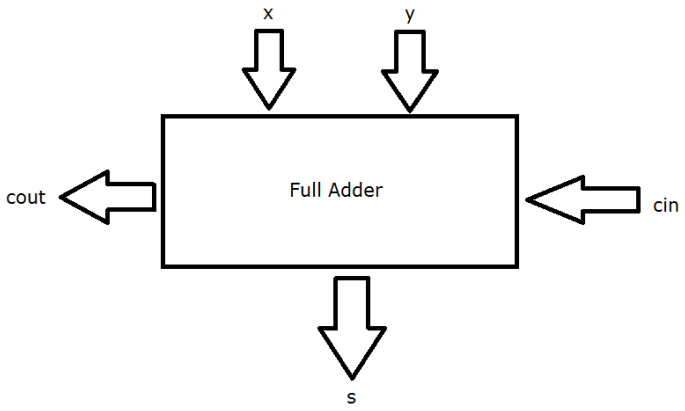
Lab02 FPGA Emulation
106061218 Cheng-En Lee

1. Emulate the full adder.

Design Specification

The idea of this experiment is 100% as same as that in lab 01. So I'll skip the details of logic diagram.

(1) Input: x, y, cin.
(2) Output: cout, s.
(3) Block diagram:



Design Implementation

(1) Logic function:

```
s = x^y^cin;
cout = (x&y) | (cin&(x^y));
```

(2) pins

I/O	*x*	*y*	*cin*	*s*	*cout*
LOC	V17	V16	W16	U16	E19

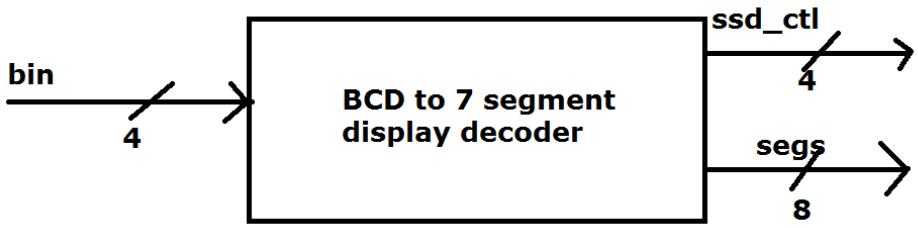
2. Seven-segment display decoder

Design Specification

(1) input:
bin [3:0] (The binary number input.)
(2) output:
segs [7:0] (Each segments of a single digit of 7-segment display.)
ssd_ctl [3:0] (Four digits of 7-segment display.)
d[3:0] (LEDs)

Since bin [3:0] is same as d[3:0], I'll skip them in the following diagrams.

(3) block diagram

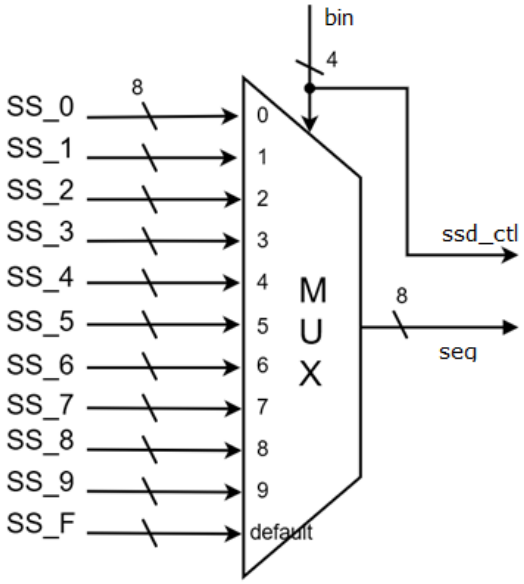


Design Implementation

We can regard this decoder as a MUX with default. The BCD - 7 segment display comparison table is showed as follow:

Decimal (SS_X)	BCD	7-SD
0	0000	00000011
1	0001	10001111
2	0010	00100101
3	0011	00001101
4	0100	10011001
5	0101	01001001
6	0110	01000001
7	0111	00011111
8	1000	00000001
9	1001	00001001
(default)	1010	00000001

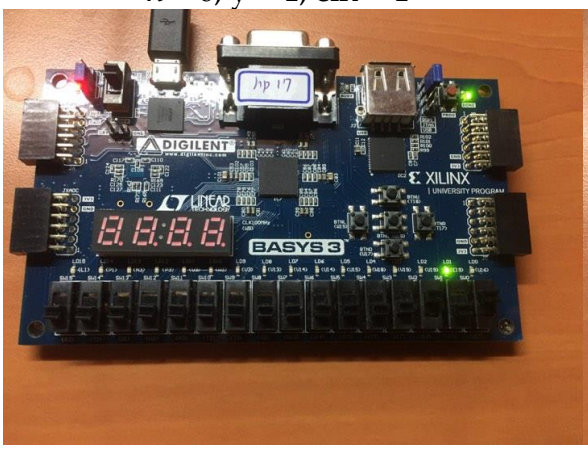
In addition, the logic diagram shows more detail:


bin
4
8
SS_0
SS_1
SS_2
SS_3
SS_4
SS_5
SS_6
SS_7
SS_8
SS_9
SS_F
0
1
2
3
4
5
6
7
8
9
default
M
U
X
ssd_ctl
8
seg

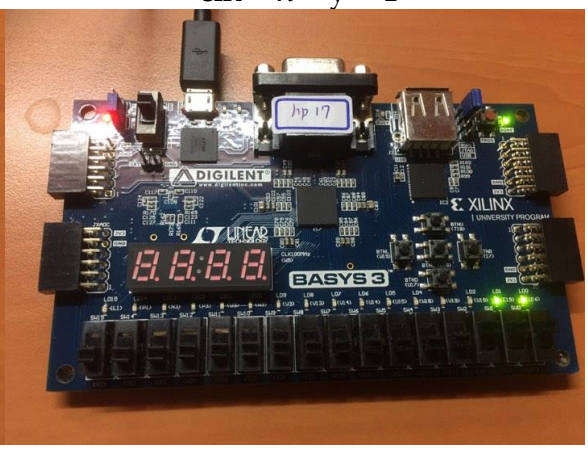
Discussion

(1) Adder

x = 0, y = 1, cin = 1

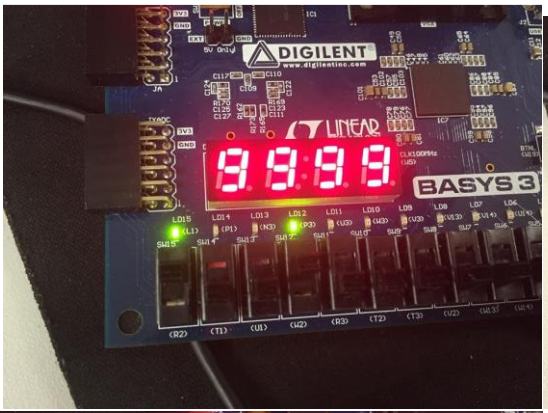


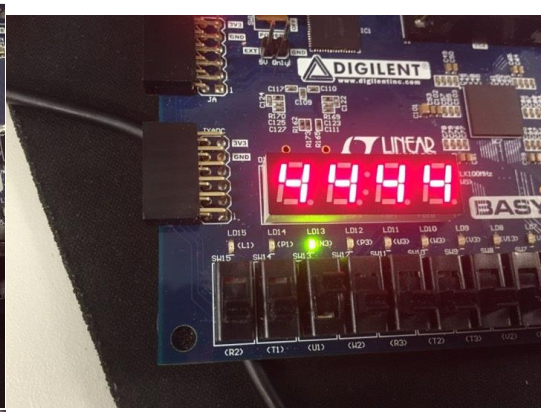
cin = x = y = 1

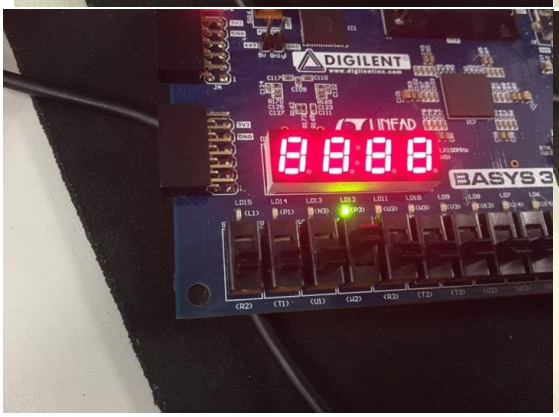


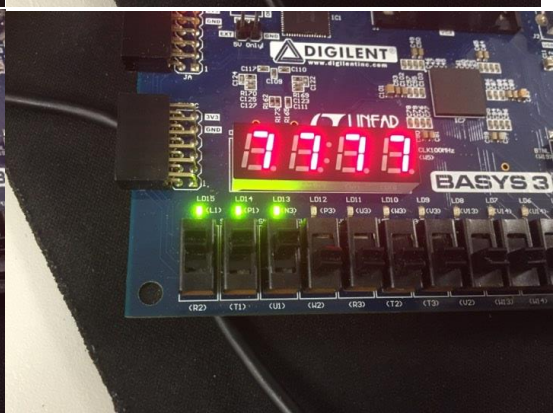
(2) 7-segment display

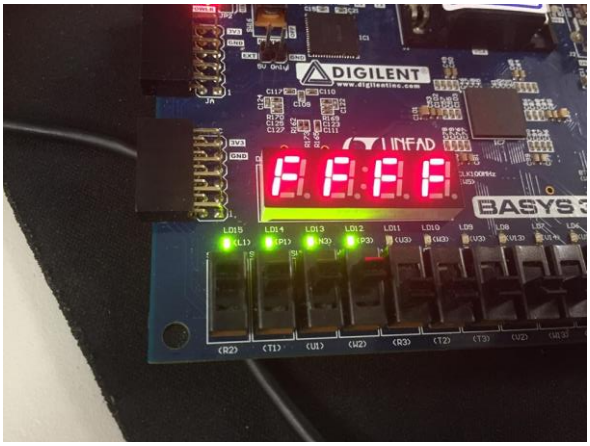
9487F











Conclusion

Be careful when you're connecting the pins. It is easy-neglected but critical and could cause bugs.

The most important thing is that this is the first lab which has two modules. The lecturer told us that it is better to have only one module in one file. Otherwise, it will be time-wasting finding the modules you've typed. Also, it's more easier to copy the codes or .v files when you have another project which is needed the same logic functions.

References

Hang-outs in class.