1 Construct a 4-bit synchronous binary up counter ($b_3b_2b_1b_0$) with the 1-Hz clock frequency from lab2 and use 4 LEDs for display.

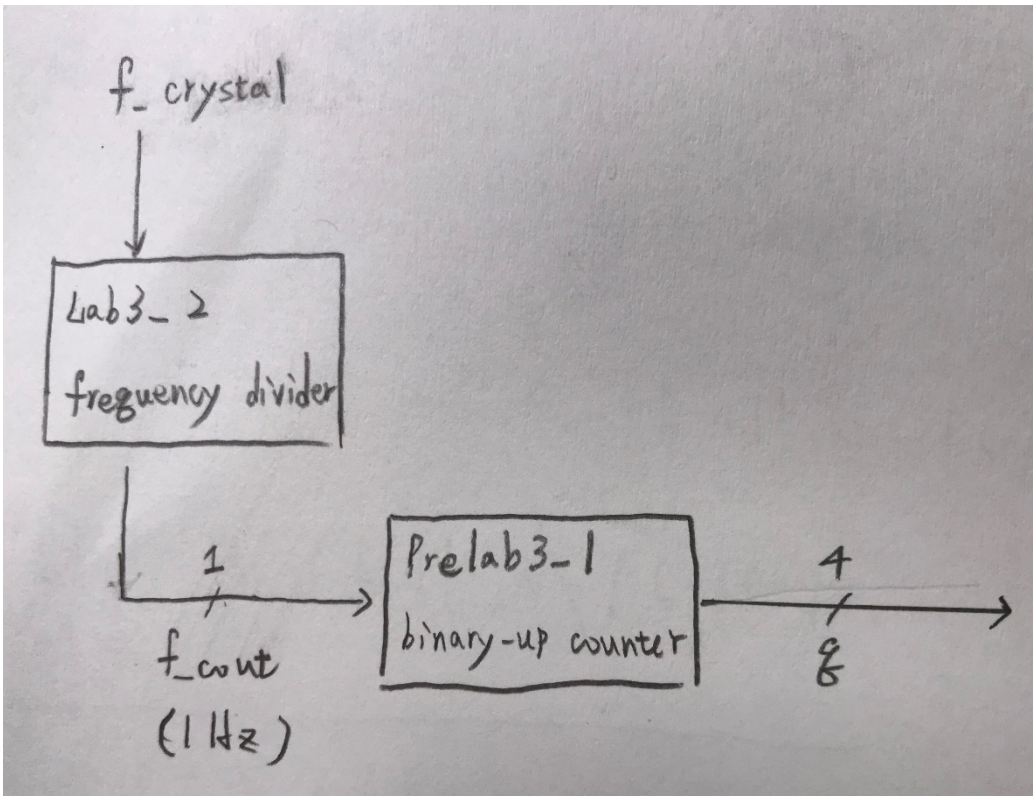
I/O	$f_{crystal}$	b_3	b_2	b_1	b_0
Site	W5	V19	U19	E19	U16

Design Specification:

Input: f_crystal, rst_n

Output: q[3:0]

Block diagram:



Design Implementation:

I/O pin assignment:

f_crystal—W5

rst_n—R2

q[3]—V19

q[2]—U19

q[1]—E19

q[0]—U16

Detail of frequency divider and binary-up counter has been illustrated at report of lab3_2 and prelab3_1.

I used a top module to include module from lab3_2 and prelab3_1 and used wire to connect two modules, such as f_out in the block diagram.

Discussion:

In this problem, we used the 1Hz frequency derived from lab3_2 to display the 4-bit binary-up-counter result from 0000 to 1111 on LED. We need 1Hz clock to control the 4-bit binary counter. As our eyes are only able to observe the different patterns of LED when they're changing in 1Hz frequency. If the frequency is too high, what we could see is only 4 always-on LEDs.

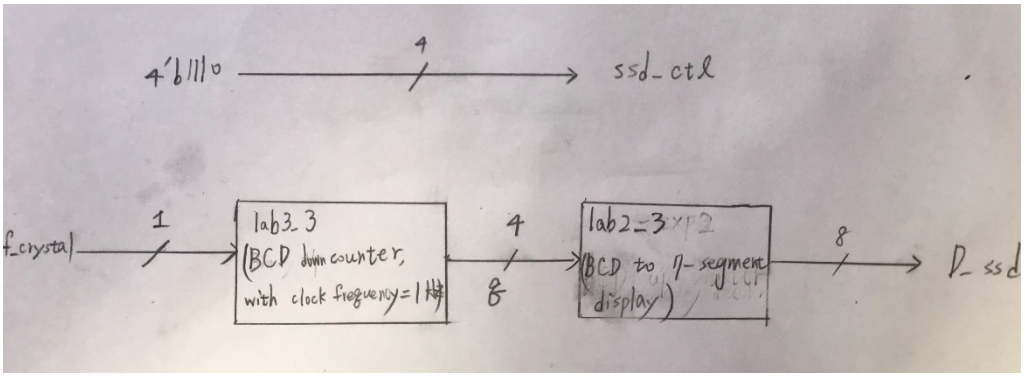
2. Combine the 4-bit synchronous binary up counter from exp1 with a binary-to-seven-segment-display decoder (from lab2-exp3) to display the binary counting in 7-segment display.

Design Specification:

Input: f_crystal, rst_n

Output: ssd_ctl[3:0], D_ssd[7:0]

Block diagram:



Design Implementation:

I/O pin assignment:

f_crystal—W5

rst_n—R2

ssd_ctl[3]—W4

ssd_ctl[2]—V4

ssd_ctl[1]—U4

ssd_ctl[0]—U2

D_ssd[7]—W7

D_ssd[6] — W6

D_ssd[5] — U8

D_ssd[4] — V8

D_ssd[3] — U5

D_ssd[2] — V5

D_ssd[1] — U7

D_ssd[0] — V7

From the module Lab3-exp3 I can get 1Hz and use it as clock for 4-bit binary up counter. And I connect this 4-bit signal to binary-to-seven-segment-display so that I can observe the output through 7-segment display. I need to use 1Hz frequency as clock frequency so that our human eyes are able to see the different patterns on 7-segment display.

And ssd_ctl is always equal to 4’b1110 because I only need one of 7-segment display to show the result and others are off.

Discussion:

I decompose this problem into two parts, which is binary-up-counter and 7-segment display respectively. I have done both of the modules in previous labs. Hence, I just used some wires to connect them to form a big module in this lab.

3. Construct a single digit BCD up counter with the divided clock as the clock frequency and display on the seven-segment display.

3.1 Construct a BCD up counter.

3.2 Construct a BCD-to-seven-segment display decoder (from lab2-exp2).

3.3 Combine the above two together.

Design Specification:

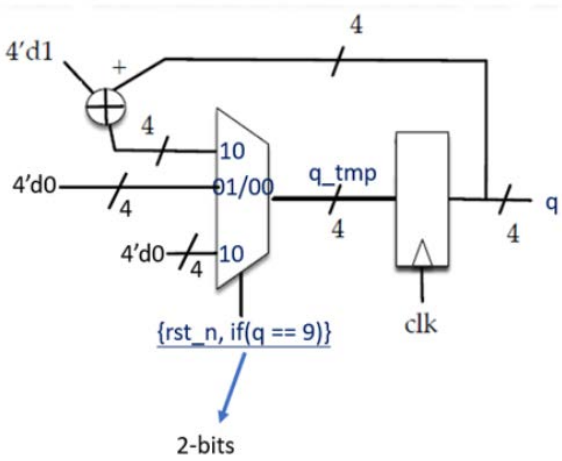
Input: f_crystal, rst_n

Output: [3:0]ssd_ctl

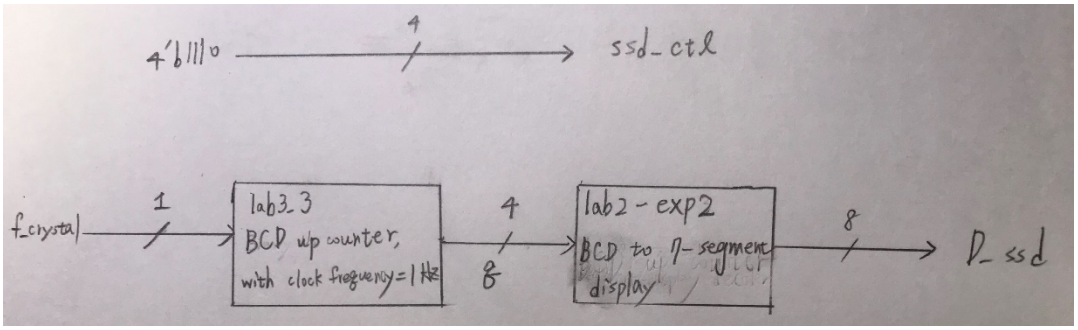
Output: [7:0]D_ssd

Block diagram:

BCD up counter:



Whole Architecture:



Design Implementation:

I/O pin assignment:

f_crystal—W5

rst_n—R2

ssd_ctl[3]—W4

ssd_ctl[2]—V4

ssd_ctl[1]—U4

ssd_ctl[0]—U2

D_ssd[7]—W7

D_ssd[6] — W6

D_ssd[5] — U8

D_ssd[4] — V8

D_ssd[3] — U5

D_ssd[2] — V5

D_ssd[1] — U7

D_ssd[0] — V7

In this part, I need to use a BCD up counter. Therefore, I modify the lab3_3 a little bit. I use the value of reset of current counter value to control the next counter value. If rst_n = 0 (don't care the current counter value), q_tmp (in the block diagram of BCD up counter) is assigned to 0 for next clock cycle. If rst_n = 1, and current value is equal to 9, q_tmp (in the block diagram of BCD up counter) is assigned to 0 for next clock cycle as well (since BCD value is not larger than 9). When it's other condition q_tmp (in the block diagram of BCD up counter) is assigned to current counter value (q in the block diagram of BCD up counter) **plus** 1. And then use BCD to 7-segment display decoder in lab2 to show the pattern on the 7-segment. And ssd_ctl is always equal to 4'b1110 as I only show the result on one of the 7-segment display

Discussion:

This problem is similar to the previous one. The previous one is binary up counter and this one is BCD up counter. So, I just modify module from the last problem a little bit and connect the 7-segment display. Then, we are able to observe the different patterns of 7-segment display.

4 Construct a single digit BCD down counter with the divided clock as the clock frequency and display on the seven-segment display.

4.1 Construct a BCD up counter.

4.2 Construct a BCD-to-seven-segment display decoder (from lab2-exp2).

4.3 Combine the above two together

Design Specification:

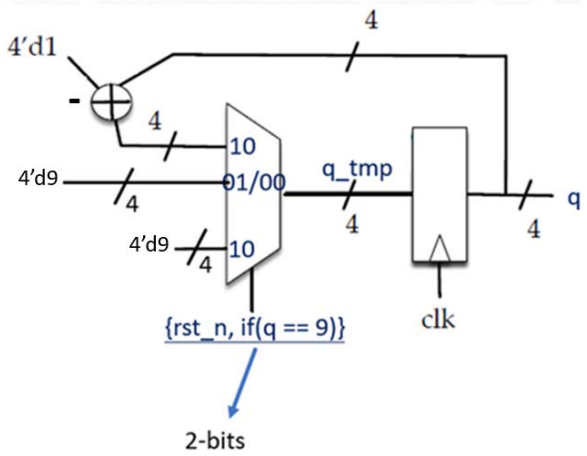
Input: f_crystal, rst_n

Output: [3:0]ssd_ctl

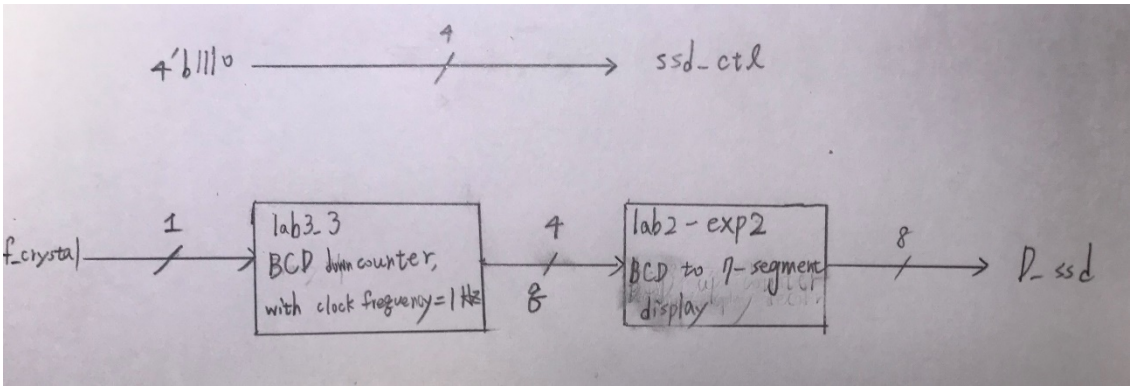
Output: [7:0]D_ssd

Block diagram:

BCD down counter:



Whole architecture:



Design Implementation:

I/O pin assignment:

f_crystal—W5

rst_n—R2

ssd_ctl[3]—W4

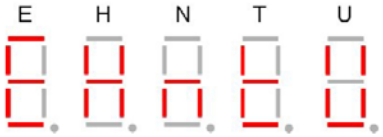
ssd_ctl[2]—V4
ssd_ctl[1]—U4
ssd_ctl[0]—U2
D_ssd[7]—W7
D_ssd[6] — W6
D_ssd[5] — U8
D_ssd[4] — V8
D_ssd[3] — U5
D_ssd[2] — V5
D_ssd[1] — U7
D_ssd[0] — V7

In this part, it's very similar to BCD up counter. If rst_n = 0 (don't care the current counter value), q_tmp (in the block diagram of BCD up counter) is assigned to 9 for next clock cycle. If rst_n = 1, and current value is equal to 0, q_tmp (in the block diagram of BCD up counter) is assigned to 9 for next clock cycle as well (since BCD value is not less than 0). When it's other condition q_tmp (in the block diagram of BCD up counter) is assigned to current counter value (q in the block diagram of BCD up counter) **minus** 1. And then use BCD to 7-segment display decoder in lab2 to show the pattern on the 7-segment. And ssd_ctl is always equal to 4'b1110 as I only show the result on one of the 7-segment display

Discussion:

This problem is similar to the previous one. The previous one is BCD up counter and this one is BCD down counter. So, I just modify module from the last problem a little bit and connect the 7-segment display. Then, we are able to observe the different patterns of 7-segment display.

5 Use the idea from pre-lab2. We can do something on the seven-segment display. Assume we have the pattern of E, H, N, T, U for seven-segment display as shown below. Try to implement the scrolling pre-stored pattern NTHUEE with the four seven-segment displays.

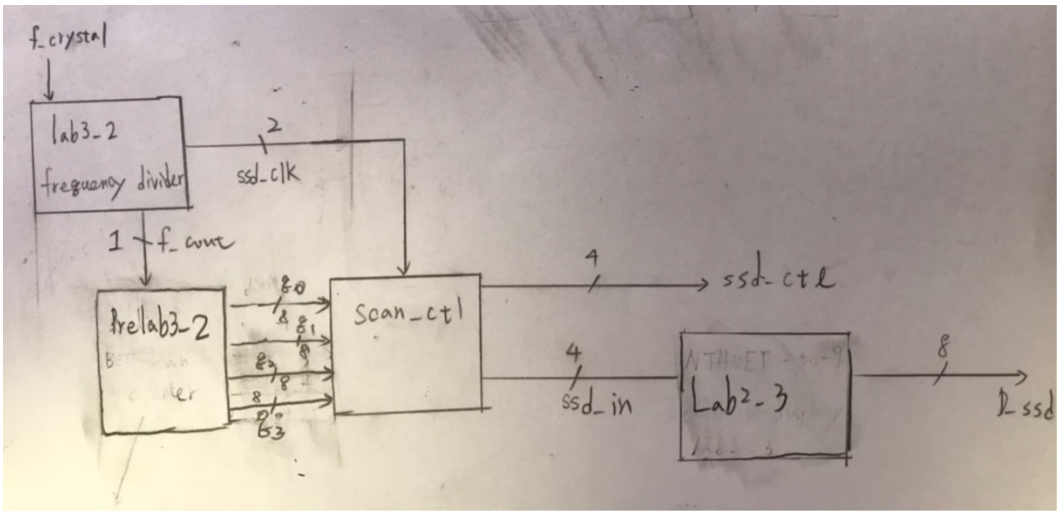


Design Specification:

Input: f_crystal, rst_n

Output: [3:0]ssd_ctl, [7:0]D_ssd

Block diagram:



Design Implementation:

I/O pin assignment:

f_crystal—W5

rst_n—R2

ssd_ctl[3]—W4

ssd_ctl[2]—V4

ssd_ctl[1]—U4

ssd_ctl[0]—U2

D_ssd[7]—W7

D_ssd[6] — W6

D_ssd[5] — U8

D_ssd[4] — V8

D_ssd[3] — U5

D_ssd[2] — V5

D_ssd[1] — U7

D_ssd[0] — V7

First all of, I used 27-bit counter get 1Hz frequency to serve as clock for shift register and 16^{th} and 17^{th} bit of counter value to serve as input for scan control, which is faster than 1Hz. Scan control uses 2-to-4 decoder to decide which 7-segment is on and other 3 ones are off (i.e. decide ssd_ctl). Scan control also takes 4 outputs from shift register, which represents characters (NTHUEE). Then, scan control's 2-to-4 decoder also decides which character is shown on 7-segment display at this point of time. Through the fast switching of 2 bits from frequency divider, our eyes would see the different characters on each 7-segment display.

Discussion:

This is quite a complicated problem at first. But I found it can be solved through dividing the problem into some smaller modules. I have done some of these smaller modules on previous lab or experiment and need to modify some of them slightly. For example, how to show characters (NTHUEE) on 7-segment display. I check what the look of each character on 7-segment display. And I assign the 1/0 to each segment of 7-segment display when I need to show this character, ex: N-> 8'b1101_0101. And I think the key module is scan_ctl, which helps me show different characters on different 7-segment display.

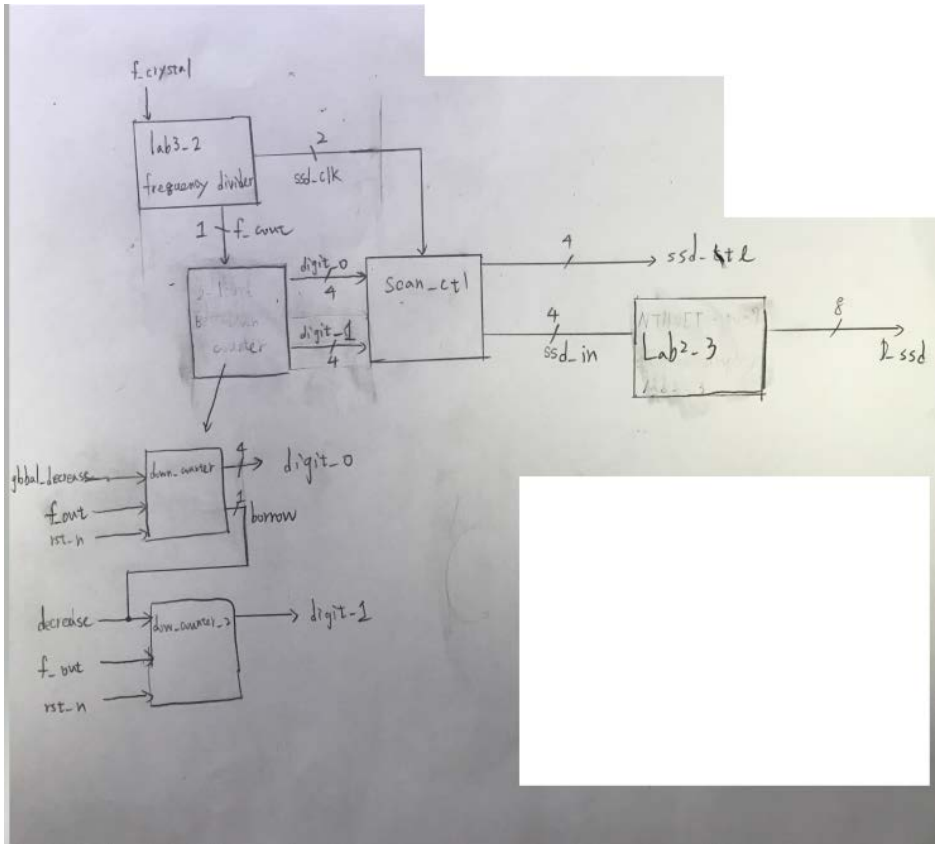
5 (Bonus) Construct a 30-second count down timer (stop at 00).

Design Specification:

Input: f_crystal, rst_n

Output: [3:0]ssd_ctl, [7:0]D_ssd

Block diagram:



Design Specification:

I/O pin assignment:

f_crystal—W5

rst_n—R2

ssd_ctl[3]—W4

ssd_ctl[2]—V4

ssd_ctl[1]—U4

ssd_ctl[0]—U2

D_ssd[7]—W7

D_ssd[6] — W6

D_ssd[5] — U8

D_ssd[4] — V8

D_ssd[3] — U5

D_ssd[2] — V5

D_ssd[1] — U7

D_ssd[0] — V7

This experiment is pretty much the same the last experiment. This experiment is to show the 30 seconds countdown timer on 7-segment display. I also use frequency divider to get 1Hz frequency for BCD down counter and much higher frequency for scan control. And 2-to-4 decoder decider which 7-segment display is on and other 3 ones are off. In this experiment, only two possible cases as there are only two digits to show. Also, since it's two digit, borrow from higher digit should be considered. When LSB goes to 0, it need to borrow from MSB (value of MSB is subtracted by 1) and keep going with 9, 8, 7…Then, we could see the result on 7-segment display.

Discussion:

The key of this experiment is also divide the problem into many modules and make sure each module works well. And I create a top module to combine and connect them well. Also, as the same as the last experiment, scan_ctl is a important module, which allows us to see the different numbers on 7-segment display.

Conclusion for Lab4:

Via this lab, I know how to design BCD up/down counter and how to use scan_ctl to show the different patterns on different 7-segment display. Most importantly, I have learned that it's important to decompose the problem into different small problems so that it would be easier to solve.

Reference for Lab4: handout given by professor, from which I learned how to design BCD up/down counter and how to use scan_ctl to show the different patterns on different 7-segment display