State Machine Design with VHDL

Introduction

In this lab you will design a state machine that displays the six digits of your BCIT ID in binary.

You will use the same hardware, inputs and outputs as in the previous lab.

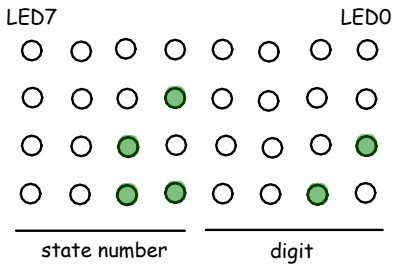
Requirements

Use `key(0)` as the clock and `key(1)` as an active-low *synchronous* reset input. Recall that the pushbuttons are active-low and the LEDs are active-high. The buttons are debounced by hardware on the board.

When the circuit is reset, the binary value 0 (all LEDs off) should be displayed on the most-significant (MS) four bits. Each rising edge of the clock should cause the binary value displayed on the MS four bits to be incremented by 1.

When the circuit is reset, the first of the 6 digits of your BCIT ID should be displayed on the least-significant (LS) four bits. Each rising edge of the clock should cause the next digit of your ID to be displayed on the LS four bits. Thus it will take six clock cycles to display all of the digits of your ID number.

For example, if your student number was A00123456, the first four patterns displayed on the LEDs would be:



If you student number is an even number, the next clock pulse should cycle the display back to the first. If your student ID is an odd number the subsequent clock pulses should hold the last display until the circuit is reset.

Your design's architecture should use exactly two selected assignment statements and one process statement (no more, no less). You may also use additional (unconditional) assignments to connect signals to outputs.

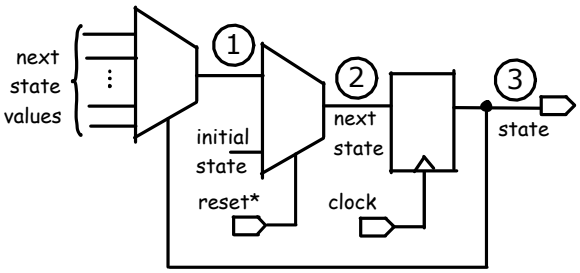
The process statement will implement a register that holds the current state. You *must* use the single-if `process` statement shown in the lecture notes. No other code may be placed inside the `process`.

The outputs must be registered – the output signals must be included in the state.

One selected assignment statement, which instantiates a 6-input multiplexer (mux), should use the current state to select what the next state would be if the reset input is not active.

The second selected assignment statement, which instantiates a 2-input mux, should use the reset input to select either the next state value as computed by the first selected assignment statement (if reset is false) or the initial state (when reset is true).

The structure of your design will thus look as follows:



You will need to declare additional signals with appropriate bus indices (e.g. `7 downto 0`) to connect the two multiplexers, the state register and the output. These are labelled ①,②, and ③ on the diagram above. You may use any signal names you want, although it's a good idea to make them meaningful, reasonably short and to document them in comments.

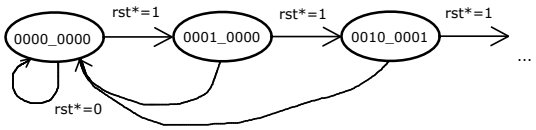
Pre-Lab Report

Submit a pre-lab report in PDF format to the appropriate dropbox on the course web site. The report

should contain the required identification information plus the following:

(1) the required identification information,

(2) a state transition diagram with each state labelled with the outputs for that state and the transitions labelled with the input condition(s) for that transition. For the example above the state transition diagram would begin:



Notes:
(1) rst* is (the active-low reset signal) KEY[1].
(2) state labels are the values of LED[7..0].

You can draw the diagram by hand and you may use a different notation and signal names as long as the diagram is unambiguous.

(3) A listing of VHDL code that implements a solution to the requirements above.

(4) Answers to the following questions:

(a) Does the output change when you press the "clock" button or release it?

(b) What sequence of button(s) do you need to press to reset your circuit?

(c) What is the minimum number of bits of state (flip-flops) needed for your design? Your answer should assume you followed the course guidelines and designed a Moore state machine with registered outputs. *Hint: the answer depends on your student ID number.*

(d) Would your design work if the most significant bits of state were not connected to the LEDs?

Procedure

Follow the same procedure as for Lab 1.

Lab Report

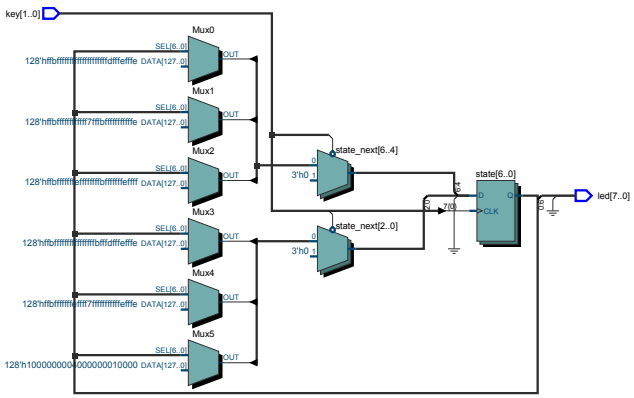
Submit a lab report in PDF format to the appropriate dropbox on the course web site. The report should contain the required identification information plus the following:

- your VHDL code with all errors corrected
- a screen capture of the Flow Summary section of the Compilation report showing the number of logic elements and registers used. For example:

Flow Summary

Flow Status	Successful - Mon Apr 04 22:03:28 2016
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Web Edition
Revision Name	lab2
Top-level Entity Name	lab2
Family	Cyclone IV E
Device	EP4CE22F17C6
Timing Models	Final
Total logic elements	19 / 22,320 (< 1 %)
Total combinational functions	17 / 22,320 (< 1 %)
Dedicated logic registers	6 / 22,320 (< 1 %)
Total registers	12
Total pins	10 / 154 (6 %)
Total virtual pins	0
Total memory bits	0 / 608,256 (0 %)
Embedded Multiplier 9-bit elements	0 / 132 (0 %)
Total PLLs	0 / 4 (0 %)

- the block diagram produced by running Tools > Netlist Viewers > RTL Viewer. For example[1]:



Bonus marks may be awarded for the design that is the most concise (fewest statements), smallest (fewest logic elements) and clearest (easiest to understand).

[1] The long constants at the mux input are because 7 bits select from 128 values.