Assignment 2 lab book

Start Date: 23/11/2021

Submission Deadline: 29/11/2021

Plan for the assignment:

Date	Work to do	Estimated working time
23/11 to 24/11	Read through the assignment guide, fully understand the requirement for the traffic light system. Plan the module structure for the basic traffic light application, implement the basic features including (three traffic light display cycle, the pedestrian button and pedestrian light with blinking function, set up a suitable interrupter and 7-segment/LED display) Since I don't have a monitor that has VGA port, So I will have to test my initial design based on the 7-segment display and the LEDs.	6 hours, 3 hours each day on average
25/11	Attend the lab at campus, test my initial design using the lab monitors. Start implementing the traffic System with extra features. Think at least 5 extra features on the software level and realistic applications.	3 hours
26/11 to 27/11	Finish off the code implementation for both basic and advanced traffic light system. Review and refactor the code for better readability and reusability. Add comments to every module.	5 hours, 2.5 hours per day
28/11	Write up the assignment report including user's guide and the programmer's guide. (6 pages at maximum)	5 hours

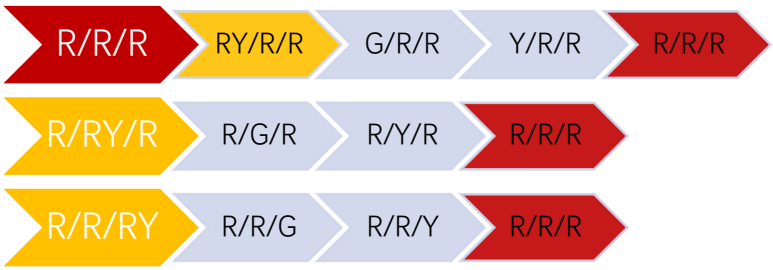
Chronological Lab record

[23/11]

- Review the Assessment 2 Traffic light guideline and specification.
  Open the assessment 2 file in vivado, Export Hardware and launch SDK, create a new application project named TrafficLights.
- To manipulate the FPGA, the first step is to initiate the GPIOs.
  To accomplish the basic tasks detailed in the specification (The basic traffic lights rotation on the monitor and on the LEDs and the pedestrian press button), my initial plan is that

only the LEDs, one of the four available push buttons, the GA regions and the 7-Segment display will need to be initialised.
So I look up the xparameters file in the bsp folder for the GPIOs' corresponding device IDs. Then I initialise them in the gpio_init.c and gpio_init.h files.

- I have decided to use the up button as the pedestrian press, because it's nice and intuitive.
- It's worth noticing that only three colours are needed for the traffic lights, they are red, yellow and green. I declared them as global variables in a header file named colours.h
- I create the following flow chart to model the cycle of the traffic light changes.



- To achieve the target traffic light display cycle, I have two approaches in mind, one is to list all the variations in a switch{} method within the main function, and going through the traffic cycle would simply be changing one "currentLightState" variable as the parament for the switch method. However, as there are three different traffic lights and I have to control them individually, the switch method within main() would be too heavy to contain all the possible cases. So my alternative approach is to initialise a variable for every VGA region and create three switch method to control the three traffic lights display, in another c file.
- I use three variables state_1, state_2, state_3 to control the three traffic lights display individually, switching between red/red yellow/ yellow/ green.
  I set up a hwTimerISR() such that the states changes every 1 second as required in the specification.
  I also implemented the 7- segment display using code from previous labs, to display the duration of 1 seconds for each traffic light display cycle.
- Testing: This is also used for testing the basic function of my implementation as I don't have a monitor that supports VGA port at the moment.

[End of 23/11, hour spent working: 3 hours 20 mins]

[24/11]

- To add a 5 seconds pedestrian phase into the cycle, I create the flow chart below for a better understanding.

- The main difficulty I encountered here was that after one pedestrian, if another pedestrian presses the passing button again while the three lights are still red, it would trigger another pedestrian phase rather than swapping back to the normal vehicle phases. To avoid such problem, I have to initialise
- Another problem: So, I implemented the pedestrian button basic function and it's working. However, after the pedestrian phase is finished, it goes to a R/R/R phase and then resume the next traffic light cycle starting with RY. This is not what the specification asks for, it should enter RY phase right after the pedestrian phase. To Solve the problem, I tried many modifications to the pedestrianButton() function, and I solved the problem by directly calling the updateState() function inside the pedestrianButton() function rather than specifying the 7-seg display and VGA region there.
- I finished my initial design by implementing LED display for the traffic light and pedestrian by calling XGpio_DiscreteWrite(&LEDs, 1, led_out) in the main() and changing the led_out value in different phases.
- The LED display is tested against the 7-segment display and helped me to test the traffic light cycle (the leftmost LEDs light up for the corresponding 9 traffic light VGA regions.
- Problem, surprisingly, the LEDs display don't match with the actual VGA display, it displays green light/yellow light other way around. Solution: I make this right by swapping the green light and yellow light's binary code for LED display. This was a human error and was easy to correct.

[End of 24/11, hour spent working: 2 hours]

[25/11]

- Attend the lab at 10am.
- Connect the FPGA board with my laptop and the VGA port of the lab monitor. Test my initial application design for the basic traffic light system. The on-screen display matches my expectation, just like the traffic light cycle flowchart above.
- Problem: while checking the 7-segment display on the FPGA, I notice that I have a countdown display from 1 to 0 for each traffic light display duration. There is a 1 millisecond display of zero. I am not sure whether this 'zero' display is what the specification asks for so I check this with Pro Arslan, and he says this is not what we want and ask me to remove it. And I eliminate the zeros by extending the interrupt counter divisor for the numberToDisplay variable so it never reaches zero.
- Now, think about the extra features to be added to my basic traffic light system on a

software level.

Here're my extra feature ideas

1) Modify the duration for the traffic lights
2) Traffic jam busy mode
3) Quarantine mode
4) Freeze the moment
5) Colour blind mode
6) Disable the pedestrian press button

- My plan is to start implementing the easiest extra feature, 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. I plan to assign the right most five slide switches on the Basys3 board for extra features 1, 3, 4, 5, 6 and the rest slide switches for numeric input, press buttons left right down for controlling the traffic jam busy mode (to be specified later). For the Quarantine mode, the first slide switch on the right is used to stop the traffic light cycle and turn all lights to read until the switch is switched off. My plan is to create a helper function in main.c to check whether the quarantine mode is on and by calling it in the main loop, I shall set VGA regions 0, 3, 6 to red and stop the cycle.
- Problem: The basic traffic light cycle wouldn't stop while the LED/7seg/VGA display are frozen, the problem is solved by adding a simple if(modeSwitch) in the hwTimerISR() and return interruptServiced = TRUE;
- On the basis of the Quarantine mode, I then implemented the Disable the pedestrian Press Button function. And also the Freeze the moment mode and colour blind display mode. At last, I implemented the rest slide switches to modify the duration of each of the traffic light cycles. No problems encountered

[End of 25/11, hour spent working: 3 hours]

[26/11]

- Today, I improved my code in terms of splitting them up in separate functions to increase readability of the code.
- I then added comments to my code including a main body description at the top of each module.
- The application is now finished and I finished it earlier than what's planned

[End of 26/11, hour spent working: 1 hours 30 mins]

[27/11] : no work to do

[20/11]

- Write up the assignment report

[End of 27/11, hour spent working: 3 hours]

Summary

In general, my initial plan and goal of the project was successfully met, I planned to spend 6 hours on the initial design of the application Traffic light system whereas in reality I spent 5 hours 20 mins on it. And I did what I planned to do in the lab session, prof. Arslan was there and he was particularly helpful with my concerns regarding the specification. After the main part of the assignment was finished, I speeded up and managed to finish it off in around 1.5 hours time after the great lab session. Leaving me plenty of time to write up the report, So I think my work and time management was great against the initial project planning.

Reference:

[1] General Assessment Guidelines for Engineering Software 3 - Assessment 2
[2] Online Binary to Hex converter at https://www.rapidtables.com/convert/number/binary-to-hex.html This website was particularly useful and helped me with some of the planning and building up the code