**DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING**
**UNIVERSITY OF BRITISH COLUMBIA**
**CPEN 211 Introduction to Microcomputers, Fall 2016**

# Lab 4: State Machines and the HEX display

***Week of October 3 to 7 (due 11:59 PM the night before your lab session)***

## 1 Introduction

You will design a state machine in Verilog and connect it to one of the seven segment LEDs on the DE1-SoC.

### 1.1 Specification

The system you design cycles through and displays the first five digits of a number—any number you want. For example, you could use five digits from your own or your lab partner's student number or phone number. Your submitted code may be stored on computers outside of Canada. If you are concerned about privacy, use a random five digit number. The goal is that each cycle, the LCD on your DE1-SoC will display one digit of this five digit number.

The clock input to your state machine should come from pushbutton switch KEY0 on your DE1-SoC. Every time you press switch KEY0, another number should appear. If the first five digits of your number is "60412", the LED should display a "6" one the first cycle, then "0" in the second cycle, then a "4" in the third cycle, etc. On the sixth cycle, it should cycle back to "6" and start again. The seven segments of the seven segment display are each controlled by each bit of HEX0[6:0]. A segment turns on if the corresponding bit is 0 and off when that bit is 1 (this might be opposite to what you would expect). The mapping from bits of HEX0 to segments can be seen on the right side of the figure below.

To add a bit of a challenge (and make the lab more fun), the user (i.e., your TA) should be able to change the "direction" of cycling through the digits using slider switch SW0 on your DE1-SoC. If this switch is "up" (corresponding to logic value "1"), the system operates as described above. If this switch is "down" (corresponding to a logic value of "0"), the system should cycle "backwards" (but still starts with the first character). So, in the above example the order would be "621406".

To make things even more interesting (and fun), the user should be able to change the slider switch during any cycle. So, for example, you might go "forwards" for 4 cycles, "backwards" for 2 cycles, and "forwards" for 4 cycles, outputting a display of "6041404126".

Your design should include a reset input controlled by pushbutton switch KEY1. Your state machine should reset on the rising edge of `clk` if KEY1 is pressed. ***Note KEY0 and KEY1 output a 1 when NOT pressed and a 0 when pressed (this is probably the opposite of what you would expect).***

Figure 1 shows the overall system you will build:

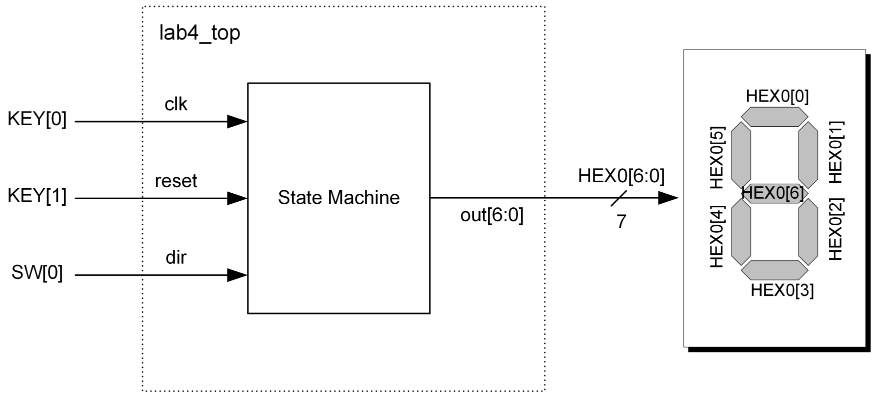

Figure 1: The circuit you will design for Lab 4

## 2 Lab Procedure

Design a state machine in Verilog to implement the circuit as described above. A sample state diagram might be something like the one shown in Figure **??** and assuming you used the first five digits of your phone number and your phone number happens to be '604-827-4116':

The reset should be synchronous. This means that when the reset signal is asserted (set to logic 1), the state machine is reset on the next rising edge of the clock. Again, remember that the actual output of the KEY1 will be 0 when you press the key and 1 when it is not pressed.

The state machine is positive-edge triggered. This means that the transition from one state to the next occurs on the rising edge of the clock. Given KEY0 is 0 when pressed and 1 when not pressed, you should think carefully about exactly *when* your circuit will see the rising edge.

The output of the state machine is connected to the signal HEX0. You should be able to figure out what should be driven on the signal HEX0 to display a given number given the information in Figure 1. You will need to determine the 7-bit value that will display the number corresponding digit you want to put on the hex display. For example, `7'b00000110` will display a "1" and `7'b1011011` will display a "2".

The output of your state machine should depend only on the state it is in (and not the current input). If you end up getting a job where you do hardware design regularly you will probably hear people talk about "Moore" state machines, which is just a fancy name for a state machine in which the output only depends upon the current state (and not also the current input).

Put your synthesizable code in a file called `lab4_top.v`. You can start with the template version on Piazza. The code provided on Piazza uses signal names that correspond with pin assignments in the pin assignments file, `DE1_SoC.qpf`, from Lab 3, which you should also use for Lab 4.

You should also create a testbench file `lab4_top_tb.v` that simulates inputs on the pushbutton and slider switch to verify that your design works. Save your waveform format in a file `lab4_wave.do`. You will need to show the results of your simulations to your TA before demonstrating the design on a DE1-SoC.

## 3 Marking

This section describes the marking rubric for Lab 4. You can use either state machine coding style shown in Slide Set 5 for this lab.

**[4 marks] Simulation (ModelSim).** Your mark for this part will be:

**0/4 marks** If you don't have a testbench at all.

**1/4 marks** If you created a testbench and it compiles, but it does not work together with your FSM module, or the simulation output suggests you may have inferred latches, or you do not include a `lab4_wave.do` file.

**2/4 marks** If you created a testbench and it compiles and works with your FSM module, you include a `lab4_wave.do` file that includes both top level signals shown in Figure 1, and the state of your state machine, but your testbench and/or FSM module includes no comments or very few comments (see below), or you do not test for changing the input `dir`.

**3/4 marks** If you do all of the above but there are bugs in your simulation such as the FSM does not change state at all.

**4/4 marks** If your testbench extensively tests your synthesizable design, your `lab4_wave.do` includes the necessary signals mentioned above, the output looks correct in the ModelSim waveform to the TA, and your Verilog includes sufficient commenting. Specifically, that means, include at least one comment per test condition – e.g., change on input `SW[0]` or `KEY[1]` – in your test script and one comment per always block, module instantiation or assign statement in your sythesizable code. You do *not* need to include a comment for each change in the clock input `KEY[0]` in your test script unless you want to. You can get 4/4 on this part even if your FSM does not change direction at the right time when dir changes because 2 marks will be deducted for that error when it shows up on the DE1-SoC below. However, if that error *only* shows up in simulation but not on the DE1-SoC then 1 mark will still be deducted from this portion.

**[6 marks] Synthesis (Quartus and DE1-SoC).** Your mark for this part will be:

| | |
|---|---|
| **0/6 marks** | If your your design does not synthesize in Quartus. |
| **2/6 marks** | If your design synthesizes but it has warnings about inferred latches or the HEX does not display anything, or the display does not change. |
| **3/6 marks** | If your design synthesizes with no inferred latches warnings and it displays something on HEX0, which changes when the clock input is pressed but what it displays does not look like numbers to your TA. Or, if your design works except the reset does not work properly on a DE1-SoC. |
| **4/6 marks** | If your design displays the expected numbers on HEX0, they change on the rising edge of the clock and your reset works BUT they do not change direction properly when the slider switches changes (e.g., there is an extra cycle before the direction changes). |
| **6/6 marks** | If your design works properly on a DE1-SoC as far as the TA can tell. |

## 4 Lab Submission

Remember the lab code including all files created by ModelSim and Quartus must be submitted by 11:59 pm the evening before your lab section using handin. Use the same procedure outlined at the end of the Lab 3 handout except that now you are submitting Lab4, so use:

```
  handin cpen211 Lab4-<section>
```

where `<section>` should be replaced by your lab section. Remember you can overwrite previous and trial submissions using `-o`.

A short (15 minute) video walking through how to submit remotely, e.g., from a laptop or home computer running Windows can be found here https://youtu.be/bxr5dq0xHzc (this was recorded for Lab 3—make sure you make appropriate changes for Lab 4).

## 5 Lab Demonstration Procedure

To reduce congestion in the lab we will be dividing each lab section into two one hour sessions. For example, for L1A the first session will run from 9 am to 10 am and the second session will run from 10 am to 11 am. We request that you show up no more than 10 minutes before the start of your assigned one hour "Lab 4 Marking Time", which will be posted on Connect at least 24 hours before your lab section along with your "Lab 4 TA". The TAs will have a randomly ordered list of lab partners and will start working their way down the list marking. If you and your partner are not present when they ask to mark you and you have not told them where you are beforehand, your name will be put to the end of the list. If this happens the TA will be under no obligation to mark you, but may do so at their own discretion and if time permits.

Your TA should have your submitted code with them and have setup a "TA marking station" where you will go when it is your turn to be marked. However, we still require that you bring your DE1-SoC, submitted code, and (if you have one) laptop to MCLD 112. This is in case either your TA did not get your submitted code in time for the lab (it takes time to organize the files), or because they ask you to use another workstation or your laptop for your demo to better manage time. If they ask you to demo using your own workstation or laptop, then you **must** demo the exact same code you submitted via handin (so be sure you have a copy of that code with you).