Modulo-*n* Counters

Introduction

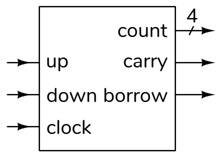
Counters whose values “wrap around” back to zero when they reach the value *n* are called modulo-*n* counters.

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* up/down counter and instantiate four instances of your module to create a four-digit up/down counter whose digits will display on the multiplexed LED display used in the previous lab. The value of *n* will depend on your student ID.

Modulo Counter Specifications

Your counter must have single-bit **clock**, **up** and **down** inputs, a 4-bit **count** output and two one-bit outputs, **carry** and **borrow**:



The counter is synchronous. This means **count** changes on the rising edge of the clock. It is:

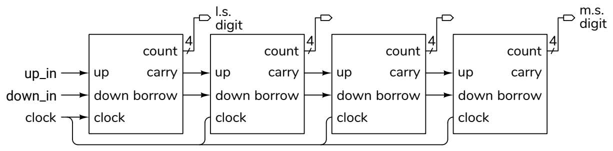
- unchanged if neither **up** nor **down** are asserted.
- incremented if only **up** is asserted
- decremented if only **down** is asserted.
- set to zero if both **up** and **down** are asserted

If **count** is incremented when its value is *n*-1 it becomes 0. If **count** is decremented when its value is 0 it becomes *n*-1.

up and **down** are combinational logic outputs that are asserted as shown below:

count	up	down	carry	borrow
n-1	1	0	1	0
0	0	1	0	1
X	1	1	1	1
others			0	0

The **carry** and **borrow** outputs allow modulo counters to be combined into a multi-digit counter by connecting the **carry** and **borrow** outputs of one digit to the **up** and **down** inputs of the next-most-significant digit:



Note that **up** and **down** are not clocks and **count** does not necessarily change in each clock cycle.

The modulo value, *n*, 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 pushbutton switches 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 CPLD 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 project also includes a **.srf** file to suppress spurious error messages. These files should allow your project to compile without warnings:

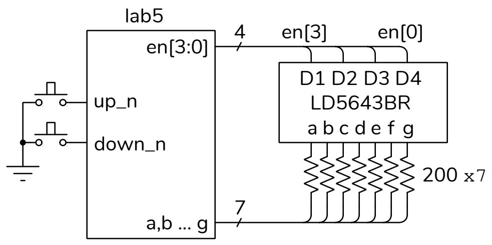
```
293000 Quartus Prime Full Compilation was successful. 0 errors, 0 warnings
```

The archive was created using Quartus version 20.1. Opening it with Quartus II version 13.0sp1 will display errors but you can open the **.qpf** project file after restoring the archive.

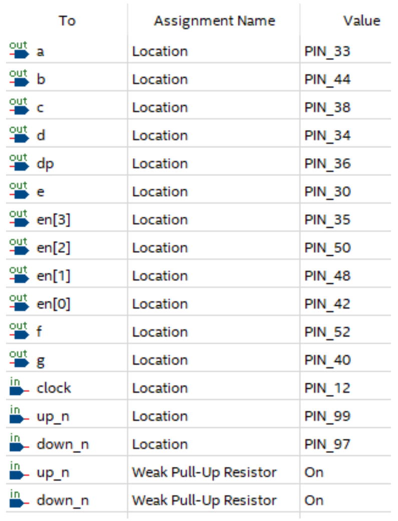
The test circuit instantiates four of your modulo-*n* counters as shown above and displays the count on the 4-digit LED display. The **up_n** pushbutton increments the count at 1 kHz while the **down_n** button decrements it at 100 Hz. Pushing both buttons resets the counters.

CPLD I/O

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



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



To	Assignment Name	Value
a	Location	PIN_33
b	Location	PIN_44
c	Location	PIN_38
d	Location	PIN_34
dp	Location	PIN_36
e	Location	PIN_30
en[3]	Location	PIN_35
en[2]	Location	PIN_50
en[1]	Location	PIN_48
en[0]	Location	PIN_42
f	Location	PIN_52
g	Location	PIN_40
clock	Location	PIN_12
up_n	Location	PIN_99
down_n	Location	PIN_97
up_n	Weak Pull-Up Resistor	On
down_n	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:m0** block to expand it, and then **File > Export...** .
      Note that this is the schematic of one of your modulo-*n* counters, not of the complete design.
2. A video of your breadboard distinctly showing:
    - the count being reset when both buttons are pushed
    - holding down the **up** button causes the count to increase and the most-significant digit wraps around when it reaches *n*-1.
    - the count being reset again when both buttons are pushed
    - holding down the **down** button causes the count to decrease and the most-significant digit wrap around to *n*-1 when it reaches 0.