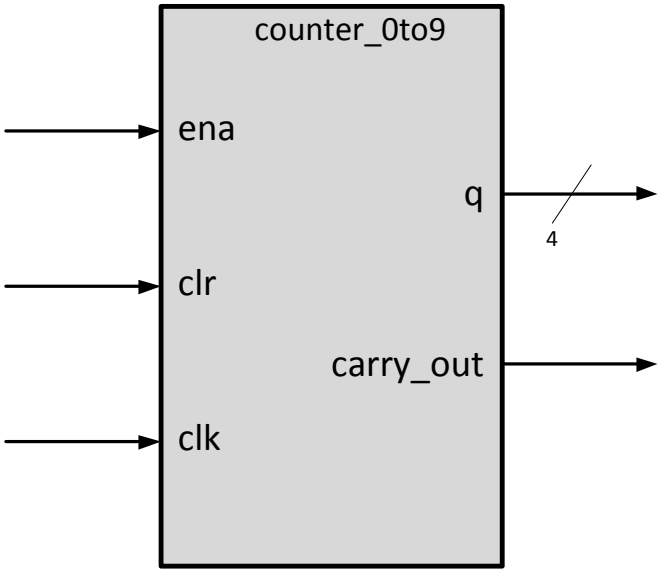
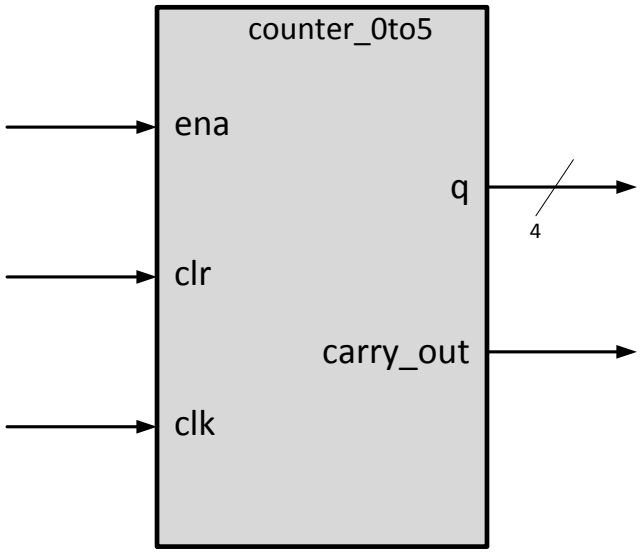
Lab 6 – Counters

In this lab you will design two different counters that will be used in a later stopwatch design. The first counter will be a 4-bit BCD counter that counts up from 0 to 9. You can use pieces of your design code from Lab 5 for the combinational block. The second counter will be a 3-bit counter that counts up from 0 to 5. This will be used for the 10 x second and 10 x minute digits functions (i.e., max number of seconds is 59 – then it increments to 00.) Both of these counters have similar input and output signals – the input signals are a clock (clk), enable (ena) and clear (clr). The output signals are the binary count bits (q) and a carry out (carry_out) [when the maximum count is reached]. For more information (truth tables, signal priority, etc.) refer to the notes of Lab 5.

Both counters should be designed each with two internal blocks (*always* constructs): one for the sequential block and one for the combinational block. This will be a **simulation only** lab. There will be no hardware implementation of these until a future lab.

The block diagrams for each are given below.





Lab steps:

1. Open the *lab6* project in directory lab6. It will contain two simulation testbench wrappers (**tb_** prefix), and the associated text files for each. **You will not need to edit these**, as they will contain the full testbench information.
2. Add a new module (**counter_0to9**) to this project using the module wizard. Refer to the block diagram above for appropriate naming of the input and output ports.
3. Create your design based on the specifications for the counter (see Lab 5) (also, follow *Recommended Implementation Style* below). Don't forget to *reg* the outputs. Before you start the Verilog coding, sketch your two internal *always* blocks for this design (counter_0to9) – the sequential and combinational blocks. Define what internal signals you will need, and note what the inputs for each block are, and what the outputs are. Use this information to aid in your design of each *always* block. Be sure to include these block diagrams in your report (no handwritten sketches will be accepted).
4. Simulate your design using the testbench files provided until it passes with no mismatches. Be sure **tb_counter_0to9** is highlighted before you run the simulator.
5. Create a truth table for the 0 to 5 counter that duplicates the ideas of the 0 to 9 counter, except that it *rolls over* at 5 rather than 9.
6. Repeat steps 2-4 above for a module: **counter_0to5** (with the max count of 5 before rolling over to 0). Use **tb_counter_0to5** for the simulation testbench. Keep the 4 bit output for now, even though only 3 are needed.

Recommended Implementation Style within the design module:

a) For a sequential *always* block, use non-blocking assigns (<=). For a combinational *always* block, use blocking assigns (=).
b) For the combinational *always* block, place the inputs for that block in the sensitivity list. For the sequential block, only the **posedge** clock signal is required.

c) The only piece needed in the sequential block is the positive edge D flip-flops. This can be done with a single assignment.
d) At the top of the combinational *always* block, define your default conditions for your outputs (i.e., when ena and clr are 0).
e) In the middle of the combinational *always* block, define your outputs when ena is 1. Follow your favorite style from Lab 5.
f) At the bottom of the combinational *always* block, define your outputs when clr is 1. This will be the highest priority, since these statements will be executed last in the *always* block, and will override any previous assignments of your output signals.

Main observations:

- Take some time to carefully look over the timing diagrams (you may need to do this anyway to debug your mismatches). Note how the outputs from the flip flops are bussed. You can expand the bus by clicking on the small arrow next to the signal name. A second click will retract the expansion.
- You can change the radix of the signal by right clicking on the signal name and selecting Radix -> [desired radix]. Then you can view the signal in unsigned decimal, hexadecimal, …
- Select the Unsigned Decimal for the radix on the output of your counters. See how the numbers of your counter increment or decrement following each positive edge of the clock. Be sure note all observations in your report.

Extra observations:

- The default view of the signal list is from the top level (i.e., the testbench wrapper). To see signals at a lower level in the hierarchy, in the *Instance and Process Name* window on the left, click on the UUT, for example. The signals in that module will appear in the adjacent *Object Name* window. You can drag and drop any of these signals to the Timing diagram window. Their timing results are not yet visible, as data from the simulation run were not stored for them. In order to see their data, you need to rerun the simulation: Simulation -> Restart … Simulation -> Run All.
- Look over the testbench files (both verilog and txt). Note that each test vector represents one clock cycle.
- Note that even though the inputs and outputs are contained in one test vector, the inputs are set up before the clock edge, and the outputs are observed (tested) after the clock edge (tb_[design_name].v).