CmpE 124 Lab 7: Traffic Lights

Jasper Favis, 012225486, CmpE 124 Summer 2018, Lab Section 02

***Abstract*—This lab report describes the use of state machines in the design of a simple pedestrian-car traffic light system. State machines streamline the design process because it allows a system to be broken down into several states. In a state machine, outputs can either depend on the state alone or on the state and the inputs.**

I. Introduction

The goal of this lab was to design a pedestrian-car traffic light system on LogicWorks using algorithmic state machines. The traffic light system was designed for a one-way car lane with a single pedestrian crossing.

II. Design methodology

The pedestrian signal light was designed using a red LED for stop and a green LED for go. The 74LS163 was used to set the duration of the green light which stayed on for fifteen clock cycles while servicing the pedestrian. The signal light sequence was designed using the ASM chart shown in Figure 1. A state-transition table was generated from this chart and the logic equations for the D input of the flip flop as well as the outputs pGRN, count, and pBSY were derived from this table. The equations were implemented with a 74LS157 (2x1 multiplexer) and some combinational logic as shown in Figure 5. A pushbutton switch was used as the pedestrian request for the crosswalk. The switch was designed to be asynchronous so that the request can be made at any time, independent of a clock. The switch consisted of two D flip flops that were used to detect a request and hold it until the crosswalk has finished servicing the pedestrian. However, to ensure the crosswalk and car signal light were never green simultaneously, a second pedestrian request input called reqPsrvnc (request pedestrian no car) was used to initiate the pedestrian light sequence only when the car traffic light was red. Figure 3 shows the circuit. The 74LS08 with inputs reqPsrv and cBSY' was used to prevent the pedestrian traffic light from turning green while the car light was in use. The 74LS20 was used to prevent both lights from turning green when both lights are red and both requests have been made. In the event this occurs, the car light will turn green and the pedestrian light will stay red until the car light has completed its sequence. The green pedestrian light was also designed to blink four times during the last eight cycles before turning red. The idea is that the LED pGRN for the green pedestrian light will be inactive when the count of the 74LS163 is at nine (1001), eleven (1011), thirteen (1101), or fifteen (1111). The 74LS138 (3x8 decoder) was used to detect each of the counts. The NAND gate at the enable input of the decoder ensures that the decoder becomes active only when the pedestrian light is in use and when the 74LS163 is halfway through its count. The XOR gate and inverter at the output of the 4-input NAND gate ensures that pGRN will only blink when the red light is off and when the 74LS163 has reached any of the counts mentioned previously. The XOR-inverter combination also prevents the LED from turning on when the decoder is deactivated and the red light is on or when the output of the 4-intput NAND gate is low and q is low. Output pGRN should be low only when q is high and blink is low. It should be high when q is low and the blink is low or when q is high and the blink is high. The circuit will never be in a state where the blink is high and q is low simultaneously. Based on this input-output combination, the logic gate required for pGRN is XNOR, but an XOR gate with an inverter at the output yields the same result.

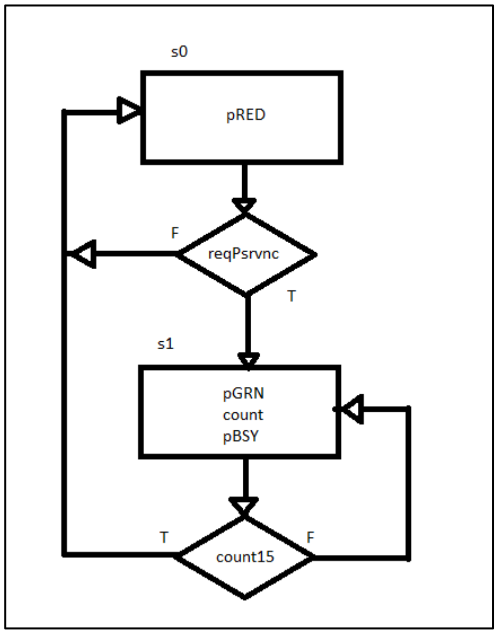

Figure 1 – Pedestrian ASM chart

The car signal light was built with a red, green, and yellow LED for stop, go, and slow respectively. The traffic light sequence is shown by the ASM chart in Figure 2. Similar to the pedestrian design, a table was generated from the chart

Jasper Favis jasperfavis@gmail.com

and equations were derived for the inputs D1 and D0 to the flip flops and for the outputs cRED, cGRN, cYLW, count, and cBSY. In addition to the flip flops, the car signal light was also built with the 74LS157 (4x1 multiplexer), 74LS139 (2x4 decoder), and combinational logic as shown in Figure 5.

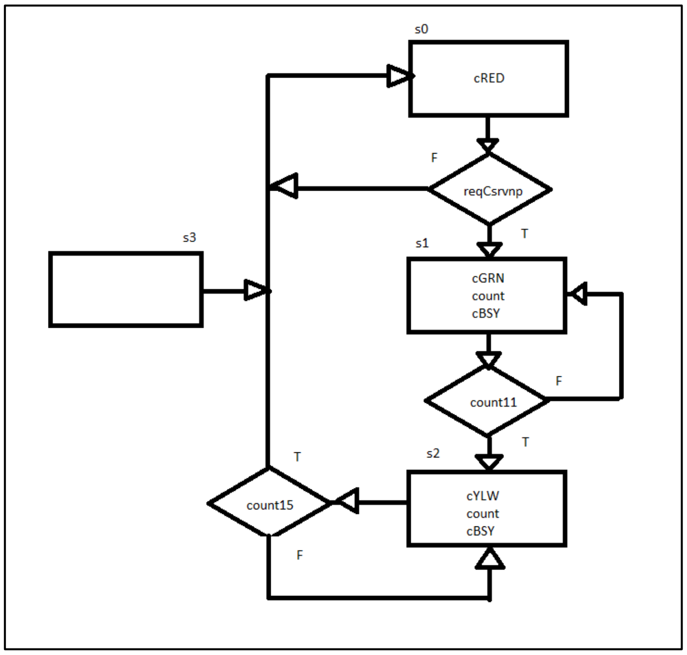

Figure 2 – Car ASM chart

A. Parts List

- DC power supply
- Breadboard
- Resistors: 470 ohms, 65k, 130k
- 5 mm LEDs: green, red, yellow, white
- Switches: SPDT, pushbutton
- 74LS163 (4-bit Binary Counter)
- 74LS00 (2 input NAND)
- 74LS04 (inverter)
- 74LS08 (2 input AND)
- 74LS20 (4 input NAND)
- 74LS32 (2 input OR)
- 74LS74 (D flip flop)
- 74LS86 (2 input XOR)
- 74LS138 (3x8 decoder)
- 74LS139 (2x4 decoder)
- 74LS153 (4x1 multiplexer)
- 74LS157 (2x1 multiplexer)

B. Truth Tables

NOT

X	OUT
1	0
0	1

NAND

X	Y	OUT
0	0	1
0	1	1
1	0	1
1	1	0

NOR

X	Y	OUT
0	0	1
0	1	0
1	0	0
1	1	0

XOR

X	Y	OUT
0	0	0
0	1	1
1	0	1
1	1	0

pGRN (Blinking)

q	blink	pGRN
0	0	1
0	1	invalid
1	0	0
1	1	1

D Flip Flop

pre	clr	D	clk	Q+	Qn+
0	0	0	→	0	1
0	0	0	→	0	1
0	0	1	→	1	0
0	0	1	→	1	0
1	0	x	x	1	0
0	1	x	x	0	1

JK Flip Flop

pre	clr	j	k	clk	Q+	Qn+
0	0	0	0	→	Q	Qn
0	0	0	1	→	0	1
0	0	1	0	→	1	0
0	0	1	1	→	$\overline{Q}$	$\overline{Qn}$
1	0	x	x	x	1	0
0	1	x	x	x	0	1

2 × 4 Decoder

INPUTS			OUTPUTS			
g	*b*	*a*	*O₃*	*O₂*	*O₁*	*O₀*
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0
0	-	-	0	0	0	0

4 × 1 Multiplexer

INPUTS							OUTPUT
g	*b*	*a*	*d₃*	*d₂*	*d₁*	*d₀*	*z*
1	0	0	x	x	x	0	0
1	0	0	x	x	x	1	1
1	0	1	x	x	0	x	0
1	0	1	x	x	1	x	1
1	1	0	x	0	x	x	0
1	1	0	x	1	x	x	1
1	1	1	0	x	x	x	0
1	1	1	1	x	x	x	1
0	x	x	x	x	x	x	0

C. *Original and Derived Equations*

1. NOT

$$f = \bar{x}$$

2. NAND

$$f = \overline{x \cdot y}$$

3. NOR

$$f = \overline{x + y}$$

4. XOR

$$f = x \oplus y$$

5. D Flip Flop

$$Q^{+} = D$$

6. JK Flip Flop

$$Q^{+} = J\bar{Q} + \bar{K}Q$$

Decoder:

$$O_0 = g\bar{b}\bar{a}$$
$$O_1 = g\bar{b}a$$
$$O_2 = gb\bar{a}$$
$$O_3 = gba$$

Multiplexer:

$$z = (g\bar{b}\bar{a})d_0 + (g\bar{b}a)d_1 + (gb\bar{a})d_2 + (gba)d_3$$

Pedestrian Equations (Before blinking light modification)

D = Q’(reqPsrvnc) + Q(count15’)
pRED = Q’
pGRN = count = pBSY = Q
reqPsrvnc = recPsrv(cBSY)

Car Equations

D1 = Q1’Q0(count) + Q1Q0’(count15’)
D0 = Q1’Q0’(reqCsrvnp) + Q1Q0(count11’)
cRED = Q1’Q0
cGRN = Q1’Q0
cYLW = Q1Q0’
count = cBSY = Q1’Q0 + Q1Q0’
reqCsrvnp = reqCsrv(pBSY’)

D. *Schematics*

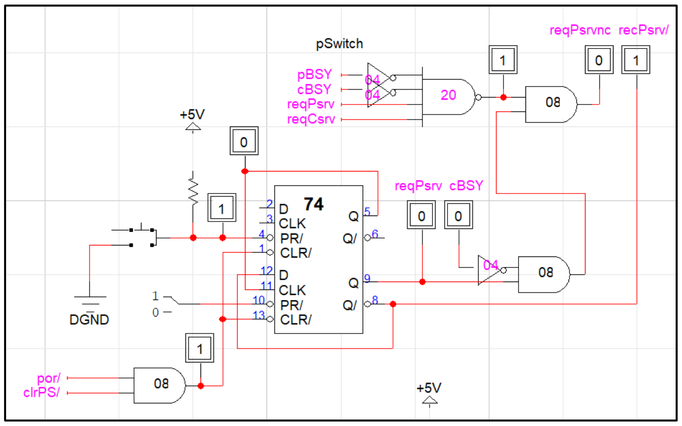

Figure 3 - reqPsrvnc

III. TESTING PROCEDURES

1. Power circuit via 5V supply.
2. Test the circuit by first requesting pedestrian service then requesting car service midway through the execution of the pedestrian signal light sequence. Toggle pSwitch then toggle sclk for ten clock cycles before toggling cSwitch. Continue toggling the sclk for the next twenty clock cycles. Observe the changes to the signals sclk, reqPsrvnc, reqCsrvnp, pRED, pGRN,

cRED, cGRN, cYLW, count11, count15, pBSY, cBSY, pCount, and cCount. Compare results with the state-transition tables listed in the appendix.

3. Test the circuit by first requesting car service then requesting pedestrian service midway through the execution of the car signal light sequence. Toggle cSwitch then toggle sclk for seven clock cycles before toggling pSwitch. Continue toggling the sclk for the next twenty-three clock cycles. Observe the changes to the signals sclk, reqPsrvnc, reqCsrvnp, pRED, pGRN, cRED, cGRN, cYLW, count11, count15, pBSY, cBSY, pCount, and cCount. Compare results with the state-transition tables listed in the appendix.
4. Test the circuit by requesting both pedestrian and car service while both traffic lights are at state zero. Toggle sclk for the next thirty cycles and observe the changes to the signals sclk, reqPsrvnc, reqCsrvnp, pRED, pGRN, cRED, cGRN, cYLW, count11, count15, pBSY, cBSY, pCount, and cCount. Compare results with the state-transition tables listed in the appendix.

IV. TESTING RESULTS

See appendix for the waveform results of step two. When pSwitch is pressed, signals reqPsrv and reqPsrvnc goes high. In the next clock cycle, pRED goes high to deactivate the red LED and pGRN goes low to activate the green LED. Output pGRN stays low for the next eight clock cycles. The LED then toggles from low to high on the ninth, eleventh, thirteenth, and fifteenth clock cycles. On the tenth cycle, reqCsrv goes high, but reqCsrvnp stays low until pRED goes low again. When reqCsrvnp goes high, cGRN becomes low and stays low for the next eleven cycles before going high as cYLW goes low. Output cYLW stays low for the remaining four clock cyles before cRED goes low again.

See appendix for the waveform results of step three. When cSwitch is pressed, reqCsrv and reqCsrvnp goes high. After the first clock cycle, cRED goes high and cGRN stays low on the next eleven clock cycles at which point cYLW goes low and cGRN goes high again. When reqPsrv goes high on the seventh clock cycle reqPsrvnc does not change until cYLW goes high and cRED goes low on the fifteenth clock cycle. Once again, pGRN stays a constant low for the first eight cycles and toggles between high and low within the last eight cycles. Afterwards, pRED goes low again.

See appendix for the waveform results of step three. When the cSwitch is toggled following the toggle of the pSwitch,while both pBSY and cBSY are low, reqPsrvnc goes back low and does not go high until the sixteenth clock cycle. During the time reqPsrvnc is low, the car traffic light goes through the same sequence described in steps two and three followed by the signal light sequence of the car traffic light.

V. CONCLUSION

A pedestrian-car traffic light system was designed in LogicWorks using state machines. D flip flops were used to control the state of the machine. The inputs to the flip flops that determined the next state of the machine were obtained from the state transition tables which were based on the ASM charts for each light. However, the circuit for the blinking light was not obtained from the chart and neither was the circuit used to prevent the lights from becoming green simultaneously after both requests have been made.

VI. APPENDICES AND REFERENCES

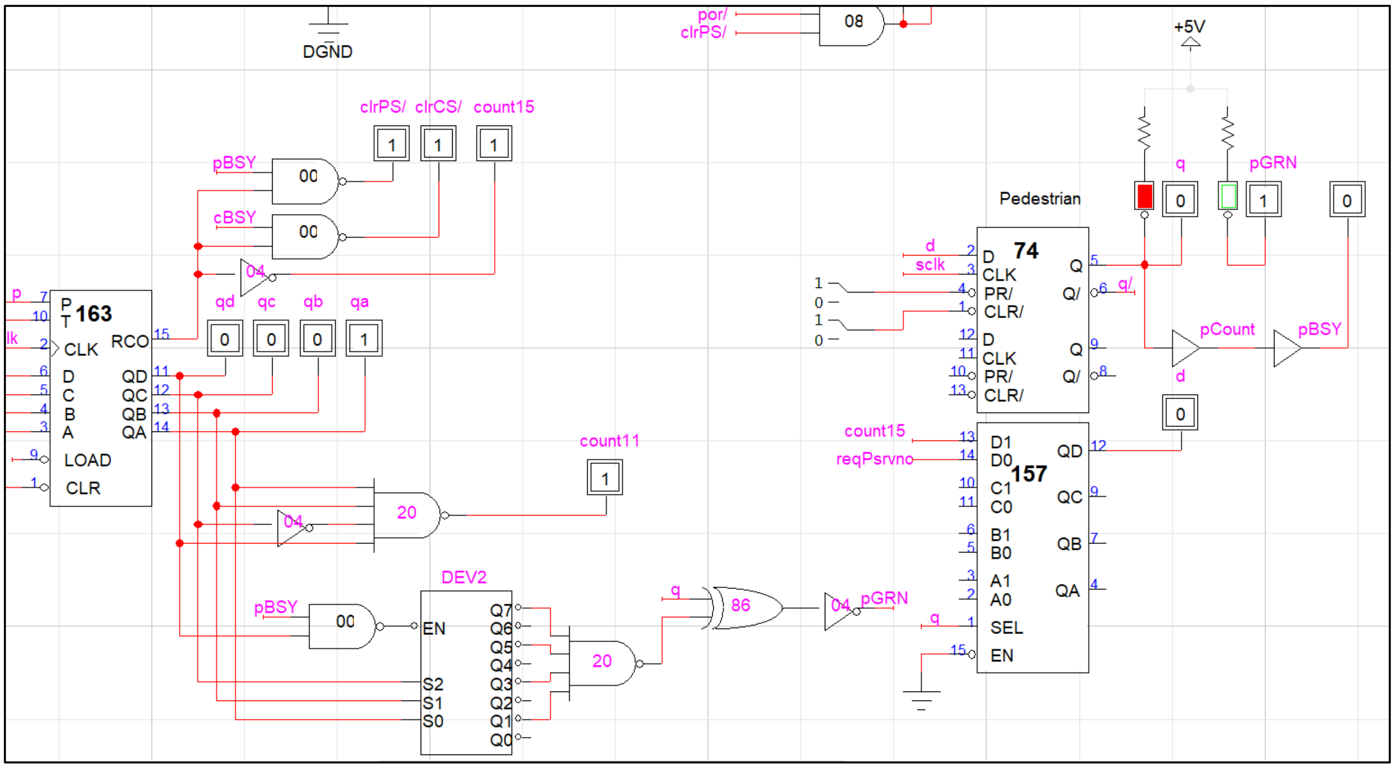

Figure 4 – Pedestrian Blink

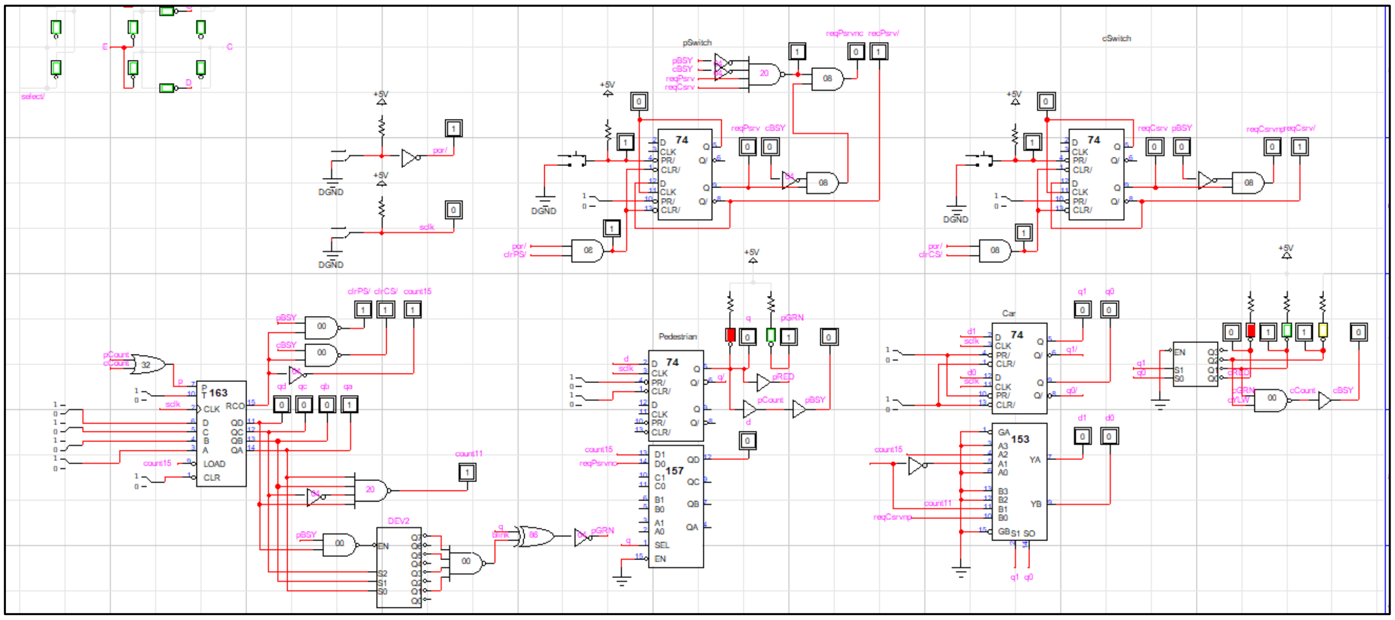

Figure 5 – Pedestrian-Car Traffic Light

Jasper Favis jasperfavis@gmail.com

State	Q	reqPsrvnc	Count15	State	Q+	pRED	pGRN	Count	pBSY
0	0	0	X	0	0	1	0	0	0
	0	1	X	1	1	1	0	0	0
1	1	X	0	1	1	0	1	1	1
	1	X	1	0	0	0	1	1	1

Figure 6 – Pedestrian State-Transition Table

State	Q1	Q0	recCsrvnp	Count11	Count15	State	Q1+ (D1)	Q0+ (D0)	cRED	cGRN	cYLW	Count	cBSY
0	0	0	0	X	X	0	0	0	1	0	0	0	0
	0	0	1	X	X	1	0	1	1	0	0	0	0
1	0	1	X	0	X	1	0	1	0	1	0	1	1
	0	1	X	1	X	2	1	0	0	1	0	1	1
2	1	0	X	X	0	2	1	0	0	0	1	1	1
	1	0	X	X	1	0	0	0	0	0	1	1	1
3	1	1	X	X	X	0	0	0	0	0	0	0	0

Figure 7 – Car State-Transition Table

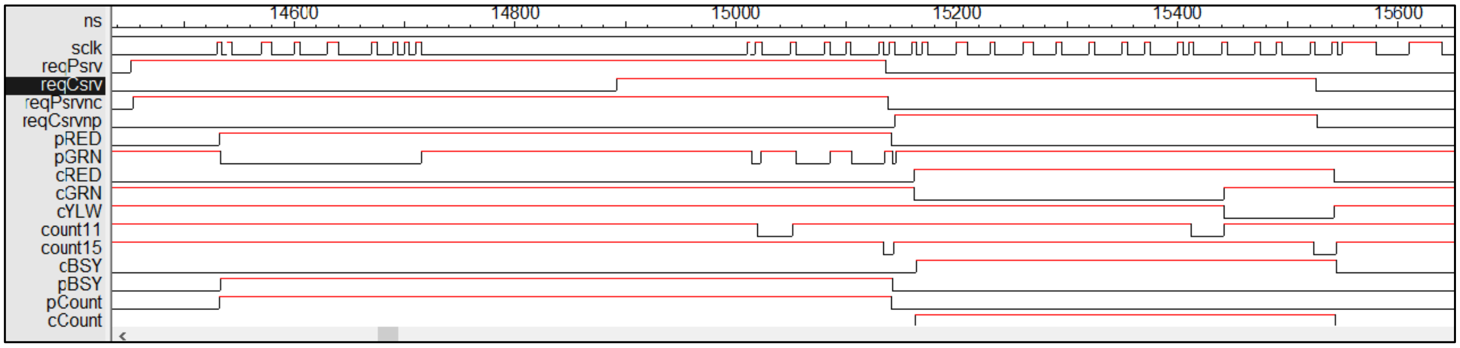

Figure 8 – Pedestrian Request then Car Request

Jasper Favis jasperfavis@gmail.com

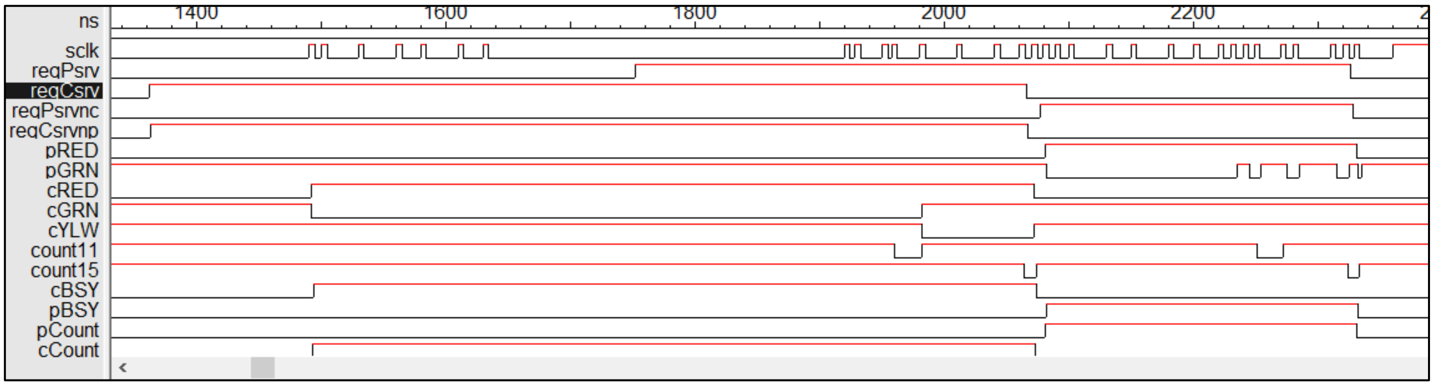

Figure 9 – Car Request then Pedestrian Request

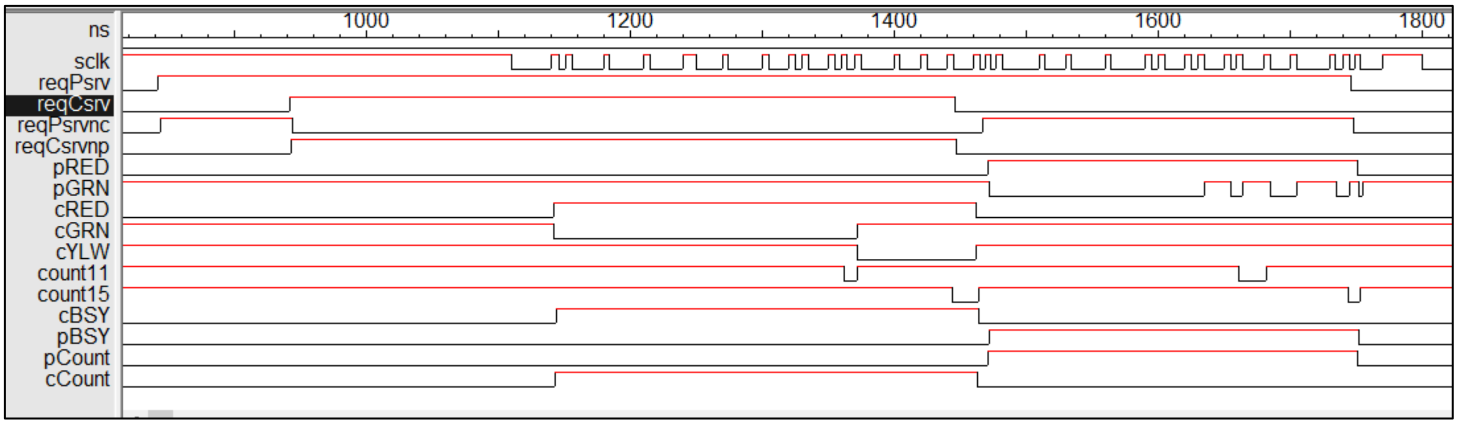

Figure 10 – Pedestrian Request and Car Request