Lab 01
done by 107061218 謝霖泳

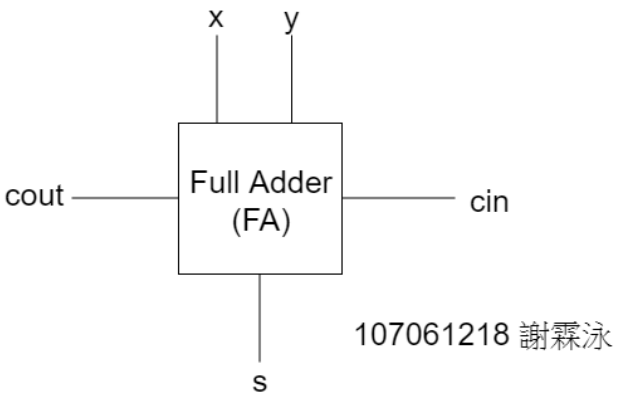
1.

➢Design Specification

For a full adder (FA):

Input: x, y, cin

Output: s, cout



➢Design Implementation

First, we can derive the truth table.

x	y	cin	S	cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Then, by using K-map, we can derive the logic equations.

S:

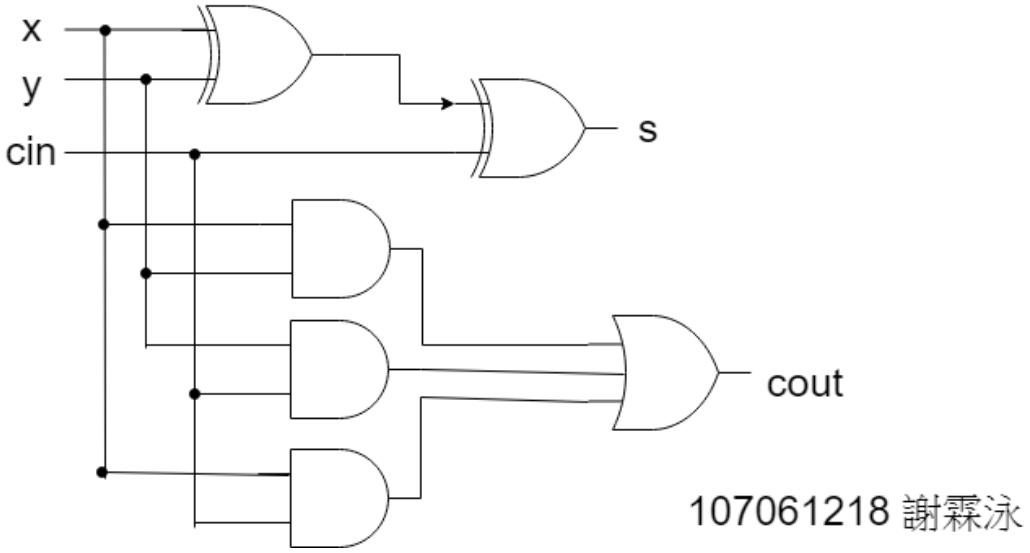
cin \ xy	00	01	11	10
0	0	1	0	1
1	1	0	1	0

S = (x ⊕ y) ⊕ cin = x ⊕ y ⊕ cin

cout:

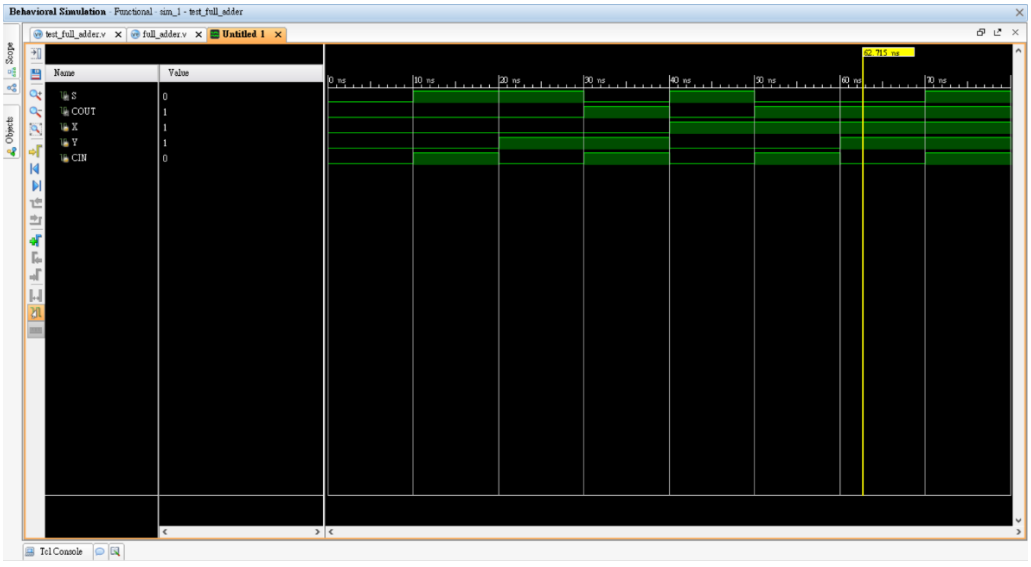
cin \ xy	00	01	11	10
0	0	0	1	0
1	0	1	1	1

cout = x&y + y&cin + x&cin



➢Discussion

A full adder is a basic design. When we want to conduct some arithmetic manipulations between two binary numbers such as addition or subtraction, full adders play an important role during the process. First, we derive the Boolean functions of s and cout, which represents the sum and the carry out to the next digit respectively, by K-maps. Then, we can draw the logic diagram as shown above. Finally, we use Verilog code to simulate our design and check the waveform. Here displays the waveform of the design.



As seen in the waveform, we can see that the design runs perfectly correct.

The yellow line, for example, shows that when x=y=1 and cin=0, cout should equal to 1 and s should equal to 0, which corresponds with our acknowledgement.

➢Conclusion

After this experiment, I've learned to design an FA and simulate the result.

2.

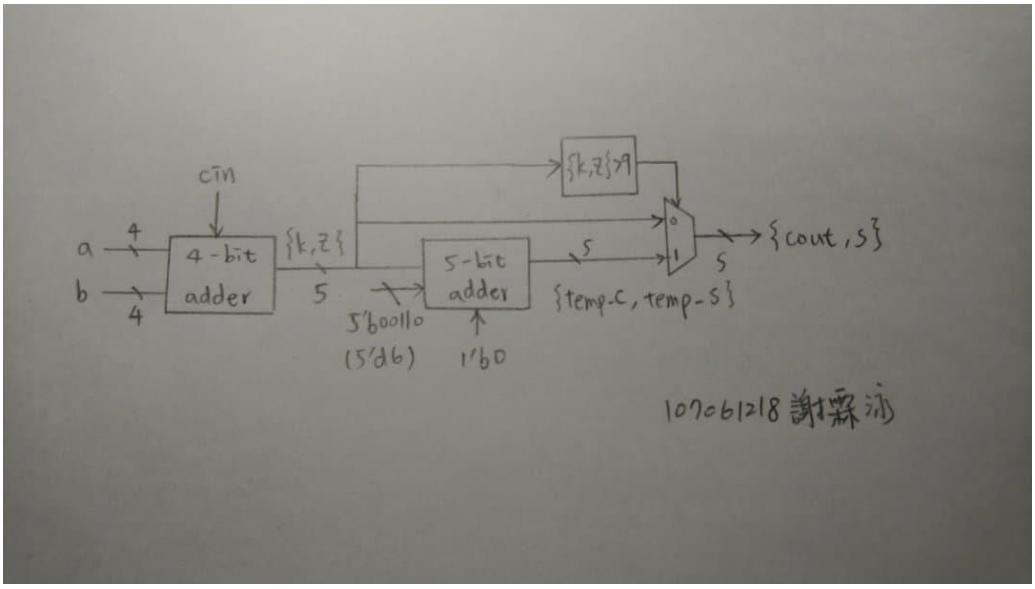
➢Design Specification

For a single digit decimal adder:

Input: a[3:0], b[3:0], cin

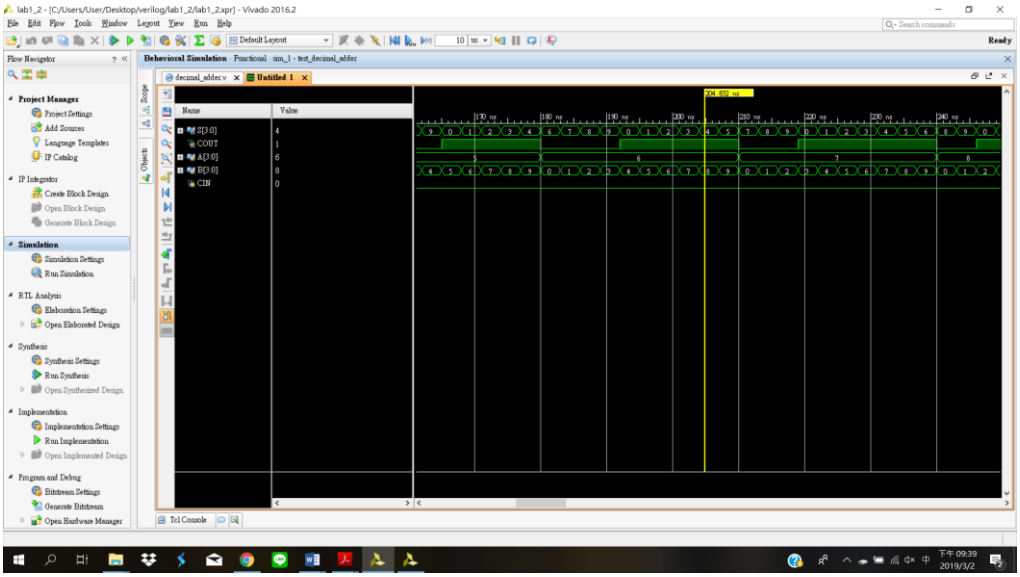
Output: s[3:0], cout

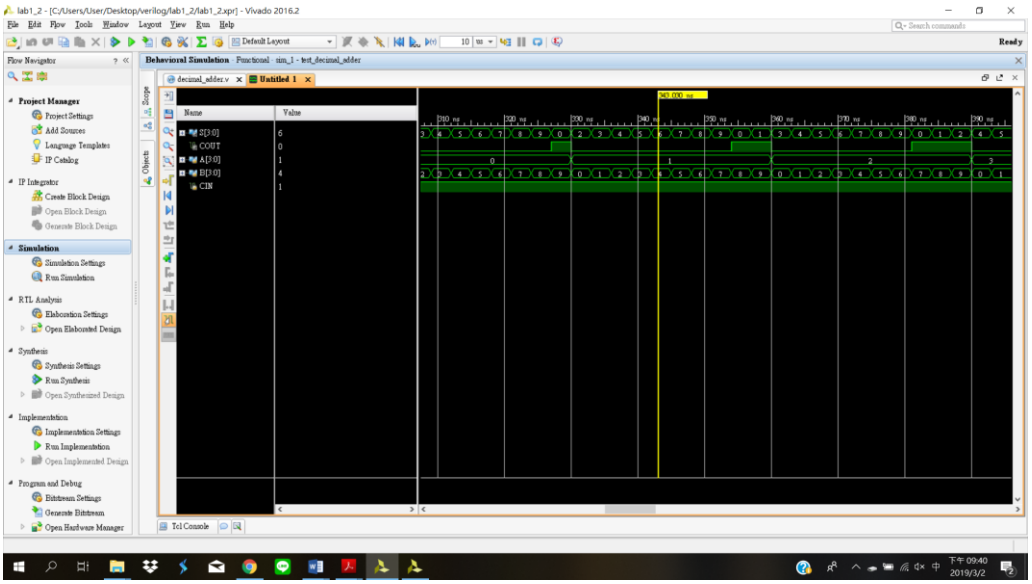
➢Design Implementation



➢Discussion

The two decimal numbers are written in BCD form. When adding them together, we should check if the result exceeds 9. If so, the carry out digit should be 1, and the remaining digit should plus 6 as the final result.





➢Conclusion

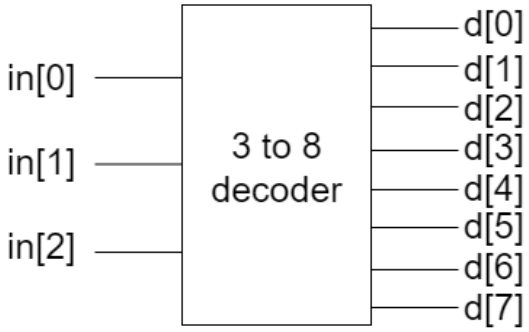
3. Design a 3-to-8-line decoder with enable (input ***in***[2:0], enable ***en*** and output ***d***[7:0]).

➢Design Specification

For a 3 to 8 decoder with enable:

Input: in[2:0], en

Output: d[7:0]



107061218 謝霖泳

➢Design Implementation

The truth table is shown here.

en	in[0]	in[1]	in[2]	d[0]	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]	d[7]
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
1	0	1	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	1

Then, we can derive the Boolean functions of d[0] to d[7].

d[0] = in[0]’in[1]’in[2]’en

d[0] = in[0]’in[1]’in[2]en

d[0] = in[0]’in[1]in[2]’en

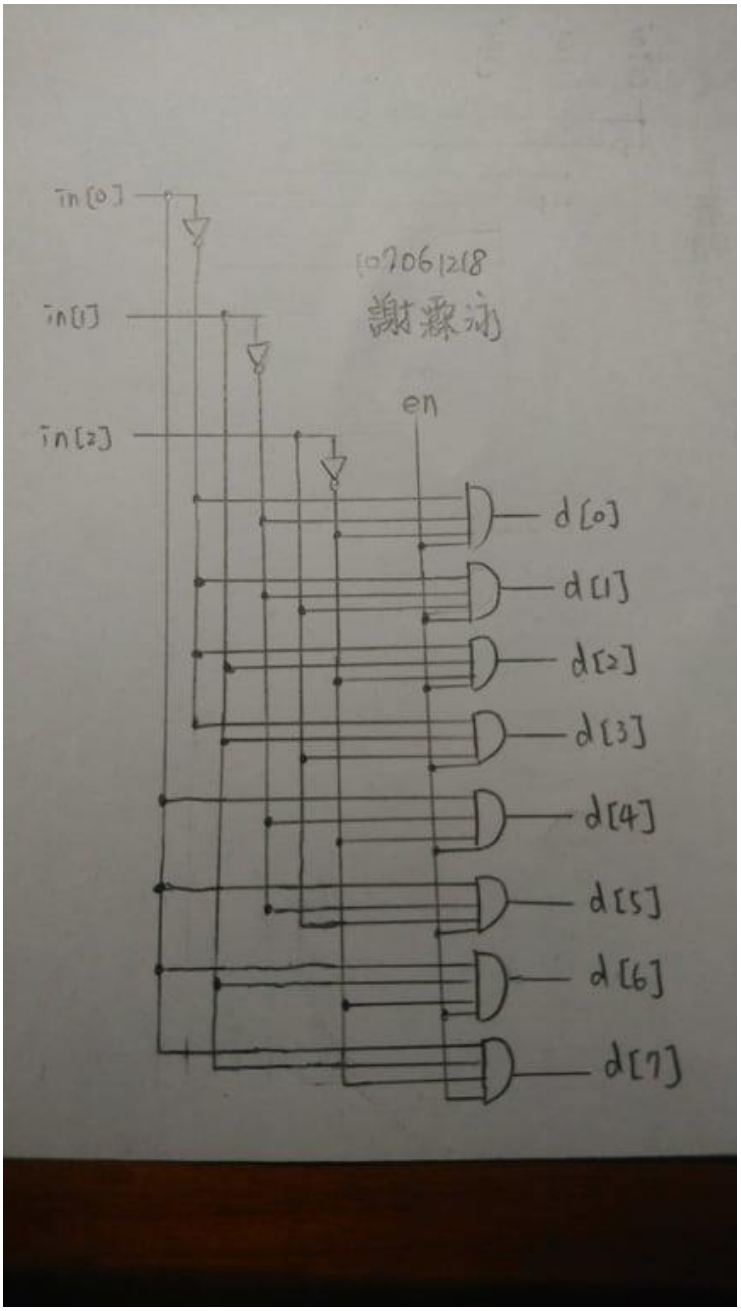
d[0] = in[0]’in[1]in[2]en

d[0] = in[0]in[1]’in[2]’en

d[0] = in[0]in[1]’in[2]en

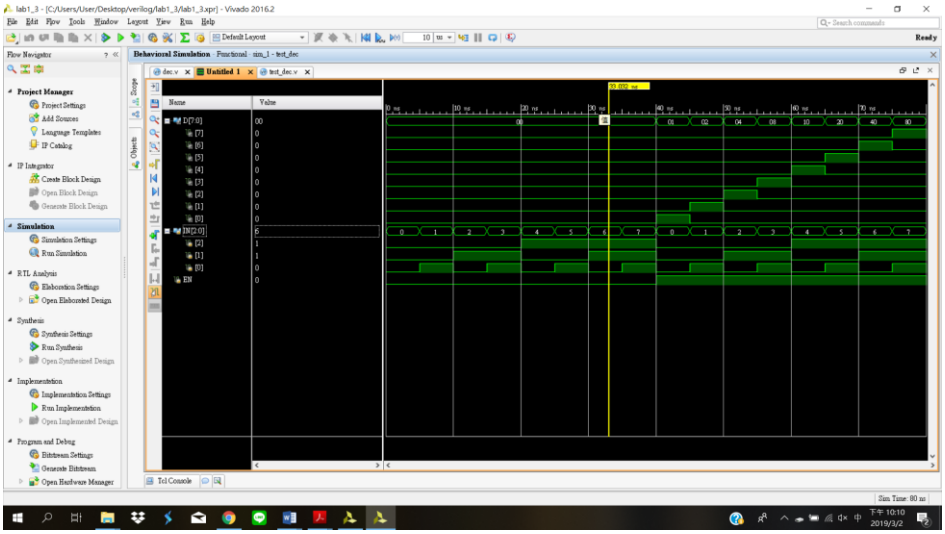
d[0] = in[0]in[1]in[2]’en

d[0] = in[0]in[1]in[2]en



➢Discussion

From the design of the 3-8 decoder, we can see that under the condition of en=1, the decoder can generate all of the minterms. Once in[2:0] is given, only one of d[0:7] will be one, which is so-called “one-hot code”.



From the left hand side waveform, it tells us that no matter how in[0:2] varies, the output always stays 0, since en, which means enable, is 0. On the other hand, when en=1, the output d performs the action of one-hot code. That is, only one d equals to 1 when a specified input is given.

➢Conclusion

After this experiment, I've learned how to design a decoder and how to generate one-hot code.