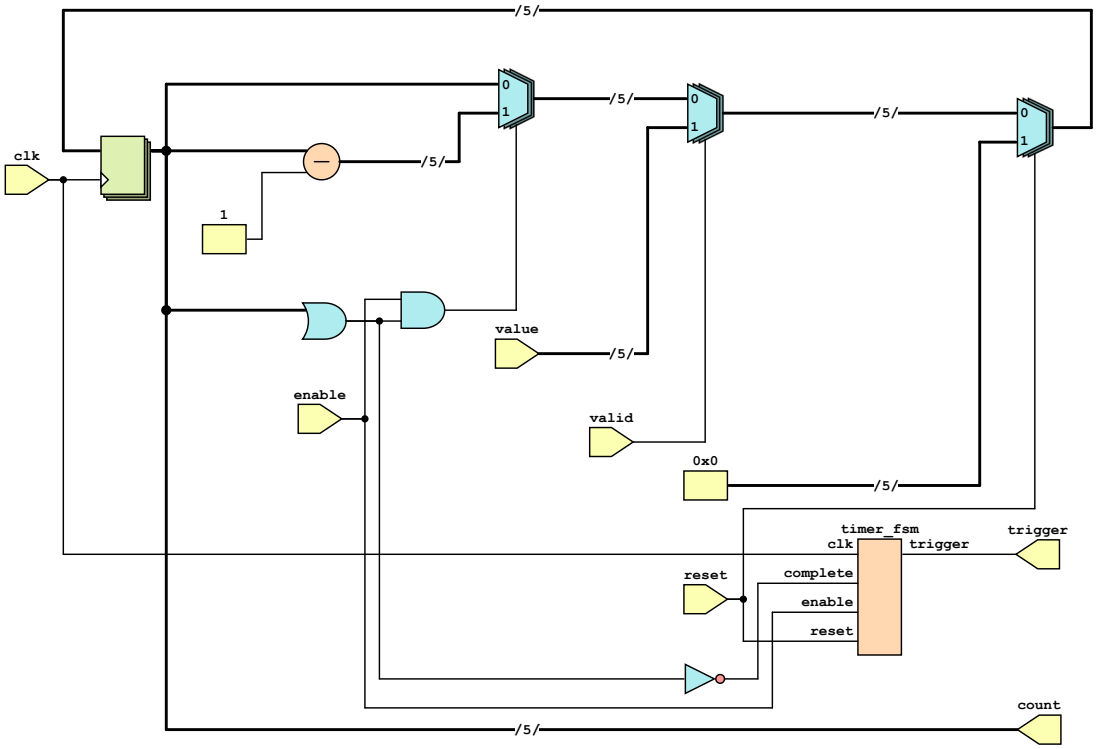
Computation II: embedded system design (5EIB0)

[redacted]

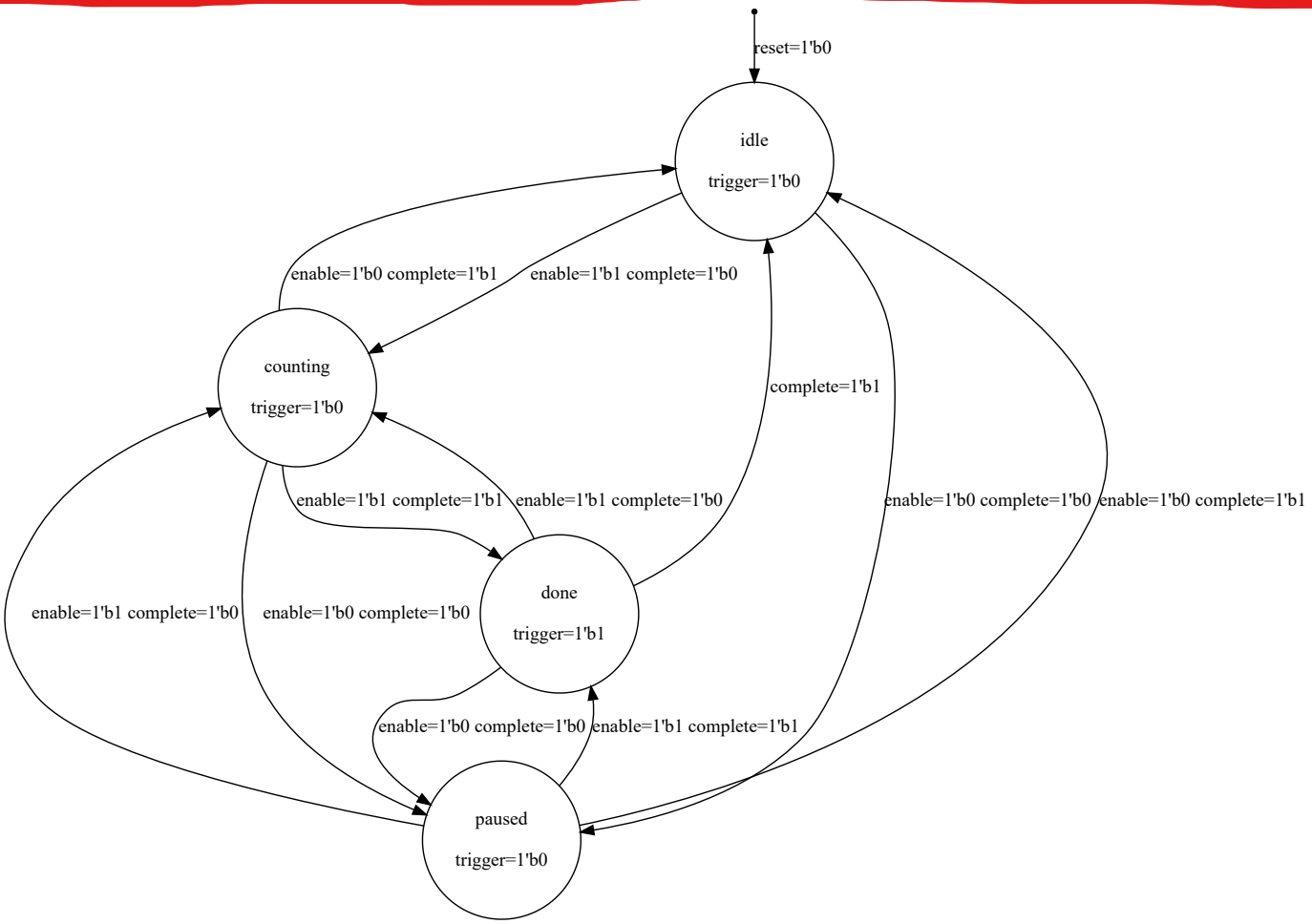
Type of work

A counter (or timer) is a very common hardware module with many uses. In this assignment you will create a Verilog implementation of a countdown timer. This timer will count down from a programmable value until it reaches zero. When it reaches zero it will produce a trigger signal. The count will only proceed it the input enable signal is high and will otherwise retain the previous value. The module can be implemented as follows:



Input signals are illustrated towards the left of the diagram with an arrow to the right. Output signals are illustrated towards the right of the diagram with an arrow to the left. Wires are shown as lines. Bold lines represent multiple wires with the bit width specified on the line between forward slashes, e.g. a bit width of 5 is represented as a /5/ on the line. Registers are illustrated as rectangular boxes with a small triangle where the clk input is located. Modules to be instantiated are shown as vertical rectangular boxes with input ports on the left and output ports on the right. The names of the modules are shown above the rectangle. The **`timer_fsm`** module needs to be created as follows.

In this assignment you will create a Verilog implementation of a Finite State Machine (FSM) with the following specifications. Including the clock (clk) and reset signals, the implemented FSM will have 4 input signals and 1 output signal.



The clk input signal is 1-bit wide, the reset input signal is 1-bit wide, the enable input signal is 1-bit wide and the complete input signal is 1-bit wide. The state elements of the FSM are to capture their inputs on the positive clock (clk) edge and reset their state when the synchronous reset signal is high. A reset can be asserted (synchronously) at any time causing all state elements to reset and hence a transtion to the initial state.

The trigger output signal is 1-bit wide. Initial values of the output signals upon reset are shown in the FSM diagram.

Your solution will be tested using bounded model checking against a golden reference implementation. The model checking software will stop at the earliest moment that your implementation diverges from the behaviour of the golden reference implementation. If a divergence takes place, a waveform will be created showing a trace of the module signals resulting in the divergent state. This waveform also shows the behaviour of the golden reference (REF) so that you can see what your implementation (DUT) should have done differently.

Please note that the golden reference implementation supersedes all other implementation descriptions. If the output from your solution implementation does not diverge from the golden reference implementation then your solution is marked as correct. Otherwise, your solution is incorrect and you should use the waveform that is produced to assist you to correct your implementation.

VPL