ELEX 2117 : Digital Techniques 2
2020 Fall Term

BCD Counters

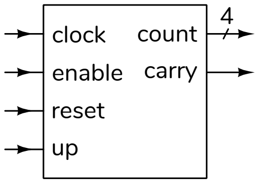
Introduction

Counters that count in base 10 rather than binary are useful when the count must be displayed as decimal digits. Each digit is represented by a 4-bit counter that counts from 0 to 9. This is known as binary-coded decimal (BCD). Multi-digit BCD counters can be built from single-digit BCD counters as described below.

In this lab you will design a single-digit BCD counter. A test module provided by the instructor will use four instances of your module to create a four-digit up/down counter whose digits will be displayed on the multiplexed LED display used in the previous lab.

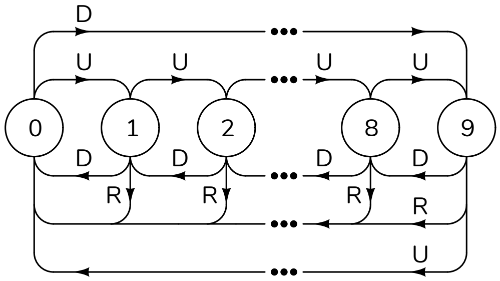
BCD Counter Specifications

Your BCD up/down counter must have **`clock`**, **`enable`** and **`reset`** inputs. An **`up`** input controls the count direction: up when asserted, down otherwise. The counter has a 4-bit BCD **`count`** output and a one-bit **`carry`** output:



The counter is synchronous. This means the count value changes only on the rising edge of the clock. As shown in the following state transition diagram, **`count`** is:

- set to zero when **`reset`** is asserted,
- incremented when **`enable`** and **`up`** are asserted,
- decremented when **`enable`** is asserted and **`up`** is not asserted:



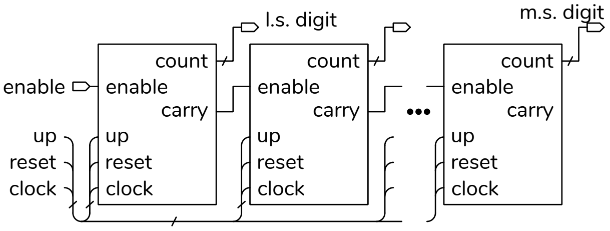
U=enable && up
D=enable && !up
R=reset

all transitions on
rising edge of clock

`carry` is a combinational logic output that is asserted when **`count`** is equal to 9 and **`enable`** and **`up`** are asserted. **`carry`** is also asserted when **`count`** is equal to 0, **`enable`** is asserted and **`up`** is not [1] :

`count`	`enable`	`up`	`carry`
`9`	`1`	`1`	`1`
`0`	`1`	`0`	`1`
`others`			`0`

The **`carry`** output allows BCD 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:



In such a multi-digit counter the **`clock`**, **`reset`** and **`up`** inputs are connected together.

Note that, unlike some other counter designs, **`count`** does not necessarily change in each clock cycle and the enable signal is not a clock.

Components

You will need the same components as in the previous lab plus two pushbutton switches (use the

[1] When counting down **`carry`** means "borrow."

normally-closed contacts as shown in the schematic below).

Procedure

Design a BCD 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 `bcdcount.sv`.

Open the archive file with Quartus and restore it to a convenient folder. Add your BCD counter code to `bcdcount.sv`. Do not rename the signals in the `bcdcount` 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 that clock frequency.

The project also includes a `.srf` file to suppress spurious error messages. These files should allow your project to compile without warnings:

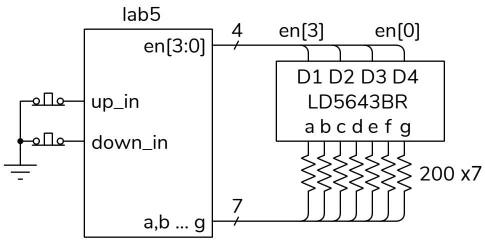
```
293000 Quartus Prime Full Compilation was successful. 0 errors, 0 warnings
```

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.

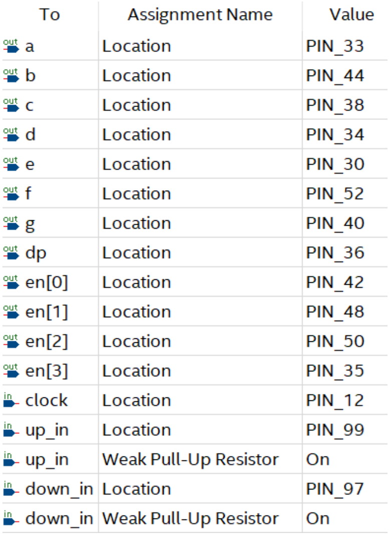
The test circuit instantiates four of your BCD counters as shown above and displays the count on the 4-digit LED display. The `up_in` pushbutton increments the count at 1 kHz while the `down_in` button decrements it at 100 Hz. Pushing both buttons resets the counters.

CPLD I/O

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

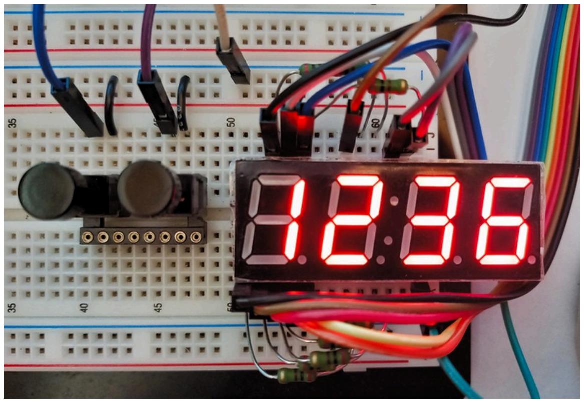


The pin assignments for the LED in the supplied project are the same as in the previous lab. The two pushbutton switch inputs have pull-up resistors configured. The pin assignments are shown below:

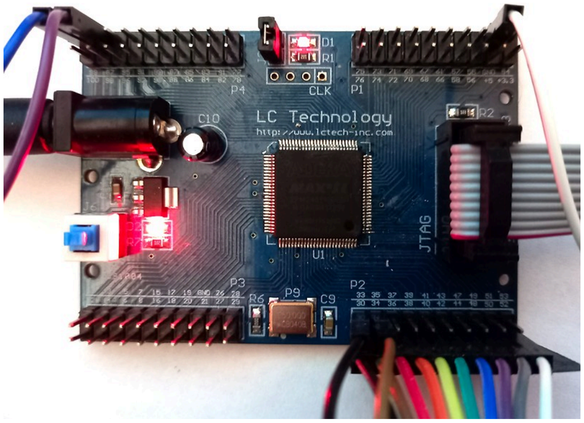


To	Assignment Name	Value
a	Location	PIN_33
b	Location	PIN_44
c	Location	PIN_38
d	Location	PIN_34
e	Location	PIN_30
f	Location	PIN_52
g	Location	PIN_40
dp	Location	PIN_36
en[0]	Location	PIN_42
en[1]	Location	PIN_48
en[2]	Location	PIN_50
en[3]	Location	PIN_35
clock	Location	PIN_12
up_in	Location	PIN_99
up_in	Weak Pull-Up Resistor	On
down_in	Location	PIN_97
down_in	Weak Pull-Up Resistor	On

The wiring to the 7-segment LED is shown below:



and the connections to the CPLD pin headers are:



.

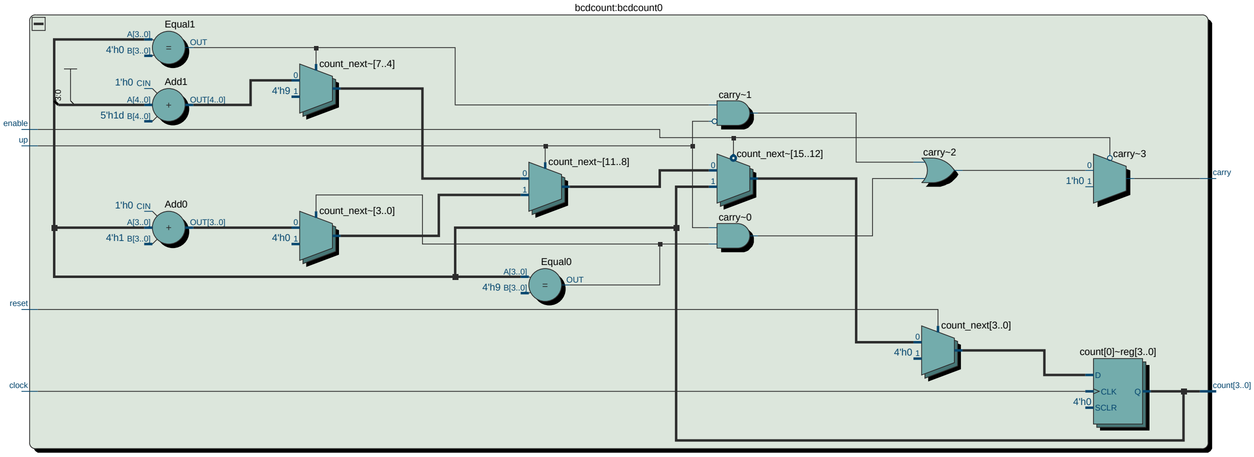


Figure 1: Example RTL Schematic for Lab 5.

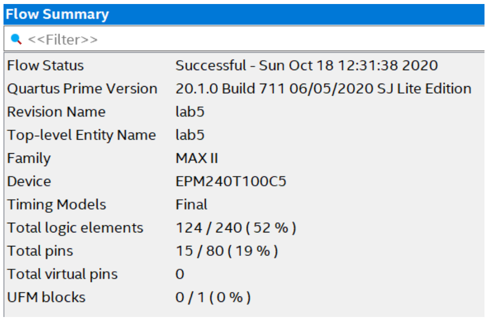
Submission

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:

   - Your name, BCIT ID, course number and lab number.
   - A listing of your `bcdcount.sv` file (not `lab5.sv`). You must follow the coding guidelines given on the "Course Information" section of the course website. Note that these may have changed.

     The listing should be included as text rather than images.
   - a screen capture of your compilation report (**Window** > **Compilation Report**) similar to this:



Flow Summary

<<Filter>>

Flow Status	Successful - Sun Oct 18 12:31:38 2020
Quartus Prime Version	20.1.0 Build 711 06/05/2020 SJ Lite Edition
Revision Name	lab5
Top-level Entity Name	lab5
Family	MAX II
Device	EPM240T100C5
Timing Models	Final
Total logic elements	124 / 240 (52 %)
Total pins	15 / 80 (19 %)
Total virtual pins	0
UFM blocks	0 / 1 (0 %)

2. The PDF file containing the schematic created by **Tools > Netlist Viewers > RTL Viewer**, clicking on the + button on the `bcdcount0` block to expand it, and then **File > Export...** . The file should look similar to Figure 1.

   Note that this is the schematic of one of your BCD counters, not of the complete design.

3. a video of your breadboard distinctly showing:

   - the count being reset when both buttons are pushed
   - the count reaching more than 1000 when the `up_in` button is pushed (about 1 second)
   - the count being reset again when both buttons are pushed
   - the count reaching less than 9900 when the `down_in` button is pushed (about 1 second)