DD LAB10

一、地點：工學501A

二、準備時間：6/15 1924-1932

三、DEMO時間：6/15 1932-1940

BF16主要概念在於透過降低數字的精度，從而減少運算資源和功耗。
在 BF16中，我們使用 1 bit 表示正值或負值，8 bits 表示指數，7 bits 表示尾數精度

Sign = +1
Exponent = $(01111100)_2 - 127 = -3$
Fraction
$= 1 + (0.0100000)_2$
$= 1 + 2^{-2}$
$= 1.25$
Value = $(+1) \times 1.25 \times 2^{-3} = +0.15625$

四、評分方式

1. 輸入一BF16浮點數，按下N17將其顯示在七段顯示器上（60%）
   (1)位移處理
   （i） 分成整數num跟小數float的部分，若exp 是正的左移，exp 是負的右移。
   （ii） 整數num選用10個bit ，因2^10=1024>999
   （iii）小數float選用14個bit ，因1/(2^14)= 1/16384 < 1/9999

```
42  reg [9:0] num,num_tmp;// 2^10 = 1024 >999
43  reg [13:0] float_point,float_point_tmp,float_point_tmp2;// 1 / 2^14 = 1 / 16384 < 1/9999

66  always@ ( posedge clk ) begin
67      num_tmp <= num;
68      float_point_tmp <=float_point;
69      exp_tmp <= exp;
70      fra_tmp <= fra;
71      if (exp_tmp <= 255 )begin   // exp is pos
72      // 10 bits + 14bits <= 10bit + 7bit + 7 bit
73          {num_tmp,float_point_tmp} <= { 10'b0000000001,fra_tmp,7'b0000000 } << exp_tmp;
74      end
75      else begin // exp is nega
76          exp_tmp2 <= ~exp_tmp+1;
77          {num_tmp,float_point_tmp} <= { 10'b0000000001,fra_tmp,7'b0000000 } >> exp_tmp2;
78      end
79  end
80
```

(2)顯示正負號與整數處理

(i)sign為1顯示-，為0不顯示(11 為empty)

```
128 always@(posedge clk) begin
129     counter <=(counter<=100000) ? (counter +1) : 0;
130     state <= (counter==100000) ? (state + 1) : state;
131       case(state)
132           0:begin
133               if ( sign == 1'b1 )
134                   seg_number <= 10;
135               else
136                   seg_number <= 11;
137               scan <= 8'b0111_1111;
138           end
139           1:begin
140               seg_number <= (num_tmp/100);
141               scan <= 8'b1011_1111;
142           end
143           2:begin
144               seg_number <= (num_tmp/10) %10;
145               scan <= 8'b1101_1111;
146           end
147           3:begin
148               seg_number <= num_tmp %10;
149               scan <= 8'b1110_1111;
```

(3)顯示小數處理

(i)對應表

Binary	dec
0.10000000000000	0.5000
0.01000000000000	0.2500
0.00100000000000	0.1250
0.00010000000000	0.0625
0.00001000000000	0.0313
0.00000100000000	0.0157
0.00000010000000	0.0079
0.00000001000000	0.0040
0.00000000100000	0.0020
0.00000000010000	0.0010
0.00000000001000	0.0005
0.00000000000100	0.0003
0.00000000000010	0.0002
0.00000000000001	0.0001

(ii)實作

```
82  always @(posedge clk) begin
83      float_point_tmp2 <= float_point_tmp;
84      float_digit_tmp[13] <= float_point_tmp2[13] * 14'd5000;
85      float_digit_tmp[12] <= float_point_tmp2[12] * 14'd2500;
86      float_digit_tmp[11] <= float_point_tmp2[11] * 14'd1250;
87      float_digit_tmp[10] <= float_point_tmp2[10] * 14'd625;
88      float_digit_tmp[9] <= float_point_tmp2[9] * 14'd313;
89      float_digit_tmp[8] <= float_point_tmp2[8] * 14'd157;
90      float_digit_tmp[7] <= float_point_tmp2[7] * 14'd79;
91      float_digit_tmp[6] <= float_point_tmp2[6] * 14'd40;
92      float_digit_tmp[5] <= float_point_tmp2[5] * 14'd20;
93      float_digit_tmp[4] <= float_point_tmp2[4] * 14'd10;
94      float_digit_tmp[3] <= float_point_tmp2[3] * 14'd5;
95      float_digit_tmp[2] <= float_point_tmp2[2] * 14'd3;
96      float_digit_tmp[1] <= float_point_tmp2[1] * 14'd2;
97      float_digit_tmp[0] <= float_point_tmp2[0] * 14'd1;
98      float_digit <= float_digit_tmp[0] + float_digit_tmp[1] + float_digit_tmp[2] + float_digit_tmp[3] + float_digit_tmp[4]
99                  +float_digit_tmp[5] + float_digit_tmp[6] + float_digit_tmp[7] + float_digit_tmp[8] + float_digit_tmp[9]
100                 +float_digit_tmp[10] + float_digit_tmp[11] + float_digit_tmp[12] + float_digit_tmp[13];
101 end
```

(iii)顯示

```
151         4:begin
152             seg_number <= (float_digit/1000) ;
153             scan <= 8'b1111_0111;
154         end
155         5:begin
156             seg_number <= (float_digit/100) %10;
157             scan <= 8'b1111_1011;
158         end
159         6:begin
160             seg_number <= (float_digit/10) %10;
161             scan <= 8'b1111_1101;
162         end
163         7:begin
164             seg_number <= (float_digit) %10;
165             scan <= 8'b1111_1110;
166         end
167         default: state <= state;
```

(4)overflow顯示F

待處理

2. 按下P17按鈕將輸入數值+7，並將其顯示在七段顯示器上（20%）

   待處理

3. 按下M17按鈕將輸入數值*3，並將其顯示在七段顯示器上（20%）

   待處理

附錄

DD LAB10 問題

1. 遇到無窮小數的時候要四捨五入還是無條件捨去呢? 都可以，容許些許誤差
2. 乘法要用原始數字去做，還是用四捨五入或無條件捨去後的數字去做呢?原始
3. 需要顯是中間過程嗎? 還是只要最終結果就好呢? 最終結果
4. LED也需要跟著sw亮嗎?不一定要
5. 乘法可以直接用乘法嗎? 還是只能用加法迭代?可直接用乘法