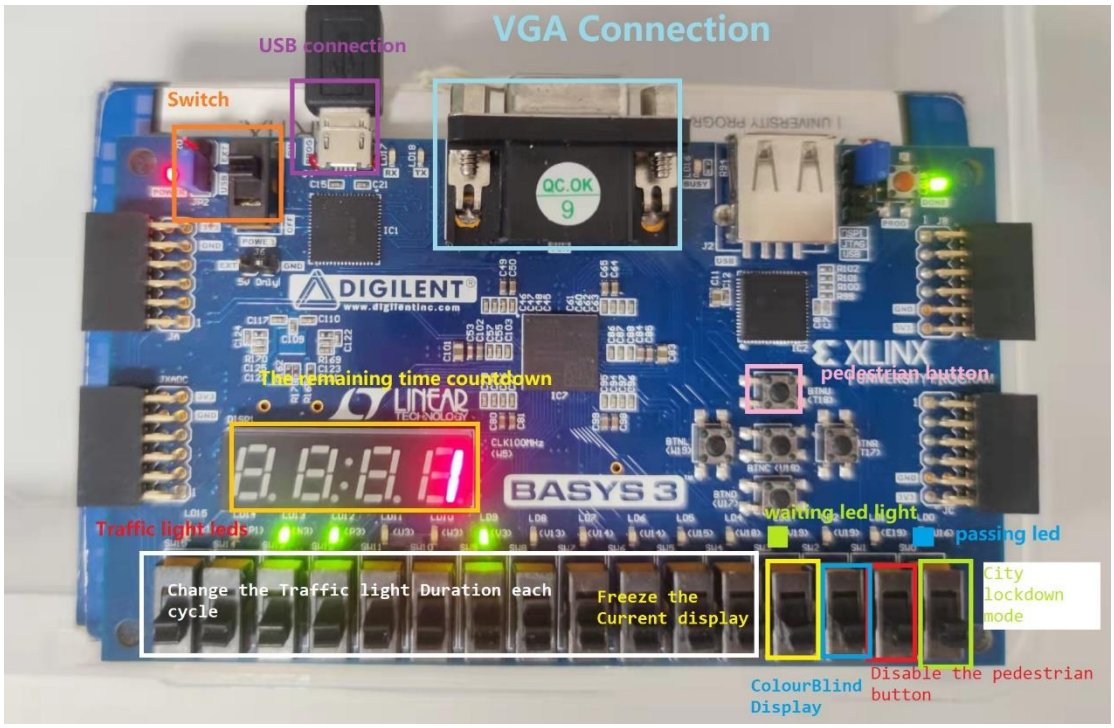

Figure 1: The FPGA Bays3 board functionality overview

Part 2.1 Basic instructions

To run the application:

I. Turning on the FPGA(BASYS 3), connecting it to PC with the Micro-USB

II. Opening the application in Vivado —> Export Hardware (Includes bitstream) —> Launch SDK.

III. Connecting board to the monitor with the VGA cable, and select VGA monitor source.

IV. Program FPGA and run the application (either basic Traffic System or the advanced Traffic System with extra features), then the VGA Regions (12 empty white squares) will be displayed on monitor (as shown in Figure 2). Region 0–2 are dedicated to display traffic light 1, regions 3-5 are for traffic light 2 and regions 6-8 are for traffic light 3. Region 10 is for the pedestrian light.

(REGION 0) RED LIGHT	(REGION 3) RED LIGHT	(REGION 6) RED LIGHT	
(REGION 1) YELLOW LIGHT	(REGION 4) YELLOW LIGHT	(REGION 7) YELLOW LIGHT	(REGION 10) PEDESTRIAN LIGHT (EITHER RED OR GREEN)
(REGION 2) GREEN LIGHT	(REGION 5) GREEN LIGHT	(REGION 8) GREEN LIGHT	

Figure 2: VGA region selection on display

Once the application is running the traffic lights will be displaying in the following sequences (Please see Figure 3) For every R/R/R phase, the current working traffic light stays at Red and the next traffic starts working from Red & Yellow. The light changes every 1 second by default.

Figure 3: The functioning cycle of the three traffic lights.

Whenever a pedestrian presses the pedestrian button, the pedestrian light will light up for 5 seconds, and within the last two seconds crossing time, the pedestrian light would continuously blink to remind the pedestrian to move faster.

The LEDs on the FPGA will also light up for the corresponding traffic light and pedestrian waiting/crossing light as shown in Figure 1.

Part 2.2 Extra Features

1. City Quarantine mode, so in a city lockdown, (from my experience in China) some of the roads are closed to limit traffic and prevent people going on streets. And some of the traffic lights will be set to red. By turning on the first slide switch on the right, you may enable the city quarantine mode and all three traffic lights will be set to red and the 7-segment display will be set to dashes. After turning the switch off, everything resumes.
2. Disable the pedestrian press button, turning on the second slide switch disables the pedestrian button, turning if off re-enables it,
3. Colour blind mode, with the third slide switch switched on, all the green light display will be set to a green-blue colour and that's the scientific traffic light signal[1] for colour blind people. Green-red blind people can't distinguish between red and green but can see red and blue.
4. Freeze the moment, you may free the current traffic light display by switching on the fourth slide switch on the right.
5. You may change the duration for each traffic light cycle by inputting time in seconds using the rest 12 slide switches, and that input duration will be added extend the default

duration of 1 second as well as the 5 seconds pedestrian crossing time.

Part 3: Programmer's Guide

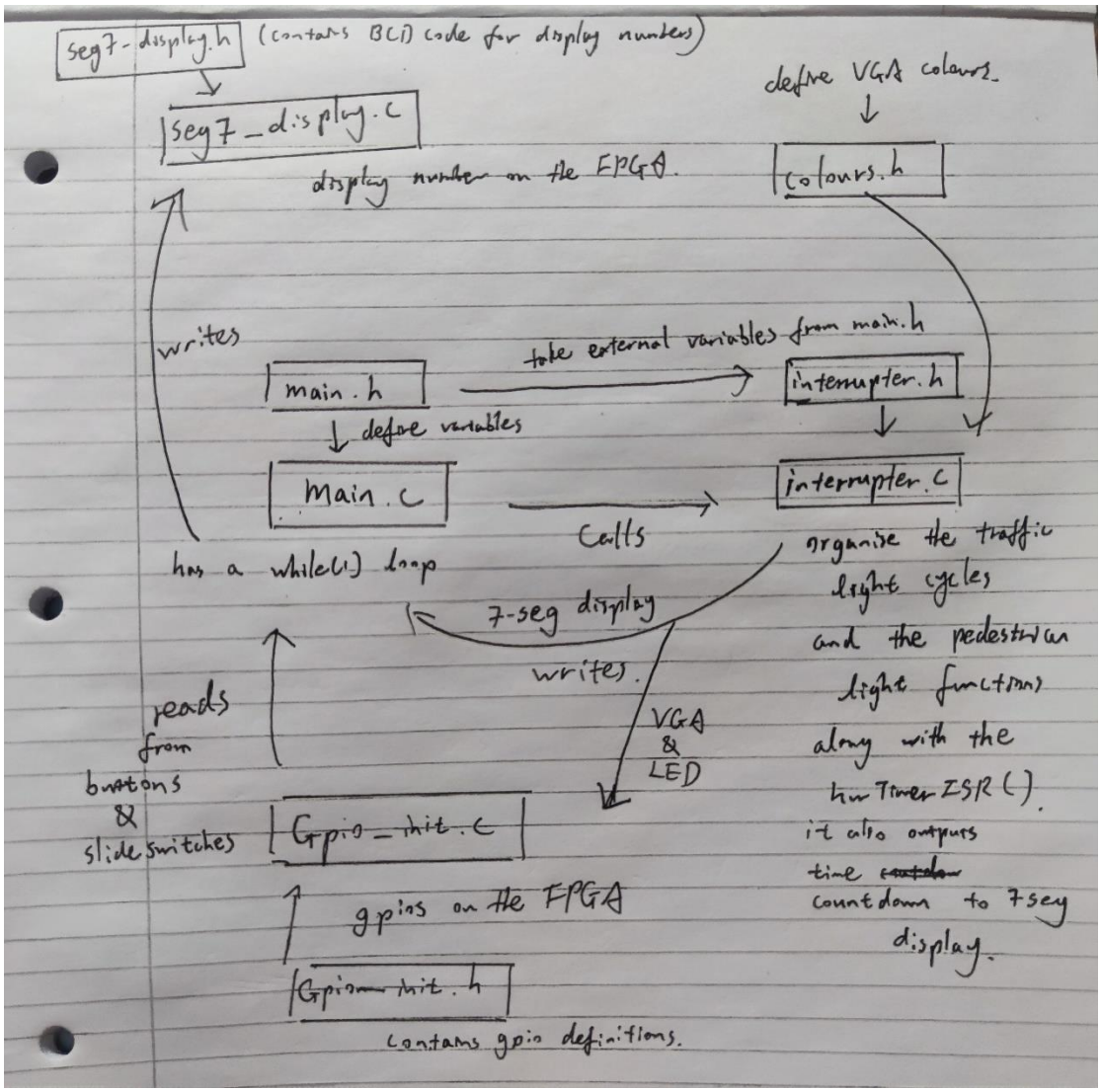

Figure 4: The basic application structure as a flow chart.

Part 3.1 Files included in the src folder of the application (with basic features)

File name	Description
Main.c	This source file contains functions used to drive the whole application. * There are 4 functions in this file: * checkButton() read input from the up button and assign 1 if the pedestrianButton is pressed * writeDisplay() writes all variables to their corresponding gpios for correct output * controlXGpios() controls the gpios; * main() to sum up the initGpio() and setUpInterruptSystem() functions along with the switchMode() to start the application

Main.h	This header file contains declarations of variables to be assigned as Colours for the three traffic lights and The binary code for the LED display to indicate the traffic light/pedestrian light/ waiting light, Value of pedestrian button, 1 if pressed, 0 if not, variables to be used to changing the traffic light cycle and variable to write the duration for traffic light cycles
Interrupter.c	This source file contains functions used to control the traffic light cycles with an interrupter * It has 6 functions: setColours() sets the VGA colour for each of the traffic light, * setLED() sets the corresponding LED for each traffic light display and the pedestrian light and the waiting light * blink() makes the pedestrian light to blink in the last two seconds of its display * enablePedestrian() controls the pedestrian light display and display duration * updateStates() controls the traffic lights display cycle * hwTimerISR(void *CallbackRef) (which is defined in the xinterruptES3.c file) method is used to call updateStates() * and enablePedestrian() whenever appropriate to control this traffic light system
Interrupter.h	This header file contains reference to the variables declared in the main.h and also definitions to the functions in the interrupter.c
Colours.h	This header file contains definitions for the binary code used for RGB colours for VGA display
Gpio_init.c	This source file contains function initGpio used to initialise gpios used in the application including Initalise the 12 VGA Display regions, initialise the LEDs, initialise the 7-seg display and initialising the pedestrian button, which is the up button
Gpio_init.h	This header file contains definitions for the gpios used in this application including the VGA region Gpios and LED Gpios and 7-segment Gpios and button Gpios
7seg_display.c	This source file contains functions used to drive the 7 segment-display. There are four functions in this file. The displayNumber() function receives an unsigned 16-bit integer. It is used to assign the digit number and the value to be displayed per digit when the timer interrupt occurs. displayDash() is a helper function for the displayNumber() to display dashes. The calculateDigits() function is used to extract the digits (of which a maximum of four can be displayed) from the number to be displayed. displayDigit() is to display the given number as SEG7_HEX_OUT on the expected SEG7_SEL_OUT
7seg_display.h	This header file contains definitions used to drive the 7 segment-display. it contains Definitions for 7-segment BCD codes, Definitions for digit selection codes and Definitions for digit selection codes

Others	platform.h: contains definitions of init_platform() and cleanup_platform(). platform_config.h: definition of hardware configuration. platform.c: declarations of init_platform() and cleanup_platform().

Part 3.2 Crucial codes to modify the application

In Main.c:

```
void writeDisplay() {
    // Write colours to the first traffic light
    XGpio_DiscreteWrite(&REGION_0_COLOUR, 1, colour_0);
    XGpio_DiscreteWrite(&REGION_1_COLOUR, 1, colour_1);
    XGpio_DiscreteWrite(&REGION_2_COLOUR, 1, colour_2);
```

Figure 5 The VGA regions on the monitor are filled with XGpio_DiscreteWrite function, the variables hold the region and colours are initialised in the main.h header file.

```
void switchMode() {
    while (1)
    {
        writeDisplay();
        // Read the value from the pedestrian button
        checkButton();

        checkSlideswitch();

        switch(modeSwitch) {
        case 0:
            displayNumber(disp_number);
            duration = 0;
            break;
        case 1:  //quarantine mode
            displayDash();
```

Figure 6 The function that's called in main(), the while(1) loop continuously reads input from the button and slide switches gpios and based on that it calls the displayNumber() function for which the disp_number variable is initialised and changed in the interrupter.c

```
void setColours()
{
    // Based on state of Traffic light 1, set the values of colour regions 0 to 2
    switch (state_1)
    {
        // RED
        case 0:
            colour_0 = RED;
            colour_1 = WHITE;
            colour_2 = WHITE;
            break;
        // RED + YELLOW
        case 1:
            colour_0 = RED;
            colour_1 = YELLOW;
            colour_2 = WHITE;
            break;
        // GREEN
        case 2:
            colour_0 = WHITE;
            colour_1 = WHITE;
            colour_2 = GREEN;
            if(modeSwitch == 4) {   //Colour blind GREEN display
                colour_2 = GREENBLUE;
            }
            break;
```

Figure 7 The simple switch code to control the VGA display of traffic light 1, where state_1 is maintained and called in updateStates() for a normal traffic light cycle between the three traffic lights.

```
void updateStates()
{
    // It is true by default to start the cycle
    if (traffic3_ok)
    {
        state_1 = (state_1 + 1) % 4; // Go to next state
        traffic3_ok = state_1 == 0 ? 0 : 1; // Check if sequence is finished
        traffic2_ok = !traffic3_ok; // If first light is finished, make light 2 start
    }

    else if (traffic2_ok)
    {
        state_2 = (state_2 + 1) % 4;
        traffic2_ok = state_2 == 0 ? 0 : 1;
        traffic1_ok = !traffic2_ok; // If second light is finished, make light 3 start
    }

    else if (traffic1_ok)
    {
        state_3 = (state_3 + 1) % 4;
        traffic1_ok = state_3 == 0 ? 0 : 1;
        traffic3_ok = !traffic1_ok; // If third light is finished, make light 1 start again
    }
```

Figure 8 The updateState() is continuously called in the hwTimerISR every time when the Traffic Time Duration reaches the set duration(1 second by default and add any extra seconds read from the slide switches) Notice that each interrupter counter count is 0.004 seconds on this Hadware setting.

```
    interruptCounter++;

    // 7-seg display countdown
    disp_number = (1+duration) - (interruptCounter) / 251;

    // Update state of the lights after 1 seconds
    if (interruptCounter == (250*(1+duration))){
        updateStates();
        interruptCounter = 0;
    }

    interruptServiced = TRUE;
}
```

Figure 9- Part of the hwTimerISR() in interrupter.c that calls updatestate() and display the remaining reduation on 7-seg display.

References

[1] Guidance on colour Blind Traffic light display https://www.quora.com/How-do-red-green-colorblind-people-determine-which-colors-are-which-on-a-horizontal-traffic-light

[2] The website that provides the colour distribution between red/ green/ blue colours https://www.colorhexa.com/0d98ba I used it to decide which colour to be displayed on VGA regions and to write the 12 digits colour code with it.