Modulo-*n* Counters

Introduction

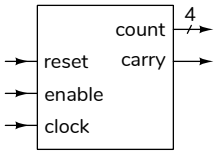
Counters whose values "wrap around" back to zero when they reach the value *n* are called modulo-*n* counters.

The most common example is a modulo-10 counter that counts from 0 up to 9 and then "wraps around" back to 0. We can build a multi-digit binary-coded decimal (BCD) counter using one modulo-10 counter for each digit.

In this lab you will design a single-digit modulo-*n* counter and instantiate four instances of your module to create a four-digit counter whose digits will display on the multiplexed LED display used in previous labs. The value of *n* will depend on your student ID.

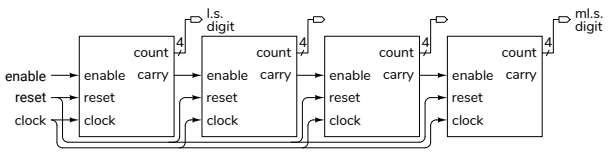
Modulo Counter Specifications

Your counter must have single-bit **`reset`**, **`enable`** and **`clock`** inputs, a 4-bit **`count`** output and a one-bit **`carry`** output:



The counter is synchronous. This means **`count`** changes on the rising edge of the clock. It is set to zero if **`reset`** is asserted, otherwise it is incremented if **`enable`** is asserted, otherwise it does not change. If **`count`** is incremented when its value is *n*-1 it becomes 0. **`carry`** is a combinational logic outputs that is asserted only when **`enable`** is asserted and **`count`** is *n*-1.

The **`carry`** output allows counters to be combined into a multi-digit counter by connecting the **`carry`** output of one digit to the **`enable`** input of the next-most-significant digit:



Note that **`enable`** is not a clock and **`count`** does not necessarily change in each clock cycle.

The modulo value, *n*, to use in your design depends on the last digit of your student ID as shown in the table below. For example, if your ID were A00123456 you would design a modulo-7 counter.

last digit of ID	*n*
0, 1, 2	5
3, 4, 5	6
6, 7	7
8, 9	9

Components

You will use the same four-digit 7-segment LED display, 200 Ω current-limiting resistors and keypad as in previous labs.

Procedure

Design a modulo-*n* counter that meets the specifications above.

Download the **`lab5.qar`** project archive file from the course web site. This project archive includes a complete project except for the Verilog code in the file **`modncount.sv`**.

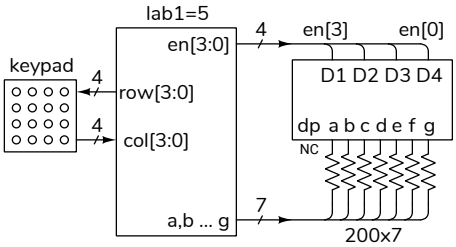
Open the archive file with Quartus and restore it to a convenient folder. Add your modulo-*n* counter code to **`modncount.sv`**. Do not rename the signals in the **`modncount`** module declaration. Change the pin assignments, if necessary, to match your circuit layout. Compile your design, program the FPGA and test it.

The project contains an **`.sdc`** file that defines the clock frequency (50 MHz) and the I/O signals as asynchronous. This allows Quartus to verify that your design operates at a 50 MHz clock frequency.

The test circuit instantiates four of your modulo-*n* counters as shown above and displays the count on the 4-digit LED display. The 1 key increments the count at 5 Hz, the 2 key at 50 Hz and the 3 key at 500 Hz. Pushing the A key resets the counters.

FPGA I/O

The connections to the FPGA are shown in the following diagram:



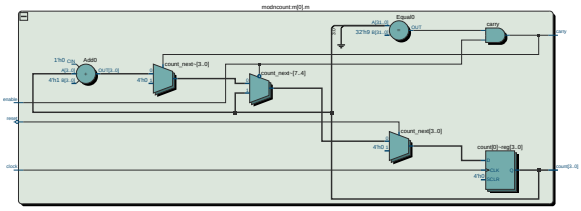
The keypad and LED pin assignments in the supplied project are the same as in the previous lab. The keypad inputs have internal pull-up resistors configured. The pin assignments are shown below:

To	Assignment Name	Value
out a	Location	PIN_49
out b	Location	PIN_65
out c	Location	PIN_54
out d	Location	PIN_50
out e	Location	PIN_46
out en[3]	Location	PIN_51
out en[2]	Location	PIN_69
out en[1]	Location	PIN_67
out en[0]	Location	PIN_60
out f	Location	PIN_71
out g	Location	PIN_58
out row[3]	Location	PIN_103
out row[2]	Location	PIN_100
out row[1]	Location	PIN_98
out row[0]	Location	PIN_86
in col[3]	Location	PIN_84
in col[2]	Location	PIN_80
in col[1]	Location	PIN_76
in col[0]	Location	PIN_74
in clk50	Location	PIN_23
in col	Weak Pull-Up Resistor	On

Submission

To get credit for completing this lab, submit the following to the Assignment folder for this lab on the course website:

1. A PDF document containing:
    - A block diagram of your modulo-*n* counter.
    - A listing of your **`modncount.sv`** file (but not **`lab5.sv`**).
    - A screen capture of your compilation report.
    - The schematic created by **Tools > Netlist Viewers > RTL Viewer**, clicking on the + button on the **`modncount:m[0].m`** block to expand it, and then **File > Export...** It may look like:



      Note that this is the schematic of one of your modulo-*n* counters, not of the complete design.

2. If you do not demonstrate your design in the lab, a video of the display and keypad showing:
    - the count being reset when A is pushed
    - holding down 1 makes the rightmost digit increase, wrap-around at the correct value and increment the next most-significant digit,
    - the other digits also wrapping and carrying when 2 and 3 are pressed,
    - the counts being reset when A is pushed.

A sample video is available on the course website.