

The background features abstract, overlapping green geometric shapes, primarily triangles and polygons, in various shades of green, creating a modern and dynamic look.

數位系統導論實驗 Lab10

Serial Squarer IP

負責助教: 王璽喆

Outline

- ▶ 課程目的
- ▶ 回顧 Sequential Circuit in Verilog
- ▶ 範例
- ▶ 作業說明及評分方式
- ▶ 附錄

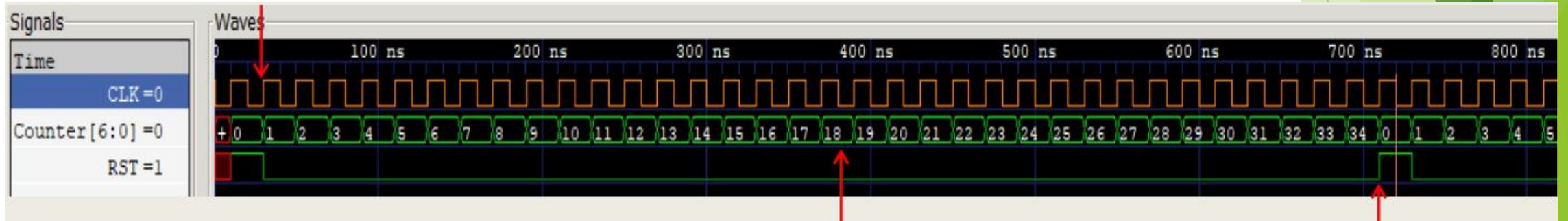
課程目的

- ▶ 學習以 Verilog 設計 32-bit optimized serial multiplier
- ▶ 學習以 Verilog 設計 32-bit modified booth multiplier
- ▶ 學習套用 32-bit booth multiplier 於 Zedboard 做為 serial squarer

回顧 Sequential Circuit in Verilog

- ▶ 右圖程式碼描述：觸發 CLK 或 RST 訊號正緣時會執行紅框內的程式碼

```
//Counter
always @(posedge CLK or posedge RST)
begin
    if(RST)
        Counter <= 6'b0;
    else
        Counter <= Counter + 6'b1;
end
```



Counter 逐漸增加直到 RST為1

RST 正緣

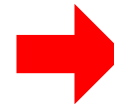
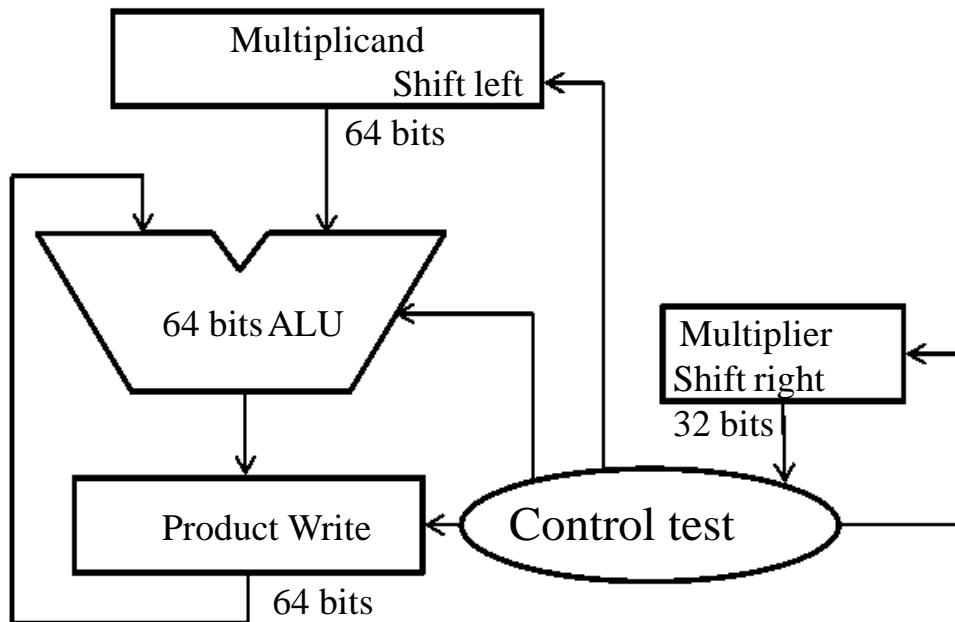
範例 - Unsigned 32-bit Serial Multiplier

- ▶ 本範例將以 Verilog 模擬 unsigned 32-bit serial multiplier，並以 testbench 產生如下圖的測試結果

```
// Successful //  
//      30 *      90 = ?  
//your answer is      2700, correct answer is      2700  
  
// Fail //  
//      30 *      -90 = ?  
//your answer is      128849016180, but correct answer is      -2700  
  
// Fail //  
//     -30 *      90 = ?  
//your answer is      386547053940, but correct answer is      -2700  
  
// Fail //  
//     -30 *     -90 = ?  
//your answer is 18446743558313478796, but correct answer is      2700  
  
// Successful //  
// 4294967295 * 4294967295 = ?  
//your answer is 18446744065119617025, correct answer is 18446744065119617025  
  
// Fail //  
// 4294967296 *      1 = ?  
//your answer is      0, but correct answer is      4294967296  
  
// Fail //  
// 4294967297 *      1 = ?  
//your answer is      1, but correct answer is      4294967297
```

Step1 -開啟範例程式

- 開啟 example 資料夾內的 32mpy.v



```
always @(posedge CLK or posedge RST)
begin
    //初始化數值
    if(RST) begin
        Product <= 64'b0;
        Mplicand <= 64'b0;
        Mplier <= 32'b0;
    end

    //輸入乘數與被乘數
    else if(Counter == 7'd0) begin
        Mplicand <= {32'b0,in_a};
        Mplier <= in_b;
    end

    //乘法與數值移位
    /* write down your design below */
    else if(Counter <= 7'd32)
    begin
        if(Mplier[0] == 1'b1)
            Product <= Mplicand + Product;
            Mplicand <= Mplicand << 1'b1;
            Mplier <= Mplier >> 1'b1;
        end
        /* write down your design upon */
    end
end
```

Step2 - 測試乘法器

- ▶ 開啟 example 資料夾內的 tb32mpy.v
- ▶ 在命令提示字元輸入指令
“iverilog -o test tb32mpy”
“vvp test”
- ▶ 確認 unsigned 32-bit serial multiplier 運算成功

```
// Successful //  
//      30 *      90 = ?  
//your answer is      2700, correct answer is      2700
```

```
// Fail //  
//      30 *      -90 = ?  
//your answer is      128849016180, but correct answer is      -2700
```

```
// Fail //  
//     -30 *      90 = ?  
//your answer is      386547053940, but correct answer is      -2700
```

```
// Fail //  
//     -30 *     -90 = ?  
//your answer is 18446743558313478796, but correct answer is      2700
```

```
// Successful //  
// 4294967295 * 4294967295 = ?  
//your answer is 18446744065119617025, correct answer is 18446744065119617025
```

```
// Fail //  
// 4294967296 *      1 = ?  
//your answer is      0, but correct answer is      4294967296
```

```
// Fail //  
// 4294967297 *      1 = ?  
//your answer is      1, but correct answer is      4294967297
```

範例-32-bit booth multiplier

- ▶ 本範例將以 Verilog 模擬 32-bit booth multiplier，並以 testbench 產生如下圖的測試結果

```
// Successful //  
//          30 *          90 = ?  
//your answer is      2700, correct answer is      2700  
  
// Successful //  
//          30 *          -90 = ?  
//your answer is     -2700, correct answer is     -2700  
  
// Successful //  
//         -30 *          90 = ?  
//your answer is     -2700, correct answer is     -2700  
  
// Successful //  
//         -30 *          -90 = ?  
//your answer is      2700, correct answer is      2700
```


Step1 -開啟範例程式

- 開啟 example 資料夾內的 32mpy_booth.v

| b_i | b_{i-1} | operation |
|-------|-----------|-----------|
| 0 | 0 | 0 |
| 0 | 1 | +A |
| 1 | 0 | -A |
| 1 | 1 | 0 |

```
24 //Multiplier
25 always @(posedge CLK or posedge RST)
26 begin
27     //初始化數值
28     if(RST) begin
29         Product <= 64'b0;
30         Mplicand <= 32'b0;
31         Mythicalbit <= 1'b0;
32         //Mplier <= 32'b0;
33     end
34
35     //輸入乘數與被乘數
36     else if(Counter == 6'd0) begin
37         Mplicand <= in_a;
38         Product <={32'b0,in_b};
39         Mythicalbit <= 1'b0;
40     end
41
42     //乘法與數值移位
43     /* write down your design below */
44     else if(Counter <=7'd32)
45     begin
46
47         if(Product[0]==1'b0 && Mythicalbit ==1'b1) //Product 最低位為0 且 Mythicalbit 為1
48         begin//做加: 把被乘數與 Product左半相加, 存回 Product左半
49             Product = Product + {Mplicand,32'b0};
50         end
51
52         if(Product[0]==1'b1 && Mythicalbit==1'b0)//Product 最低位為1 且 Mythicalbit 為0
53         begin//做減: 把被乘數與 Product左半相減, 存回 Product左半
54             Product = Product - {Mplicand,32'b0};
55         end
56
57         Mythicalbit = Product[0];
58
59         Product = Product >>> 1;
60
61     end
62     /* write down your design upon */
63 end
```

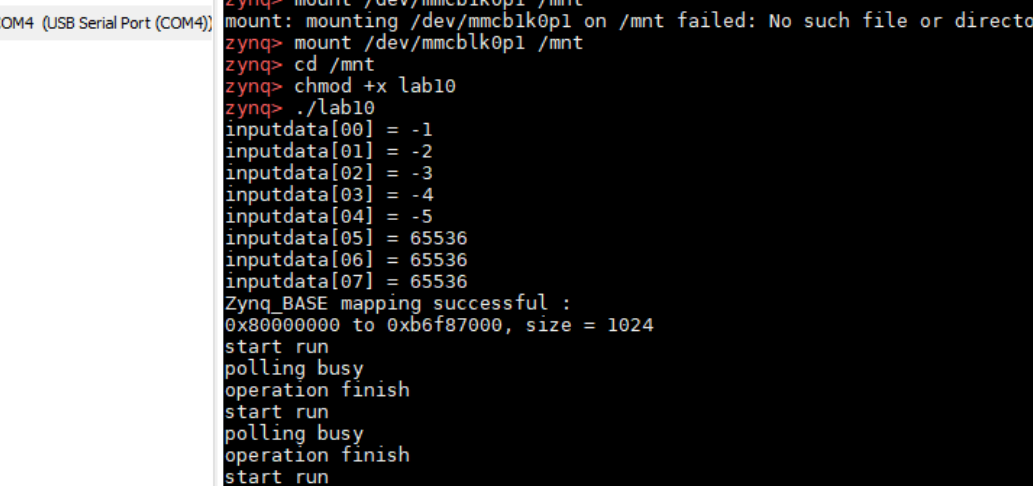
Step2 - 測試乘法器

- ▶ 開啟 example 資料夾內的 tb32mpy_booth.v
- ▶ 在命令提示字元輸入指令
"iverilog -o test tb32mpy_booth.v"
"vvp test"
- ▶ 確認 32-bit booth multiplier 運算成功

```
// Successful //  
//  
//your answer is 30 * 90 = ? 2700, correct answer is 2700  
  
// Successful //  
//  
//your answer is 30 * -90 = ? -2700, correct answer is -2700  
  
// Successful //  
//  
//your answer is -30 * 90 = ? -2700, correct answer is -2700  
  
// Successful //  
//  
//your answer is -30 * -90 = ? 2700, correct answer is 2700
```

範例-booth squarer on FPGA

- ▶ 使用FPGA做平方運算，並取回結果如右圖
- ▶ 請同學打開BOOTH資料夾，將LAB10執行檔放入SD卡，並將Zedboard開機
- ▶ 開啟booth.xpr，將bit檔燒錄至Zedboard
- ▶ 使用mobaXterm執行LAB10，可得結果如右



The screenshot shows the Zynq IDE interface. On the left, there is a sidebar with icons for Sessions, Tools, and Macros. The main area displays a terminal window titled "COM4 (USB Serial Port (COM4))". The terminal shows the following commands and output:

```
zynq> mount /dev/mmcblk0p1 /mnt
mount: mounting /dev/mmcblk0p1 on /mnt failed: No such file or directory
zynq> mount /dev/mmcblk0p1 /mnt
zynq> cd /mnt
zynq> chmod +x lab10
zynq> ./lab10
inputdata[00] = -1
inputdata[01] = -2
inputdata[02] = -3
inputdata[03] = -4
inputdata[04] = -5
inputdata[05] = 65536
inputdata[06] = 65536
inputdata[07] = 65536
Zynq BASE mapping successful :
0x80000000 to 0xb6f87000, size = 1024
start run
polling busy
operation finish
start run
polling busy
operation finish
start run
polling busy
operation finish
start run
polling busy
operation finish
start run
polling busy
operation finish
start run
polling busy
operation finish
start run
polling busy
operation finish
start run
polling busy
operation finish
start run
polling busy
operation finish
start run
polling busy
operation finish
accelerator result[00] = 1
accelerator result[01] = 4
accelerator result[02] = 9
accelerator result[03] = 16
accelerator result[04] = 25
accelerator result[05] = 4294967296
accelerator result[06] = 4294967296
accelerator result[07] = 4294967296
zynq>
```

範例-booth squarer on FPGA(1/3)

- ▶ 本範例將以 booth multiplier做修改，使得乘法器能在FPGA上做平方運算

```
1 module lab(CLK, RST, din,addr, ctrl,Partial_Product, Product_Valid);
2 input CLK, RST;
3 input [2:0]ctrl;
4 input [15:0]addr;
5 input signed [31:0] din;
6 output reg [31:0] Partial_Product;
7 output reg Product_Valid;
8
9 reg signed[31:0] in_a; // Multiplicand
10 reg signed[31:0] in_b;// Multiplier
11
12 reg [31:0] Mplicand;
13 reg signed [63:0] Product;
14 //reg [31:0] Mplier;
15 reg [6:0] Counter ;
16 reg sign;
17 reg Mythicalbit;
18
19 //Counter
20 always@(posedge CLK or negedge RST)
21 begin
22     if(!RST)
23         Counter <= 6'b0;
24     else if(ctrl[0])
25         Counter <= 6'b0;
26     else if(Counter<=6'd33)
27         Counter <= Counter +6'b1;
28 end
29
30 always@(posedge CLK or negedge RST)
31 begin
32     if(!RST) begin
33         in_a <=32'd0;
34         in_b <=32'd0;
35     end
36     else if(addr[15:8]==8'h00)begin
37         in_a <=din;
38         in_b <=din;
39     end
40 end
41
```

```
42 always@(posedge CLK or negedge RST)
43 begin
44     if(!RST)begin
45         Partial_Product<=32'd0;
46     end
47     else if(ctrl[2])begin
48         Partial_Product<=Product[31:0];
49     end
50     else if(ctrl[1])begin
51         Partial_Product<=Product[63:32];
52     end
53 end
54
```

```
100 //判斷輸出
101 always @(posedge CLK or negedge RST)
102 begin
103     if(!RST)
104         Product_Valid <=1'b0;
105     else if(Counter>=6'd32)
106         Product_Valid <=1'b1;
107     else
108         Product_Valid <=1'b0;
109 end
110
```

```
55 //Multiplier
56 always @(posedge CLK or negedge RST)
57 begin
58     if(!RST) begin
59         Product <= 64'b0;
60         Mplicand <= 32'b0;
61         Mythicalbit <= 1'b0;
62         //Mplier <= 32'b0;
63     end
64
65     else if(Counter == 6'd0) begin
66         Mplicand <= in_a;
67         Product <={32'b0,in_b};
68         Mythicalbit <= 1'b0;
69     end
70
71     /* write down your design below */
72     else if(Counter <=7'd32)
73     begin
74         if(Product[0]==1'b0 && Mythicalbit ==1'b1) //Product
75         begin
76             Product = Product + {Mplicand,32'b0};
77         end
78
79         if(Product[0]==1'b1 && Mythicalbit==1'b0)//Product
80         begin/
81             Product = Product - {Mplicand,32'b0};
82         end
83
84         Mythicalbit = Product[0];
85
86         Product = Product >>> 1;
87
88     end
89     /* write down your design upon */
90 end
91
92
93
94
```

範例-booth squarer on FPGA(2/3)

- ▶ 在TOP檔增加控制信號

```
67 assign ctrl = (hwrite_reg && (haddr_reg == 32'h80000100)) ? hwdata[2:0] : 3'b000;
```

- ▶ 選擇要寫入什麼資料

```
86 assign hrddata_es1 = (haddr_reg == 32'h8000_0104) ? {31'h0000_0000, ready} : dout;  
87 assign hreadyout_es1 = (cen_wait == 1'b0) ? 1'b1:1'b0;  
88 assign hresp_es1      = 2'b00;
```

- ▶ 加入booth squarer

```
100 lab lab(  
101     .CLK(g_hclk_es1),  
102     .RST(hreset_n),  
103     .ctrl(ctrl),  
104     .addr(haddr_reg[15:0]),  
105     .din(hwdata[31:0]),  
106     .Partial_Product(dout),  
107     .Product_Valid(ready)  
108 );  
109
```

範例-booth squarer on FPGA(3/3)

► 對應TOP檔所準備之C Code

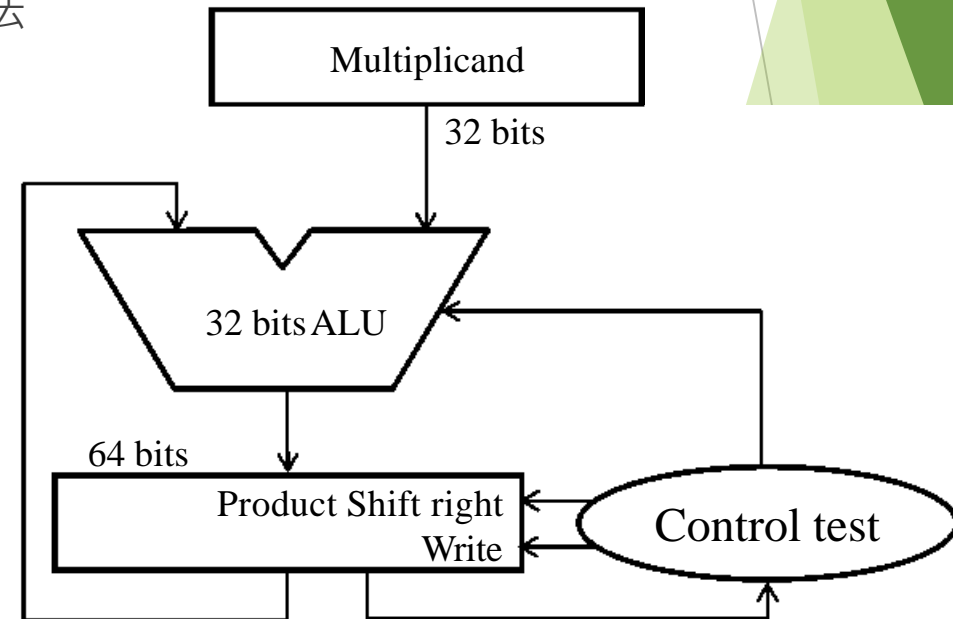
```
8  #define ZYNQ_BASE          0x80000000  //base address for 64 sequence of integer
9  #define CTRL_OFFSET        0x00000100  //for ctrl signal c
10 #define STATUS_OFFSET      0x00000104  //for status signal

80  for(i = 0 ; i < DEPTH ; i++){
81      *(io + i) = A[i];
82      printf("start run\n");
83      *(io+(CTRL_OFFSET>>2)) = 0x1;
84      *(io+(CTRL_OFFSET>>2)) = 0x0;
85
86      printf("polling busy\n");
87
88      while ( *(io+(STATUS_OFFSET >> 2))==(volatile unsigned int)0)
89      {
90          printf("0x%x\n",*(io+(STATUS_OFFSET >> 2)));
91      }
92
93      printf("operation finish\n");
94
95      *(io+(CTRL_OFFSET>>2)) = 0x2;
96      *(io+(CTRL_OFFSET>>2)) = 0x0;
97
98      result[i] = *(io+i);
99      result[i] = result[i]<<32;
100
101      *(io+(CTRL_OFFSET>>2)) = 0x4;
102      *(io+(CTRL_OFFSET>>2)) = 0x0;
103
104      result[i] = result[i] + *(io+i);
105  }
```

作業說明

- ▶ 作業Part1：按照範例操作並以 testbench 成功測試 unsigned 32-bit serial multiplier及 32-bit booth multiplier
- ▶ 作業Part2：參考範例與附錄，完成 unsigned 32-bit optimized serial multiplier，並使用 testbench (tb32mpy_opt.v) 測試以下幾筆乘法

1. 30×90
2. 30×-90
3. -30×90
4. -30×-90
5. $4294967295 \times 4294967295$
6. 4294967296×1
7. 4294967297×1



作業說明

- 作業Part3：參考下表完成 32-bit modified booth multiplier，並使用 testbench(tb32mpy_booth_mod.v) 測試以下幾筆乘法

1. 30×90
2. 30×-90
3. -30×90
4. -30×-90

| b_{i+1} | b_i | b_{i-1} | operation |
|-----------|-------|-----------|-----------|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | +A |
| 0 | 1 | 0 | +A |
| 0 | 1 | 1 | +2A |
| 1 | 0 | 0 | -2A |
| 1 | 0 | 1 | -A |
| 1 | 1 | 0 | -A |
| 1 | 1 | 1 | 0 |

作業說明

- ▶ 作業Part4：修改提供之Vivado專案將乘法器換成本週之32-bit modified booth multiplier，修改C Code傳入以下數字並平方：

1. 30
2. 59
3. 90
4. 125
5. -30
6. -59
7. -90
8. -125

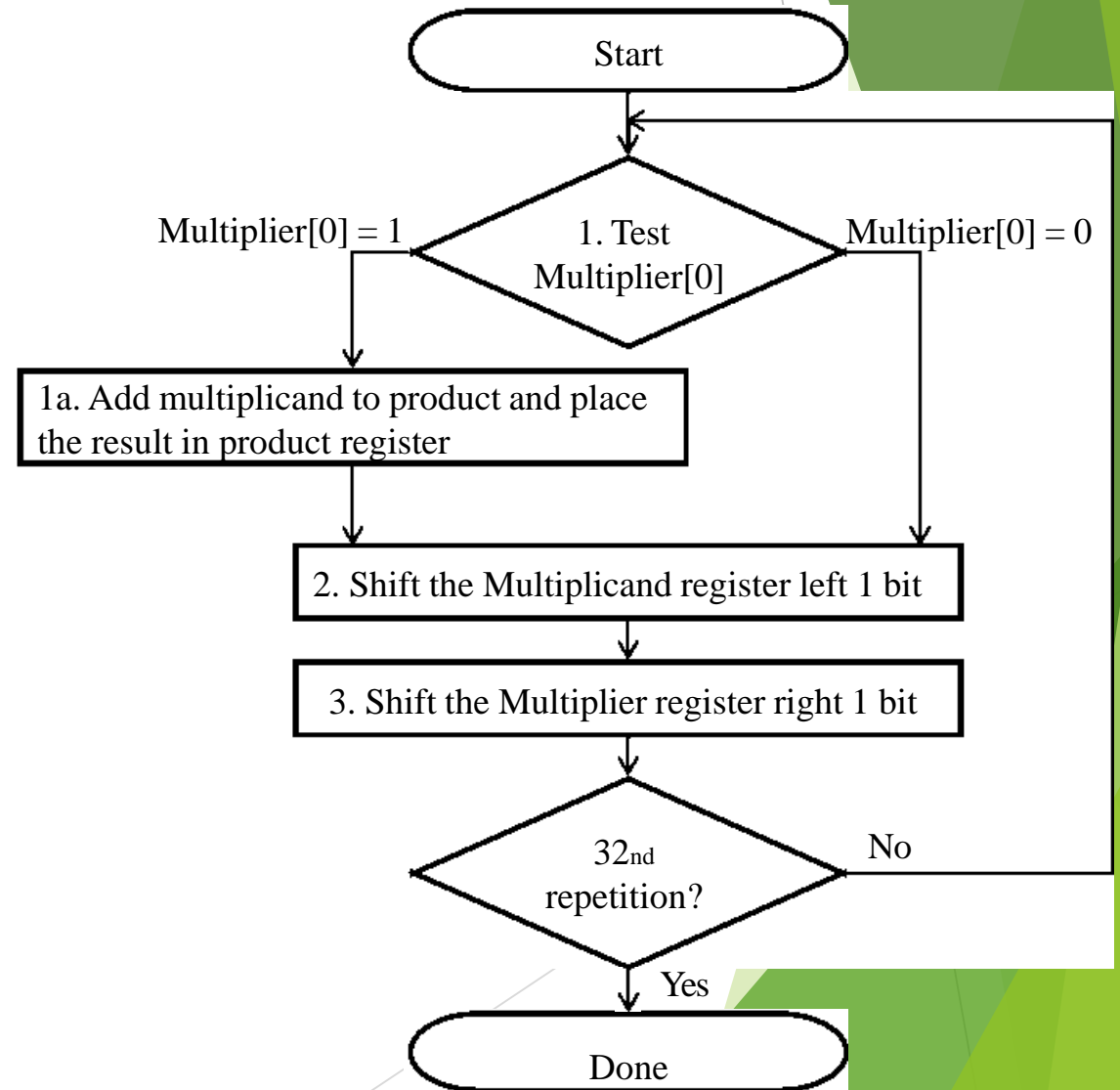
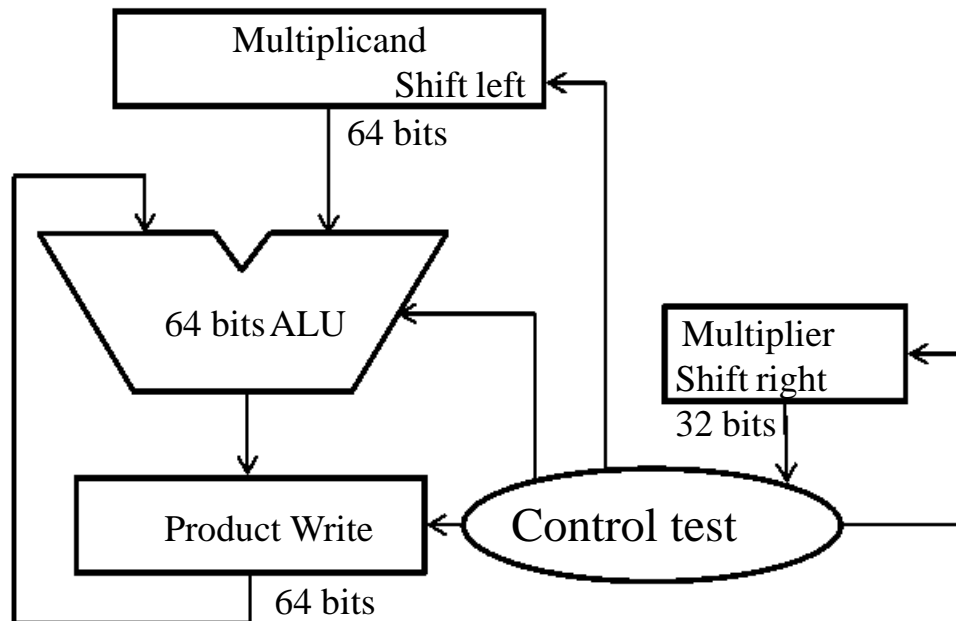
ANS: 900,3481,8100,15625

課程評分

- ▶ Demo 方式：Demo 開始時助教會公布隨堂測驗考題，請於時限內呈現作業並作答考題
- ▶ Demo 日期：5/20，5/22
- ▶ Demo 地點：工程一館206

- ▶ 評分方式：作業80%，隨堂測試20%
- ▶ 作業分數配分：part1 20%，part2 20%，part3 20%，part4 20%

附錄 - Serial Multiplier



附錄 - Optimized Serial Multiplier

