CPEN 211 Introduction to Microcomputers, 2018
Lab Proficiency Test #2

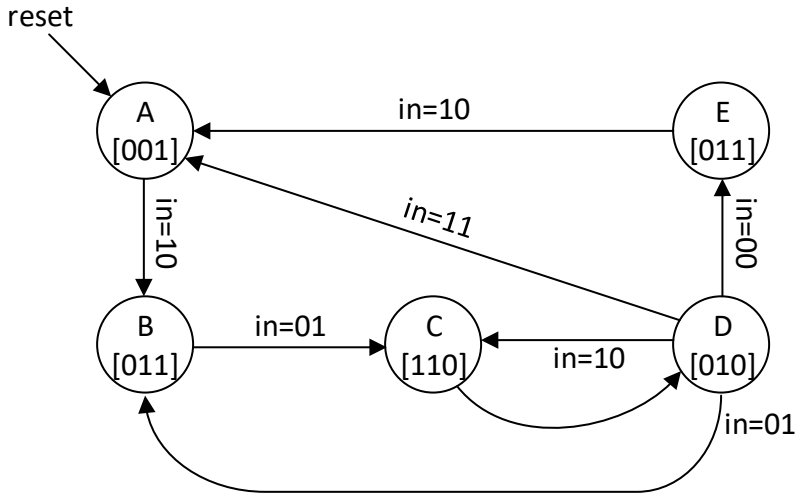
Question 1 [3 marks]: Create a file named "**q1.v**" and inside it write **synthesizable** Verilog that implements the finite state machine illustrated in the figure below. Like the examples in class, state transitions occur on the rising edge of input "`clk`" and the reset is synchronous (occurs on the rising edge of clk) and is active high (`reset` equal to `1'b1` means reset). Transitions from a state to itself are not explicitly shown. The condition for an unlabeled edge is always true. The input "`in`" is 2-bits wide. The output "`out`" is 3-bits wide. The output for each state is shown in square brackets. For example, when in state A, `out` should be 001, and if `in` is 10, then the next state should be B. The auto grader used to mark your answer will assume your top-level module is called "`top_module`" with inputs "`clk`", "`reset`" and "`in`", output "`out`", and that "`top_module`" is declared as follows:

```
module top_module(clk,reset,in,out);
```

Your q1.v file **must** include definitions for any modules instantiated inside `top_module` (even those from the slides or textbook). **Upload your Verilog file named "q1.v" via the "Lab Proficiency Test #2" assignment on Canvas before 6:50 pm as the submission site closes at exactly 6:50 pm.** Do **NOT** "zip" your submission. The file you upload for this question **must** be called "q1.v" or the autograder script will not mark it.

Your solution will get zero if any of the following are true:

1. Your **last** "Lab Proficiency Test #2" attempt on Canvas does not include "q1.v",
2. Your q1.v file does not compile using ModelSim (e.g., due to syntax errors),
3. Your q1.v file does not contain a module called `top_module` with inputs/outputs as above,
4. Your `top_module` cannot be simulated (e.g., due to missing module definitions in q1.v),
5. The Verilog used by your `top_module` is not synthesizable by Quartus.
6. The Verilog used by your `top_module` has any inferred latches.
7. Your `top_module` output "`out`" does not exactly match the output of the state machine below for some sequence of values for inputs "`clk`", "`reset`" and "`in`".



CPEN 211 Introduction to Microcomputers, 2018
Lab Proficiency Test #2

Question 2 [3 marks]: Create a file named "**q2.v**" and inside it write **synthesizable** Verilog that implements the circuit specified as follows. Your top-level module **must** be called "`fab`" and have the exact inputs and outputs shown in the lines of code below:

```
module fab(clk, reset, s, in, op, out, done);
  input clk, reset, s;
  input [7:0] in;
  input [1:0] op;
  output [7:0] out;
  output done;
```

In module `fab` write Verilog for a datapath and finite state machine controller that together implement the instructions in the table below:

Instruction	Value on input "**op**"	Operation (may take more than one cycle)
INIT **[1 mark]**	2'b00	Initialize `A` to zero, `B` to one and `C` to "`in`". Copy `B` to `out` and set `done` = 1. `A`, `B` and `C` are 8-bits wide.
FIB **[1 mark]**	2'b01	Compute sum = "value in `A`" + "value in `B`" then, if `C` is not zero, set `A` to the prior value of `B`, set `B` to sum, and update `C` to be `C` – 1. When `C` equals zero set `done` = 1 and copy `B` to `out`. Use normal (2's complement) addition for "+" and do NOT take any special actions in case the result `A`+`B` overflows.
FACT **[1 mark]**	2'b10	If `C` is not zero, set `B` = `B * C` and `C` = `C - 1`. When `C` equals zero set `done` = 1 and copy `B` to `out`. Use the "*" operator in Verilog to compute `B * C` and use the lower 8-bits of the result when updating `B`.

Your controller should reset to a wait state on the next rising edge of "`clk`" if "`reset`" is 1. In the wait state `done` should be 0. Assume inputs "`in`", "`op`" and `s`" are set on the same cycle. Once "`s`" is set to 1 these inputs will not change until your controller sets "`done`" to 1. Your controller should stay in the wait state until the input "`s`" is 1 and there is a rising edge of "`clk`". The final value of B computed by the instruction INIT, FIB or FACT should appear on output "`out`" sometime after "`s`" is set to 1. When the value on "`out`" is correct, your controller should set "`done`" to 1 and then wait for "`s`" to be set to 0 and a rising edge of `clk`. Then, your controller should return to the wait state. The autograder will only check the value of `out` when `done` is 1. An interface checker is here: https://cpen211.ece.ubc.ca/2018/lpt-2/q2_checker_xw8.html (you may need to type out this URL if clicking or cutting and pasting doesn't work). Ensure "`q2_check_tb`" prints "INTERFACE OK". The `q2_check_tb` testbench includes simple tests for INIT, FIB and FACT. The autograder will try additional test cases. **Inferred latch and/or synthesis errors in Quartus will result in up to 2 marks deducted for Question 2.** Your q2.v file **must** include definitions for any modules instantiated inside `fab`. Upload "q2.v" via "Lab Proficiency Test #2" on Canvas **before** 6:50 pm.

WARNING: Remember to include "q1.v" when submitting "q2.v" on Canvas. To do this, in the "File Upload" area on Canvas click on + Add another file so that you can upload two files. **DO NOT** "zip" your ".v" files before submission. If you resubmit q1.v and/or q2.v and Canvas renames the new files to "q1-N.v" or "q2-N.v", where N is an integer this is OK.