Traffic Signal Controller

Laboratory Report for Exercise #9
EE241 Digital Design

<name>
Wilkes University
May 29, 2008

Abstract

A traffic light controller for an intersection of three roads (five corners) was designed using conventional sequential design techniques, and implemented with Low Power Schottky TTL logic. The design was simulated with Logicworks 3 but the simulation results were incorrect. However, the circuit was built and tested, and found to operate in a correct manner matching the requirements specified for the intersection.

1. Background

Brook Road, Laburnum Avenue, and Fauquier Avenue form an important intersection in northern Richmond, Virginia. The first two roads are heavily used, but Faquier has only light traffic. So, a traffic light controller design for this intersection was needed that would be sensitive to traffic conditions. The design was to be used for instruction in the class EE241, to be implemented by students, so reasonable simplifications were considered acceptable, including use of the same time for Green lights for both of the main streets and Fauquier, and the Yellow timing could also be used if desired for Fauquier Green timing, although this was not done.

2. Specification

The intersection is shown in Figure 1. Fauquier is to get a Green signal only when an automobile is detected on the inductive sensor embedded in the pavement of that road, and that Green cycle may add to the overall light cycle time. The signals are not synchronized to other signals. No delay between a light turning Red in one direction is needed before it turns Green for another direction.

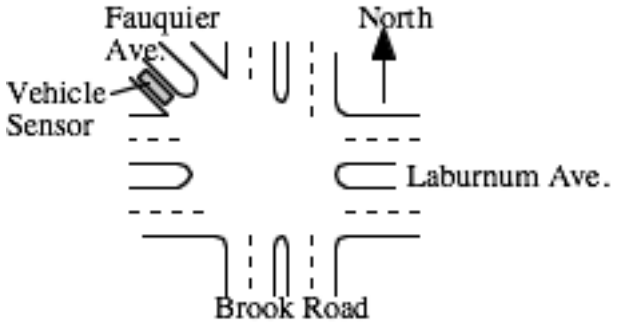

Figure 1 The intersection

3. Design

The system outputs are the signals to the traffic lights. Since both directions of a given street get the same signal, the different output signals can be simplified to those listed in Table 1. Furthermore, the Red signals can be considered combinational functions of the Green and Yellow signals, simplifying the design of the sequential machine that controls the traffic signal. There is only one input to the system, the vehicle sensor, identified as signal XS. However, timers are needed for the Yellow as well as the Green lights. In the interest of simplicity, the decision was made to give Fauquier Ave. the same length Green as the other streets. These timers can be implemented as monostable multivibrators (one – shots) outside the sequential controller proper. The timer triggers are identified as TG and TY for Green and Yellow respectively, and their outputs as XG and XY. The resulting block diagram of the overall system is shown in Figure 2.

Table 1 Output signals

Output Signal Name	Output Signal Identity
B_R	Brook Road Red
B_Y	Brook Road Yellow
B_G	Brook Road Green
L_R	Laburnum Ave. Red
L_Y	Laburnum Ave. Yellow
L_G	Laburnum Ave. Green
F_R	Fauquier Ave. Red
F_Y	Fauquier Ave. Yellow
F_G	Fauquier Ave. Green

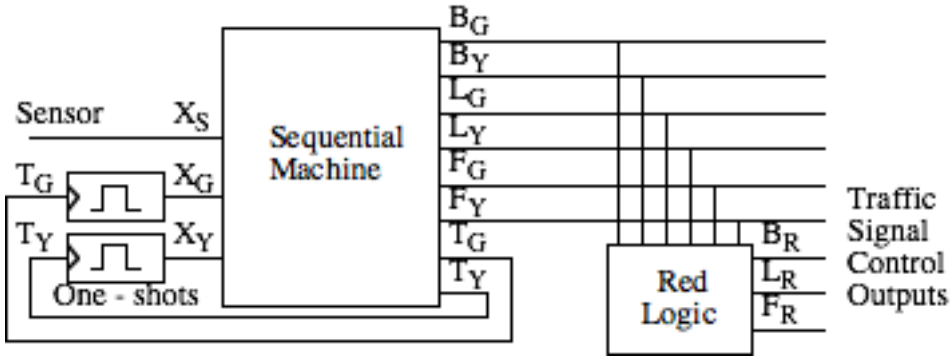

Figure 2 System Block Diagram

3.1 Output logic design

The design of the Red Logic is very straightforward. Each Red is on whenever neither the Green nor yellow is on for that direction. This gives the Red output equations given in Table 2.

Table 2 Red Output Equations

BR = (BG + BY)'
LR = (LG + LY)'
FR = (FG + FY)'

3.2 Timer design

Design of the one-shots is a matter of choosing the time for each, and then selecting the component values for R and C to give the required delay. Since the design is to use 74LS parts, the 74LS123 was selected because it incorporates two one shots into a single package. The time delay is specified as a function of the R and C values connected to each section of the device is given by equation 1. Since no special guidance for times was given, the design was developed for Green times of approximately 10 seconds and a Yellow time of about 5 seconds. These times could be easily changed by the adjustment of the resistance values.

$$t = .45\ R\ C \tag{1}$$

The component values chosen are given in Table 3. Because the capacitor values are large, electrolytic capacitors must be used. Since these have poor high frequency characteristics, it is necessary for small capacitors having good high frequency characteristics to be used in parallel with them.

Table 3 One – shot timer component values

Timer	time (sec)	R (Ω)	C (μF)
Green	10	22K	1,000
Yellow	5	12K	1.000

3.3 Sequential machine design

The sequential machine design is specified by the state diagram shown below in Figure 3. The sequence continues through states A to D unless the sensor, checked when the yellow phase of Laburnum Avenue ends, is active, adding states E and F to the sequence. Thus, Fauquier follows Laburnum, an arbitrary choice allowed by an absence of a specific sequence in the specification. The state diagram was converted into the form of a symbolic state table as shown in Table 4.

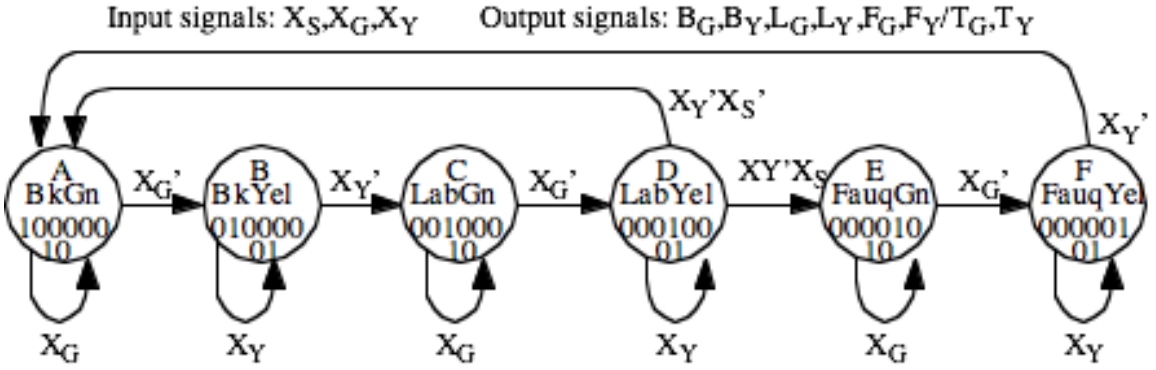

Figure 3 Traffic Signal State Diagram

At this point, four different design approaches were considered. These were: Classical state machine design, One Hot state machine design. Almost One – Hot state machine design, and Counter based machine design

Table 4 Symbolic State table for the Sequential Machine

State	Input signals: X_S,X_G,X_Y 000	001	011	010	110	111	101	100	Outputs
A	B	B	A	A	A	A	B	B	10000010
B	C	B	B	C	C	B	B	C	01000001
C	D	D	C	C	C	C	D	D	00100010
D	A	D	D	A	E	D	D	E	00010001
E	F	F	E	E	E	E	F	F	00001010
F	A	F	F	A	A	F	F	A	00000101

3.3.1 Classical State machine design

This approach minimizes the number of state variables. With only six states, this would allow the use of only three latches. Because the latch device to be employed is the 74LS74 dual D latch, two devices would be needed, meaning one would be wasted. Still, this would be a smaller number than for the One – Hot designs.

A state assignment was made by simply assigning state variables to the states in Grey Code order. Checking alternate assignments for optimality was not done pending comparison with the other design approaches. The symbolic state table was then converted into the state table shown in Table 5. This table was then converted into the Karnough Maps developed for AND-OR or BAND-NAND logic shown in Figure 4 .

Table 5 State table for Classical State Design

State variables State $Y_2Y_1Y_0$	Input signals: X_S,X_G,X_Y 000	001	011	010	110	111	101	100	Outputs
A 000	001	001	000	000	000	000	001	001	10000010
B 001	011	001	001	011	011	001	001	011	01000001
C 011	010	010	011	011	011	011	010	010	00100010
D 010	000	010	010	000	110	010	010	110	00010001
E 110	111	111	110	110	110	110	111	111	00001010
F 111	000	111	111	000	000	111	111	000	00000101

The state variable excitation equations derived from Figure 4 are given in Table 6, and the Output Equations in Table 7. Assuming inverted inputs are available, the number of gates required is 20 for state excitation and output variables, with 56 inputs total. No gate needs to have more than four inputs. Using SSI TTL, this logic could be implemented with seven combinational devices (2 x 74LS00, 3 x 74LS10, 2 x 74LS20).

Table 6 State Variable Equations for Classical Sequential Design

$D_2 = Y_2Y_0' + Y_2X_Y + Y_1X_SX_Y'$ (terms a,b,c)

$D_1 = Y_2Y_0' + Y_1X_Y + Y_2'Y_0X_Y' + Y_1X_SX_G'$ (terms a,d,e,f)

$D_0 = Y_1'X_G' + Y_2'Y_0X_G + + Y_2Y_0X_Y + Y_2Y_1Y_0'X_Y'$ (terms g,h,I,j)

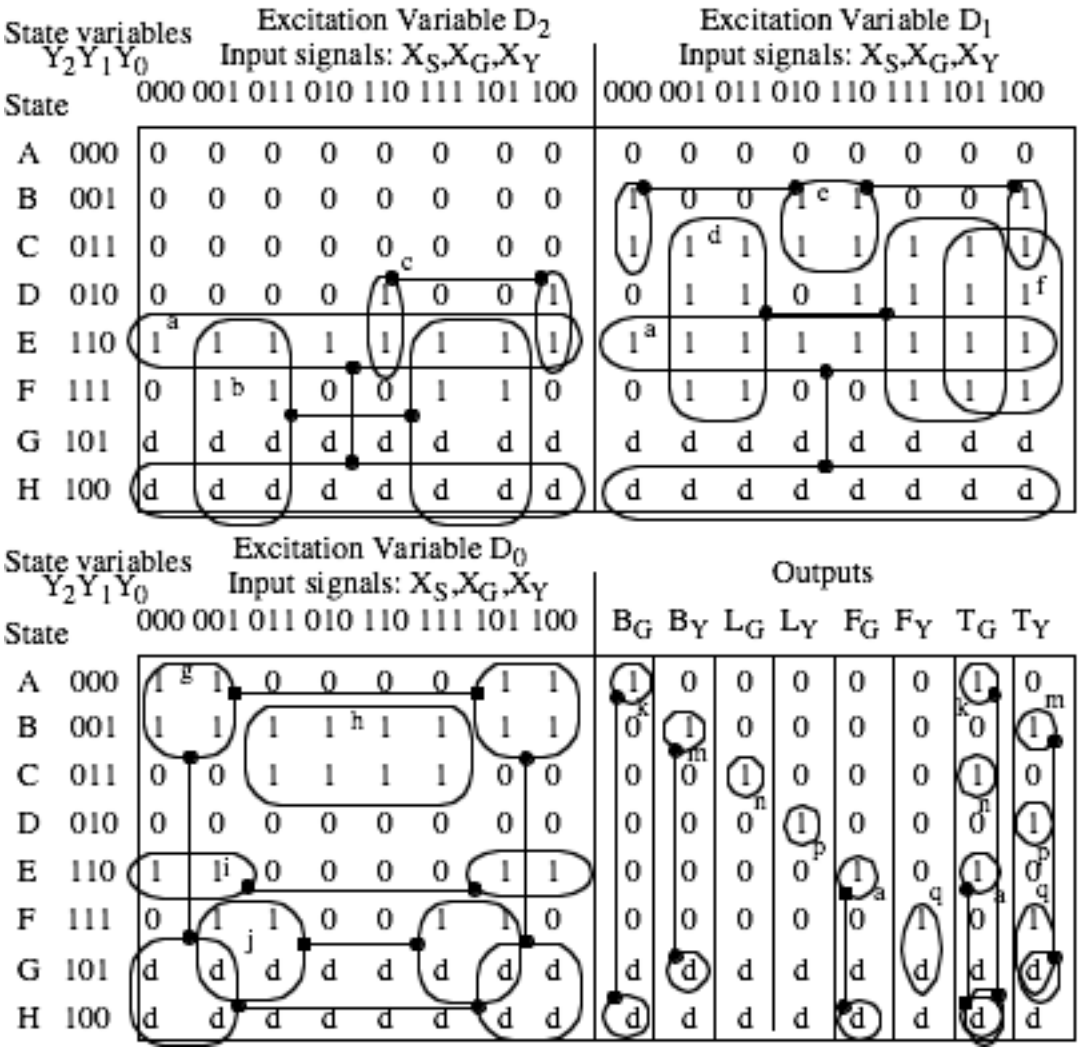

Figure 4 Karnough Maps for Classical Sequential Design

Table 7 Output Equations for Classical Sequential Design

$B_G = Y_1'Y_2'$	(term k)
$B_Y = Y_1'Y_0$	(term m)
$L_G = Y_2'Y_1Y_0$	(term n)
$L_Y = Y_2'Y_1Y_0'$	(term p)
$F_G = Y_2Y_0'$	(term a)
$FY = Y_2Y_1$	(term q)
$TG = Y_1'Y_2' + Y_2Y_0' + Y_2'Y_1Y_0$	(terms a,k,n)
$TY = Y_1'Y_0 + Y_2Y_1 + Y_2'Y_1Y_0'$	(terms m,p,q)

For the classical design, one final issue is the question of what happens if the machine happens to start up in one of the "Don't care" states, G or H. State H always goes to state E or F, but state G stays in G as long as X_Y is 1, turning on Yellow lights in all three directions, and then goes to A. This is assumed to be acceptable.

3.3.2 One-Hot machine design

In One – Hot design, one state variable is assigned to each state. For this sequential machine, that requires 6 latches, or two 74LS74 devices. In any given state, the state variable corresponding to the state is "1" and all other state variables are "0". It is usually necessary to initialize the machine on power – up to a desired legitimate state. The state variable assignments are given in Table 8 along with the excitation equations, which can be derived directly from the state table. Note that in this scheme, the output variables B_G to F_Y correspond directly to the state variables Q_A to Q_F respectively, so no separate output logic is needed for these outputs. The combinational logic needed requires 21 gates having 47 inputs (exclusive of inverters, of which at least one is needed). No gate needs to be larger than three input. The combinational logic requires six devices (2 x 74LS10, 4 x 74LS00) plus an inverter (74LS04) for the input plus the three 74LS74 dual latches, for a total of 10 devices. In addition, power-up reset is needed, so this approach is no better and perhaps somewhat worse than classical design.

Table 8 One – Hot state variable assignment and equations

State	Variables	Excitation and Output Equations
A	100000	$D_A = Q_A X_G + Q_D X_S' X_Y' + Q_F X_Y'$
B	010000	$D_B = Q_A X_G' + Q_B X_Y$
C	001000	$D_C = Q_B X_Y' + Q_C X_G$
D	000100	$D_D = Q_C X_G' + Q_D X_Y$
E	000010	$D_E = Q_D X_S X_Y' + Q_E X_G'$
F	000001	$D_F = Q_F X_Y + Q_E X_G$
		$T_G = Q_A + Q_C + Q_E$
		$T_Y = Q_B + Q_D + Q_F$

3.3.3 Almost One-Hot design

This method is similar to One-Hot, but QA is omitted, and State A has all state variables equal to zero. This can be more convenient if reset zeros all state variables rather than setting one of them to a 1. One latch (for Q_A) is in effect replaced by logic that detects that all latches are zero, or any logic AND or NAND gates that would have used Q_A are expanded to a large enough size to have the inverted state variables to be substituted. This would require at the very least a large gate (74LS30) and would save one latch. So, this approach was judged as inferior to those above and not explored further.

3.3.4 Counter Based Machine Design

The nature of the state diagram suggests that implementation using a Medium Scale Integration (MSI) counter might be efficient, since the sequence is an ordered progression and always returns to the first state. A counter such as the 74LS162 with a synchronous count enable and a synchronous clear is needed. With some modification, a counter with an asynchronous clear could be used instead. The light outputs can be generated using a decoder such as the 74LS42. Combinational logic is needed for the Count enable and Clear signals, as well as the timer outputs. Figure 5 gives a diagram of the system. Table 9 gives the state assignments. The functions are derived from the state diagram, and are given in Table 10. We want to count when in an even state when X_G goes off, or in an odd state whenever X_Y goes off, unless we want to reset instead.

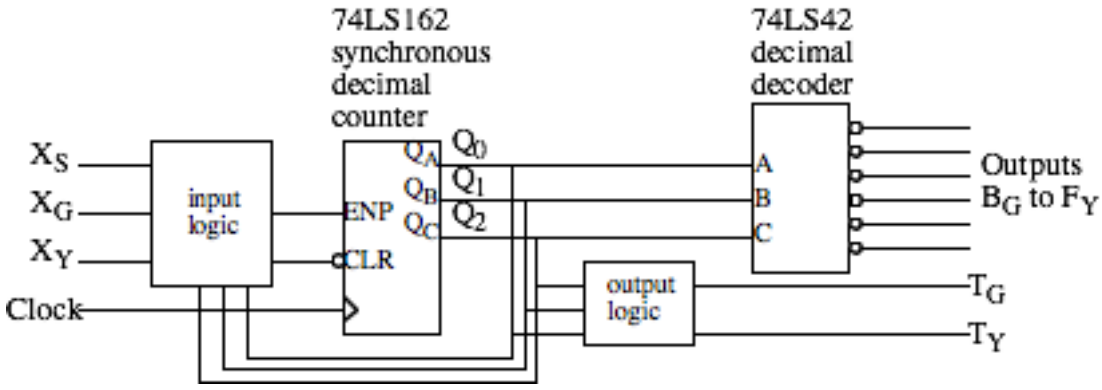

Figure 5 Counter based machine design diagram

Table 9 Synchronous Counter Design

State	Variables (Q3 to Q0)
A	0000
B	0001
C	0010
D	0011
E	0100
F	0101

Table 10 Count Enable, Clear, Output functions for the synchronous counter design

$ENP = Q_0'X_G' + Q_0X_Y'CLR$

$CLR=Q_1Q_0X_S'X_Y' + Q_2Q_0X_Y'$

$TG = Q_0'$

$TY= Q_0$

This design requires a device each for the counter and decoder, and six gates with 16 inputs not counting inverters. The largest gate has four inputs. The combinational logic requires two devices (1 x 74LS20, 1 x 74LS10) plus an inverter for the clear signal and another for the input from the sensor. However, the 74LS42 decimal decoder has active low outputs; if that is a problem, another inverter device is needed. Overall, this design needs only 6 devices. This efficiency suggests that the classical design might be made considerably more efficient by using a binary rather than a Grey code order for the state variable assignment.

3.3.5 Binary ordered classical design

The classical approach was repeated with a binary state ordering. This design resulted in 15 gates and 39 inputs (exclusive of inverters), a considerable reduction. The state table is shown in Table 10, and the Karnough Maps in Figure 6.

Table 10 A state table with binary assignment of state variables

State variables		Input signals: X_S,X_G,X_Y								
State	$Y_2Y_1Y_0$	000	001	011	010	110	111	101	100	Outputs
A	000	001	001	000	000	000	000	001	001	10000010
B	001	010	001	001	010	010	001	001	010	01000001
C	010	011	011	010	010	010	010	011	011	00100010
D	011	000	011	011	000	100	011	011	100	00010001
E	100	101	101	100	100	100	100	101	101	00001010
F	101	000	101	101	000	000	101	101	000	00000101

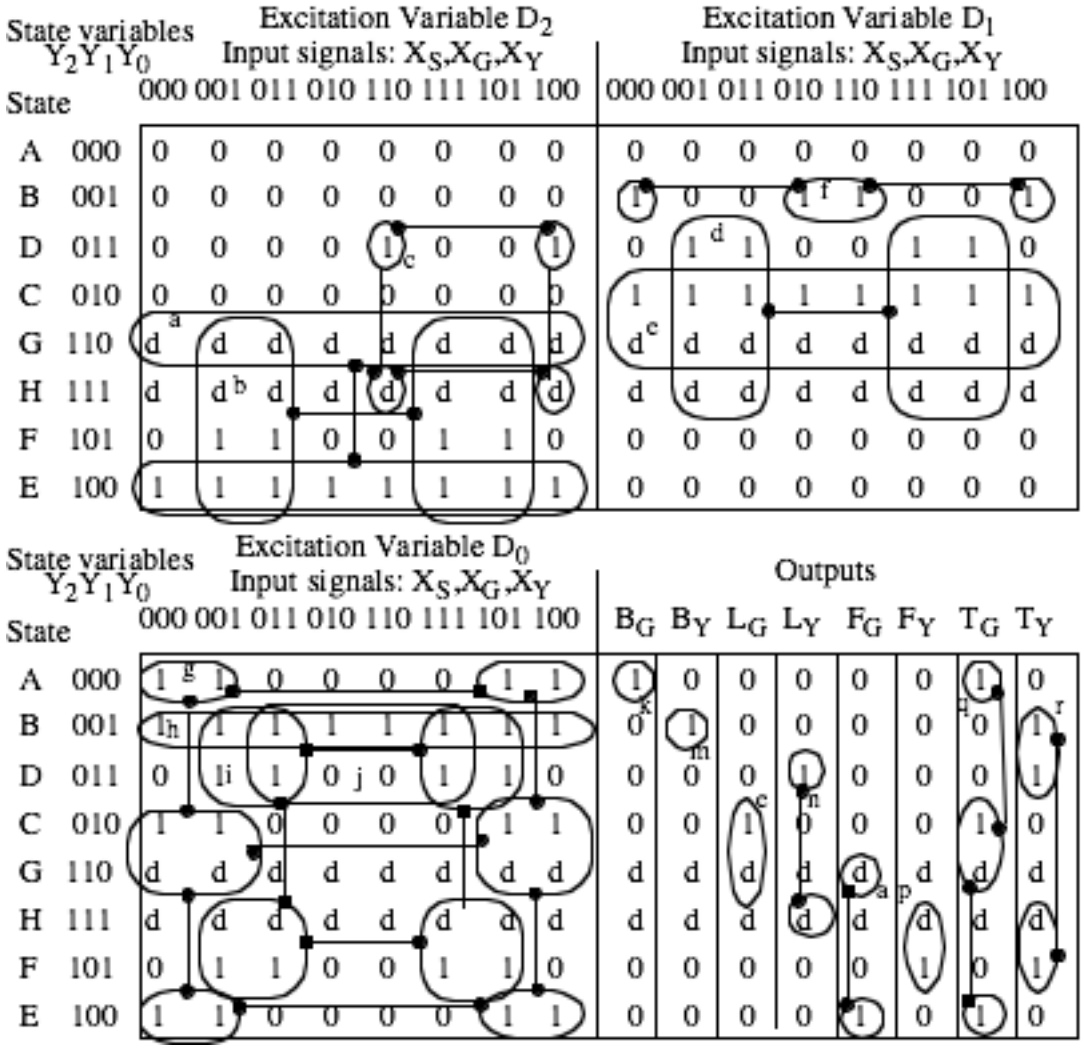

Figure 6 Karnough Maps for binary assignment of state variables.

The state variable excitation equations and the output equations derived from the Karnough Maps are given in Table 11. These functions can be implemented in TTL using x devices (1 x 74LS20, 2 x 74LS10, 2 x 74LS00) plus perhaps an inverter or two necessary to give inversions for single gate circuits if implemented with NAND logic.

Table 11 State variable excitation and output equations for binary assignment

$D_2 = Q_2Q_0' + Q_2X_G + Q_2Q_0X_SX_Y'$ (terms a,b,c)
$D_1 = Q_1Q_0' + Q_1X_Y + Q_2'Q_1'Q_0X_Y'$ (terms d,e,f)
$D_0 = Q_2'Q_1'Q_0X_Y' + Q_0'X_G' + Q_0X_Y$ (terms f,g,h)
$B_G = Q_2'Q_1'Q_0'$ (term i)
$B_Y = Q_2'Q_1'Q_0$ (term j)
$L_G = Q_1Q_0'$ (term d)
$L_Y = Q_1Q_0$ (term k)
$F_G = Q_2Q_0'$ (term a)
$F_Y = Q_2Q_0$ (term m)
$T_G = Q_0'$ (term n)
$T_Y = Q_0$ (term p)

3.3.6 Implementation

The binary ordered classical machine design was the design chosen for implementation. The schematic of the circuit is shown in Figure 7. The device identification and power and ground connections are given in Table 12. There are several point to note. The function F_Y was implemented using NOR logic (By DeMorgan's Theorem) instead of AND to utilize the fourth gate on the device to be used to generate the red signals B_R, L_R, and F_R. There were two extra 2 input NAND gates and two inverters needed, so the extra BAND gates were used as inverters to generate LG and FG. Finally, a 3 input AND gate was used for B_G, B_Y, and L_Y instead of NAND to eliminate the need to invert these signals.

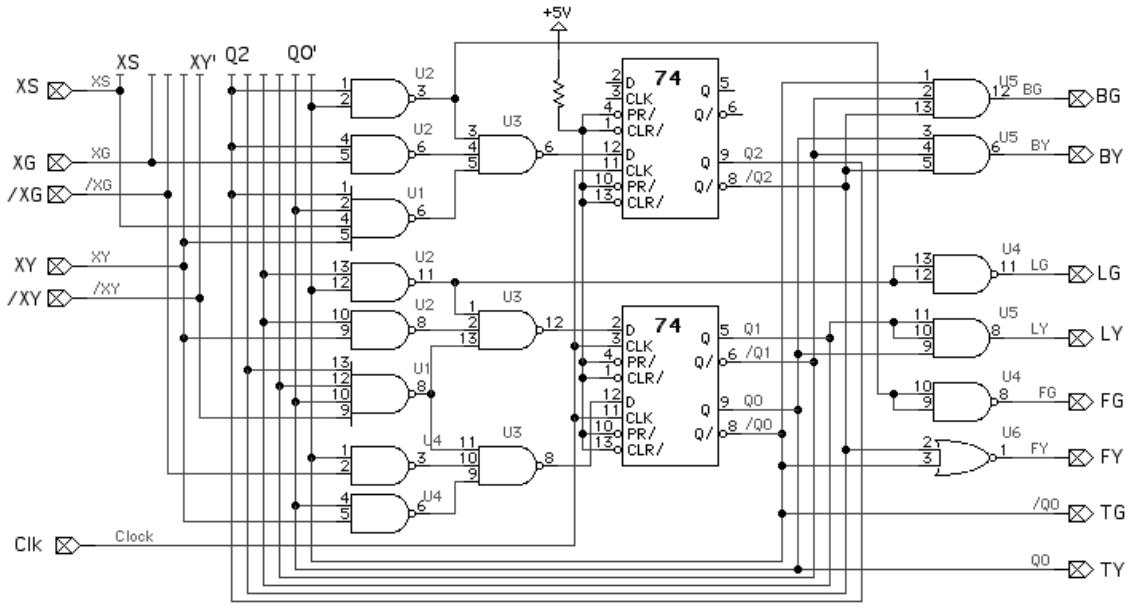

Figure 7 Traffic Signal Sequential Design Schematic

Table 12 Traffic Signal Sequential Controller Power and Ground Connections

Device	Type	Vcc	Gnd
U1	74LS20	14	7
U2	74LS00	14	7
U3	74LS10	14	7
U4	74LS00	14	7
U5	74LS11	14	7
U6	74LS02	14	7
U7	74LS74	14	7
U8	74LS74	14	7

Figure 8 shows the design of the Red Signal output logic, and Figure 9 shows the design of the One-Shot timing circuits.

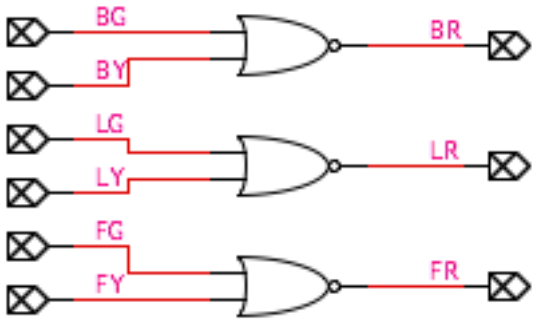

Figure 8 Red signal output logic

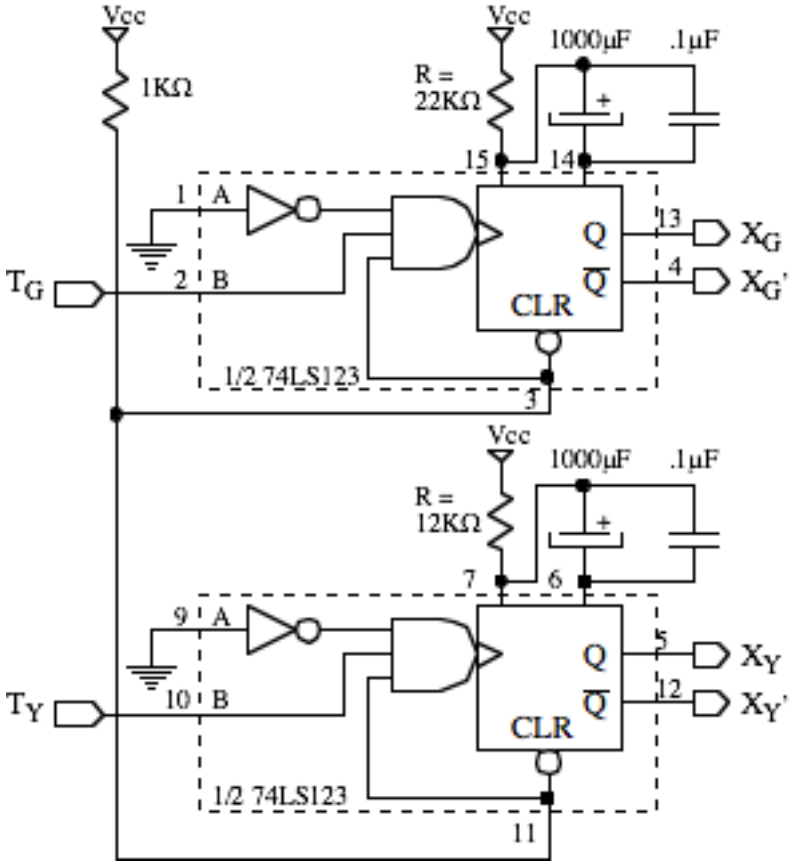

Figure 9 One – Shot timing circuit

4. Procedure

When this laboratory exercise was performed, the version of Logicworks available to simulate the system was Logicworks 3, which did not include One-Shot multivibrator. So it was not possible to test the design as a complete system. Instead, binary switches were used for inputs X_G and X_Y, with inverters to generate X_G' and X_Y', as shown in Figure 10. A binary switch was used to generate X_S. The switches were manipulated to generate a timing sequence. Errors found were then used to correct problems in the design.

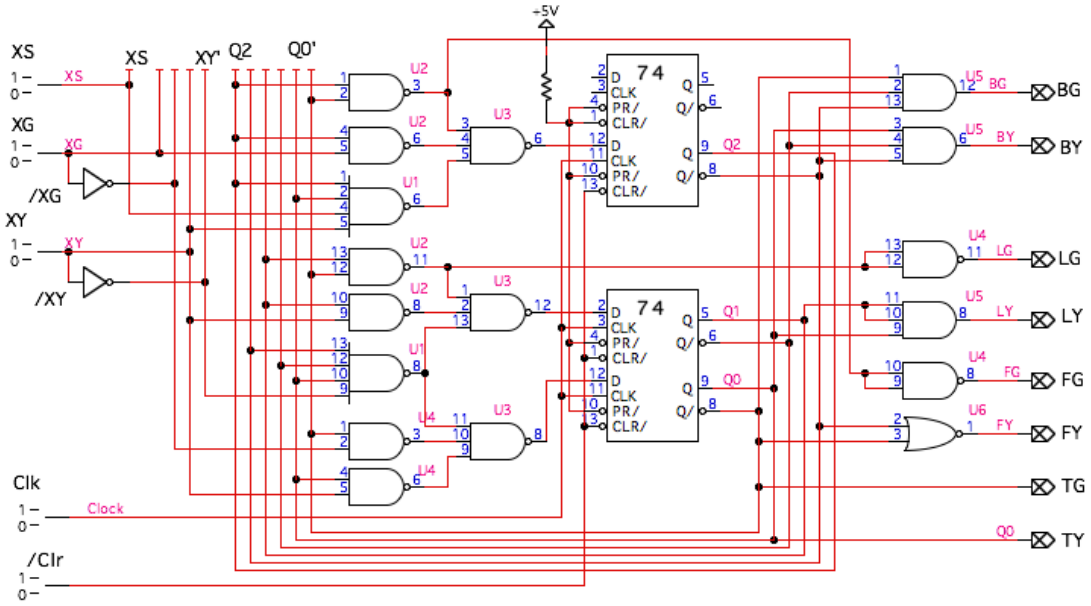

Figure 10 Simulation Test Circuit

The design was then implemented using LSTTL logic on a solderless breadboard, with LEDs used to represent the traffic signals. The correct operation of the circuit was observed, although occasional problems with the One-shots were observed, as described later.

5. Results

The simulation results in the form of a timing diagram are shown in Figure 11. In this version of the report, there is still a fault in the design; the cycle goes from Brook Road Yellow to Laburnum Yellow without first going to Laburnum Green. (This may be because the one-shots were not quite properly manipulated. There is also some reason to believe that the Logicworks 3 7474 part is not working correctly; for Q_0 both Q and /Q outputs had 0 value at the same time, which obviously is not correct. (This needs to be re-done in Logicworks 5.)

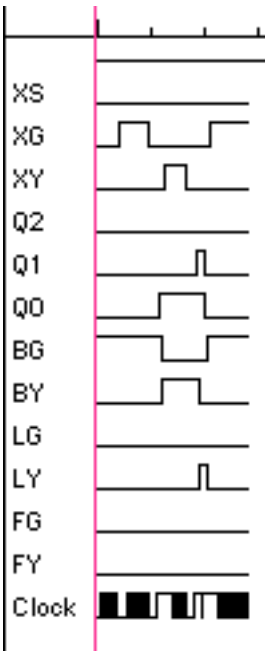

Figure 11 Timing diagram for simulation of the system

The actual circuit worked, following the normal two street cycle when the sensor was inactive, and including the Fauquier Avenue Green and Yellow when the sensor was active. There were occasions when the one-shots retriggered when they should not have, despite careful bypassing.

6. Conclusions

The actual circuit worked as expected, and was seen to operate correctly on May 30, 2008, even though the simulated circuit failed. The reasons for the simulation exercise failure may have to do with problems in Logicworks 3. At this time no clear reason for the problem has been found.

The design used was one of several possible approaches. This particular design could have also been built using a single GAL with perhaps one or two small SSI devices, and that would have been more efficient, but it could not have been simulated since Logicworks cannot represent the function of a GAL with a program generated using WinCUPL, and the Abel student edition that was supported by Logicworks is no longer available.

Reference

Fairchild, "DM74LS123 Dual Retriggerable One-Shot with Clear and Complementary Outputs," accessed from: http://www.jameco.com/Jameco/Products/ProdDS/46480.pdf on May 29, 2008.

Comments:

This example is a design oriented laboratory exercise at the more complex end of the scale. Notice that the design documentation, including the decomposition of the overall system into three parts, the exploration of different design approaches, and the final detailed design steps constitute most of the report. For that reason, sub-headings are used, and these are numbered.

Usually reports have a better balance of size between major headings, making the use of subheadings less necessary, but here's an example to the contrary.

Good engineering design involves considering alternate ways to do things. It is usually helpful to the reader to see very explicitly the approaches that were considered and how they compare with the method used. Often these can be compared using a "metric" such as the number of devices, number of gates, or number of gate inputs. Ultimately, it's about cost, and these metrics can be used as surrogates for the quantities important if a design is manufactured in quantity. If an actual SSI based design was to be built, the number of devices would be most important. If it was to be built with CMOS custom silicon, the number of inputs (with allowances for latches and buffers and such) would be most important, because that scales closely with the number of transistors, and perhaps wafer device area. If the design was to go into a Field programmable gate Array, the number of gates is probably the most important metric. It might have been useful to include a table comparing the different methods with respect to these metrics. If the alternative designs had been implemented further, those details might well have been relegated to an appendix, and even the K-maps might well have been put there as well.

Notice the different graphics. The image that became Figure 7 was converted into a bit mapped graphic, then had to be stretched to fit the area. It doesn't look good. (In the process, it was also converted to grey scale rather than color, and stored as a TIFF file.) It requires 292 KB of storage. Figure 10, with very similar content, was converted into a PICT file by the original program Logicworks 3 and then directly loaded into Word by insertion, and came out looking much better. It requires only 16 KB, and still includes color. The difference: the second graphic was not converted into pixels, it remained in a line oriented format. That is much to be preferred! If you use very many images, especially if they have sufficiently good resolution to print well, your document in electronic form will get quite large. That means it's an early candidate to throw away when people try to clear space, and will be slow to download. Another example is Figure 11, captured by screen shot as an image, then stretched to larger size. Because it is simple and most lines are vertical and horizontal, its limited resolution does not affect readability nearly so much as the process used to obtain Figure 7. It was also much smaller, at 4KB.

Notice that the State tables are listed as Tables, but the Karnough Maps are figures. The latter perhaps could be justified as tables, but they do have the drawn gates.

(In this draft of this manual, the lab report above is an example only. It has not actually been tested yet, and is hastily provided in its current form until the laboratory (and, especially the simulation) work on it can be completed. Clearly one would prefer to show good simulation results. The graph showing the results should be annotated to point out significant features, and that has been deferred for now until Logicworks 5 can be used to get good results.)