106010006 電機21 黃詩瑜

Lab 1: Verilog HDL

1. Design and implement a full adder. (*s*+*cout*=*x*+*y*+*cin*)

Specification:

input : x, y, cin

output : s, cout

1.1 Write the logic equation.

觀察truth table找出logic equation

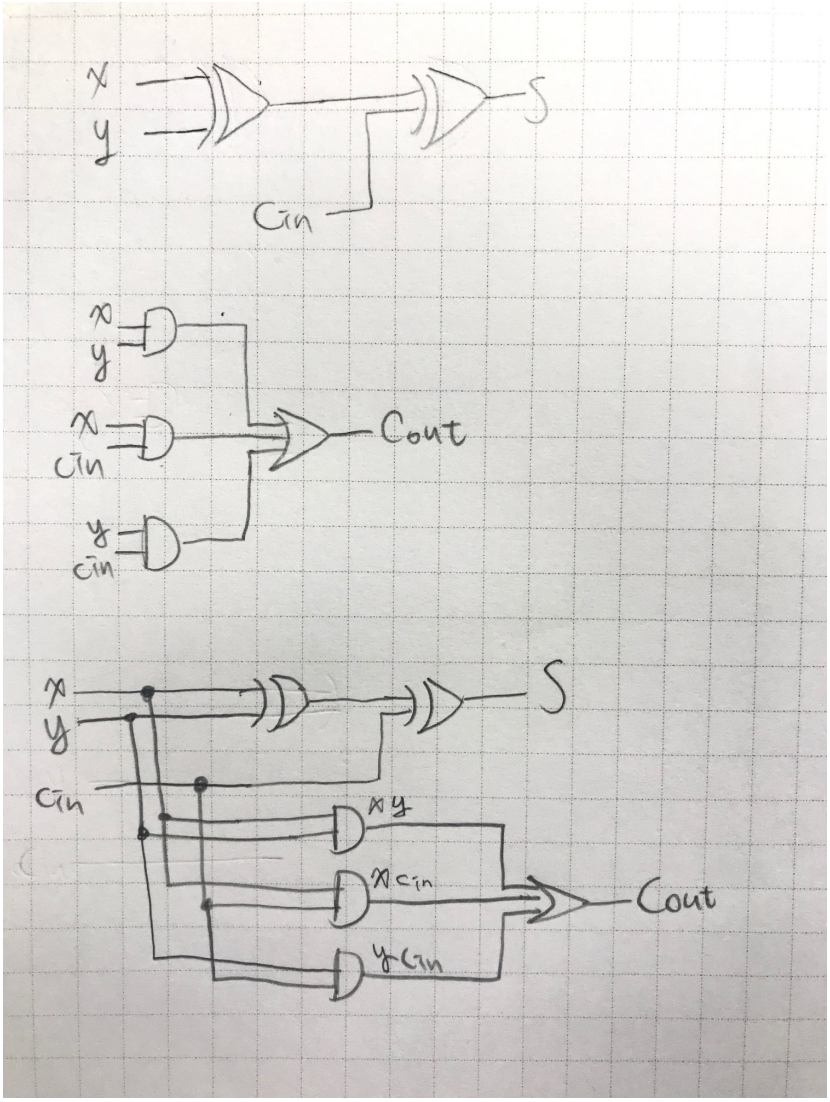
x	y	cin	cout	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Truth table

cout = x y + x cin+ y cin

$$S = x \oplus y \oplus cin$$

1.2 Draw the related logic diagram.



1.3 Verilog RTL representation with verification.

Verilog Module:

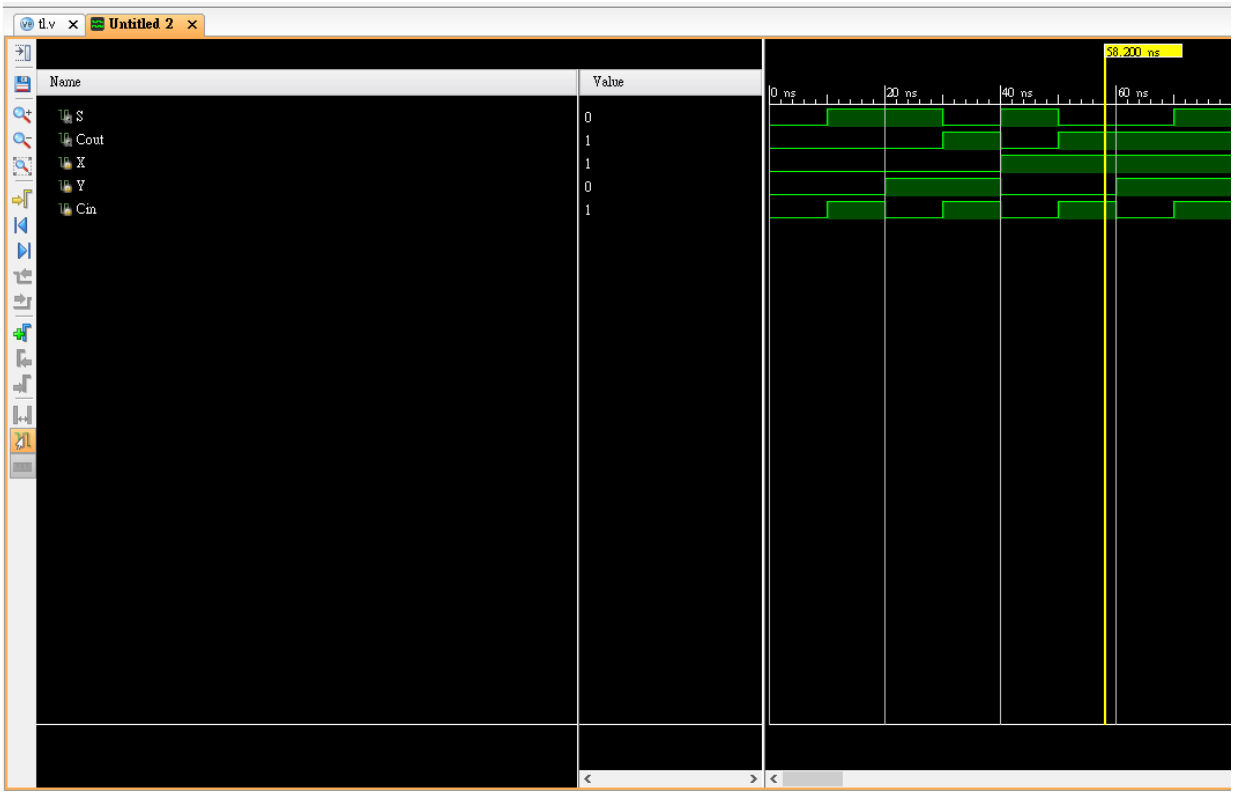
```
module lab1(
```

```
input x, y, cin,
output s, cout
);
assign s=x^y^cin;
assign cout= x&y | x&cin | y&cin;
endmodule
```

Testbench:

```
module test_lab1;

wire S,Cout;

reg X,Y,Cin;

lab1 U0(.cout(Cout),.s(S),.x(X),.y(Y),.cin(Cin));

initial

begin

  X=0;Y=0;Cin=0;

  #10 X=0;Y=0;Cin=1;

  #10 X=0;Y=1;Cin=0;

  #10 X=0;Y=1;Cin=1;

  #10 X=1;Y=0;Cin=0;

  #10 X=1;Y=0;Cin=1;

  #10 X=1;Y=1;Cin=0;

  #10 X=1;Y=1;Cin=1;

end
```

endmodule



討論：Full adder是邏輯設計的基礎，只要找出equation，verilog就能寫出來了

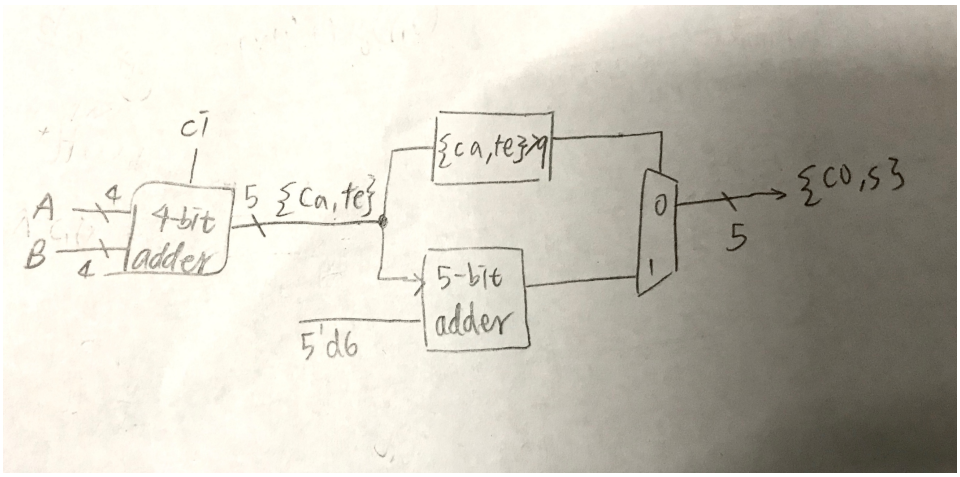
2. Design a single digit decimal adder with input A(a3a2a1a0), B(b3b2b1b0), Cin(ci), and output S(s3s2s1s0) and Cout(co).

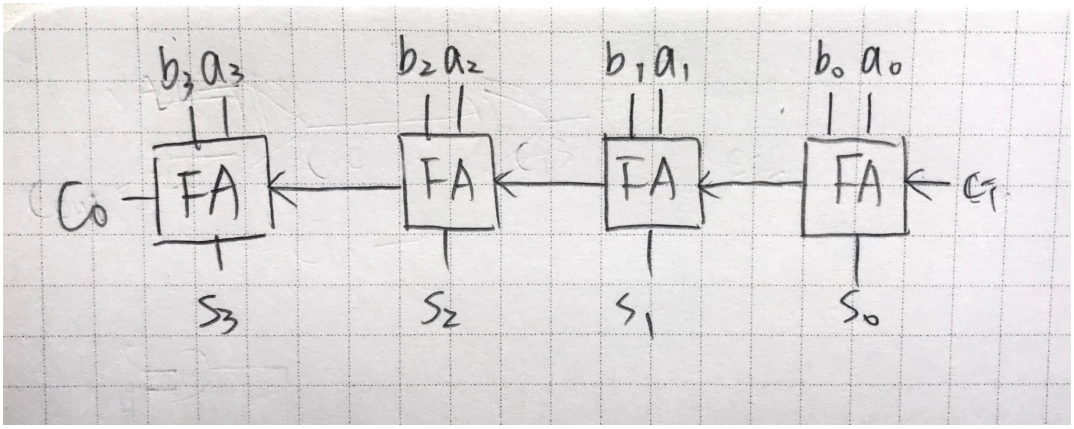
Specification:

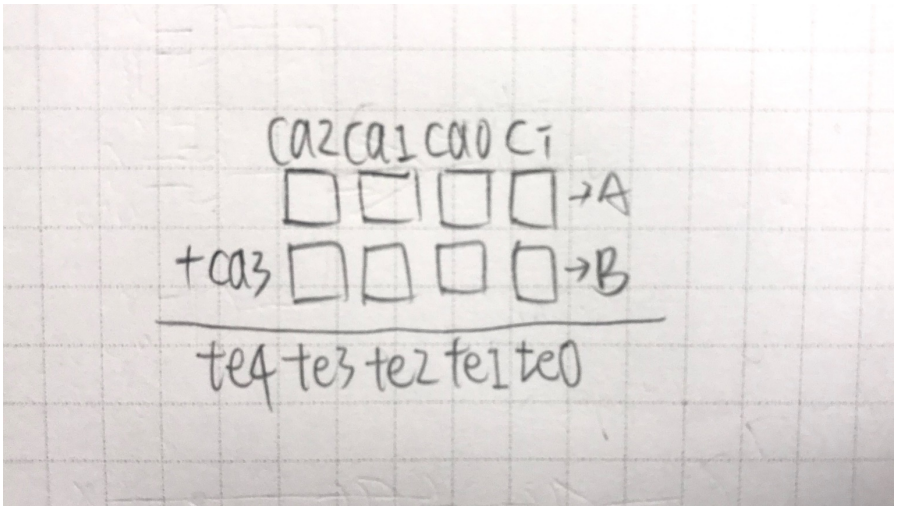
input : [3:0]A, [3:0]B, ci

output : co

output : [3:0]s







Verilog Module:

module lab2(

input [3:0]A, B,

```
input ci,
output reg co,
output reg [3:0]s
);
wire [2:0]ca;
wire [4:0]te;

assign te[0] = A[0]^B[0]^ci;
assign ca[0] = (A[0]&ci)|(B[0]&ci)|(A[0]&B[0]);
assign te[1] = A[1]^B[1]^ca[0];
assign ca[1] = (A[1]&ca[0])|(B[1]&ca[0])|(A[1]&B[1]);
assign te[2] = A[2]^B[2]^ca[1];
assign ca[2] = (A[2]&ca[1])|(B[2]&ca[1])|(A[2]&B[2]);
assign te[3] = A[3]^B[3]^ca[2];
assign te[4] = (A[3]&ca[2])|(B[3]&ca[2])|(A[3]&B[3]);

always @*
if(te>9)begin
{co, s} = te + 6;
//co = 1;
end
else begin
{co, s} = te;
//s = te;
//co = 0;
```

```
end

endmodule
```

Testbench:

```
module testlab2;

reg [3:0]A, B;

reg [1:0] Ci;

wire Co;

wire [3:0]S;


lab2 U1 (.co(Co),.s(S),.ci(Ci[0]),.A(A),.B(B));

initial

begin

for (A = 4'b0000; A<= 4'b1001; A = A+ 4'b0001)begin

      for (B = 4'b0000; B<= 4'b1001; B = B+ 4'b0001)begin

            for (Ci = 2'b00; Ci<= 2'b01; Ci = Ci+ 2'b01)

                  #10;

      end

end


end

endmodule
```

討論：這題與前一題的觀念類似，主要還是用二進位的加法，最後再以兩位數的 BCD 輸出，但是要注意的是兩個 4-bit BCD(0~9)相加超過 10 的話，會進位，co 會變成 1，而原本的 4-bit BCD(te)要再加 6，如果相加未超過 10，則 co 是 0，原本的 BCD 則不變，且因為兩個 4-bit 的數值相加可能會產生溢位所以 te 要設成 5-bit。
這題 debug 很久，像助教詢問後，了解到我當初的方程式為 s=te+6，s 與 te 的 bits 不同，所以產生錯誤，而 testbench 的問題則是當 Ci 設為 1 個 bit 跑 for 迴圈時，觀察產生的 waveform 會發現 A 與 B 都是 0，我猜測是因為 1 個 bit 的 Ci 造成無限迴圈，若將 Ci 改成 2 個 bits，結果即是正確的。

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

Specification:

input : [2:0]in, en

output : [7:0]d

3.1 Logic equation

In2	In1	In0	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Truth table

D0 = ~in2 & ~in1 & ~in0 & en

D1 = ~in2 & ~in1 & in0 & en

D2 = ~in2 & in1 & ~in0 & en

D3 = ~in2 & in1 & in0 & en

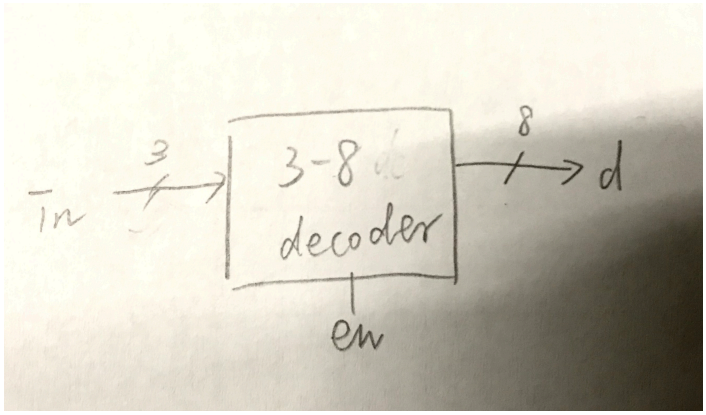
D4 = in2 & ~in1 & ~in0 & en

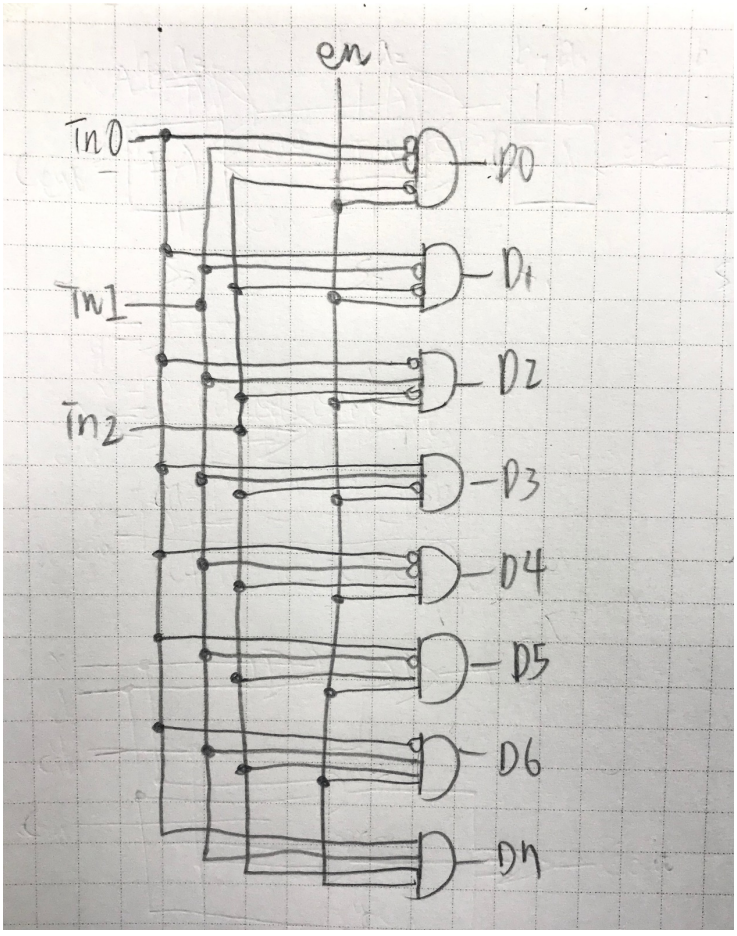
D5 = in2 & ~in1 & in0 & en

D6 = in2 & in1 & ~in0 & en

D7 = i2 & i1 & i0 & en

3.2 Logic schematic





3.3 Verilog RTL representation with verification

Verilog Module:

```
module lab3(

input [2:0]in,

input en,

output[7:0]d

);

assign d[0] = ~in[2] & ~in[1] & ~in[0] & en;

assign d[1] = ~in[2] & ~in[1] & in[0] & en;


assign d[2] = ~in[2] & in[1] & ~in[0] & en;

assign d[3] = ~in[2] & in[1] & in[0] & en;

assign d[4] = in[2] & ~in[1] & ~in[0] & en;

assign d[5] = in[2] & ~in[1] & in[0] & en;

assign d[6] = in[2] & in[1] & ~in[0] & en;

assign d[7] = in[2] & in[1] & in[0] & en;

endmodule
```

Testbench:

```
module testlab3;

reg [3:0]In;
```

```
reg [1:0]En;

wire [7:0]D;


lab3 U2(.d(D),.in(In[2:0]),.en(En[0]));

initial

begin

for (In = 4'b0000; In<= 4'b0111; In = In + 4'b0001)

for (En = 2'b00; En<= 2'b01; En = En + 2'b01)

#10;

end

endmodule
```

討論：decoder 有一個 enable 接腳，enable 可以掌控整個電路的功能，當 en 為 0 時，無論輸入狀態為何，均無法導致任何解碼輸出，D0~D7 均為 0，僅在 en 為 1 時允許解碼的功能隨著輸入而變化。將得出的 equation 用 assign 寫入 code 中即可完成。

結論：之前修邏輯設計時，那堂課的教授並沒有教verilog，所以我是從頭開始學，上網看了一些教學影片，也有去圖書館借書，發現每個人的code都寫得不太一樣，而且自己使用的是mac沒有vivado能用，只能等實驗課的時候上機操作。上機後，一開始出現了很多錯誤，後來是詢問了助教，熟悉verilog的基本觀念後，我也找到了自己習慣的寫法，發現其實並不難，重點是解題思路清晰且邏輯正確後，自然就能把code寫出來了。