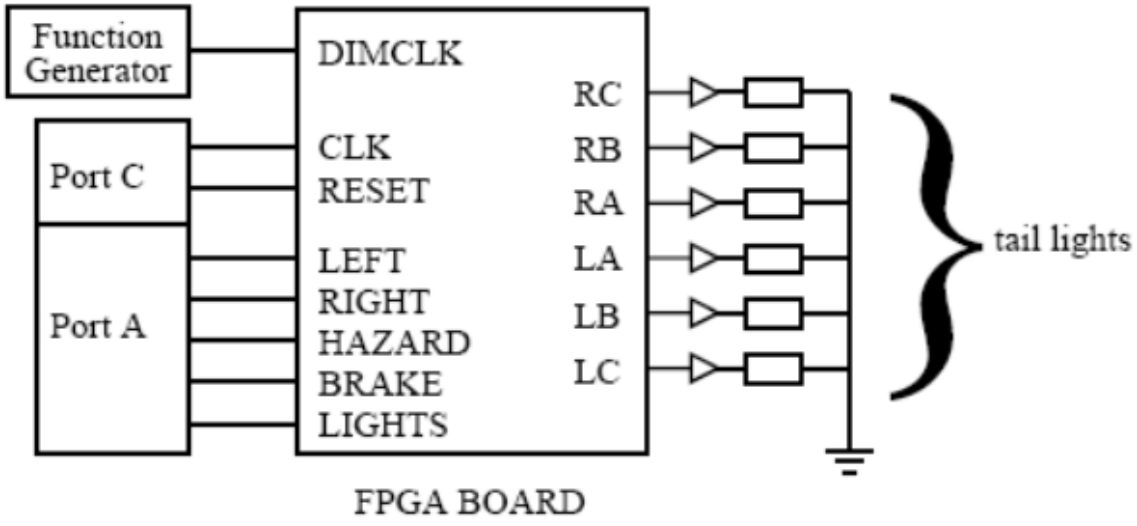
Project: 1965 Ford Thunderbird Tail Light Controller

Objective

My objective was to build a circuit that initiates the tail lights of a 1965 Ford Thunderbird. Among the tail light functions are left and right turn signals, brake lights, flashing hazards, and running lights. This project was done with accordance to a lab I had in my ECE 152A Digital Design Principle's class.



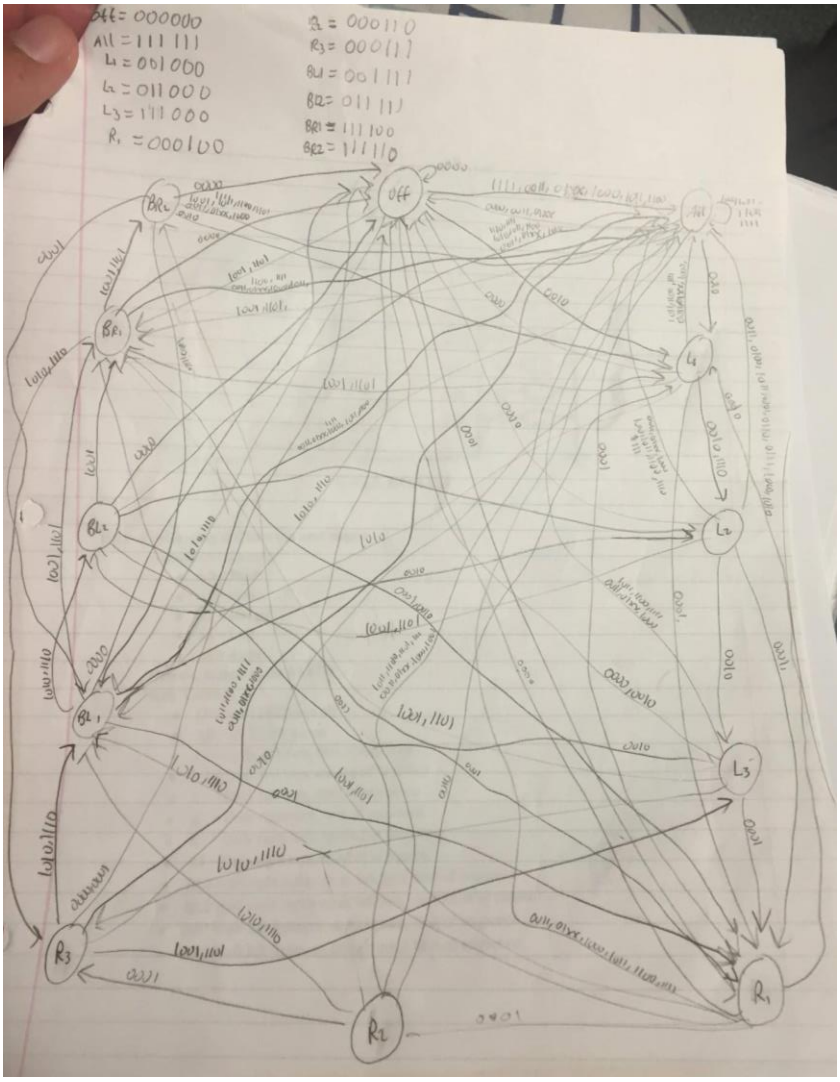
State Explanation of Chart (on next page):

- Inputs (x0x1x2x3x4):
  - x0 = Reset
  - x1 = Brake
  - x2 = Hazard
  - x3 = Left
  - x4 = Right
- Outputs: Signals to the 6 LEDS
- State Variable:
  - OFF = all LEDS are off
  - All = all LEDS are on
  - L1 = Only left turn, first of sequence
  - L2 = Only left turn, second of sequence
  - L3 = Only left turn, third of sequence
  - R1 = Only right turn, first of sequence
  - R2 = Only right turn, second of sequence
  - R3 = Only right turn, third of sequence
  - BL1 = all right lights on & left turn, first of sequence
  - BL2 = all right lights on & left turn, second of sequence
  - BR1 = all left lights on & right turn, first of sequence
  - BR2 = all left lights on & right turn, first of sequence

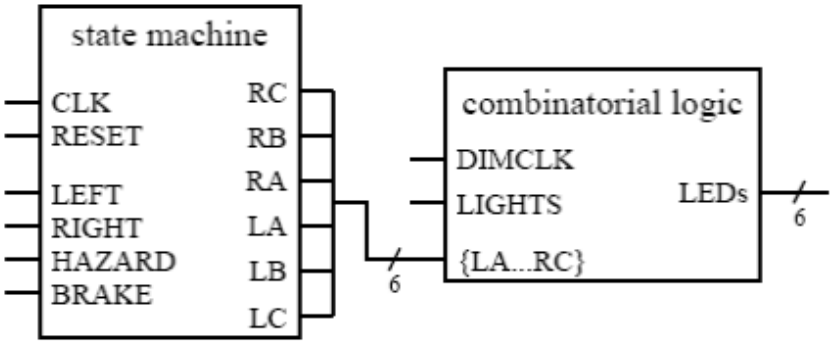
State Table

<table>
<tr><td colspan="10">LET: x0 = RESET, x1 = Break, x2 = Hazard, x3 = Left, x4 = Right</td></tr>
<tr><td rowspan="3">Present State</td><td colspan="8">Next State: x0x1x2x3x4=</td><td></td><td>Present Output</td></tr>
<tr><td>00000</td><td>00001</td><td>00010</td><td>00011</td><td>00100</td><td>00101</td><td>00110</td><td>00111</td><td rowspan="2">1----</td><td rowspan="2">LcLbLaRaRbRc</td></tr>
<tr><td>01000</td><td>01001</td><td>01010</td><td>01011</td><td>01100</td><td>01101</td><td>01110</td><td>01111</td></tr>
<tr><td rowspan="2">OFF</td><td>OFF</td><td>R1</td><td>L1</td><td colspan="5">All</td><td rowspan="26">OFF</td><td rowspan="2">000000</td></tr>
<tr><td>All</td><td>BR1</td><td>BL1</td><td>All</td><td>All</td><td>BR1</td><td>BL1</td><td>All</td></tr>
<tr><td rowspan="2">All</td><td>OFF</td><td>R1</td><td>L1</td><td colspan="5">OFF</td><td rowspan="2">111111</td></tr>
<tr><td>All</td><td>BR1</td><td>BL1</td><td rowspan="24">All</td><td>BR1</td><td>BL1</td><td>All</td></tr>
<tr><td rowspan="2">L1</td><td>OFF</td><td>R1</td><td>L2</td><td colspan="3">All</td><td rowspan="2">001000</td></tr>
<tr><td>All</td><td>BR1</td><td>BL2</td><td>BR1</td><td>L2</td><td>All</td></tr>
<tr><td rowspan="2">L2</td><td>OFF</td><td>R1</td><td>L3</td><td colspan="3">All</td><td rowspan="2">011000</td></tr>
<tr><td>All</td><td>BR1</td><td>All</td><td>BR1</td><td>All</td><td>All</td></tr>
<tr><td rowspan="2">L3</td><td>OFF</td><td>R1</td><td>OFF</td><td colspan="3">All</td><td rowspan="2">111000</td></tr>
<tr><td>All</td><td>BR1</td><td>R3</td><td>BR1</td><td>R3</td><td>All</td></tr>
<tr><td rowspan="2">R1</td><td>OFF</td><td>R2</td><td>L1</td><td colspan="3">All</td><td rowspan="2">000100</td></tr>
<tr><td>All</td><td>BR2</td><td>BL1</td><td>BR2</td><td>BL1</td><td>All</td></tr>
<tr><td rowspan="2">R2</td><td>OFF</td><td>R3</td><td>L1</td><td colspan="3">All</td><td rowspan="2">000110</td></tr>
<tr><td>All</td><td>All</td><td>BL1</td><td>All</td><td>BL1</td><td>All</td></tr>
<tr><td rowspan="2">R3</td><td>OFF</td><td>OFF</td><td>L1</td><td colspan="3">All</td><td rowspan="2">000111</td></tr>
<tr><td>All</td><td>L3</td><td>BL1</td><td>L3</td><td>BL1</td><td>All</td></tr>
<tr><td rowspan="2">BL1</td><td>OFF</td><td>R1</td><td>L2</td><td colspan="3">All</td><td rowspan="2">001111</td></tr>
<tr><td>All</td><td>BR1</td><td>BL2</td><td>BR1</td><td>BL2</td><td>All</td></tr>
<tr><td rowspan="2">BL2</td><td>OFF</td><td>R1</td><td>L3</td><td colspan="3">All</td><td rowspan="2">011111</td></tr>
<tr><td>All</td><td>BR1</td><td>All</td><td>BR1</td><td>All</td><td>All</td></tr>
<tr><td rowspan="2">BR1</td><td>OFF</td><td>R1</td><td>R1</td><td colspan="3">All</td><td rowspan="2">111100</td></tr>
<tr><td>All</td><td>BR2</td><td>BL1</td><td>BR2</td><td>BL1</td><td>All</td></tr>
<tr><td>BR2</td><td>OFF</td><td>R3</td><td>L1</td><td colspan="3">All</td><td>111110</td></tr>
</table>

State Chart



The dimming effect was implemented as a separate combinatorial circuit as shown below:



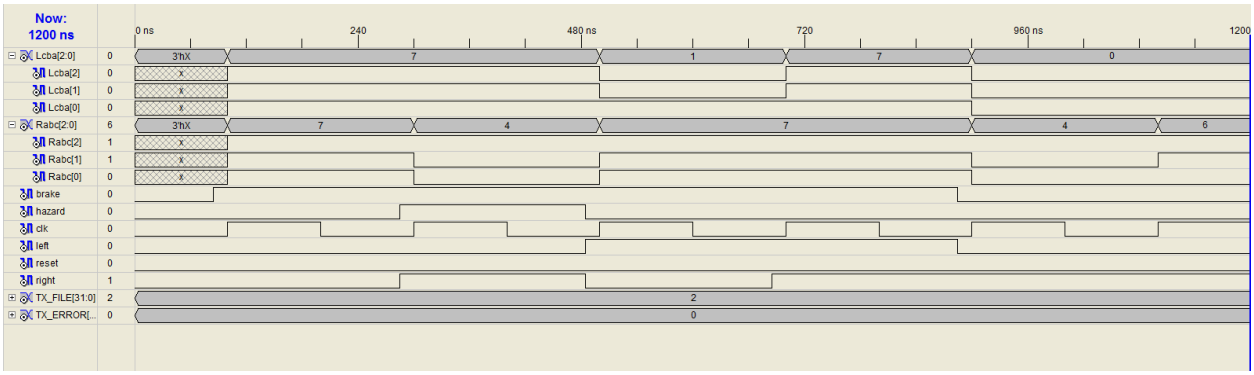
Verilog Code

Summary

My Verilog code model is a simple brute force approach, taking all of the large don't cares cases before running each condition did shorten the code down. The reset, hazard, and priorities are optimized and all the remaining cases are addressed with brute force. After the state is decided, I have a dimmer controller module that checks whether the light should be dimmed or not – and if it should be then it oscillates the light using a toggle signal that is switching between on and off at a frequency that is half of the input dimClk frequency.

Code: Please check the ".v" or the code PDF file in the repository.

Simulations



My simulation only tests the state selector since that's the only part of the circuit which involves conditional, adaptive, logic. The dimmer part is simply brute oscillation. The cases I tested were:

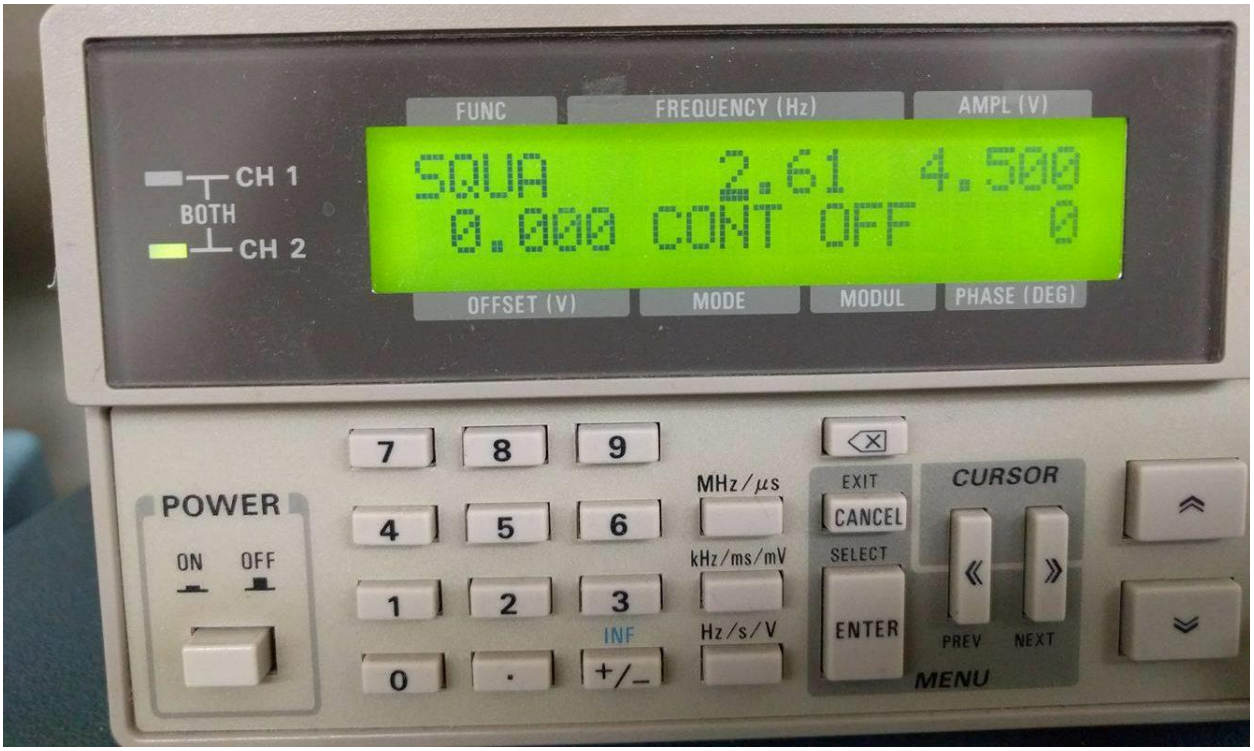
Case [x0x1x2x3x4]	Reason
01000	Check simple brake and see if it follows through.
01101	Checks brake, hazard, and right. Also checks priority management.
01010	Checks brake and left. We can see the simulation of the left turn and right being all on
01011	Checks brake with hazard (right and left together)
00001	Checks simple right turn sequence

I chose to show these simulations because I believe these extensively show the different combinations that my circuit can handle. I did a lot more simulations to make sure that the circuit can accurately handle different combinations. The ones depicted above show the hard and intensive combinations.

Next Page →

Pictures

Clock



Dimming Clock

Circuit Board