邏輯設計實驗Lab3結報

105060012張育菘

**1. Frequency Divider: Construct a 27-bit synchronous binary counter. Use the MSB of the counter, we can get a frequency divider which provides a 1/227 frequency output (fout) of the original clock (fcrystal, 100MHz). Construct a frequency divider of this kind.**

**1.1 Write the specification of the frequency divider.**

**1.2 Draw the block diagram of the frequency divider.**

**1.3 Implement the frequency divider with the following parameters.**

| **clk** | **clk\_out** | **rst** |
| --- | --- | --- |
| **W5** | **U16** | **V17** |

Design Specification

input : rst, clk;

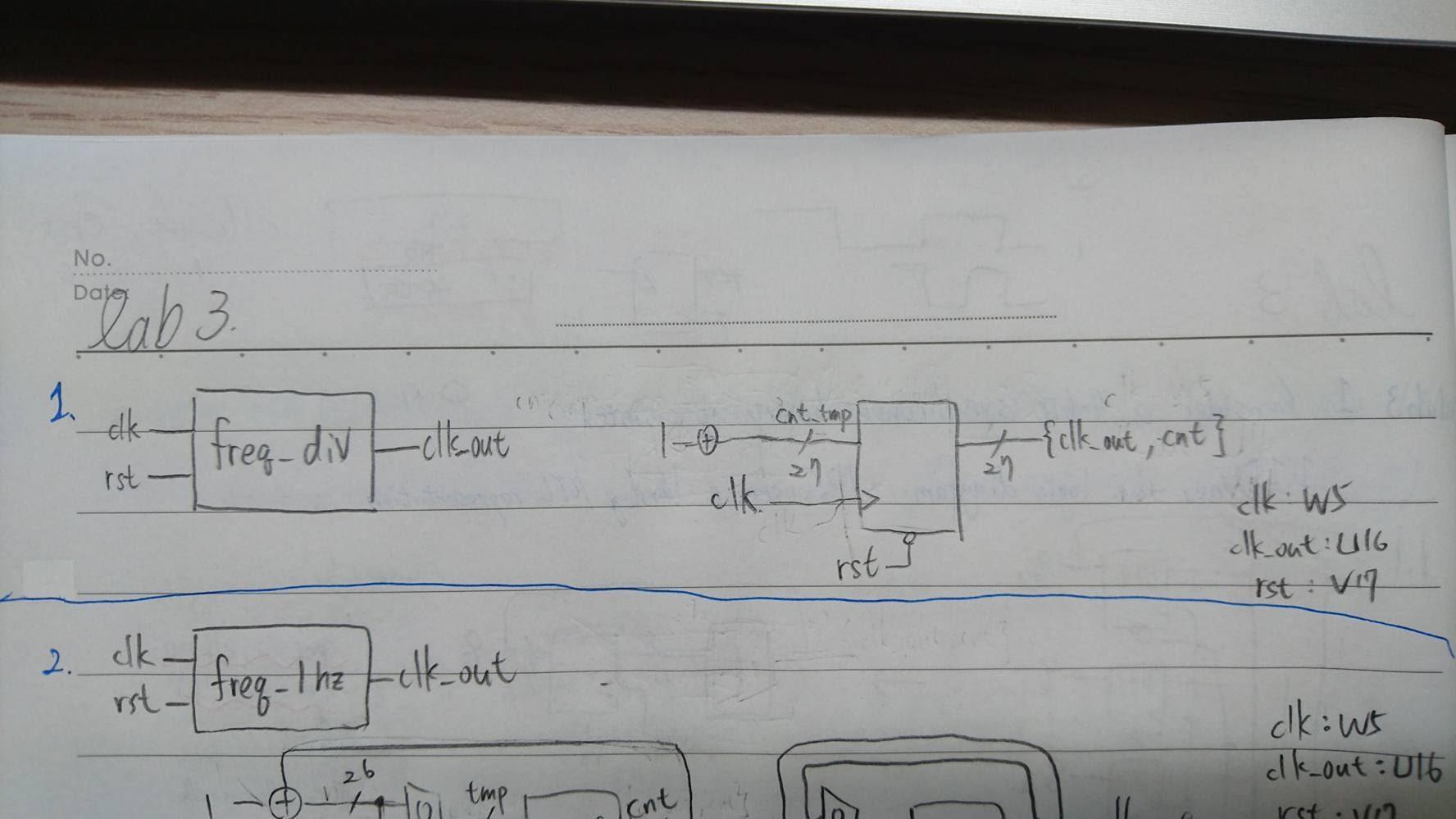
output : clk\_out;

reg clk\_out;

reg [25:0]cnt;

reg [26:0]cnt\_tmp;

block diagram :



Design Implementation

Logic function :

always@\* cnt\_tmp={clk\_out,cnt}+1'b1;

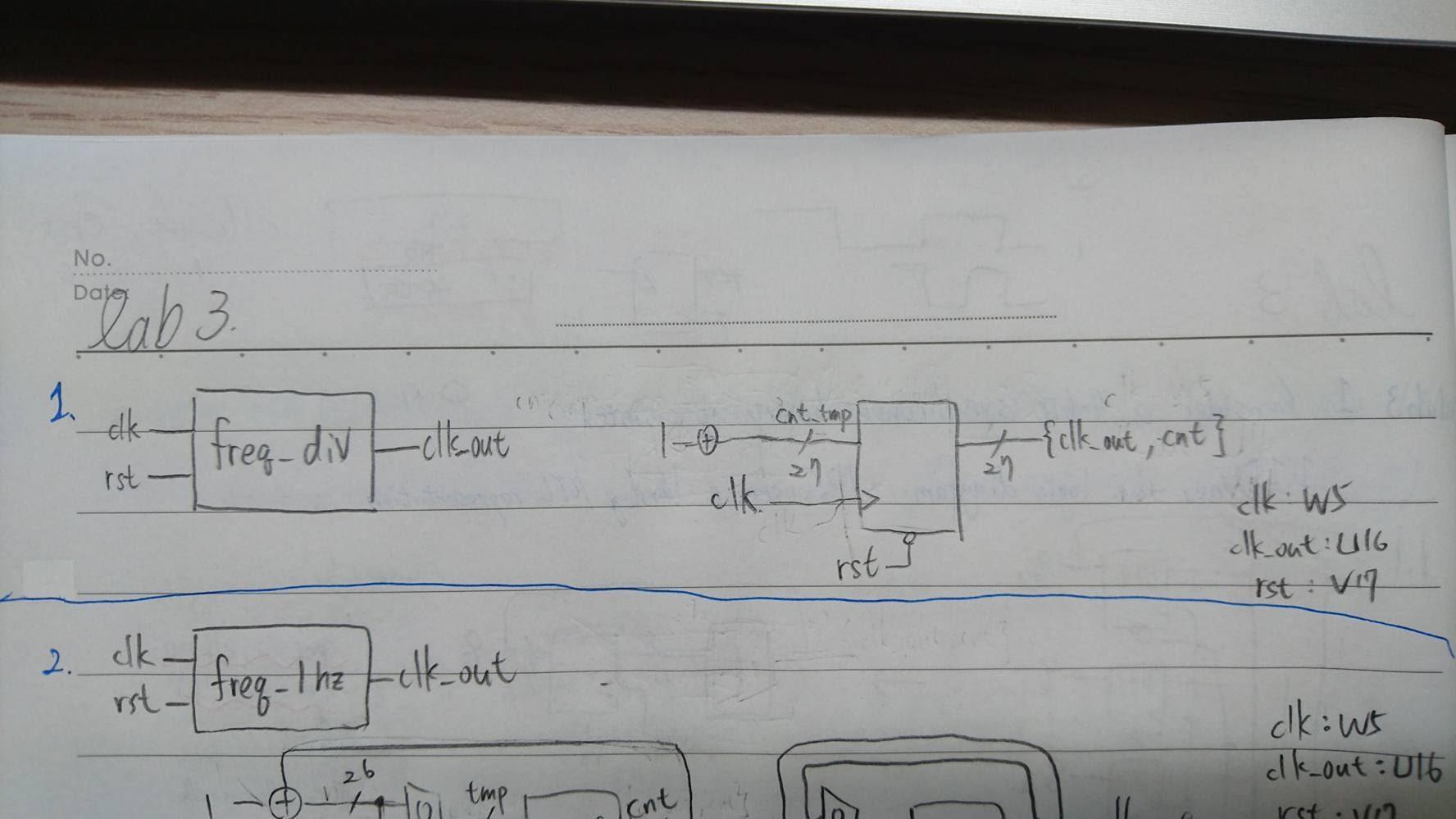
always@(posedge clk or negedge rst) //DFFs

if(~rst){clk\_out,cnt}<=27'b0; //當rst=0時，{clk\_out,cnt} = 0(從0開始數)

else{clk\_out,cnt}<=cnt\_tmp;

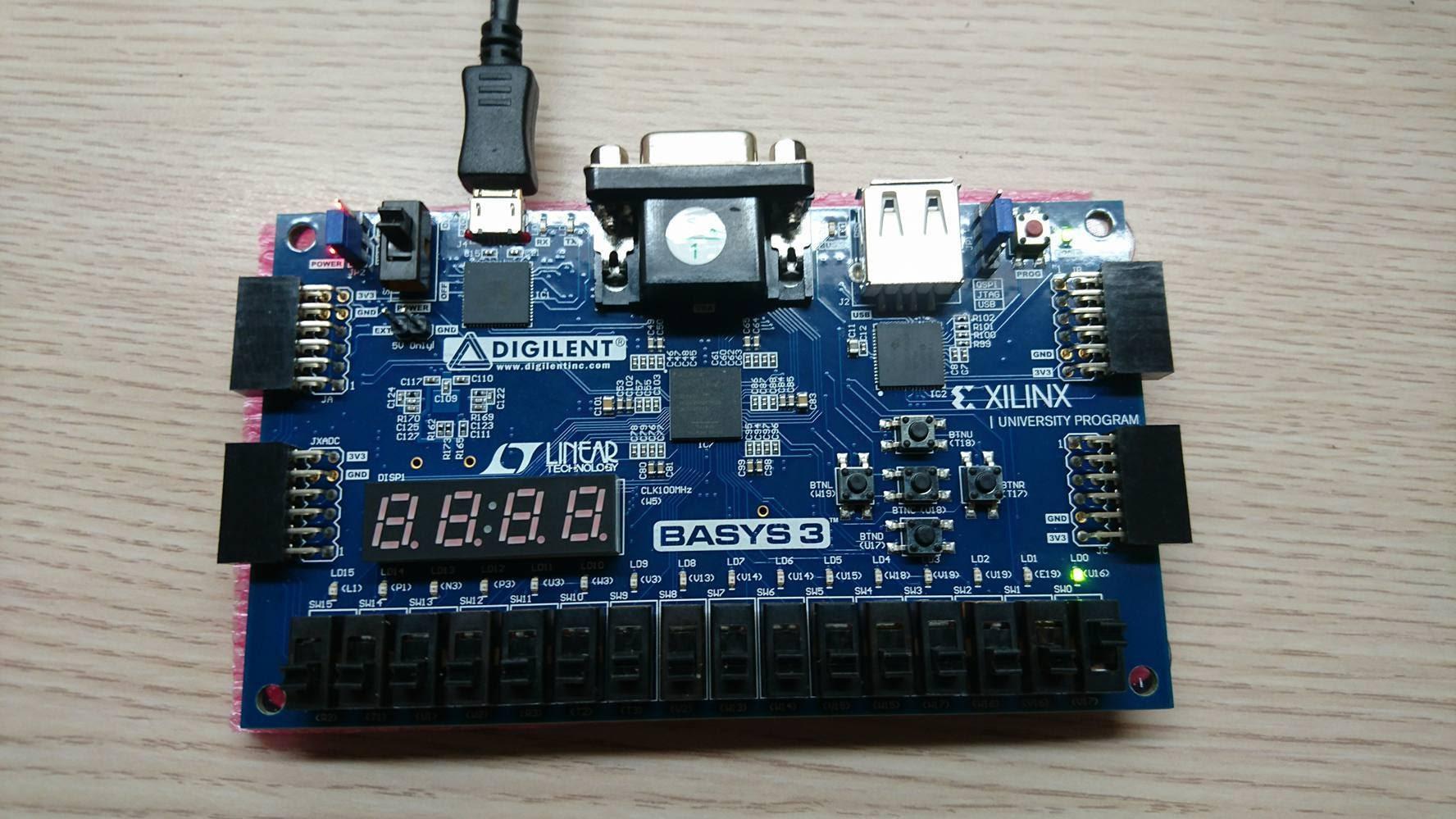
//當rst=1 & clk從0→1時，把cnt\_tmp存入{clk\_out,cnt}

Logic diagram :



Result

當rst=1時，其閃爍的頻率為clk的1/227倍



**2.Frequency Divider: Use a count-for-50M counter and some glue logics to construct a 1 Hz clock frequency. Construct a frequency divider of this kind.**

**2.1 Write the specification of the frequency divider.**

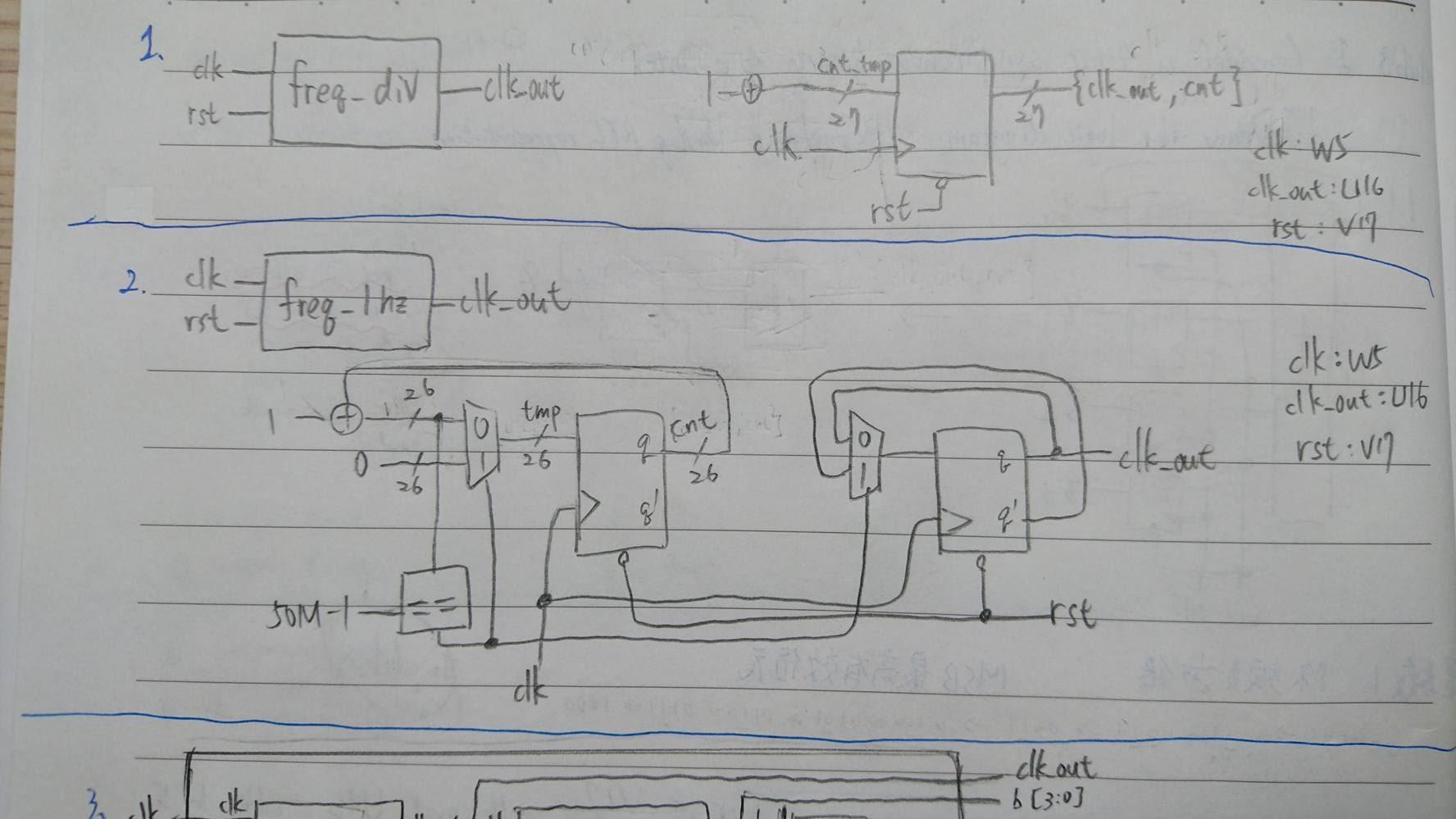
**2.2 Draw the block diagram of the frequency divider.**

**2.3 Implement the frequency divider with the following parameters.**

| **clk** | **clk\_out** | **rst** |
| --- | --- | --- |
| **W5** | **U16** | **V17** |

Design Specification

input : rst, clk;

output : clk\_out;

reg clk\_out;

reg [25:0]cnt;

reg [26:0]cnt\_tmp;

Design Implementation

Logic function :

always@\* cnt\_tmp={clk\_out,cnt}+1'b1;

always@(posedge clk or negedge rst)

if(~rst) {clk\_out,cnt} <= 27'b0;

else

begin

{clk\_out,cnt} <= cnt\_tmp;

if(cnt==49999999)

//當cnt數道49999999時，clk\_out從0→1或是從1→0，也讓cnt重數

begin

cnt<=26'b0;

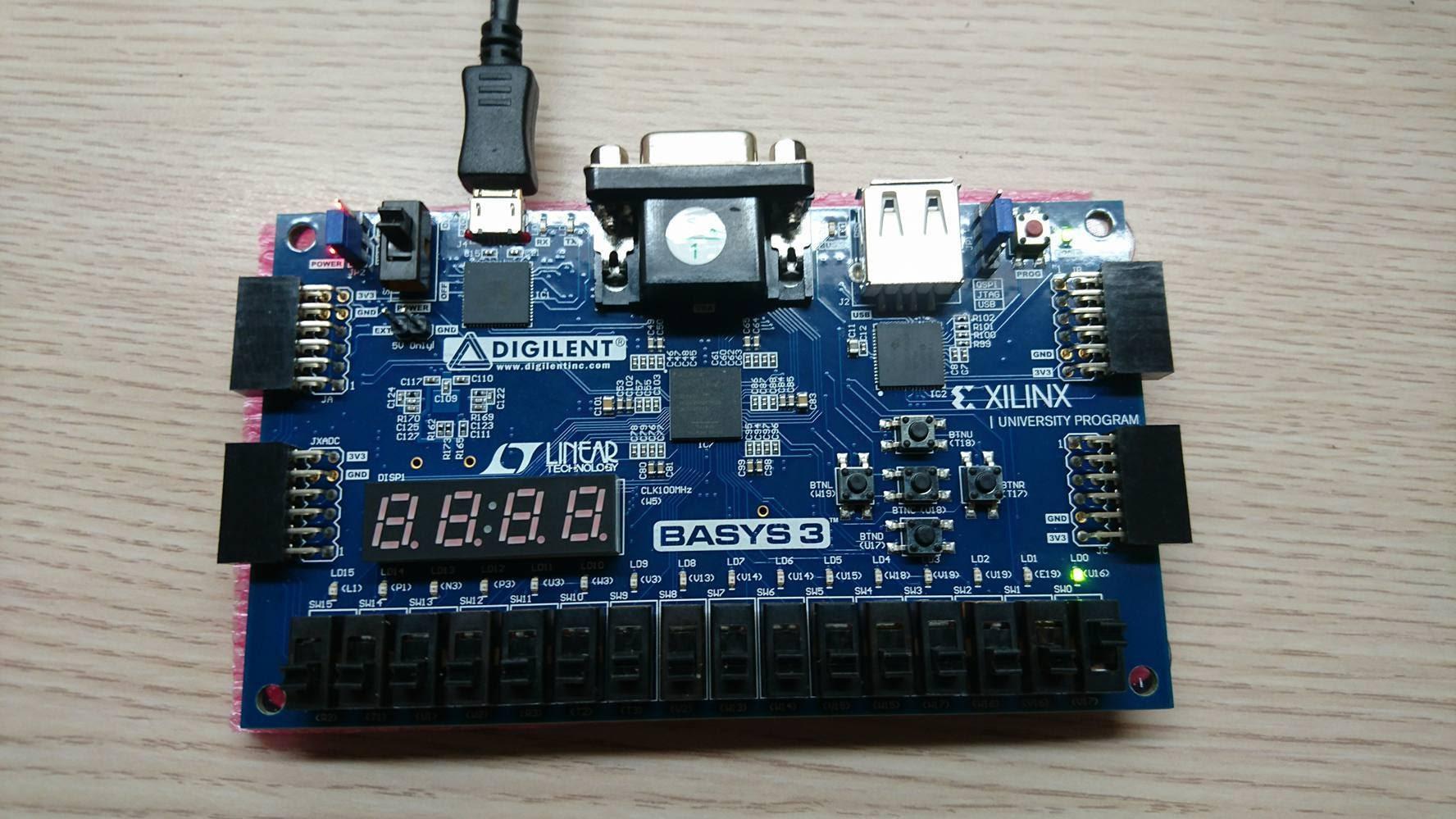
clk\_out<=~clk\_out;

end

end

Result

當rst=1時，其閃爍的頻率為clk的1/(5\*107)倍



Discussion

1. 除頻的方法可以用多個bit進行，因為越高位的頻率因加法的特性會變低
2. 在作DFFs時需要next\_state和state，因此在做的時候拿cnt\_tmp當作我的next\_state

**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.**

**3.3 Combine the above two together.**

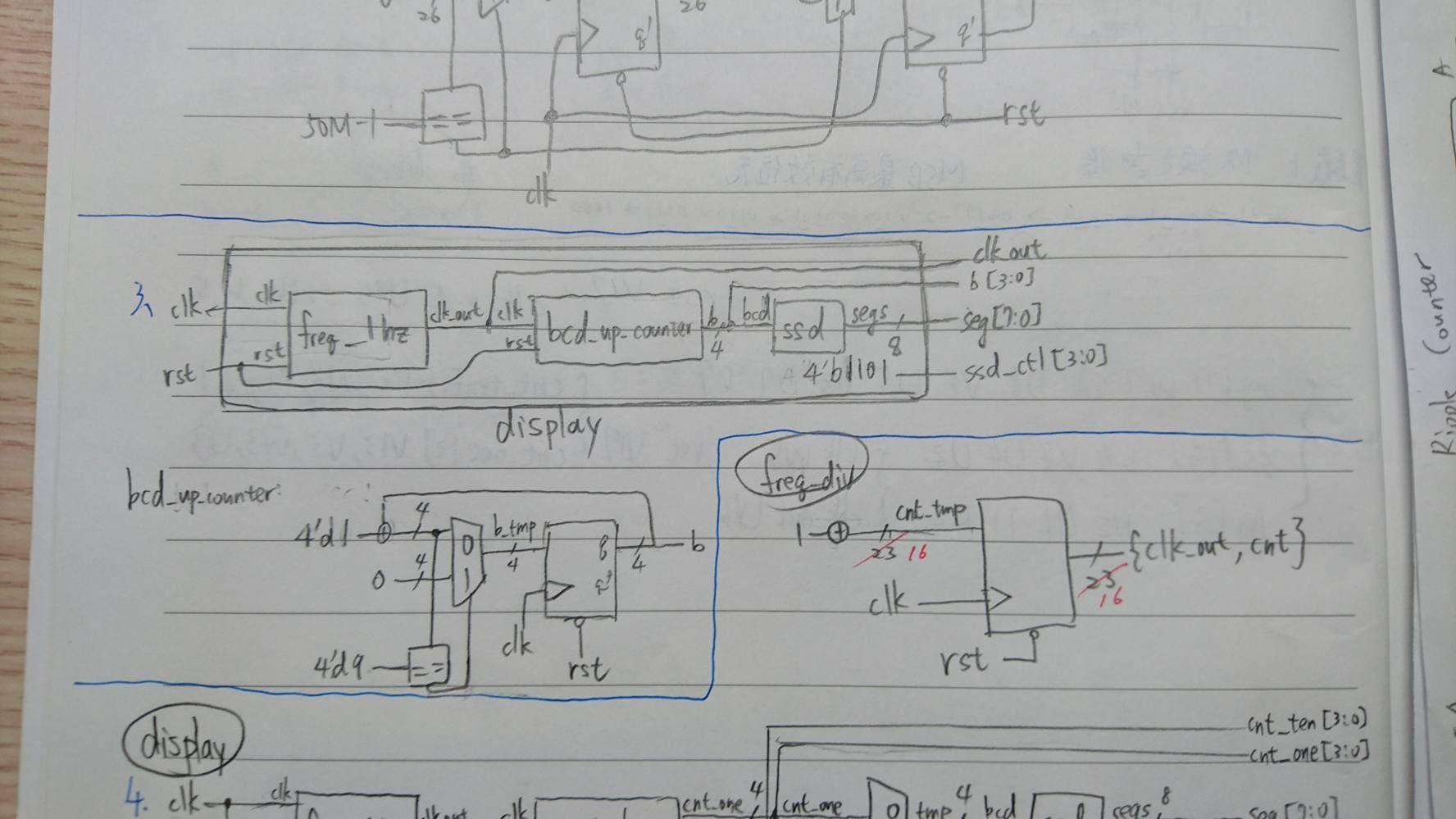
| **clk** | **clk\_out** | **b[3]** | **b[2]** | **b[1]** | **b[0]** |
| --- | --- | --- | --- | --- | --- |
| **W5** | **U16** | **P3** | **N3** | **P1** | **L1** |

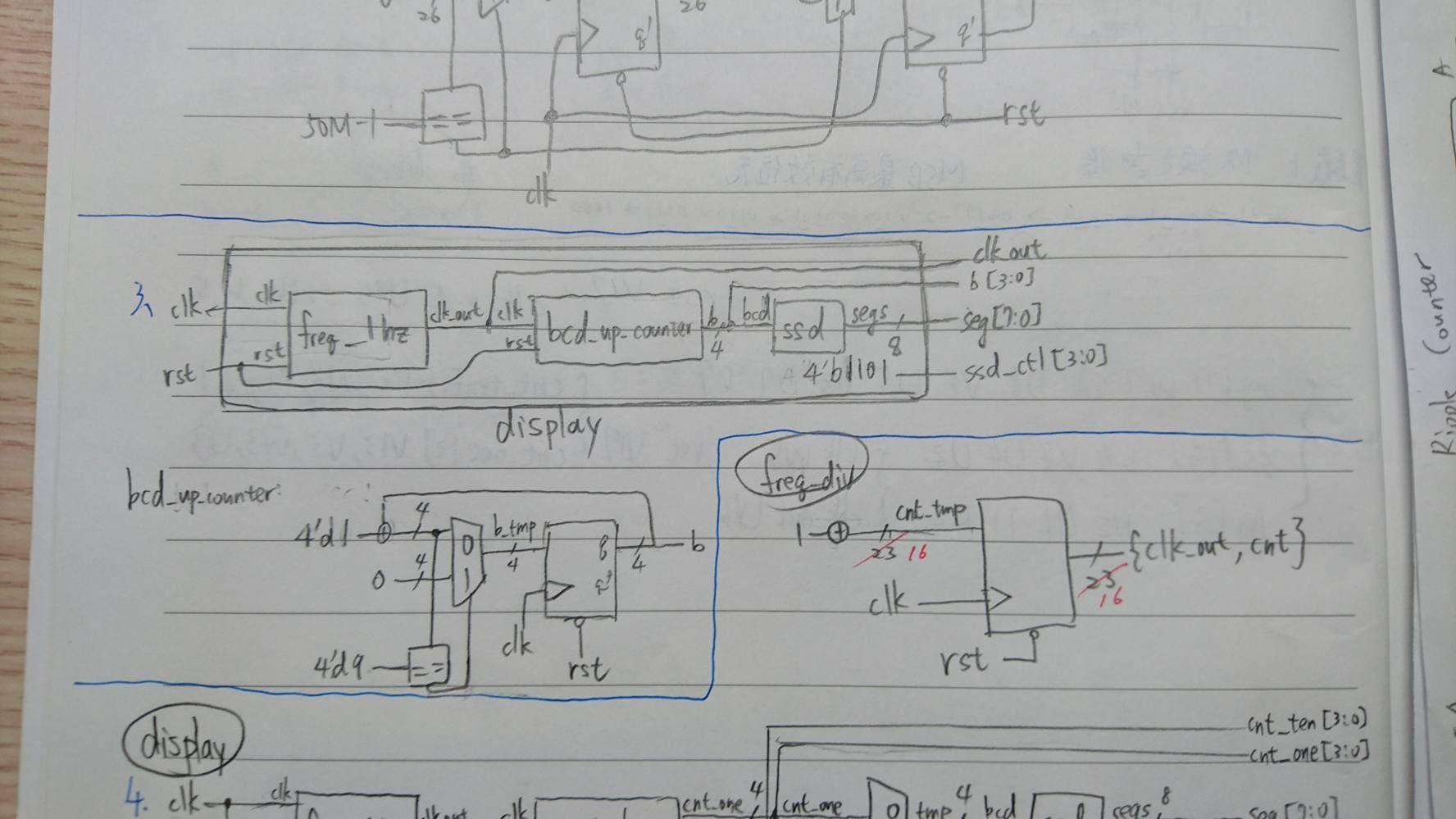
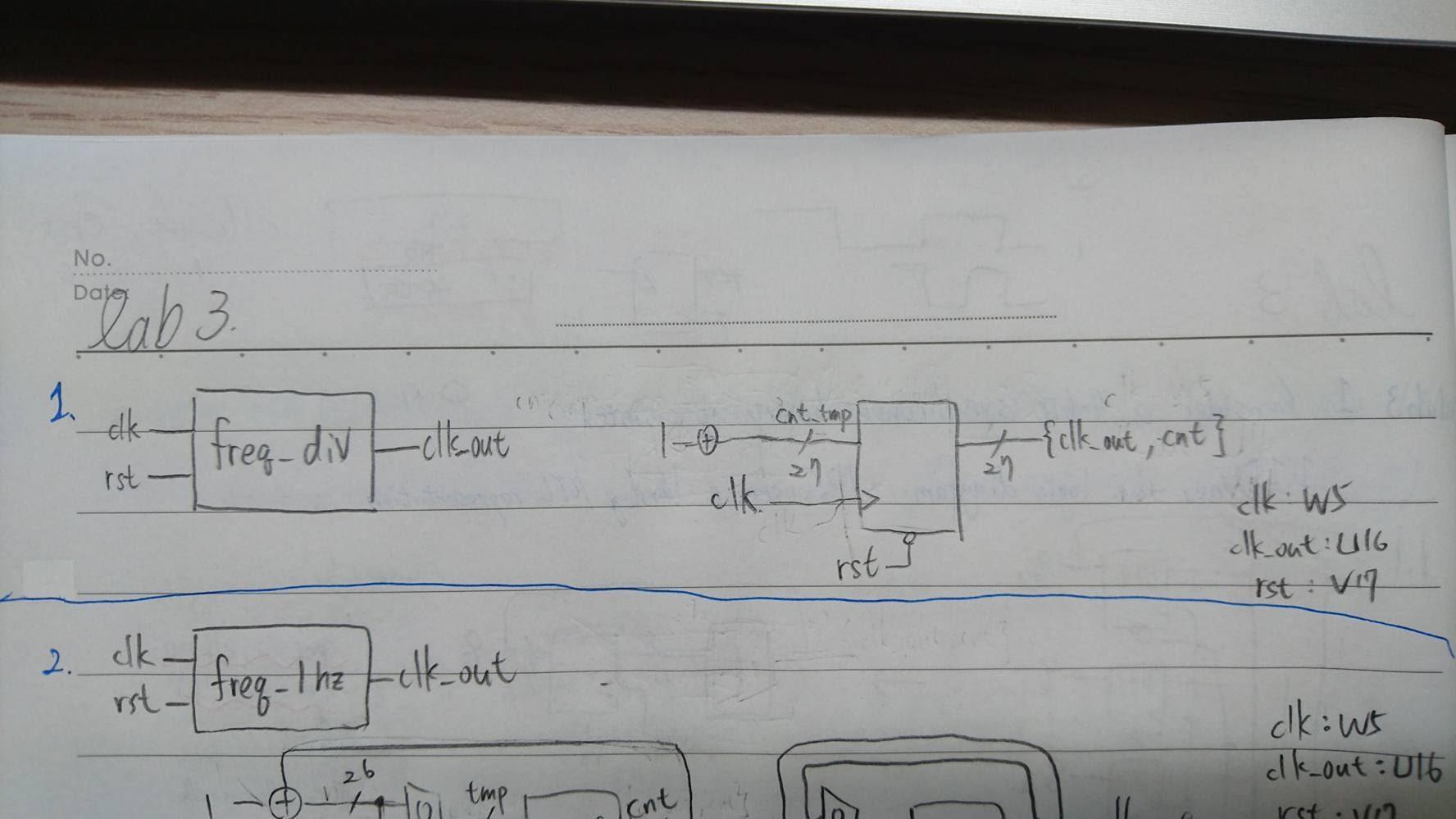
Design Specification

input : clk,rst;

output : clk\_out,[3:0]b, [3:0]ssd\_ctl, [7:0]seg;

block diagram :



(freq\_1hz)

Design Implementation

Logic function :

1. display：

此為top module，專門拿來呼叫其他小moudule的，並把freq\_1hz除頻過後的clk\_out作為bcd\_up\_counter的clk，讓bcd\_up\_counter在每1hz時把數值加一。並只讓顯示器第3個數字亮(ssd\_ctl = 4'b1101)。

1. freq\_1hz：

其運作原理同第二題，當cnt數到49999999時，clk=~clk以及讓cnt從0開始重數，而clk\_out即是除頻過後的產物，頻率為1hz。

1. bcd\_up\_counter：

先設一個b\_tmp[3:0]作為DFFs的next\_state，而b[3:0]作為DFFs的state。當b數到9時，把b的值歸零，用此達到從0數到9後再從0開始的效果。而其reset的值在此設為0。

always@\*

b\_tmp=b+1'b1;

always@(posedge clk or negedge rst)

if(~rst) b <= 4'b0;

else

begin

b <= b\_tmp;

if(b==9)

begin

b<=4'b0;

end

end

1. ssd：

當bcd 為 4'd0: segs = 8'b00000011; //七段顯示器顯示0

當bcd 為 4'd1: segs = 8'b10011111; //七段顯示器顯示1

當bcd 為 4'd2: segs = 8'b00100101; //七段顯示器顯示2

當bcd 為 4'd3: segs = 8'b00001101; //七段顯示器顯示3

當bcd 為 4'd4: segs = 8'b10011001; //七段顯示器顯示4

當bcd 為 4'd5: segs = 8'b01001001; //七段顯示器顯示5

當bcd 為 4'd6: segs = 8'b01000001; //七段顯示器顯示6

當bcd 為 4'd7: segs = 8'b00011111; //七段顯示器顯示7

當bcd 為 4'd8: segs = 8'b00000001; //七段顯示器顯示8

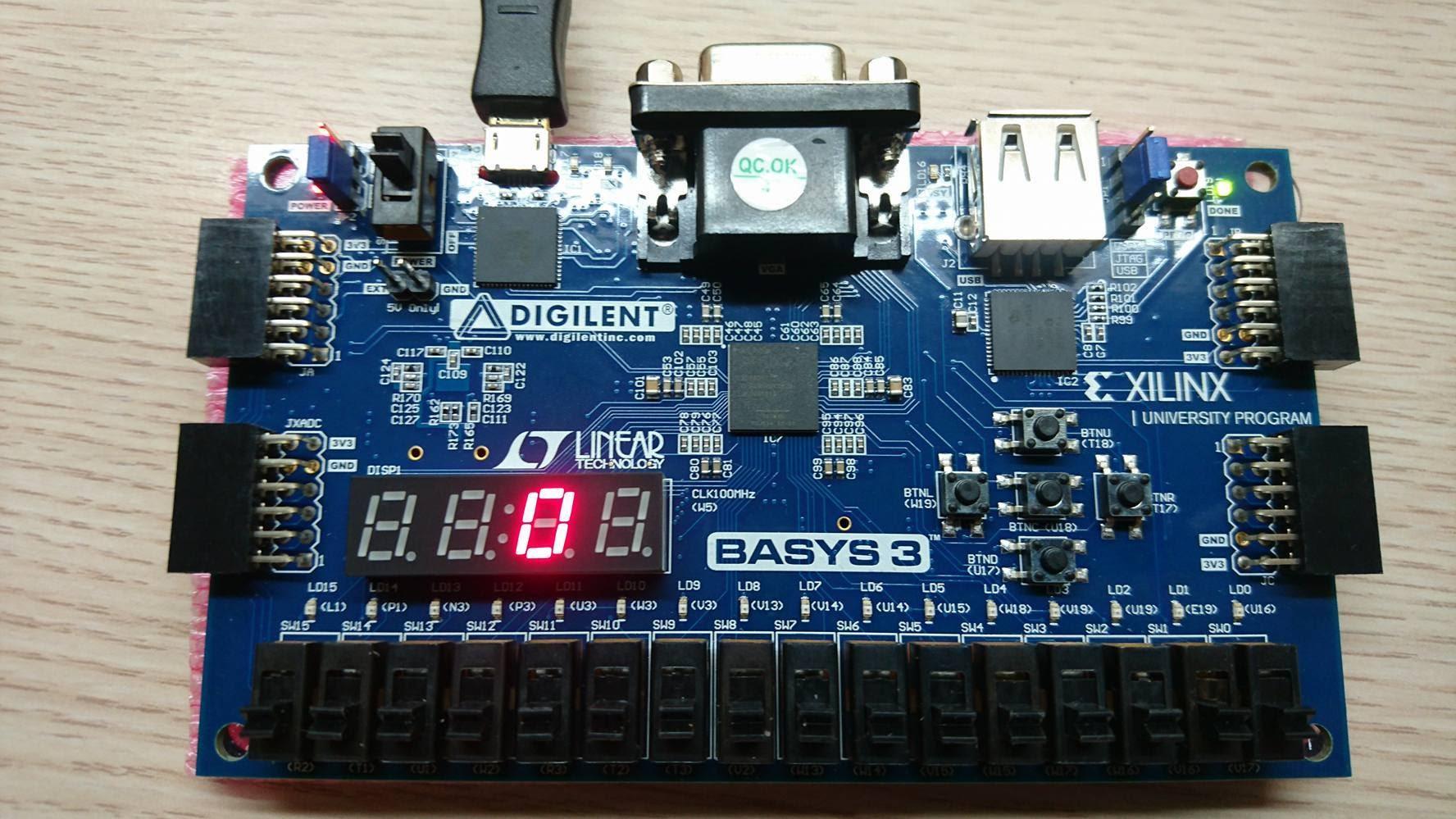
當bcd 為 4'd9: segs = 8'b00001001; //七段顯示器顯示9

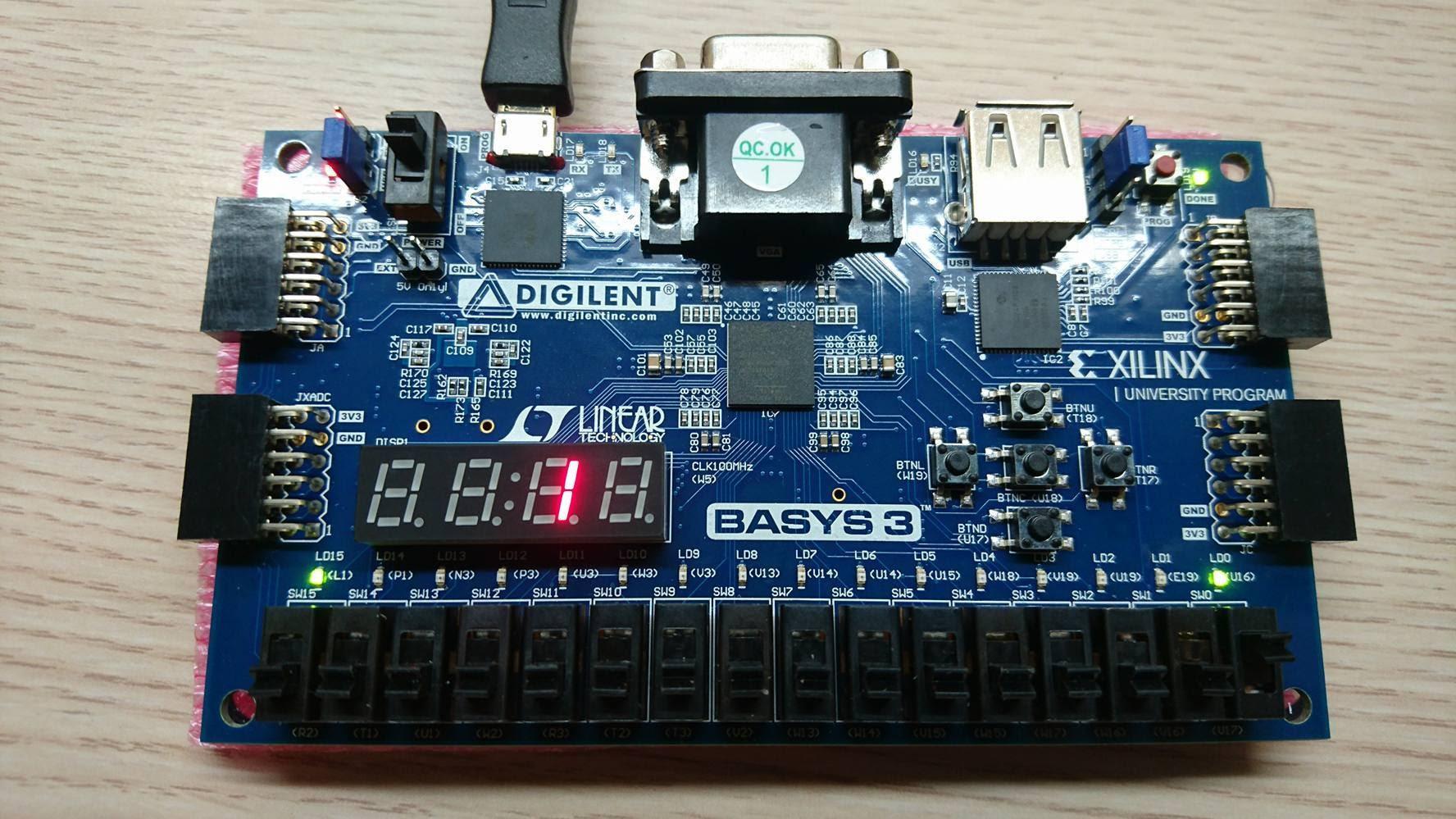
default: segs = 8'b00000000; //七段顯示器顯示8.

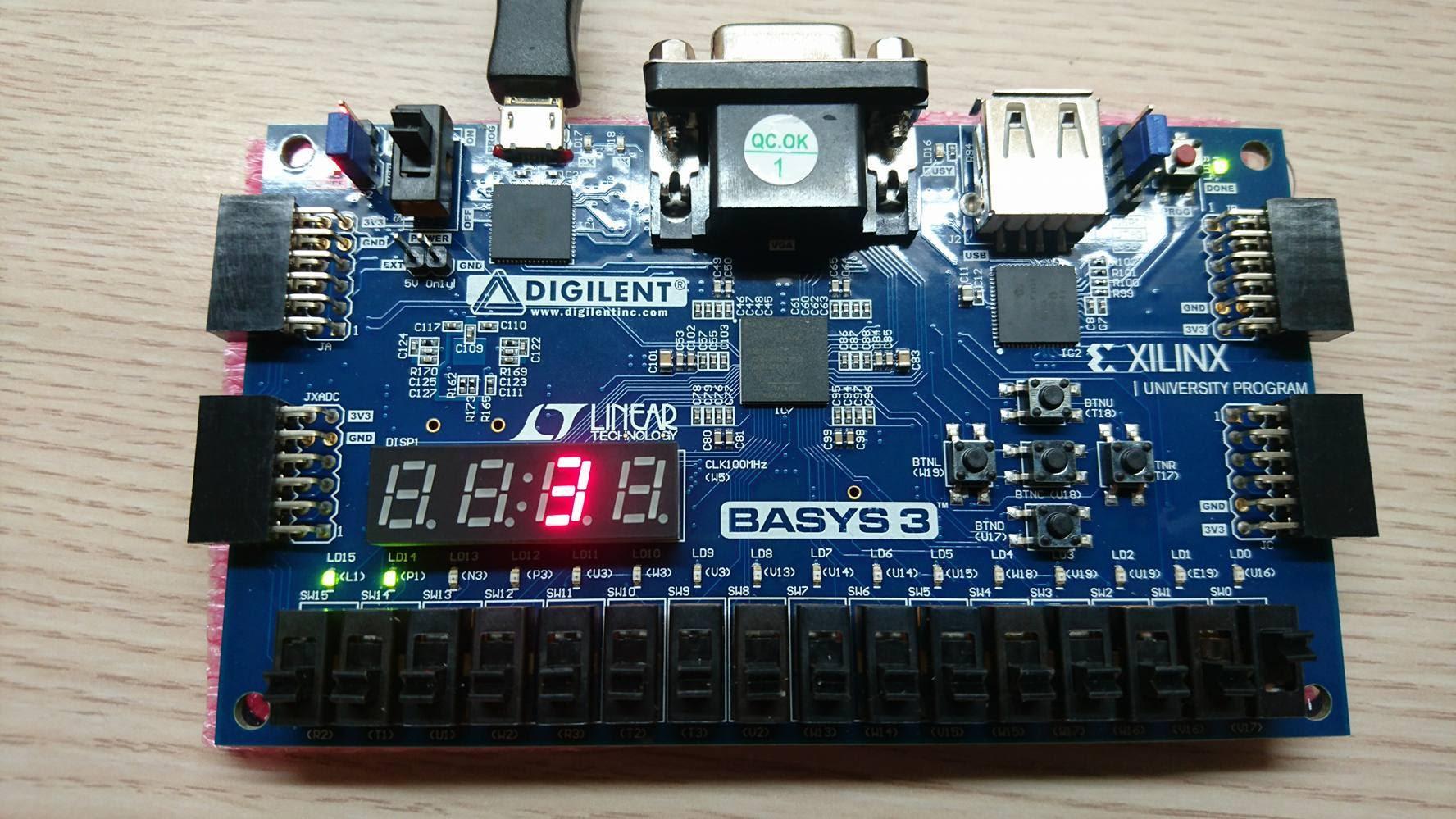
Result(顯示器只有第3個數字亮)

1. rst=0→初始化設為0
2. rst=1→每當U16的LED燈亮的霎那，七段顯示器顯示的數字開始往上數

，顯示b的燈也跟著往上跳一個數字







Discussion

1. 在呼叫其他小module時，括號內的I/O變數為大module的I/O變數；括號外的為被呼叫的小module的I/O變數。Ex. ssd U2(.bcd(b),.segs(seg)); ”U2”為名稱、”b”為display的I/O變數、”bcd”為ssd的I/O變數。
2. 最好用一個top module包小module，這樣在跑實驗時比較不會跑出不是理想中的結果。

**4.** **(Bonus) Construct a 30 seconds count down timer (stop at 00).**

| **clk** | **clk\_out** | **cnt\_ten** | **cnt\_ten** | **cnt\_ten** | **cnt\_ten** |
| --- | --- | --- | --- | --- | --- |
| **W5** | **U16** | **cnt\_ten** | **cnt\_ten** | **cnt\_ten** | **cnt\_ten** |

| **cnt\_one[3]** | **cnt\_one[2]** | **cnt\_one[1]** | **cnt\_one[0]** |
| --- | --- | --- | --- |
| **V13** | **V3** | **W3** | **U3** |

Design Specification

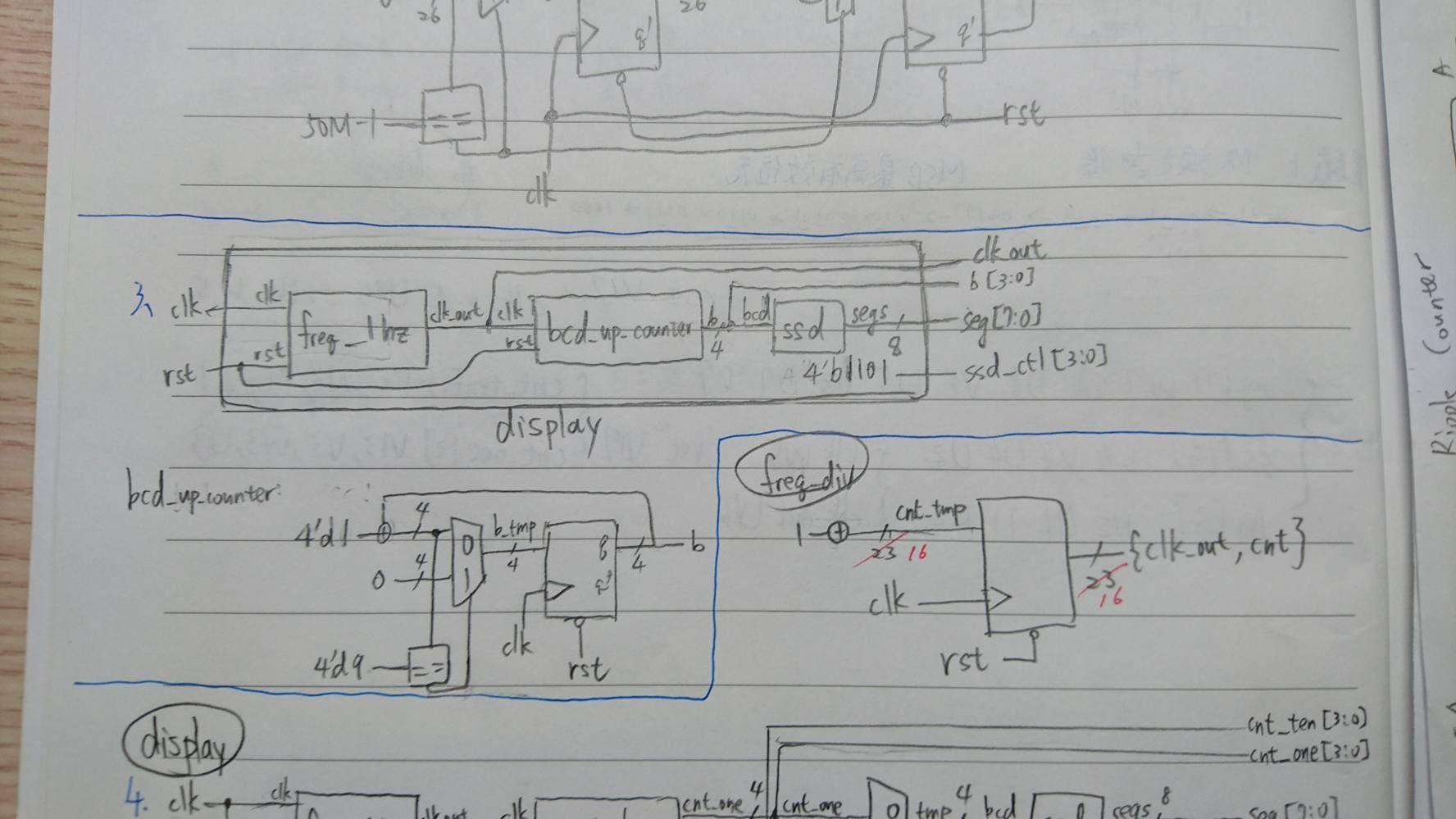
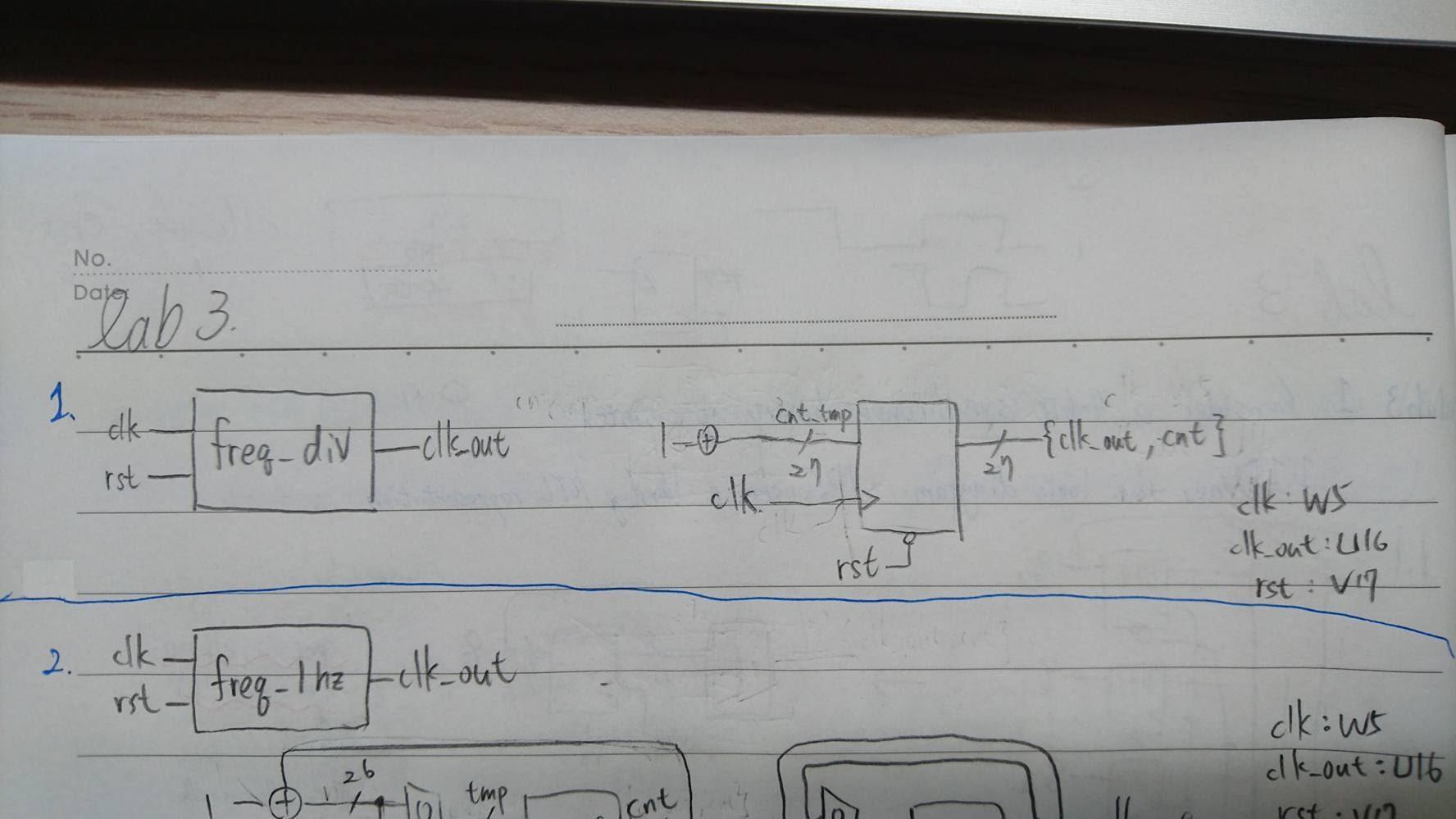
input : clk,rst;

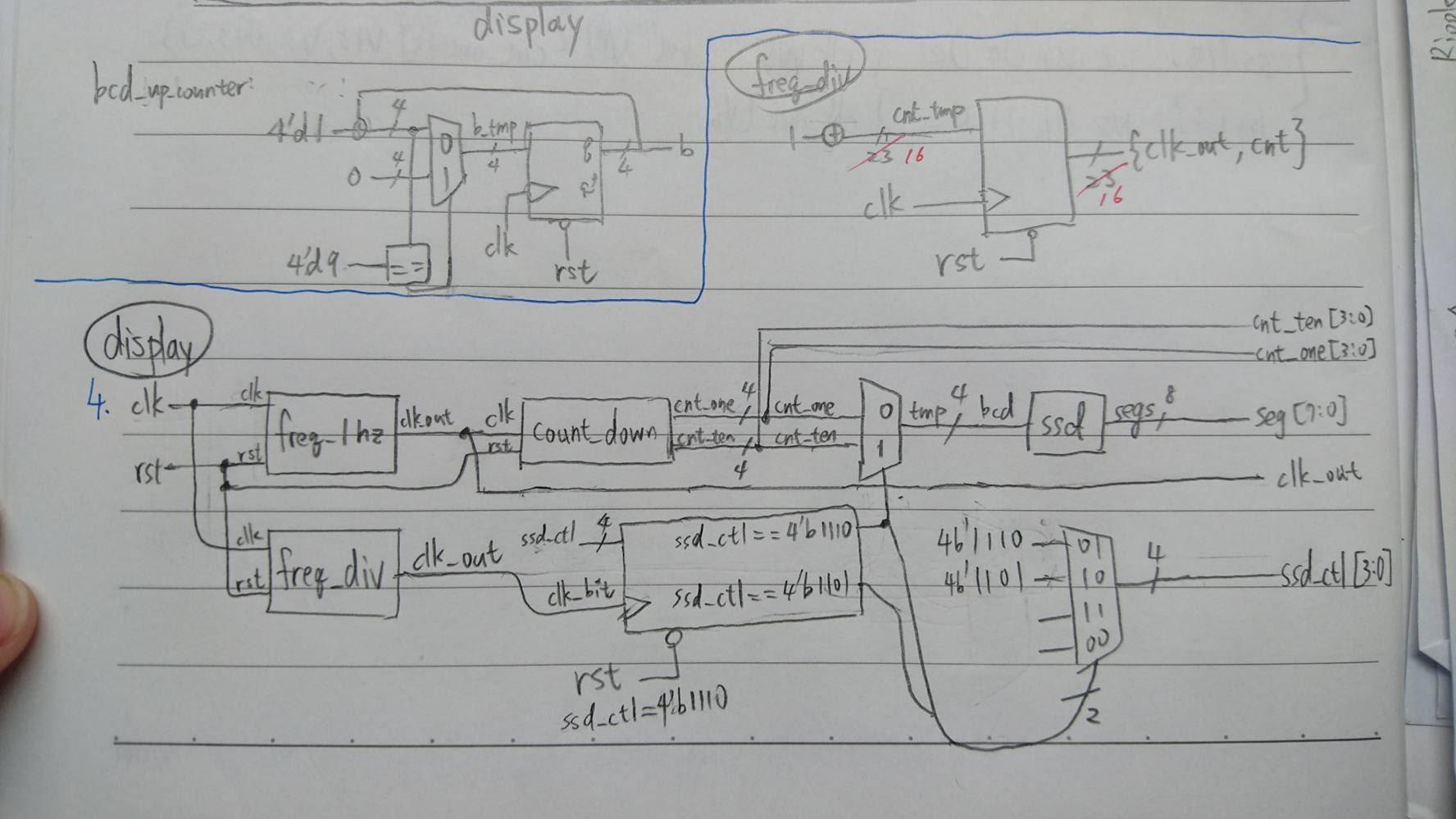
output : clk\_out, [3:0]cnt\_one,[3:0]cnt\_ten, [3:0]ssd\_ctl, [7:0]seg;

wire clk\_bit;

reg [3:0]tmp, [3:0]ssd\_ctl;

block diagram :

 (freq\_1hz)



Design Implementation

Logic function :

1. display：

此為top module，專門拿來呼叫其他小moudule的，並把freq\_1hz除頻過後的clk\_out作為count\_down的clk，讓count\_down在每1hz時把數值減一。

還有把經過freq\_div除頻過後的clk\_out(週期為內建時間的216倍)作為自己的clk\_bit。

內部還有一個多功器，功能即是當顯示其顯示十位數時讓顯示器的第2個數字亮；當顯示其顯示個位數時讓顯示器的第1個數字亮。

always@(posedge clk\_bit or negedge rst)

begin

if(ssd\_ctl == 4'b1110)

//當顯示器第一個數字亮時同時把cnt\_ten存入tmp以及讓顯示器第1個數字亮的訊號存入ssd\_ctl，讓下一次clk\_bit從0→1時，顯示此結果。

begin

tmp <= cnt\_ten;

ssd\_ctl <= 4'b1101;

end

else if(ssd\_ctl == 4'b1101)

begin

tmp <= cnt\_one;

ssd\_ctl <= 4'b1110;

end

else;

end

1. freq\_1hz：

其運作原理同第二題，當cnt數到49999999時，clk=~clk以及讓cnt從0開始重數，而clk\_out即是除頻過後的產物，頻率為1hz。

1. freq\_div：

其運作原理同第一題，cnt\_tmp為一16bits的數，因此其最高為即為clk\_out，即是除頻過後的產物，週期為clk的216倍。

1. count\_down：

這是同時數兩個變數，個位數cnt\_one跟十位數cnt\_ten進行往下數。當cnt\_one=0時把cnt\_ten的數值減一以及cnt\_one的數值變成9；當cnt\_one跟cnt\_ten都為0時，讓兩個數都維持0。而初始化(rst=0)就是把樹設為30(cnt\_ten=3 & cnt\_one=0)。

1. ssd：

當bcd 為 4'd0: segs = 8'b00000011; //七段顯示器顯示0

當bcd 為 4'd1: segs = 8'b10011111; //七段顯示器顯示1

當bcd 為 4'd2: segs = 8'b00100101; //七段顯示器顯示2

當bcd 為 4'd3: segs = 8'b00001101; //七段顯示器顯示3

當bcd 為 4'd4: segs = 8'b10011001; //七段顯示器顯示4

當bcd 為 4'd5: segs = 8'b01001001; //七段顯示器顯示5

當bcd 為 4'd6: segs = 8'b01000001; //七段顯示器顯示6

當bcd 為 4'd7: segs = 8'b00011111; //七段顯示器顯示7

當bcd 為 4'd8: segs = 8'b00000001; //七段顯示器顯示8

當bcd 為 4'd9: segs = 8'b00001001; //七段顯示器顯示9

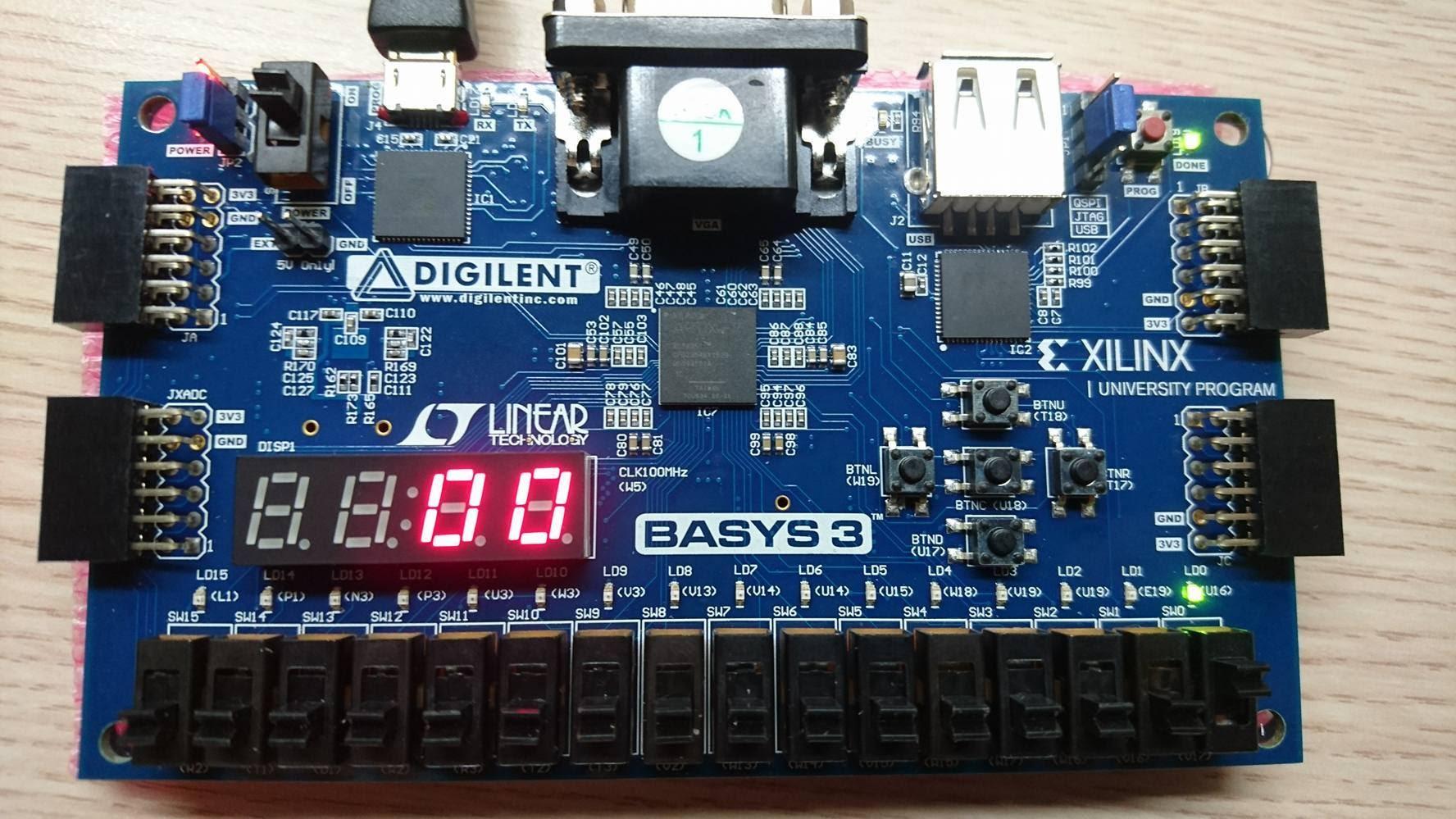
default: segs = 8'b00000000; //七段顯示器顯示8.

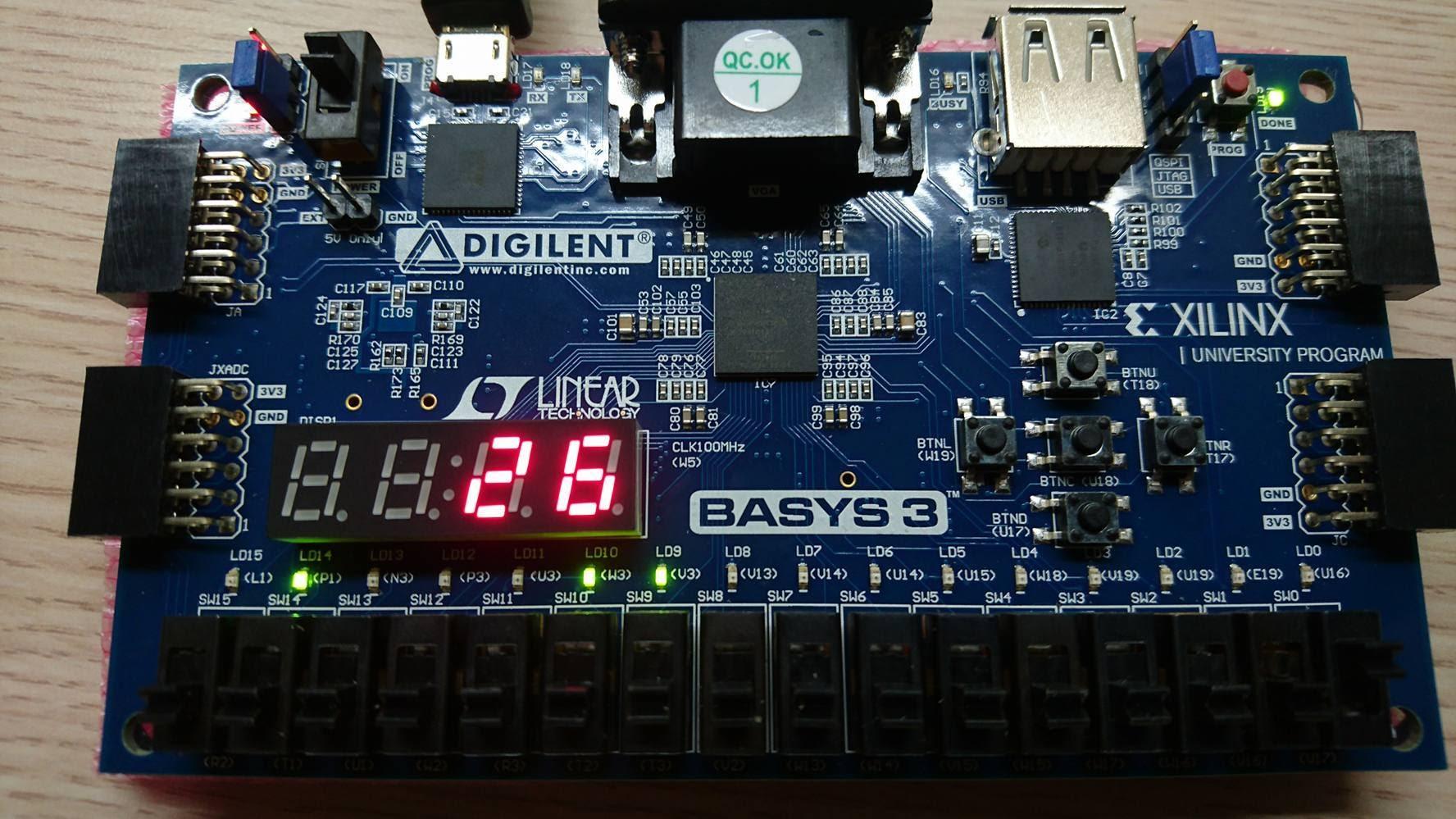
Result(顯示器只有第3個數字亮)

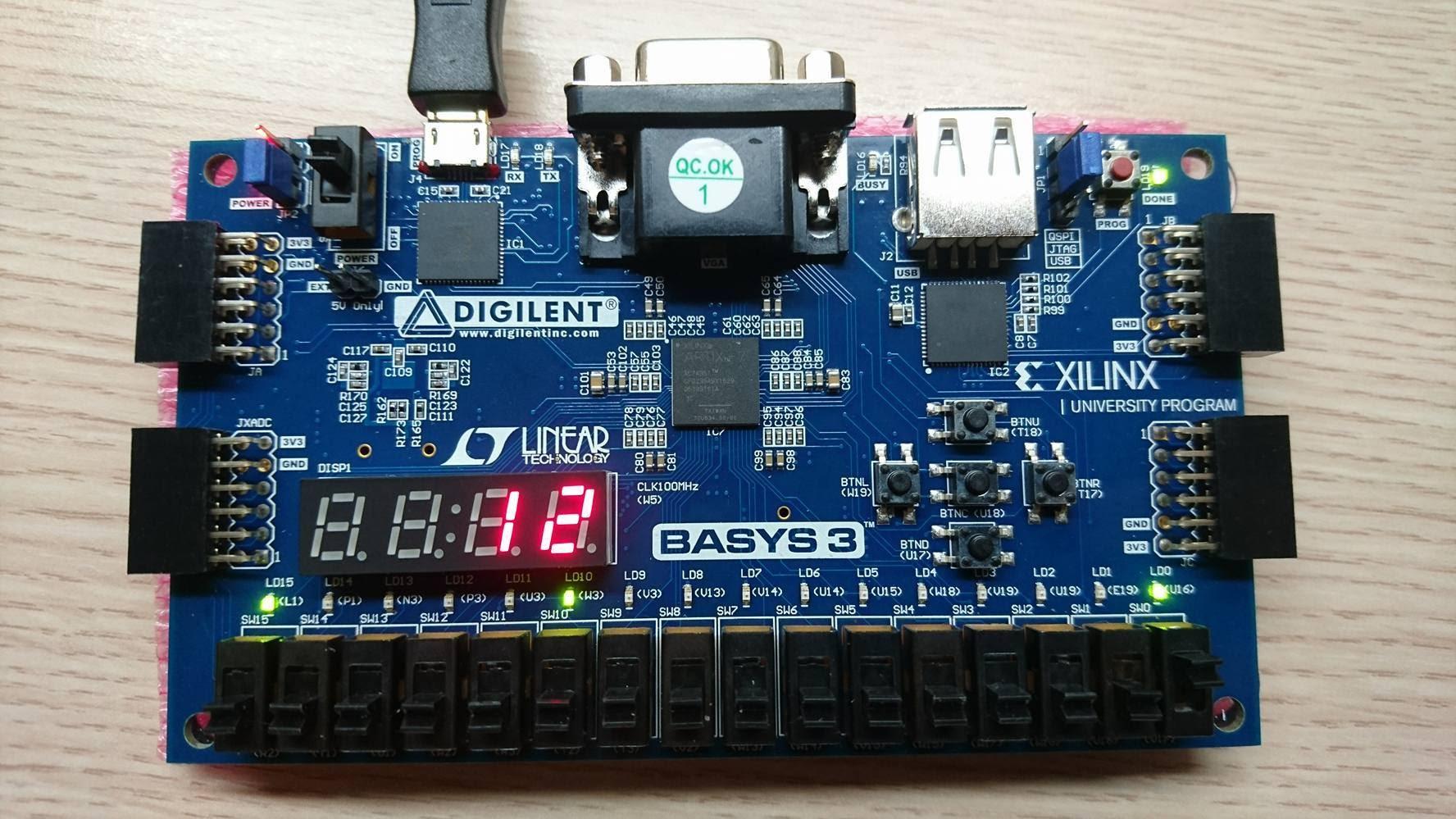
1. rst=0→初始化設為30
2. rst=1→每當U16的LED燈亮的霎那，七段顯示器顯示的數字開始往下數

，顯示cnt\_one以及cnt\_ten的燈也跟著往下跳一個數字

1. 當數到00時，顯示器保持00。







Discussion

1. 這次實驗用的clk\_bit的用意為顯示兩個不同的數，因為七段顯示器一次只能顯示一種數次，因此需要一個頻率介於內建時間到1hz的clk\_bit產生同時顯示兩個數的錯覺。
2. clk\_bit不能為內建的clk，因為此轉換時間太快了，兩個數的殘像讓眼睛無法分辨兩個數，就像兩個數若有似無的疊在一起似的，導致我們在觀察時看到不對的結果。

**Conclusion**：

這次的lab更深入帶我們了解到如何除頻，以及如何同時顯示兩個不同的數，雖然過程中挫折不斷，但是打完真的是成就感滿滿，真的很開心。