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

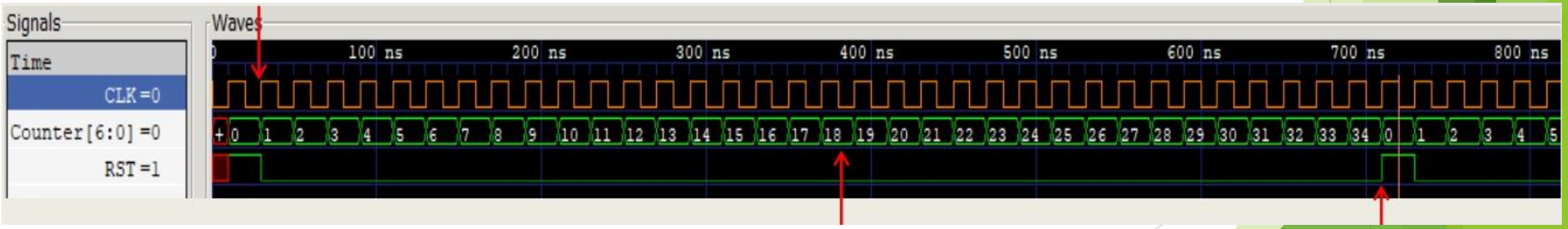
課程目的

- 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

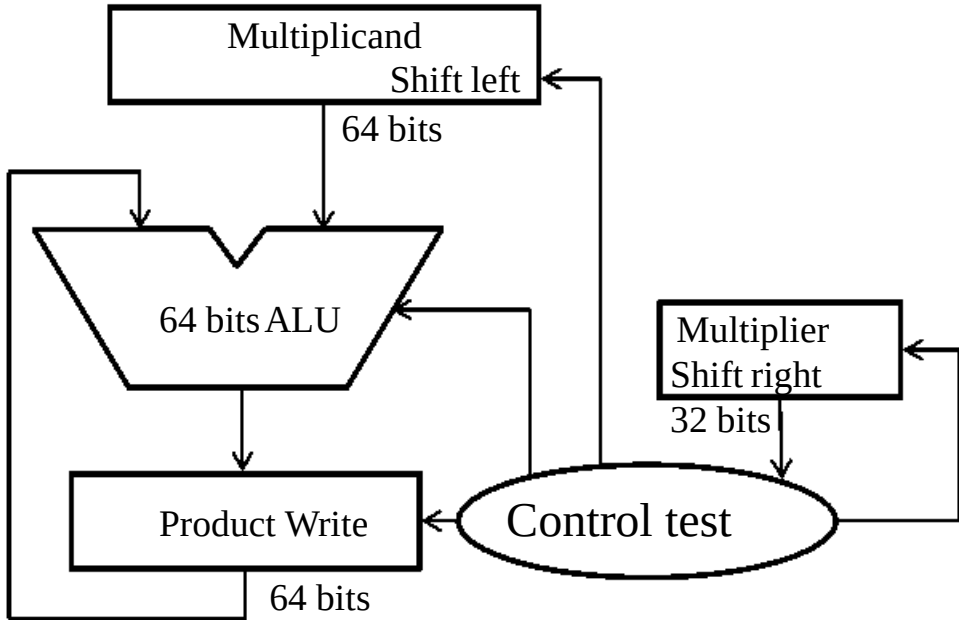
範例 – Unsigned 32-bit Serial Multiplier

- 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
```

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

```
//Multiplier
always @(posedge CLK or posedge RST)
begin
    //初始化數值
    if(RST) begin
        Product  <= 64'b0;
        Mplicand <= 32'b0;
        Mythicalbit <= 1'b0;
        //Mplier   <= 32'b0;
    end

    //輸入乘數與被乘數
    else if(Counter == 6'd0) begin
        Mplicand <= in_a;
        Product <={32'b0,in_b};
        Mythicalbit <= 1'b0;
    end

    //乘法與數值移位
    /* write down your design below */
    else if(Counter <=7'd32)
    begin

        if(Product[0]==1'b0 && Mythicalbit ==1'b1) //Product 最低位為0 且 Mythicalbit 為1
        begin//做加: 把被乘數與 Product左半相加, 存回 Product左半
            Product = Product + {Mplicand,32'b0};
        end

        if(Product[0]==1'b1 && Mythicalbit==1'b0)//Product 最低位為1 且 Mythicalbit 為0
        begin//做減: 把被乘數與 Product左半相減, 存回 Product左半
            Product = Product - {Mplicand,32'b0};
        end

        Mythicalbit = Product[0];

        Product = Product >>> 1;

    end
    /* write down your design upon */
end
```

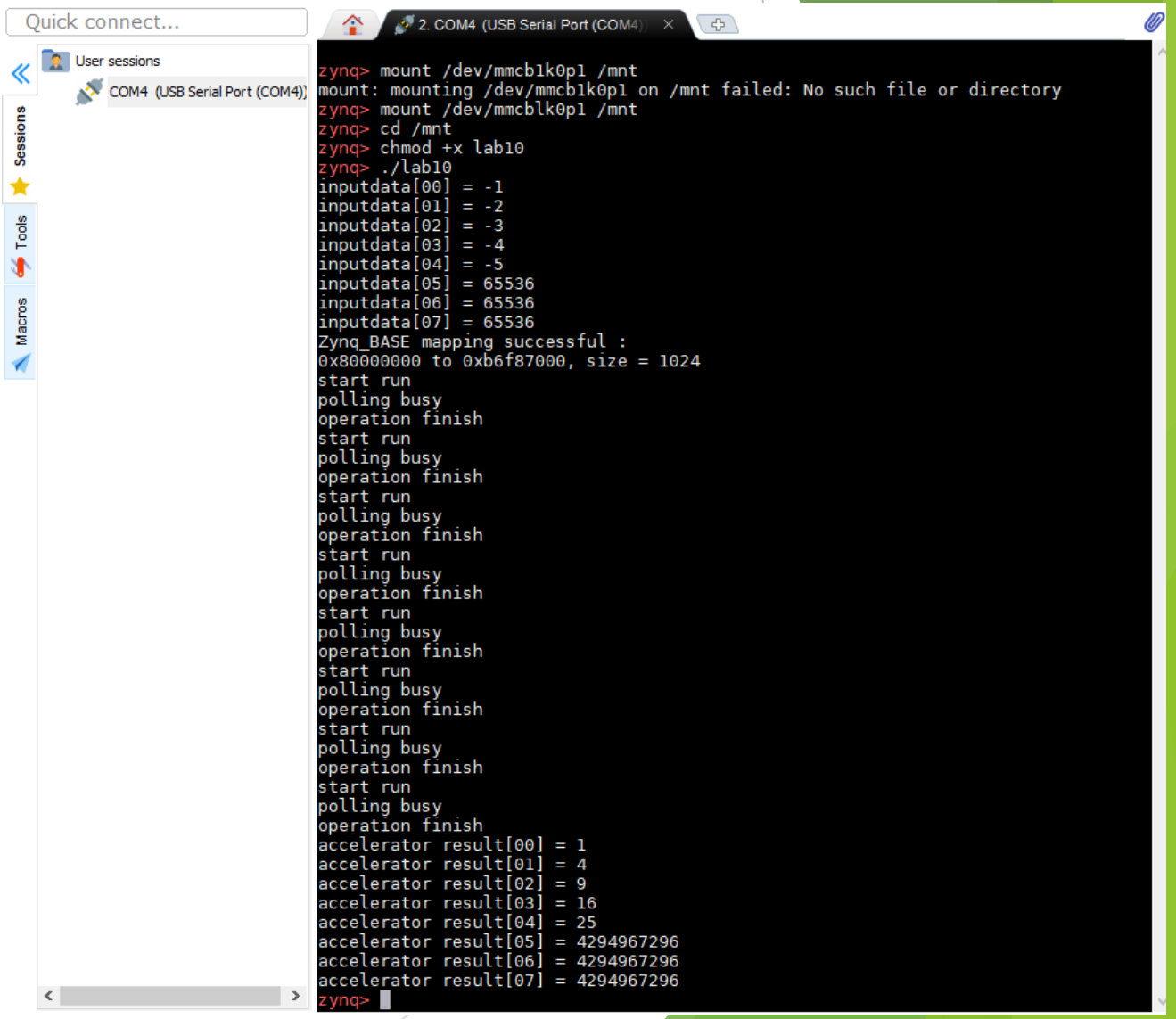
Step2 – 測試乘法器

- 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 //
//                    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
```

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



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

```
module lab(CLK, RST, din,addr, ctrl,Partial_Product, Product_Valid);
input CLK, RST;
input [2:0]ctrl;
input [15:0]addr;
input signed [31:0] din;
output reg [31:0] Partial_Product;
output reg Product_Valid;

reg signed[31:0]    in_a;   // Multiplicand
reg signed[31:0]    in_b;// Multiplier

reg [31:0] Mplicand;
reg signed [63:0] Product;
//reg [31:0] Mplier;
reg [6:0]   Counter ;
reg sign;
reg Mythicalbit;

//Counter
always@(posedge CLK or negedge RST)
begin
    if(!RST)
        Counter <=6'b0;
    else if(ctrl[0])
        Counter <=6'b0;
    else if(Counter<=6'd33)
        Counter <= Counter +6'b1;
end

always@(posedge CLK or negedge RST)
begin
    if(!RST) begin
        in_a <=32'd0;
        in_b <=32'd0;
    end
    else if(addr[15:8]==8'h00)begin
        in_a <=din;
        in_b <=din;
    end
end
```

```
always@(posedge CLK or negedge RST)
begin
    if(!RST)begin
        Partial_Product<=32'd0;
    end
    else if(ctrl[2])begin
        Partial_Product<=Product[31:0];
    end
    else if(ctrl[1])begin
        Partial_Product<=Product[63:32];
    end
end
```

```
//判斷輸出
always @(posedge CLK or negedge RST)
begin
    if(!RST)
        Product_Valid <=1'b0;
    else if(Counter>=6'd32)
        Product_Valid <=1'b1;
    else
        Product_Valid <=1'b0;
end
```

```
//Multiplier
always @(posedge CLK or negedge RST)
begin

    if(!RST) begin
        Product  <= 64'b0;
        Mplicand <= 32'b0;
        Mythicalbit <= 1'b0;
        //Mplier   <= 32'b0;
    end


    else if(Counter == 6'd0) begin
        Mplicand <= in_a;
        Product <={32'b0,in_b};
        Mythicalbit <= 1'b0;
    end


    /* write down your design below */
    else if(Counter <=7'd32)
    begin

        if(Product[0]==1'b0 && Mythicalbit ==1'b1) //Product
        begin
            Product = Product + {Mplicand,32'b0};
        end

        if(Product[0]==1'b1 && Mythicalbit==1'b0)//Product
        begin/
            Product = Product - {Mplicand,32'b0};
        end

        Mythicalbit = Product[0];

        Product = Product >>> 1;

    end
    /* write down your design upon */
end
```

範例-booth squarer on FPGA(2/3)

- 在TOP檔增加控制信號

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

- 選擇要寫入什麼資料

```
assign hrdata_esl = (haddr_reg == 32'h8000_0104)? {31'h0000_0000, ready} : dout;
assign hreadyout_esl = (cen_wait == 1'b0)? 1'b1:1'b0;
assign hresp_esl     = 2'b00;
```

- 加入booth squarer

```
lab lab(
    .CLK(g_hclk_esl),
    .RST(hreset_n),
    .ctrl(ctrl),
    .addr(haddr_reg[15:0]),
    .din(hwdata[31:0]),
    .Partial_Product(dout),
    .Product_Valid(ready)
);
```

範例-booth squarer on FPGA(3/3)

- 對應TOP檔所準備之C Code

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

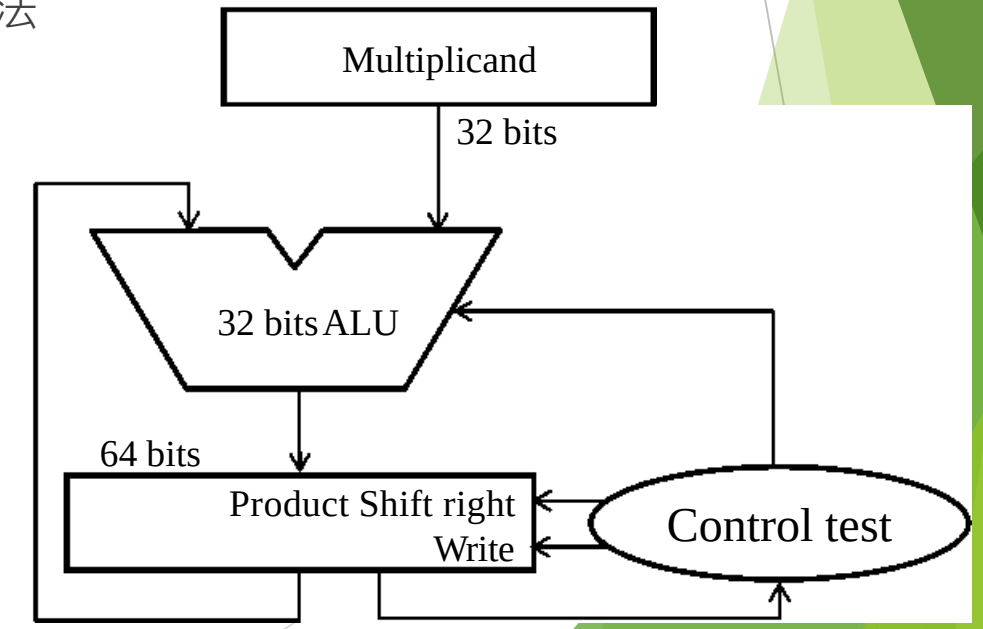
```c
    for(i = 0 ; i < DEPTH ; i++){
        *(io + i) = A[i];
        printf("start run\n");
        *(io+(CTRL_OFFSET>>2)) = 0x1;
        *(io+(CTRL_OFFSET>>2)) = 0x0;

        printf("polling busy\n");

        while ( *(io+(STATUS_OFFSET >> 2))==(volatile unsigned int)0)
        {
            printf("0x%x\n",*(io+(STATUS_OFFSET >> 2)));
        }

        printf("operation finish\n");

        *(io+(CTRL_OFFSET>>2)) = 0x2;
        *(io+(CTRL_OFFSET>>2)) = 0x0;

        result[i] = *(io+i);
        result[i] = result[i]<<32;

        *(io+(CTRL_OFFSET>>2)) = 0x4;
        *(io+(CTRL_OFFSET>>2)) = 0x0;

        result[i] = result[i] + *(io+i);
    }
```

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 × 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

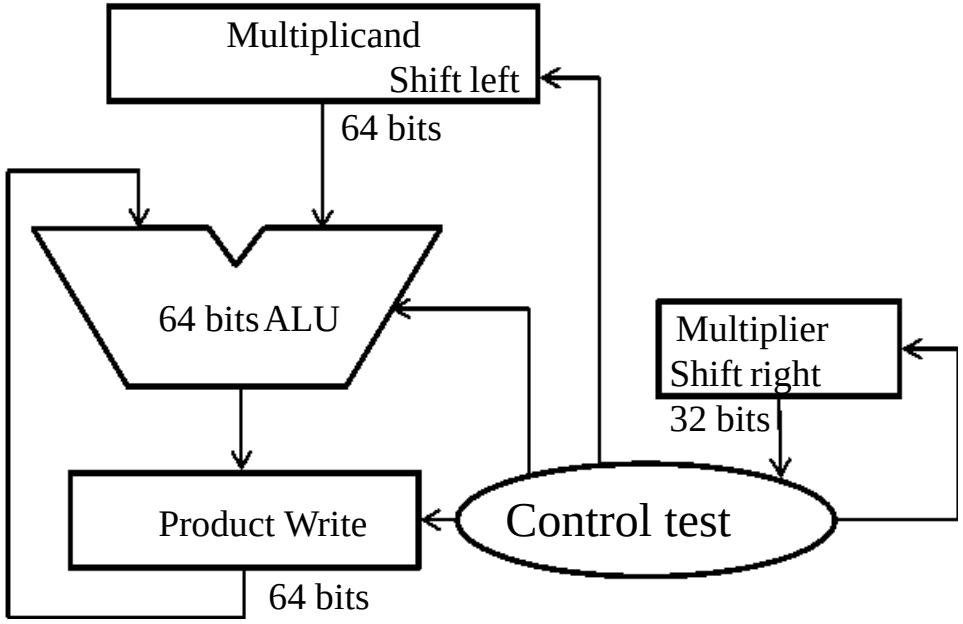
ANS: 900,3481,8100,15625

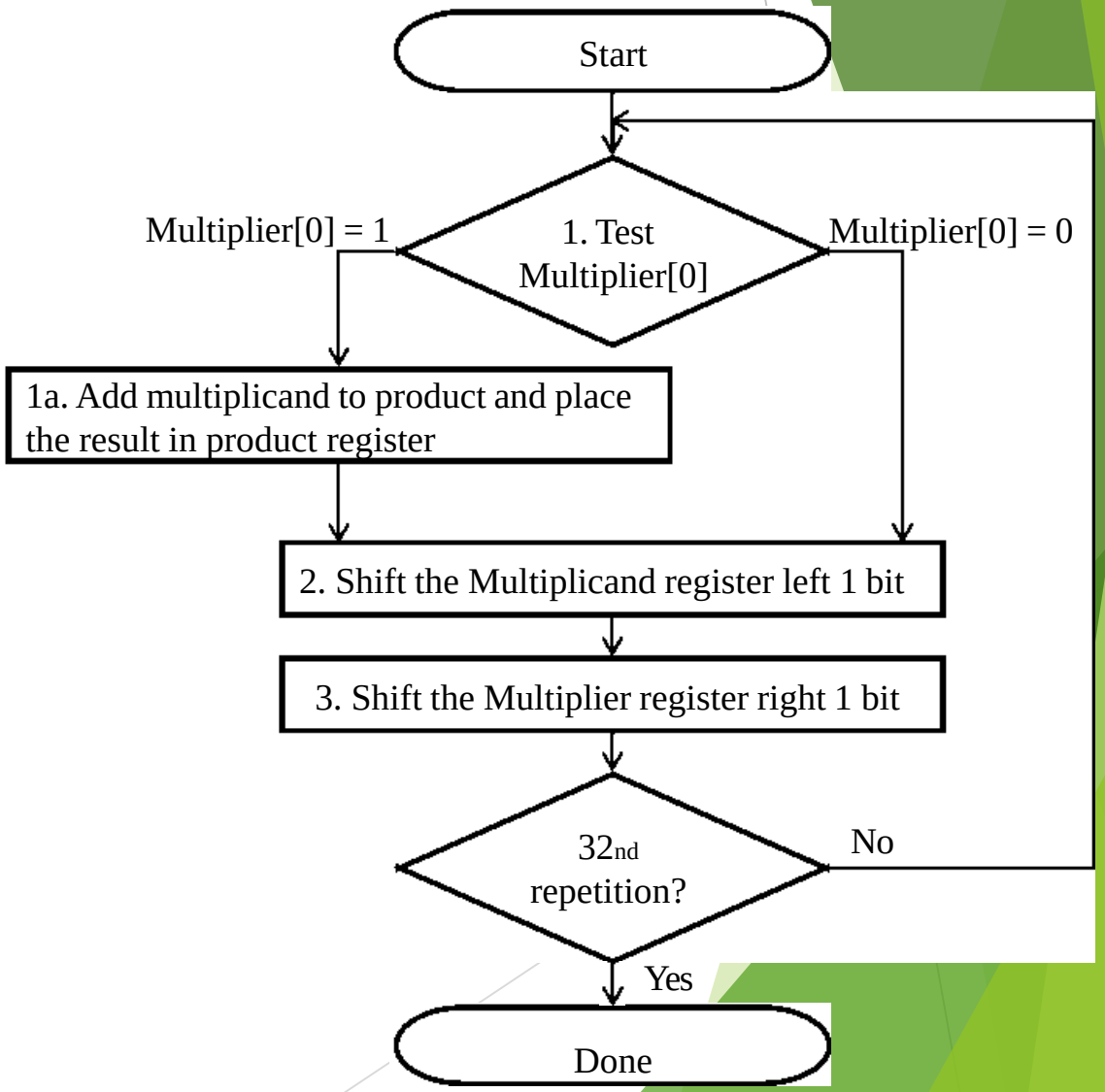
課程評分

- 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