Lab 3
Brendan Bovenschen A20323473, Landon Fox A20319340

Section 1: Introduction

In this lab, we've implemented the tail lights of a Pontiac firebird displayed on a row of LEDs. Since the firebird's taillights have a "ripple" effect, we must use sequential logic to time this perceived movement. Each of our states took roughly the same amount of time to transfer since very little combinational logic was executed. Because of this, most of our effort was directed toward the translation of our state diagram to our state machine. Finite State Machines are essential to digital design and are used in most human interactable circuits. Since this is our first project involving sequential logic, our system contains very little combinational logic and limited state branches. The tools within our HDL allow us to focus more on our design choices rather than develop on a lower level. Although we've worked with state machines at a lower level of abstraction in our lectures, SystemVerilog allows for the higher-level concept of cases to be used instead of the process of converting our states into combinational logic along with digital flip flops. This make's our design more human-readable while our HDL computes our state machine. We are unable to see the lower workings of this circuit, but given that the process to convert these states into sequential logic is algorithmic, we can assume SystemVerilog will properly implement our design concepts.

Section 2: Baseline Design

For this design, we have utilized a finite state (seen in Figure 3) machine to control the state functionality of a Pontiac firebird. Our final implementation involves two switches along with a reset button to mimic the turn and hazard signals. The FSM we developed has 10 states, with an initial state of O (Off) where no lights are on and it stays in this state until either input (L or R) is thrown. If an input of 1 from L is received, the left sequence will begin starting with state L1 which will light up LA, then it will go to the next state of L2 which lights up LA and LB, finally finishing the sequence off at state L3 which lights up LA, LB, and LC, afterward it returns to state O. This can be seen within figure 1. As the time progresses and the reset switch is not pressed, the process runs through each left state until it reaches 3 where it is returned to the starting position. This process is repeated for both the right system and the hazard system which follows the same ripple as both L and R but occurs at the same time. The reason for implementing a separate hazard signal rather than creating two separate state machines for left and right is so that both will occur at the same time. This may create more logic than the other design, but it allows for the ripple to stay simultaneous rather than risk an offset between each side of the LEDs.

Section 3: Detailed Design

SystemVerilog allows us to use cases for each state to control the functions of our control system. Within each of these segments, we can determine what the returned data will be along with the next segment to travel to. The origin state (O) starts with none of the six output lights active as no input has been detected or the reset button is thrown. Each of the state transitions occurs whenever the clock reaches the rising edge. Logic within these cases never changes the current state but sets the next state for when this clock signal transpires. The sequences of the left lights, right lights, and hazard lights once started will continue to run until the sequences are complete, always moving to the next stage in the sequence without any inputs. After the sequences are complete they each return to O, the sequences can be stopped at any time in their runtime with the use of the reset button. The left sequence starts when input L is positive, R is inactive, and the current state is Off. This transfers the current state to L1 which activates only LA initially. To create the user-perceived ripple, we must iterate through each state in this chain of states. L2 activates LA along with LB and L3 returns the same outputs along with LC. Once the end of this cycle is achieved, the state is returned to O where further inputs are awaited. This general process is identical for R1-3 which controls RA-C and H1-3 which is capable of delivering signals to all six outputs for a double simultaneous ripple. Our board implementation was achieved with a left and right switch for the inputs L and R, while a button was used for the reset function. When both of these switches are active, the hazard state is activated. Since LA-C and RA-C control 6 LEDs traveling from the middle, this can properly display the desired optical effect.

Section 4: Testing Strategy

Our limited number of states required us to test only a little amount of cases to properly evaluate each part of our design. To begin, we first made sure our reset input would stop all functions of the FSM. With this action operating, we could then test the transition between other state sections. We utilized both a waveform graph (Figure 1) and a timing chart (Figure 2) to initially see the state changes before implementation. First, from the waveform, we can see that the reset switch halts all other processes in our design while returning the current state to Off. Once this is lifted, the user input can be interpreted by the system. In this example, we first test the left cycle, right cycle, and the hazard cycle which all function as expected (waiting to begin the next cycle based on user input after the previous has finished). To accurately see when these events take place, we can analyze our timing diagram with measurable points of action. Each state occurs on the rising edge of our clock, so the clock signal must reach zero and one again before moving on to the next state. This data is the exact time sequence as the waveform graph, but it allows us to see directly when each event takes place within our design. Finally, to see if the abstracted machine displays properly, we were able to test out different combinations of input during our implementation. In the snapshots shown in figure 4, we first begin by going through each cycle (left, right, and hazard) and then terminating these with our reset switch. The analysis of this data allows us to prove that our system responds properly to each possible case.

Section 5: Evaluation

From our experience with our implementation as well as the amount of data collected during testing, we could show that our arrangement produces a proper result for the intended purpose of this lab. We were able to create and display each state within our HDL programming as well as define unique outputs based on these states. The FSM remained at the center of our attention, therefore little combinational logic was utilized and since our FSM consisted of few states, there was little room for design variation. However, we could have approached the hazard sequence differently. Instead of relying on the input of both our left and right signal, we may have triggered these lights with a separate hazard signal. The hazard state would be controlled by this new signal and most likely implemented with a switch. Although this is closer to the input of a real vehicle, it includes unnecessary inputs with a more complicated state system along with the involvement of a longer test cycle. Through this lab, we could experience more with timing as our components rely on our main clock to operate in sync with what is expected.

Figure 1: Waveform Graph

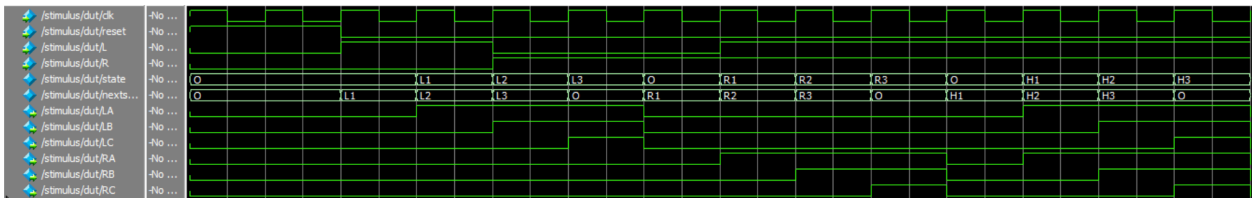


Figure 2: State, Output, Timing Chart

Time	Clk	Res	L	R	LA	LB	LC	RA	RB	RC
5	0	1	0	0	0	0	0	0	0	0
10	1	1	0	0	0	0	0	0	0	0
15	0	1	0	0	0	0	0	0	0	0
20	1	0	0	0	0	0	0	0	0	0
25	0	0	1	0	0	0	0	0	0	0
30	1	0	1	0	0	0	0	0	0	0
35	0	0	1	0	1	0	0	0	0	0
40	1	0	0	0	1	0	0	0	0	0
45	0	0	0	1	1	1	0	0	0	0
50	1	0	0	1	1	1	0	0	0	0
55	0	0	0	1	1	1	1	0	0	0
60	1	0	0	1	1	1	1	0	0	0
65	0	0	0	1	0	0	0	0	0	0
70	1	0	1	1	0	0	0	0	0	0
75	0	0	1	1	0	0	0	1	0	0
80	1	0	1	1	0	0	0	1	0	0
85	0	0	1	1	0	0	0	1	1	0
90	1	0	1	1	0	0	0	1	1	0
95	0	0	1	1	0	0	0	1	1	1
100	1	0	1	1	0	0	0	1	1	1
105	0	0	1	1	0	0	0	0	0	0
110	1	0	1	1	0	0	0	0	0	0
115	0	0	1	1	1	0	0	1	0	0
120	1	0	1	1	1	0	0	1	0	0
125	0	0	1	1	1	1	0	1	1	0
130	1	0	1	1	1	1	0	1	1	0
135	0	0	1	1	1	1	1	1	1	1
140	1	0	1	1	1	1	1	1	1	1

Figure 3

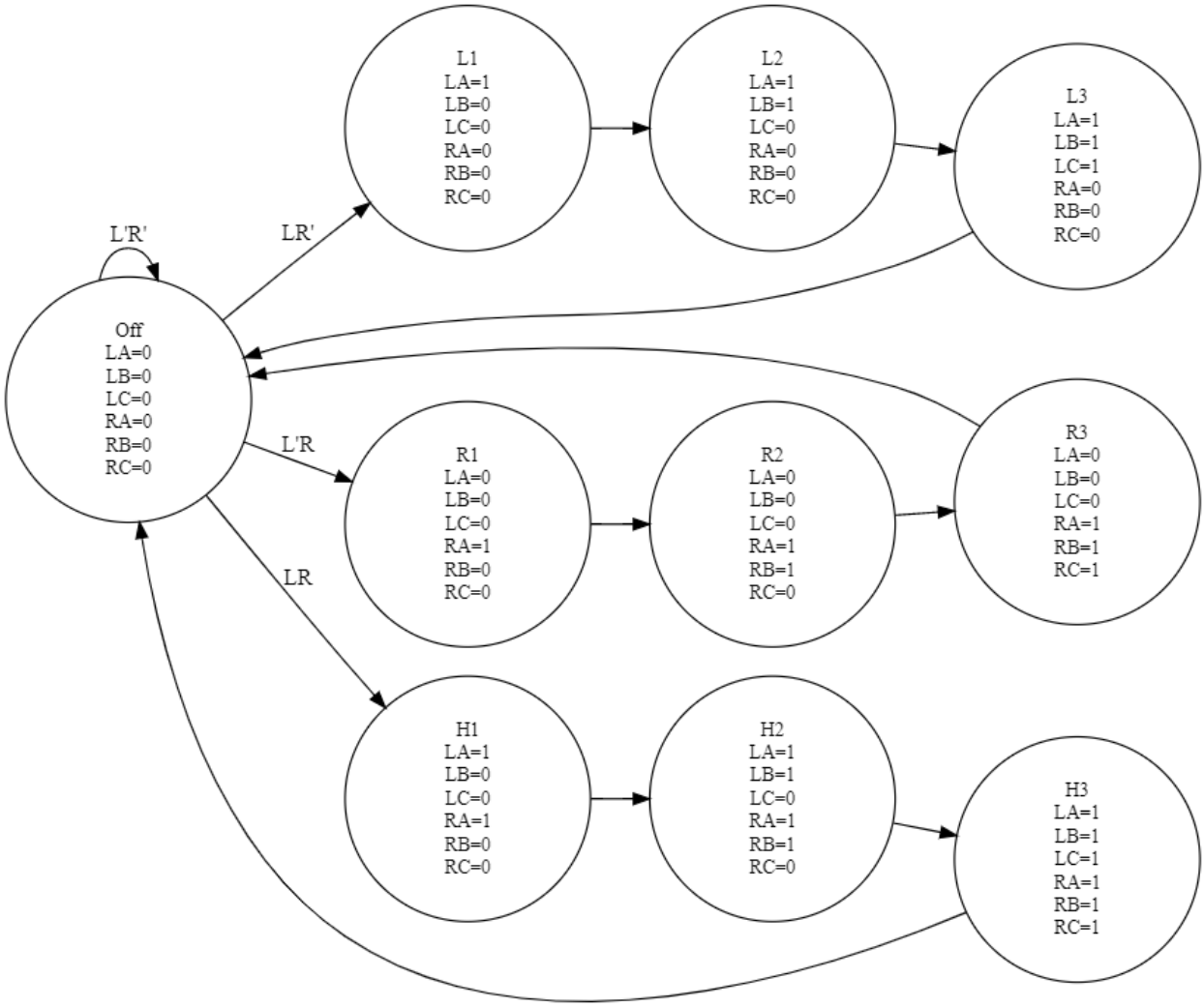
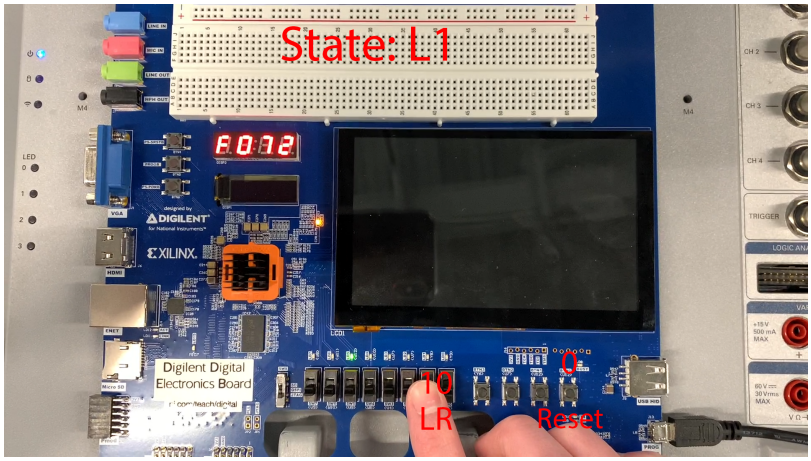
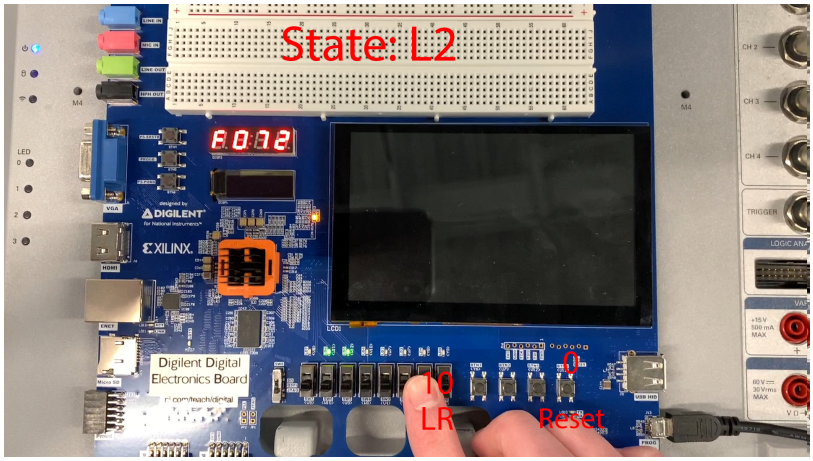

L1
LA=1
LB=0
LC=0
RA=0
RB=0
RC=0
L2
LA=1
LB=1
LC=0
RA=0
RB=0
RC=0
L3
LA=1
LB=1
LC=1
RA=0
RB=0
RC=0
L'R'
LR'
Off
LA=0
LB=0
LC=0
RA=0
RB=0
RC=0
L'R
LR
R1
LA=0
LB=0
LC=0
RA=1
RB=0
RC=0
R2
LA=0
LB=0
LC=0
RA=1
RB=1
RC=0
R3
LA=0
LB=0
LC=0
RA=1
RB=1
RC=1
H1
LA=1
LB=0
LC=0
RA=1
RB=0
RC=0
H2
LA=1
LB=1
LC=0
RA=1
RB=1
RC=0
H3
LA=1
LB=1
LC=1
RA=1
RB=1
RC=1

Figure 4 Picture of Each State

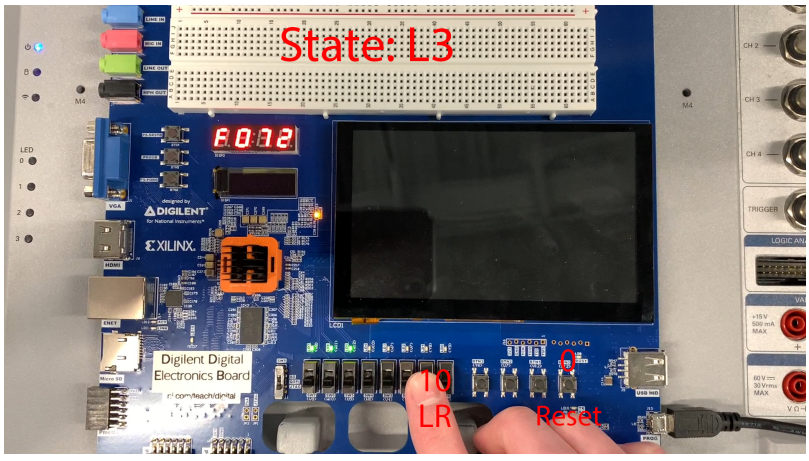
4.1



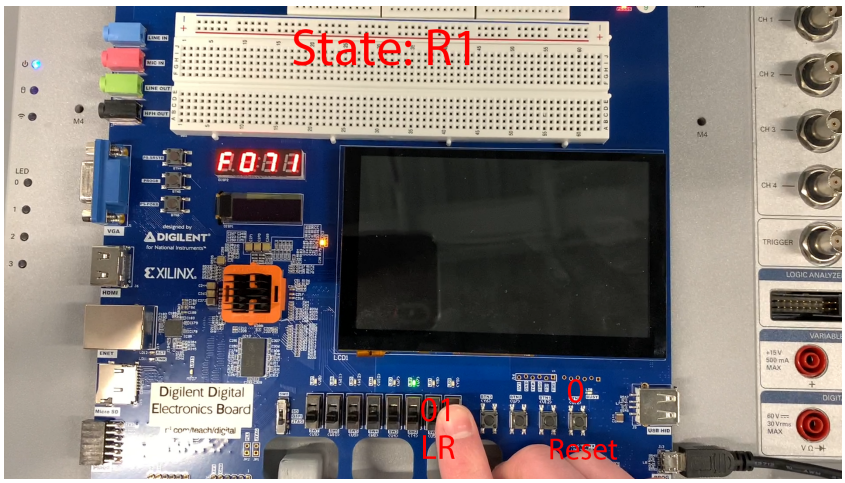
4.2



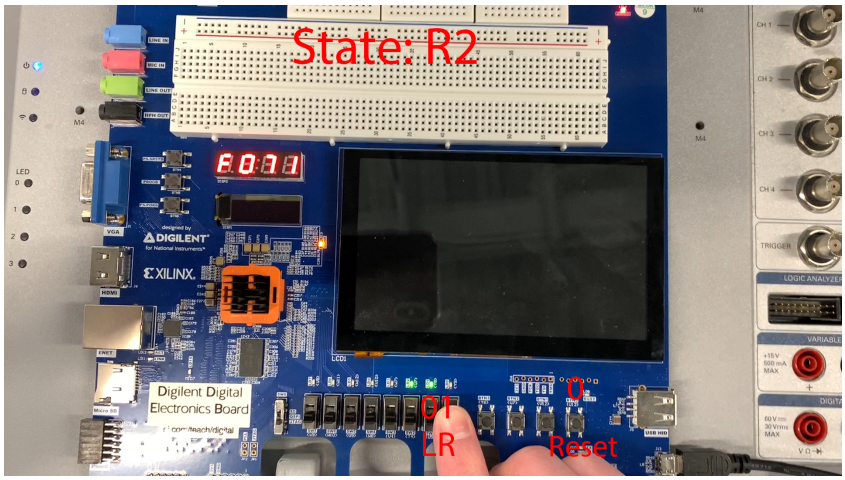
4.3



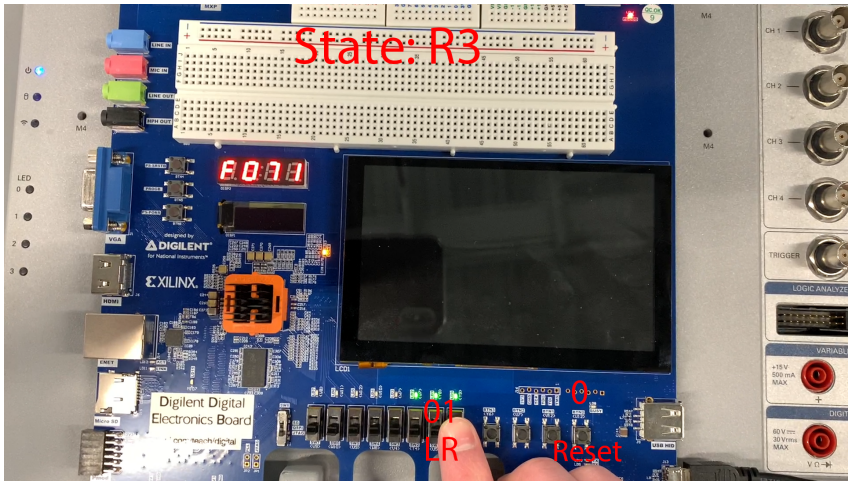
4.4


State: R1
Digilent Digital
Electronics Board
01
LR
0
Reset

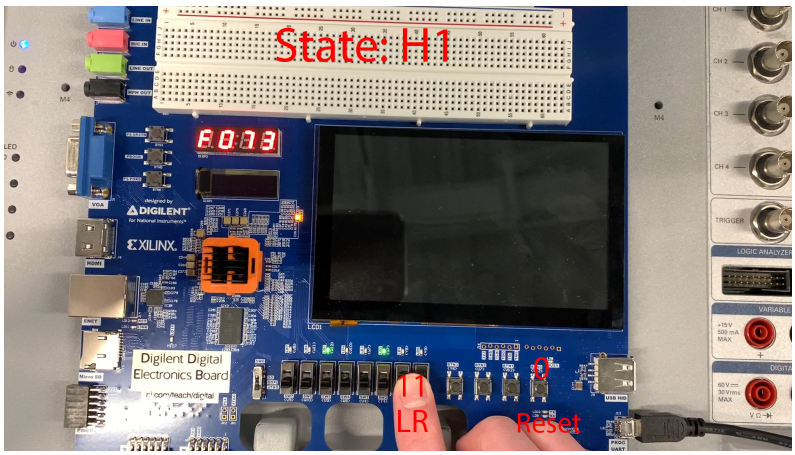
4.5


State: R2
Digilent Digital
Electronics Board
01
LR
0
Reset

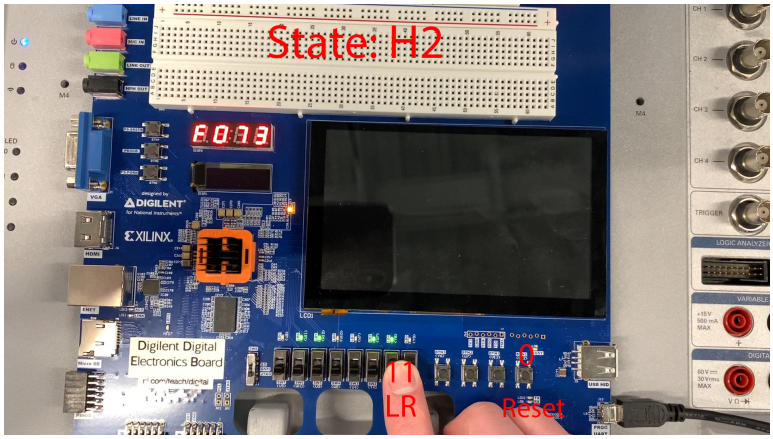
4.6


State: R3
Digilent Digital
Electronics Board
01
LR
0
Reset

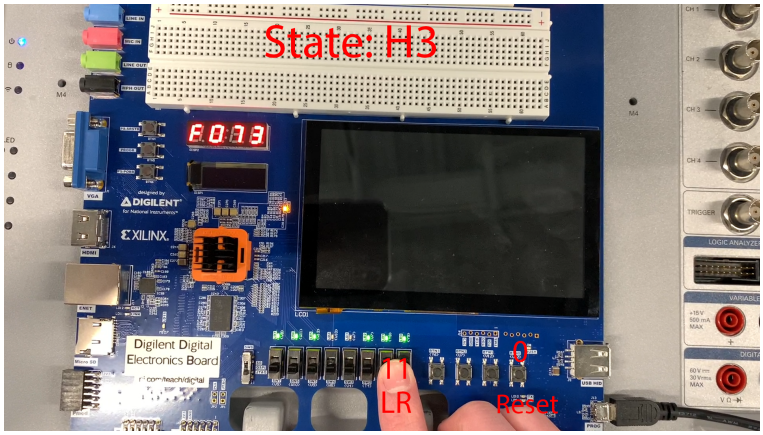
4.7


State: H1
Digilent Digital
Electronics Board
11
LR
0
Reset

4.8


State: H2
Digilent Digital
Electronics Board
11
LR
0
Reset

4.9


State: H3
Digilent Digital
Electronics Board
11
LR
0
Reset

4.10