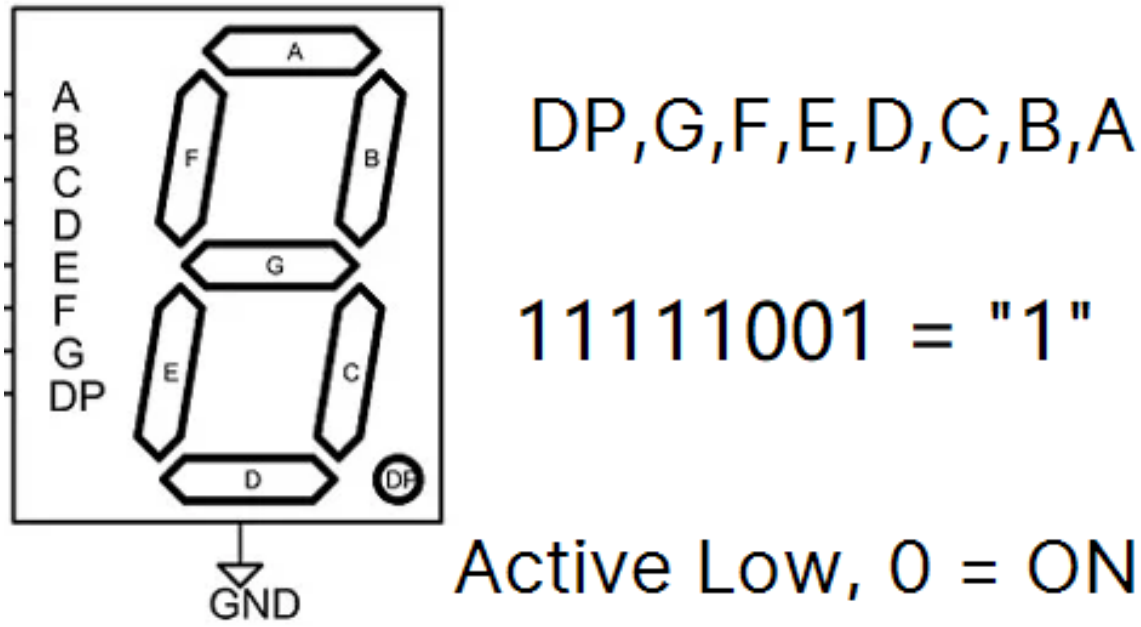
Lab 6: EE457 Digital Logic Lab – EE457 sliding sign

This design will require you to design your own FSM-Finite State Machine, we learned about state machines in Lab Exercise 5 where you did the control for the multiplier. Lab Exercise 5 will provide a good reference for creating a state diagram/bubble diagram and the structure for your VHDL. You will want to use multiple processes in this design. Look at your lab5 VHDL design. You had a process (clocked process) for creating the registers for holding the states of the Finite State Machine (FSM), a process for the next state logic, a Mealy process, and a Moore process, you do not have to have a Mealy and Moore process for this lab, design it so it makes sense to you. You are free to implement the state transitions, outputs, etc as you wish.

In this lab you will need to create the FSM to control what is displayed on the 7 segment displays (HEX5,4,3,2,1,0). You may use the generic counter (provided as gen_counter.vhd) as the baseline for this design as you will need to create a 1 second delay from the 50Mhz clock (MAX10_CLK1_50) and utilize the seven_segment_cntrl. There are other ways you could drive the 7 segment display without the seven_segment_control.vhd if you'd like, and you may write your own counter if you like. Note that If you do you use the seven_segment_control, you will have to modify the case statement slightly to display a blank display (all 7 segments off) and any other display features listed below.



A common problem when dealing with "human-scale" time like this 1s timer is that you don't actually want to run Modelsim for 1second of wall time, so consider implementing a generic into your block so that your testbench code can run fast (say 5us per change) and the hardware design can run at the intended human-scale of 1s.

Do create a component for the statemachine design, the full FSM should be implemented in the **de10_lite_base.vhd** file and the instantiate the counter and seven segment controls if you see fit (gen_counter.vhd and seven_segment_cntrl.vhd if using these.
Design Requirements:

1) Your design will start out displaying "EE457" on the 7 segment displays utilizing HEX5 to HEX1 and then every second shift right or left or stop depending on the keys pressed.

2) HEX0 will display a numeric representation of the current state of your FSM

3) You will need an asynchronous reset for the clocked process, use KEY(0) for this, it will be 0 when pressed.

4) During the reset state of your FSM, you will display "EE4570" on HEX5 through HEX0, HEX0 is a 0 since you are in state 0, the reset state.

5) The EE457 will slide to the right every 1 second, as the EE457 slides to the right, the upper HEX display will be blank (off) – I am going to represent this as a "b" below a. Reset: HEX display will be EE4570

b. 1 second later the HEX display will be bEE451

c. 1 second later the HEX display will be bbEE42

d. 1 second later the HEX display will be bbbEE3

e. 1 second later the HEX display will be bbbbE4

f. 1 second later the HEX display will be bbbbb5

g. 1 second later the HEX display will be 7bbbb6

h. 1 second later the HEX display will be 57bbb7

i. 1 second later the HEX display will be 457bb8

j. 1 second later the HEX display will be E457b9

k. 1 second later the HEX display will be EE4570 – you are back at the reset state

*Note: You may chose to implement this in a way that requires fewer than 10 states. If so, your last digit may be different. If so, explain your solution in your lab report. So long as the digits EE457 scrolls appropriately and your solution properly described, you'll get full credit for a different solution.

l. The design continues to shift until reversed, halted, or reset.

6) You will also need to reverse the shifting with sw(0) a. When sw(0) is up the EE457 will shift left (instead of right) and fill the lower HEX displays with blank/off. When sw(0) is down the sign will go the opposite direction. HEX0 will continue to display your current state. – will probably count backwards (hint)

7) KEY(1) will be used to halt the shifting. When KEY(1) is pressed (will be '0') the display will stop shifting. When you release KEY(1) the display will then continue to shift in whatever direction based on KEY(1)

8) You absolutely need to register the KEY(1) and SW(0) signals with a 2register synchronizer being clocked with the clock_50 input signal prior to using the signals in your actual logic. Also asynchronously clear these registers with the KEY(0)/asynchronous clear input signal using the reset synchronizer described in class. MAX10_CLK1_50 is a 50Mhz clock input.

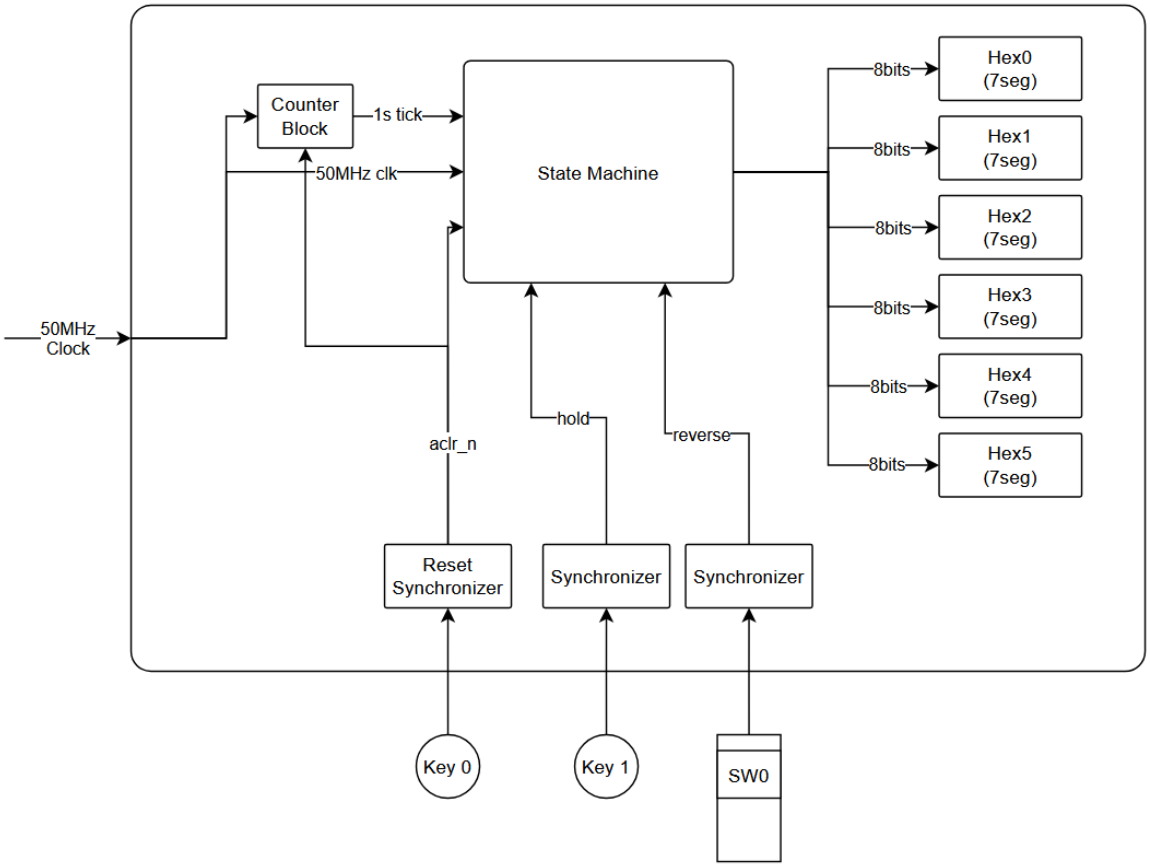
9) You must simulate backwards, forwards, reset (during run time) and normal operation in Modelsim so focus on your test bench file to get the inputs register values correct.

10) in addition to piping the data to the 7 segments you must also create a separate enumerated type for the seven segment display (This is for Modelsim debugging).

The Test Bench will allow you to test your design, but you need to build the test functionality. You will need to add a generic for the COUNTS_PER_TICK to both the base file's entity, and modify the instantiation in the testbench to match. For the synthesis (Quartus) case, the default value you assign will be used.

The lab write up must include a state / bubble diagram and a block diagram and follow the lab report guidelines from the syllabus/example that was placed on D2L. You must include Modelsim simulation screen shots in your lab report showing how you are testing the design.

Here is a suggested block diagram:



Steps to create:

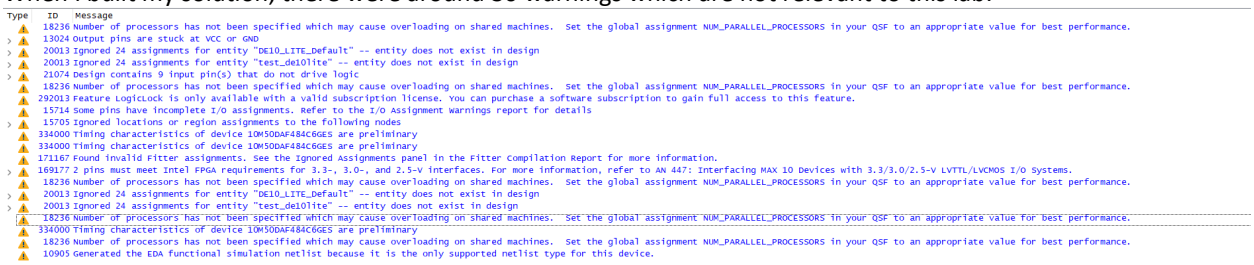
Step 1. Create a 1 Second tick signal. You may pipe the signal to an LED. This should have a generic to allow changing the frequency of the tick from 1s down to 5us. Suggest default value be 1s (for synthesis), and you can override in your testbench to 5us for simulation.
Some possible next steps, but ultimately the path to get to the solution is yours.
Step 2. Create 1 state and ensure that EE457 Appears on the seven segment displays.
Step 3. Create the remaining state machine states and attempt to get the project to move between all 10 states.
Step 4. Implement the reverse and reset.

Running your design on hardware is optional (but fun, and worth experiencing if you're serious about digital logic design!), but your design must compile and I will run it on hardware and provide feedback.

Lab Submission instructions:
1) rename the de10_lite_base_f2024 folder to lab6 if you haven't done so already.
2) Once your design is working, **delete** the following folders, which just have output stuff from your builds and simulation that take up space in the zip file:
db/
output_files/

work/
3) Zip up your project folder (see syllabus for testing instructions to make sure you don't have absolute paths, missing files etc)
4) See syllabus for lab report format and expected contents
5) Submit lab6.zip and your lab report via Canvas.

Other notes:
When I built my solution, there were around 30 warnings which are not relevant to this lab:



In the industry, I'd suggest that you go through these and fix or suppress them as appropriate. Some of these, like the one reporting no clocks defined are critical errors for something headed to production or engineering test. We'll talk more about timing and constraints later this semester.