Improve Skill：

Lab01

1. 用 shift 取代加減法或乘法可以節省很多資源(除法比較複雜沒辦法)：

例如：a*3 = (a <<< 1) + a

(a)注意括號，否則他可能會判定成 a <<< 1 + a = a <<< (1 + a)

(b)注意<<< or << (有 or 無號數)

2. 加法乘法除法如果放在 always 的 if 判斷式中可能會多用很多資源，可以就盡量拆出來。比如說：

以下這個

```
always@(*) begin
  if(opt[2] == 1) begin
    out_n = ((inputdata3<<<1) + inputdata3 - inputdata0*inputdata4);
    if(out_n[9] == 1) begin
      out_n = ~(out_n)+1;
    end
    else if(out_n[9] == 0)begin
      out_n = out_n;
    end
  end
  else if(opt[2] == 0) begin
    avg = (inputdata0+inputdata1+inputdata2+inputdata3+inputdata4)/5;
    num1 = inputdata0;
    num2 = inputdata1*inputdata2;
    num3 = avg*inputdata3;
    out_n = (num1 + num2 + num3)/3;
  end
  else begin
    avg = avg;
    num1 = num1;
    num2 = num2;
    num3 = num3;
    out_n = out_n;
  end

end
```

代替成以下這個

```
Division Division1(.in1(inputdata0+inputdata1+inputdata2+inputdata3+inputdata4),.in2(10'd5),.out(avg));
Adder Adder1(.in1(inputdata0),.in2(inputdata1*inputdata2),.in3(avg*inputdata3),.out(out2_temp));
Adder Adder2(.in1((inputdata3<<<1)),.in2(inputdata3),.in3(~(inputdata0*inputdata4)+10'd1),.out(out1));
Division Division2(.in1(out2_temp),.in2(10'd3),.out(out2));
always@(*) begin
  if(opt[2] == 1) begin
    out_n = out1;
    if(out_n[9] == 1) begin
      out_n = ~(out_n)+1;
    end
    else if(out_n[9] == 0)begin
      out_n = out_n;
    end
  end
  else if(opt[2] == 0) begin
    out_n = out2;
  end
  else begin
    out_n = out_n;
  end

end
```

```
module Adder(in1,in2,in3,out);
  input wire  signed[9:0]in1,in2,in3;
  output wire signed [9:0]out;
  assign out = in1 + in2 + in3;
endmodule


module Division(in1,in2,out);
  input wire  signed[9:0]in1,in2;
  output wire signed [9:0]out;
  assign out = in1 / in2;
endmodule
```

(a)要注意語法上引用函數不能放在 always 內

(b)子函式腳位如果要用 constant 引入，常數 size 的位數要 matchc 好

比如說以上例子，Division1 要接「10’d5”」在定義成[9:0]的 in2，不能用「6’d5”」之類。

3.減少暫存器可以大大減少面積。

4.else case 寫完整除了避免 Latch，面積也會變小(?

5.負數用 2 補數