Lab 2 Report

Group 3: Dennis Zang (704766877), Daniel Park (104809832),
Samuel Pando (904809319)

October 30, 2020

Traffic Light Controller

[Components and Implementation Logic]

This design makes use of the “traffic” module to simulate traffic light state changes. A state machine implemented with an always block dependent on the clock controls the various states the traffic light system enters during its runtime. Within the always block, a timer restricts when a state can be changed. All state updates are handled here; the actual updates to the traffic light system (based on state) are handled in a separate always block non-reliant on the clock.

```
always @ (posedge clk) begin
    // Decrement the Timer
    stateTimer = stateTimer - 1;

    // Handle state transitions
    if (stateTimer == 0) begin
        if (state == main) begin
            state = mainExtended;
            if (trafficSensor == 1)
                stateTimer = 3'b011;
            else
                stateTimer = 3'b110;
        end else if (state == mainExtended) begin
            state = mainYellow;
            stateTimer = 3'b010;
        end else if (state == mainYellow) begin
            if (walkReg == 1) begin
                state = walk;
                stateTimer = 3'b011;
            end else begin
                state = side;
                stateTimer = 3'b110;
            end
        end else if (state == side) begin
            if (trafficSensor == 1) begin
                state = sideExtended;
                stateTimer = 3'b011;
            end else begin
                state = sideYellow;
                stateTimer = 3'b010;
            end
        end else if (state == sideExtended) begin
            state = sideYellow;
            stateTimer = 3'b010;
        end else if (state == sideYellow) begin
            state = main;
            stateTimer = 3'b110;
        end else if (state == walk) begin
            state = side;
            stateTimer = 3'b110;
            walkReg = 0;
        end
    end
end
```

Figure 1.1: The state machine. All state update logic is isolated in this always block. A timer is set upon state change, and updates can only occur once the timer has expired for the current state.

This alternative always block continually listens for inputs (i.e. when a walk button is pressed or the sensor for the side street is activated). The walk button is not a continuous input; therefore when noticed, an internal register is changed to store the value. On the other hand, the traffic sensor is always triggered when a car is present, so saving its value is not necessary. Instead, the traffic sensor input is sampled at key moments and influences the successive state.

The clock based always block operates based on an internal timer. This timer is decremented at each clock tick and state changes can only be made when the timer hits zero. The state and timer are changed simultaneously, ensuring the timer never passes zero. This forces the state transitions to "wait" a specified amount of time. For testing purposes, the timescale was set to be small, but can be scaled upwards so the system waits for seconds rather than nanoseconds.

An idiosyncracy of our implementation is the handling of special cases where a walk button is pressed or a traffic sensor notices cars. This was handled with separate intermediate states that maintained the previous output. The topic is further explored in the [Challenges Faced] portion.

Finally, all states are stored as parameters so that updates can be made easily. Parameters are also used to store the different values for each of the traffic light objects (i.e. the color of the lights, the on/off state of the walk sign, etc). Finally, a reset signal is implemented to easily set the system back to its initial state for easy testing.

[Waveforms/Testbench]

Testing was accomplished with the "traffic_tb" module. Given that there are three possible deviations from the normal traffic cycle based on the two inputs, we made a total of 8 test cases reflecting all possible combinations. The deviations are whether there is a walk request, whether the main sequence should be shortened, and whether the side sequence should be lengthened. Each test case tests an entire traffic cycle.

In its undisturbed state, we expect the main street light to start with green, turn to yellow, and then to red, alternating states with the side street light (who will start at red and go to green when its counterpart turns red). With a walk request, an additional 3 seconds of both red lights should occur with the walking lamp on. With a traffic signal at the end of a base interval on the main sequence green light, we would reduce the total green light duration to 9 seconds. With a traffic signal at the end of a base interval on the side sequence green light, we would extend the

total green light duration to 9 seconds. Under all 8 tests, we verified that each behavior and deviation worked as intended.

The order, combination of deviations, and durations of each test case are listed in the comments of the testbench code.

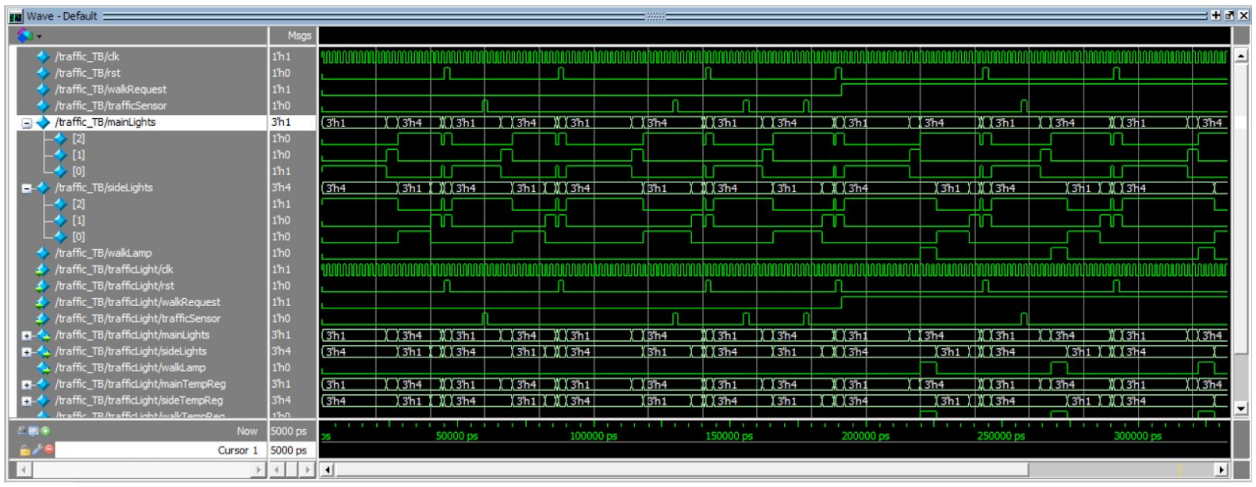

Figure 1.2: A general overview of some of the tests done. An example of an entire sequence tested is no deviations from normal, which is tested from 0ps-44000ps above. The test cases can be differentiated between to some degree by noting where rst is set.

[Challenges Faced]

One of the main difficulties with this lab was designing an implementation that could receive input at any time, but only updated its output at certain timeframes. As explained previously, this was achieved through a combination of a clock-based always block and a non-clock-based always block. However, correctly implementing this design initially led to problems such as infinite loops and an inability to be simulated.

Another issue lay in the handling of traffic sensor requests (to extend or shorten the main street light). Our implementation relied on additional states and the flexibility of the timer for these requests. Specifically, we had intermediate states that maintained the previous states' outputs, but had their own end-state transitions. Of course, their time could easily be set in the process of transitioning, so the same state could be held for different lengths of time.

For example the same state was recycled when defining whether the main street remained green for 3 seconds or 6 seconds longer with the only difference being the number assigned to the timer. On a higher level of abstraction, we can see the number of "states" of the state

machine exceeds the number of states defined in the code, but the timer veils that and allows a more straightforward implementation with less defined states.

Member Contributions:

Dennis Zang: Testbench/Specification Testing

Daniel Park: Logic Design/Code

Samuel Pando: Report/Analysis