EE 209 Lab 7 – A Walk-Off

1 Introduction

In this lab you will complete the control unit and datapath for a simple crosswalk controller that was discussed in class. You should work on this lab INDIVIDUALLY!

2 What you will learn

This lab is intended to teach you how to implement state machines, use datapath components and perform a non-trivial digital design.

3 Background Information and Notes

1. **The Crosswalk Control Unit (FSM)**

   The crosswalk system we will design in this lab is the same as the design covered in lecture. Pedestrians should be allowed to walk for 8 ticks of the clock, followed by a blinking hand (do not start walking) for 16 ticks of the clock. During these 16 ticks, a solid hand will blink on for a tick and then off for a tick while a counter counts down from 8 to 1 decrementing every **two** clocks. Then the system should display a solid hand continuously for 16 ticks indicating the red light period of the intersection.

   The outputs of the entire design are described in the table below:

Output	Description
WALK	Output to control when the "walk" icon should appear on the display. For our system we will use LED 3 and a 7-Segment display with the letter "G" for "Go" to indicate the walk state.
HAND	Output to control when the "hand" icon should appear (either during its blinking phase or solid phase). For our system we will use LED 1 and a 7-Segment display with the letter "H" for "Hand" to indicate the blinking or no-walk state.
NUM_ON	Output to control whether the "don't walk countdown timer" should be displayed.
NUM(3:0)	Value of the "don't" walk countdown timer.

Table 1 - Crosswalk System Outputs

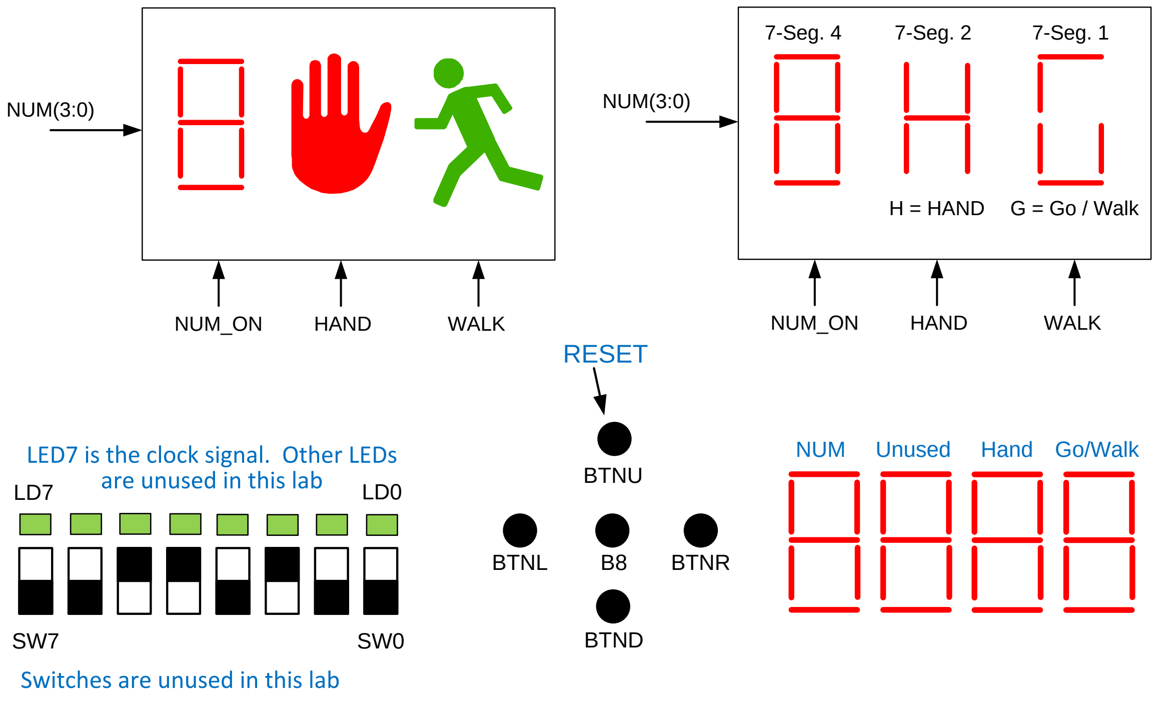


Figure 1 - Traditional Crosswalk Display and our modified FPGA I/O display and mapping.
Note: The signals going *into* the above blocks are being produced by (i.e. *outputs* of) your design.

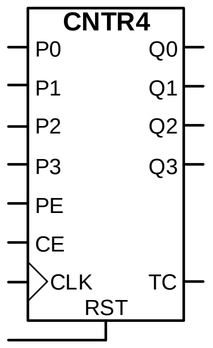
There are several possible ways this might be implemented. However, they all require a counter of some form. While we did present a possible solution in lecture, **we strongly encourage you to consider alternate ways of implementing this design but you may choose any one you want.**

Possible options:

1. If we think about the entire sequence we see it requires 40 clock cycles (8 in walk + 16 in Blinking + 16 in NoWalk). We could use a counter to simply count from 0-39 and then restart. In this case, it may be that no distinct FSM is need…the counter itself serves as the state. From the current count we could infer the state/outputs that we need to generate.
2. We could use 3 separate counters (one for each phase: Walk, Blinking, NoWalk). This might make the rest of the logic easier. However, it would lead to a larger design in all likelihood. However you are not going to be graded on size of your design, only correctness.
3. The lecture design

2. The Crosswalk Datapath

To help implement the overall system you will need to use a 4-bit counter. We've started this component for you in cntr4.v. You will need to complete it. A block diagram and function table are shown below. In addition, we also provide a 4-bit adder (adder4.v). This adder is likely necessary because we have an up counter, while we want a single digit counting down from 8-1_{10}. Think how you can achieve this conversion using the current count and the adder. Use these to help produce the needed outputs.



CLK	RST	PE	CE	Q*
0,1	-	-	-	Q
↑	1	-	-	0000
↑	0	1	-	P[3:0]
↑	0	0	1	Q+1
↑	0	0	0	Q

TC = Q3•Q2•Q1•Q0•CE

Figure 2 - Block Diagram and Function Table for 4-bit "CNTR4" Counter

4 Procedure

You will design and implement the crosswalk system including the control unit / FSM and the datapath. A testbench will be provided to give some confidence that your system is working. Then you will implement the design on the FPGA board to see it in action.

1. Download the `cwalk` project .zip file from Blackboard. Extract the files to a folder.

2. The crosswalk project has a completed top-level file to interface your design to the FPGA, a skeleton Verilog design file (cwalk.v) and a skeleton control unit / state machine schematic (cwalk_fsm.v). It also contains components you may use like: cntr4.v (incomplete), adder4.v (complete), and dff1.v (complete). The top-level FPGA interface file (cwalk_top.v) generates a slower clock signal, the reset signal, as well as converting all the outputs to appropriate 7-Segment display values (i.e. you will output a signal like WALK and our top-level schematic will convert that to "G" for Go). No changes should be made to this file.

3. Complete the cntr4.v module by adding 4-bit wide 2-to-1 muxes such that the counter implements the function table shown in Figure 2. You may then use any number of 4-bit counter instantiations in your overall crosswalk design.

4. Try to sketch out your design on paper and consider what approach you want to use. You may choose any approach you like. It just needs to produce the correct output sequence as specified in the requirements. Spend some time on this and make sure you believe it will work before you start coding in Verilog.

5. Implement your design. Add comments at the top of your file describing the approach you chose to implement and a bit about how it works.

   If you need to create a state machine, do it in the cwalk_fsm.v. (You are welcome to change the input/outputs of the module as needed). If you don't need to create a state machine, just leave that file blank.

6. In cwalk.v, put together your entire design along with the counter(s), adder(s), state machines, etc.

7. Simulate your design using the provided testbench. Click over to "Simulation", select the "cwalk_tb" and use it to simulate your design. (Note: This simulates your cwalk design, not the top-level design which is complete and valid). Right-click on "Simulate Behavioral Model".."Process Properties". Ensure "Simulation Run Time" is 1500ns or more. Now double-click Simulate Behavioral Model. The testbench runs for 1500 ns which is enough time to go through at least one full Walk, Blinking, Don't Walk sequence. Verify the correctness of your design, finding problems, fixing your code, and re-running the simulation as needed.

8. Click back to "Implementation". Synthesize, Implement and Generate the Programming File for your design.

9. Connect the programming cable to the FPGA board and download/program your design. Ensure it works as you expect.

10. Demonstrate your design to your TA/instructor & get their initials.

5 Review

None.

6 Lab Report

Name(s): ____

Due: _____________ Score: ________

(Detach and turn this sheet along with any other requested work or printouts)

1. TA/Instructor initials: _________
2. Submit your top-level file cwalk.v and your FSM if you needed it: cwalk_fsm.v.

7 EE 209 Lab 7 Grading Rubric

Student Name: ____

Item	Outcome	Score	Max.
Check off by TA	Yes / No		3
Output correctness			
• Appropriate counter design + FSM (if needed)	Yes / No		2
• WALK Output	Yes / No		1
• HAND Output	Yes / No		1
• NUM_ON Output	Yes / No		1
Datapath Correctness			
• NUM generation (i.e. Adder connections) correct	Yes / No		2
SubTotal			10
Late Deductions (-1 pts. per day)			
Total			10
Open Ended Comments:			