Experiments for the

Lab9500

Experiment 217
- Broadside Loadable Down Counter -

Introduction: In a simple traffic light, each state is timed. It is appropriate that our traffic light run continuously and that every state be timed appropriately. The traditional way to make a timer is to set a counter to the number of units to be counted and then count down to zero. So it is proposed that a suitable timer can be implemented by making a down-counter which can initialized (loaded) with the number of units to count. For a traffic light, a resolution of one second would appear to be appropriate.

I should be noted, however, that if zero is the terminal state, and if the counter is reloaded following zero, then the timed interval will in fact be N + 1 seconds, where N is the number loaded into the down-counter. Accordingly, to make N the number of seconds counted, the terminal state must be a count of one.

For the traffic light, a long green time of 32 seconds is sufficient. This requires a five-bit counter. The easy way to make counters is with T-type flip-flops. Here are the T definitions for any down-counter:

Call the LSB of the counter D0, and the MSB, D4.
T_{D0} = 1 ; toggles on every count
T_{D1} = /D0 ; toggles when D0, the preceding bit, is false
T_{D2} = /D1*/D0 ; toggles when preceding bits D1 and D0 are false
T_{D3} = /D2*/D1*/D0 ; toggles when preceding bits D2, D1 and D0 are false
T_{D4} = /D3*/D2*/D1*/D0 ; toggles when preceding bits D3 - D0 are false
T_{DN} = /DN-1* …*/D0 ; toggles when all preceding bits D_{N-1} - D0 are false

Let us define a signal that goes true when the terminal count is reached, namely, one. Call it ONE. When ONE is false, we are at a count other than 00001 and the definitions for the T inputs of the counter are as above, except now we would AND in /ONE. That is, the regular T definitions are enabled when ONE is false.

When the counter reaches the terminal count, if we want to preset a bit on the next count, we need to toggle it, since it is a zero for the terminal count (00001). The exception is the LSB, which is one at the terminal count. The logic for “toggling” this bit will be the opposite.

Actually, the terminal state, called ONE, has another important function. It is the signal that tells the state machine to change states. So that ONE becomes true, the very next one-second clock

will a) broadside load a new count, and b) increment the traffic light state counter. What gets loaded into the down-counter depends upon the current state. For example, if we are in either amber state then the next state will be one of the overlapping reds and the counter will be initialized to two seconds. If we are in either of the green states, then the next state is going to be an amber state and the time three seconds must be loaded. Note, that for the upper bits that are zero and will remain zero, they will NOT toggle and there is not need to include product terms for these loads.

In summary, normal counting is enabled when ONE is false. When ONE is true, there will be product terms depending upon the current state that will either load in a fixed count such as two or three seconds or a variable count for the long green and short green times which will be the contents of two registers.

Experimentation

Design a down-counter with a terminal count of ONE. Combine it with the simple six-state traffic light machine and the 1 Hz clock generated from the 60Hz. This requires implementing the 1Hz clock from 60Hz.

This piece really coordinates all of the pieces that make up the traffic light. The one-Hertz clock has already been done. The state machine stepped by a pushbutton has already been done. Combine the two so that the traffic light steps every second. Now make the down counter and make the load something like four seconds. Condition the clock for the traffic light state machine with the variables, ONE. Now, the timer is working, and the state machine should step every four seconds. The final step, of course, is to let the state of the state machine determine what gets loaded into the down-counter. Have fun!