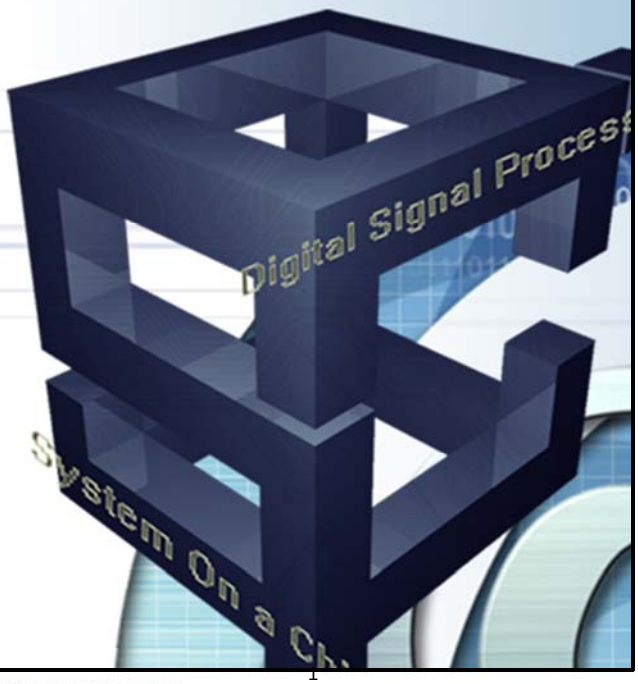
Verilog HDL設計範例



授課老師：黃穎聰 老師
TA：
徐偉傑 LAB723

Outline

1. 八位元暫存器
2. 雙向輸入輸出腳暫存器
3. 資料選擇系統
4. 存入位址控制系統
5. 堆疊系統
6. 程式計數器系統之一
7. 程式計數器系統之二

1. 八位元暫存器
2. 雙向輸入輸出腳暫存器
3. 資料選擇系統
4. 存入位址控制系統
5. 堆疊系統
6. 程式計數器系統之一
7. 程式計數器系統之二

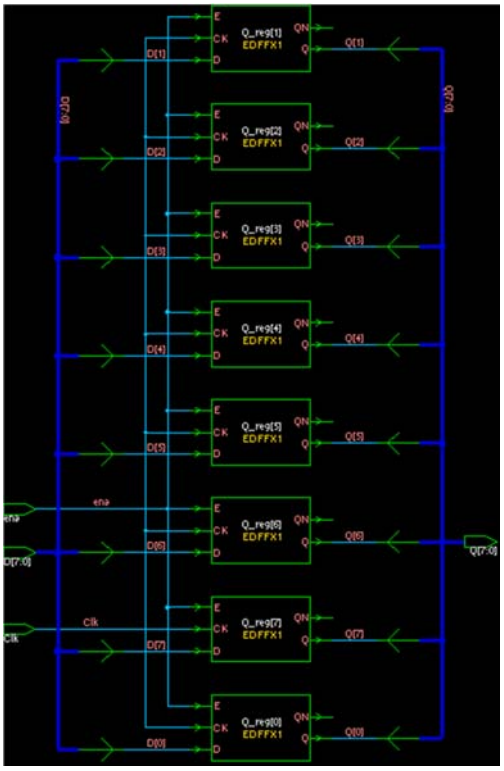
八位元暫存器-1/2

Verilog HDL Code：

```
module reg_8vl (D, Clk, ena, Q);
   input    [7:0] D;
   input    Clk, ena;
   output   [7:0] Q;
   reg      [7:0] Q;
   always @(posedge Clk )
   begin
    if (ena)
        Q <= D;
    end
endmodule
```

Symbol view：

Schematic-1：



八位元暫存器-2/2

Verilog HDL Code：

```
module reg_8v1 (D, Clk, ena, Q);
   input    [7:0] D;
   input    Clk, ena;
   output   [7:0] Q;
   reg      [7:0] Q;
   always @(posedge Clk )
   begin
    if (ena)
        Q <= D;
    end
endmodule
```

Schematic-2：

註：此圖為其中之1 bit暫存器電路圖，QN Pin為浮接。

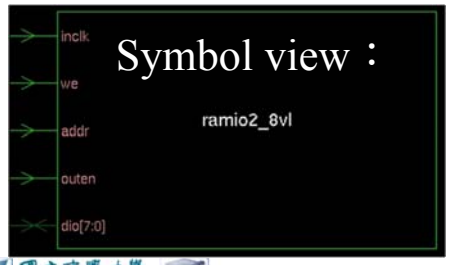
1. 八位元暫存器
2. 雙向輸入輸出腳暫存器
3. 資料選擇系統
4. 存入位址控制系統
5. 堆疊系統
6. 程式計數器系統之一
7. 程式計數器系統之二

雙向輸入輸出腳暫存器-1/3

Verilog HDL Code(part 1)：

```
module ramio2_8vl (dio,inclk,we,addr,outen);
   inout   [7:0] dio;
   input   inclk, we, addr, outen;
   reg     [7:0] Q0, Q1;
   reg     [7:0] ramo;
   reg     addrtem;

   always @(posedge inclk )
    begin
          if (we)
           begin
             case (addr)
                 0 : Q0 <= dio;
                 1 : Q1 <= dio;
             endcase
           end
   end
```

Symbol view：



Verilog HDL Code(part 2)：

```
 always @(posedge inclk)
  begin
   addrtem <= addr;
  end

 always @(addrtem or Q0 or Q1)
  begin
    case (addrtem)
      0 :  ramo = Q0;
      1 :  ramo = Q1;
    endcase
 end

 bufif1 (dio[0], ramo[0], outen);
 bufif1 (dio[1], ramo[1], outen);
 bufif1 (dio[2], ramo[2], outen);
 bufif1 (dio[3], ramo[3], outen);
 bufif1 (dio[4], ramo[4], outen);
 bufif1 (dio[5], ramo[5], outen);
 bufif1 (dio[6], ramo[6], outen);
 bufif1 (dio[7], ramo[7], outen);
endmodule
```

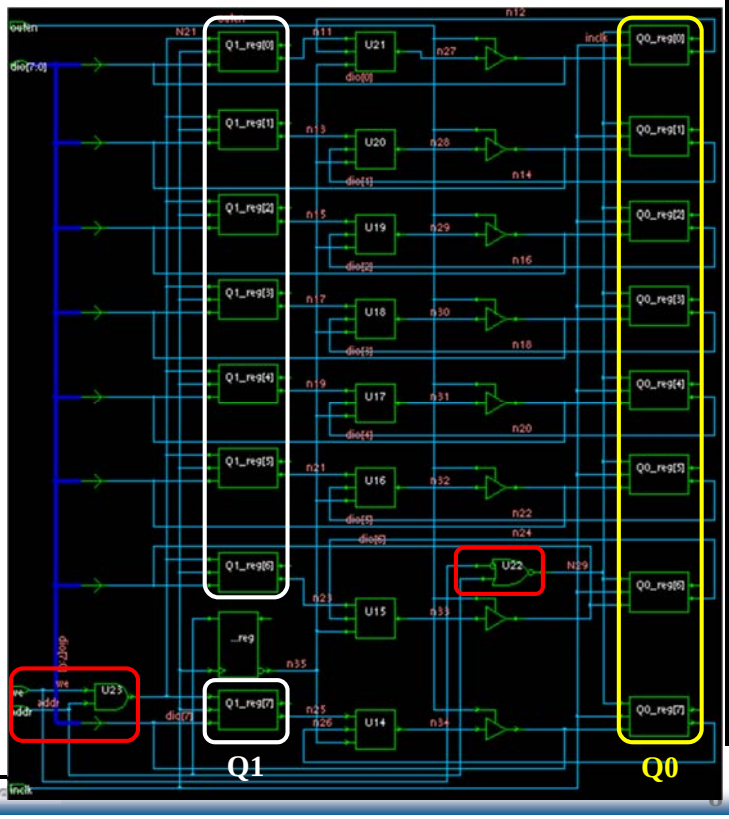
We: determine the direction
Addr: determine Q0 or Q1

雙向輸入輸出腳暫存器-2/3

Verilog HDL Code(part 1)：

```
module ramio2_8vl (dio,inclk,we,addr,outen);
   inout   [7:0] dio;
   input   inclk, we, addr, outen;
   reg     [7:0] Q0, Q1;
   reg     [7:0] ramo;
   reg     addrtem;

   always @(posedge inclk )
    begin
          if (we)
           begin
             case (addr)
                 0 : Q0 <= dio;
                 1 : Q1 <= dio;
             endcase
           end
   end
```

Schematic-1：



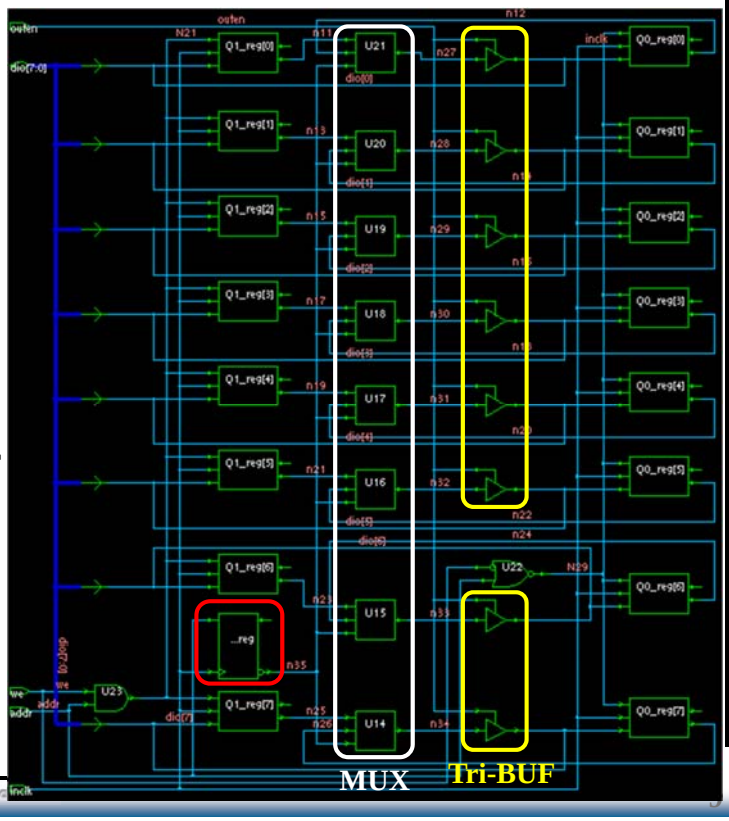
雙向輸入輸出腳暫存器-3/3

Verilog HDL Code(part 2)： Schematic-2：

```
always @(posedge inclk)
 begin
  addrtem <= addr;
 end

always @(addrtem or Q0 or Q1)
 begin
   case (addrtem)
     0 :  ramo = Q0;
     1 :  ramo = Q1;
   endcase
 end

bufif1 (dio[0], ramo[0], outen);
bufif1 (dio[1], ramo[1], outen);
bufif1 (dio[2], ramo[2], outen);
bufif1 (dio[3], ramo[3], outen);
bufif1 (dio[4], ramo[4], outen);
bufif1 (dio[5], ramo[5], outen);
bufif1 (dio[6], ramo[6], outen);
bufif1 (dio[7], ramo[7], outen);
endmodule
```

MUX

Tri-BUF



1. 八位元暫存器
2. 雙向輸入輸出腳暫存器
3. 資料選擇系統
4. 存入位址控制系統
5. 堆疊系統
6. 程式計數器系統之一
7. 程式計數器系統之二

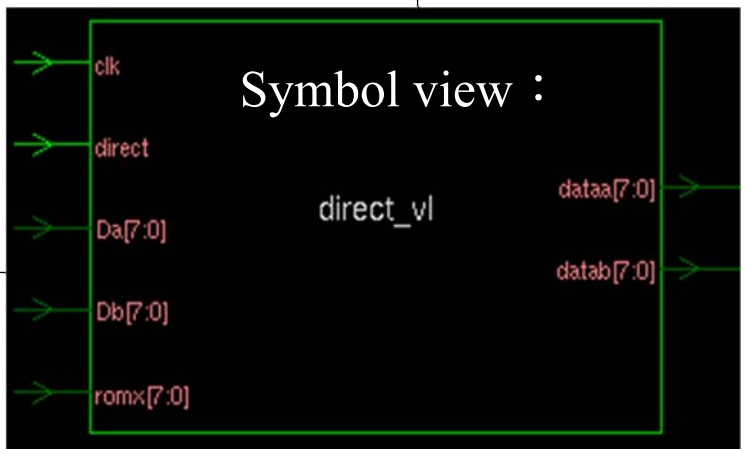
資料選擇系統-1/2

Verilog HDL Code：

```
module direct_vl (Da, Db, romx, clk, direct, dataa, datab);
input [7:0]  Da, Db, romx;
input  clk, direct;
output [7:0] dataa, datab;
reg    [7:0] dataa, datab;
always @(posedge clk)
  begin
    dataa <= Da;
  end
always @(posedge clk)
  begin
    case (direct)
      0 : datab <= Db;
      1:  datab <= romx;
    endcase
  end
endmodule
```

Da -> dataa
Db / romx -> datab

Symbol view：



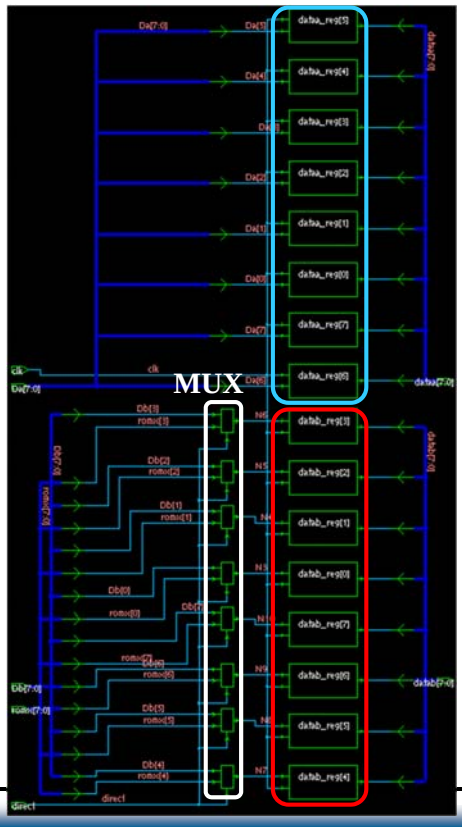
資料選擇系統-2/2

Verilog HDL Code：

```
module direct_vl (Da, Db, romx, clk, direct, dataa, datab);
input [7:0]  Da, Db, romx;
input  clk, direct;
output [7:0] dataa, datab;
reg    [7:0] dataa, datab;
always @(posedge clk)
  begin
    dataa <= Da;
  end
always @(posedge clk)
  begin
    case (direct)
      0 : datab <= Db;
      1:  datab <= romx;
    endcase
  end
endmodule
```

MUX

Schematic：



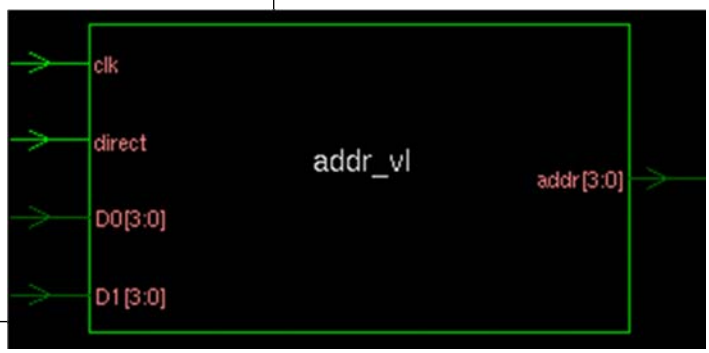
1. 八位元暫存器
2. 雙向輸入輸出腳暫存器
3. 資料選擇系統
4. 存入位址控制系統
5. 堆疊系統
6. 程式計數器系統之一
7. 程式計數器系統之二

存入位址控制系統-1/2

Verilog HDL Code：

```
module addr_vl (D0, D1, clk, direct, addr);
input [3:0]  D0, D1;
input clk, direct;
output [3:0] addr;
reg [3:0] addr, temp1, temp2;

 always @(direct or D0 or D1)
  begin
      case (direct)
         0 :  temp1 = D0;
         1 :  temp1 = D1;
      endcase
  end
 always @(posedge clk)
  begin
      addr <= temp2;
      temp2 <= temp1;
  end
endmodule
```

Symbol view：

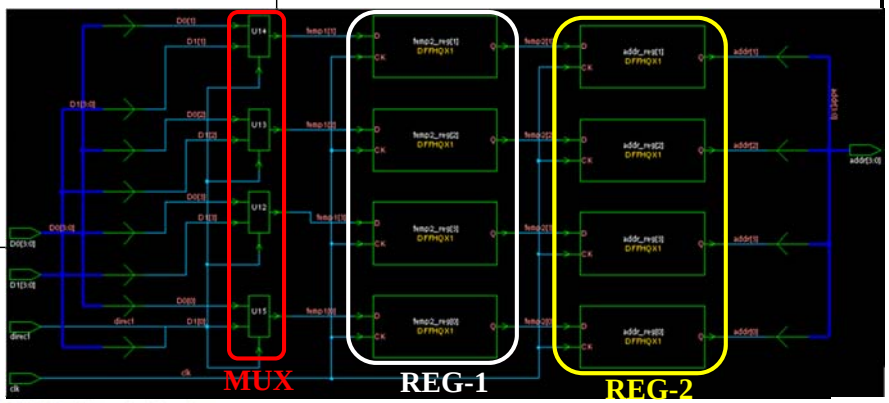


存入位址控制系統-2/2

Verilog HDL Code：

```
module addr_v1 (DO, D1, clk, direct, addr);
input [3:0]  DO, D1;
input clk, direct;
output [3:0] addr;
reg [3:0] addr, temp1, temp2;

 always @(direct or DO or D1)
  begin
      case (direct)        MUX
          0 :  temp1 = DO;
          1 :  temp1 = D1;
      endcase
  end
 always @(posedge clk)
  begin                    REG-2
     addr <= temp2;
     temp2 <= temp1;
  end                      REG-1
endmodule
```

Schematic：



1. 八位元暫存器
2. 雙向輸入輸出腳暫存器
3. 資料選擇系統
4. 存入位址控制系統
5. 堆疊系統
6. 程式計數器系統之一
7. 程式計數器系統之二

堆疊系統-1/4

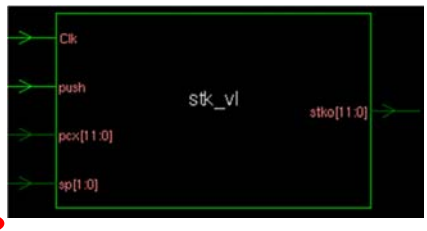
Verilog HDL Code：

```
module stk_vl (pcx, Clk, push, sp, stko);
  input   [11:0] pcx;
  input   Clk, push;
  input   [1:0] sp;
  output  [11:0] stko;
  reg     [11:0] Q0, Q1, Q2, Q3;
  reg     [11:0] stko;
  always @(posedge Clk )
   begin
        if (push)
         begin
           case (sp)
               0 : Q0 <= pcx;
               1 : Q1 <= pcx;
               2 : Q2 <= pcx;
               3 : Q3 <= pcx;
           endcase
         end
   end
  always @(sp or Q0 or Q1 or Q2 or Q3)
  begin
    if (sp == 0)
     stko = Q0;
    else if (sp == 1)
     stko = Q1;
    else if (sp == 2)
     stko = Q2;
    else
     stko = Q3;
  end
endmodule
```

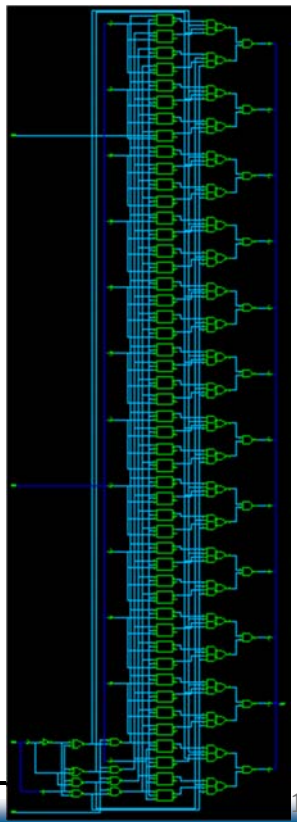
Sequential logic

Combinational logic

Symbol view：



Schematic：

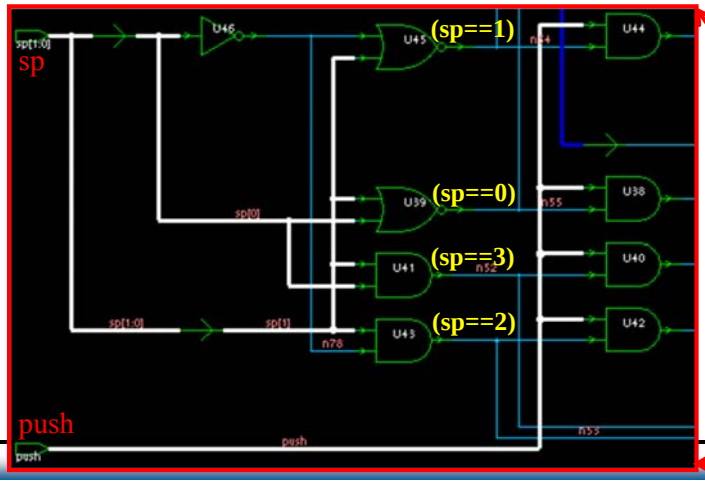


堆疊系統-2/4

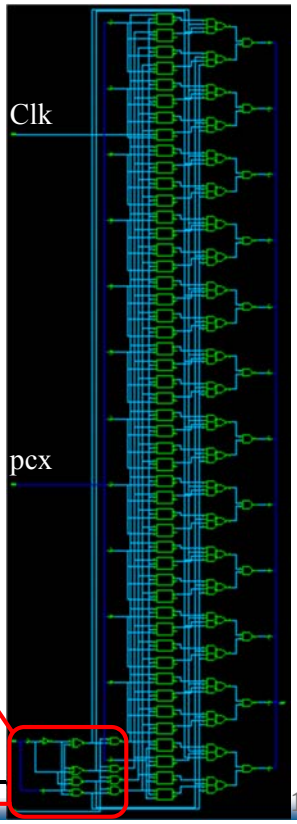
Verilog HDL Code(part1)：

```
  always @(posedge Clk )
   begin
        if (push)
          begin
            case (sp)
                0 : Q0 <= pcx;
                1 : Q1 <= pcx;
                2 : Q2 <= pcx;
                3 : Q3 <= pcx;
            endcase
          end
   end
```

Decoder logic



Schematic：



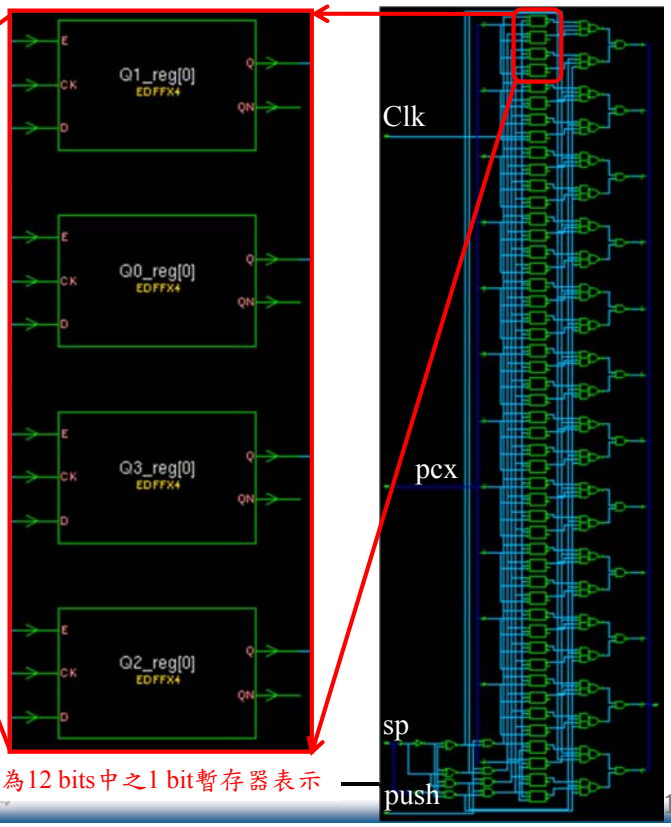
堆疊系統-3/4

Verilog HDL Code(part2)：

```
always @(posedge Clk )
 begin
      if (push)
       begin
         case (sp)
             0 : Q0 <= pcx;
             1 : Q1 <= pcx;
             2 : Q2 <= pcx;
             3 : Q3 <= pcx;
         endcase
       end
 end
```

Register

Schematic：



註：此圖為12 bits中之1 bit暫存器表示

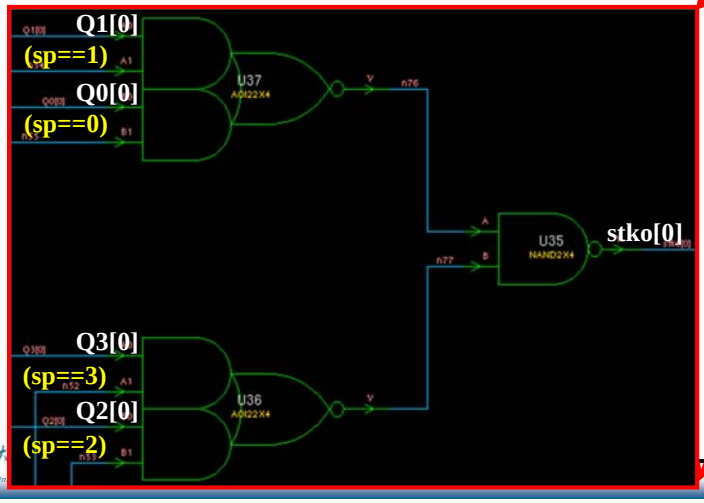
堆疊系統-4/4

Verilog HDL Code(part3)：

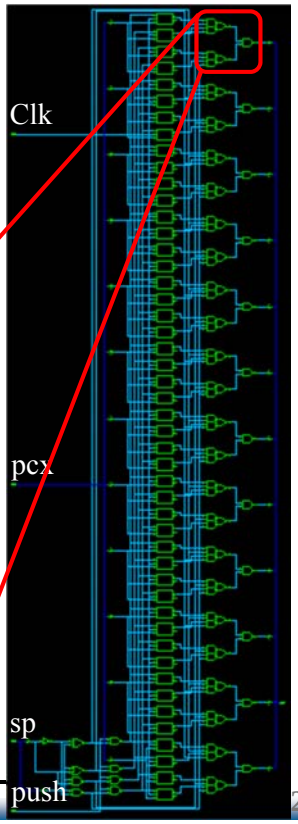
```
always @(sp or Q0 or Q1 or Q2 or Q3)
 begin
   if (sp == 0)
    stko = Q0;
   else if (sp == 1)
    stko = Q1;
   else if (sp == 2)
    stko = Q2;
   else
    stko = Q3;
 end
endmodule
```

4:1 Multiplexer

↓註：此圖為12 bits的**stko**中之1 bit表示



Schematic：



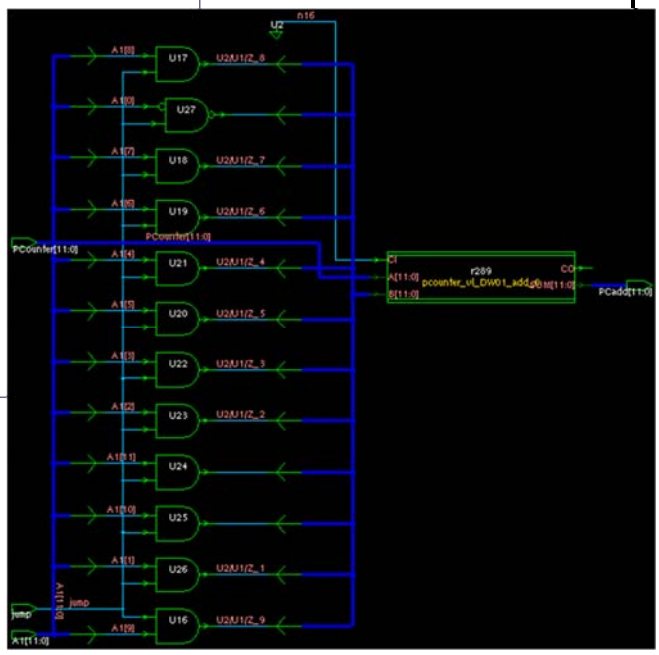
1. 八位元暫存器
2. 雙向輸入輸出腳暫存器
3. 資料選擇系統
4. 存入位址控制系統
5. 堆疊系統
6. 程式計數器系統之一
7. 程式計數器系統之二

程式計數器系統之一-1/3

Verilog HDL Code：

```
module pcounter_vl (PCounter, A1, jump, PCadd);
 input    [11:0] PCounter, A1;
 input    jump;
 output   [11:0] PCadd;
 reg      [11:0] PCadd;
 always @(jump or A1)
  begin
        if (jump == 0)
          PCadd = PCounter + 1;
        else
          PCadd = PCounter + A1;
  end
endmodule
```

Schematic：



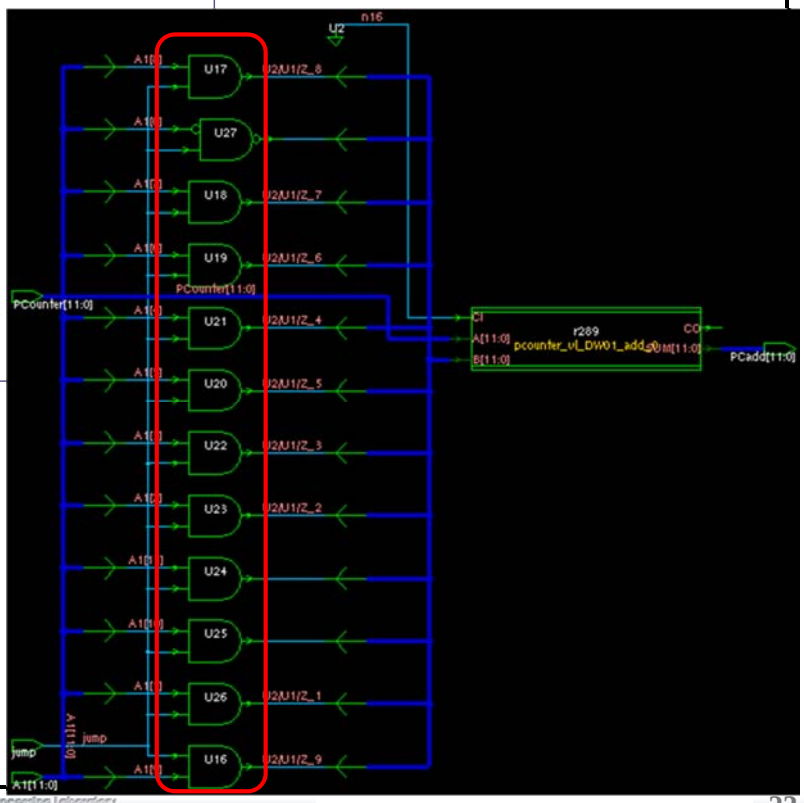
Symbol view：

程式計數器系統之一-2/3

Verilog HDL Code：

```
module pcounter_v1 (PCounter, A1, jump, PCadd);
 input    [11:0] PCounter, A1;
 input    jump;
 output   [11:0] PCadd;
 reg      [11:0] PCadd;
 always @(jump or A1)
  begin
        if (jump == 0)
          PCadd = PCounter + 1;
        else
          PCadd = PCounter + A1;
  end
endmodule
```

Schematic：



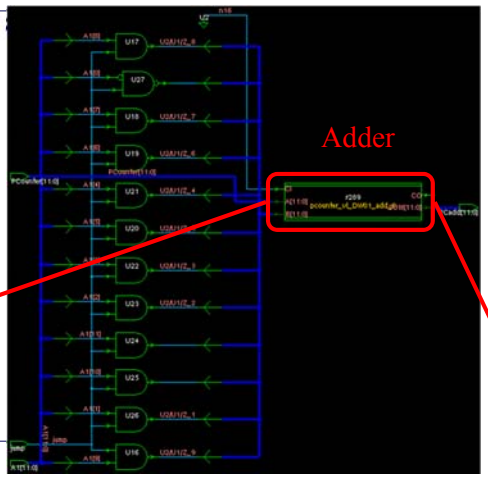
程式計數器系統之一-3/3

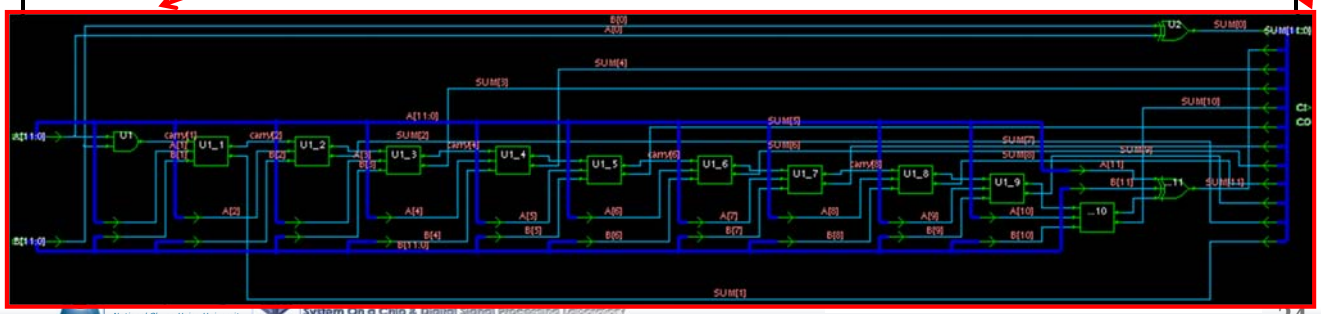
Verilog HDL Code：

```
module pcounter_v1 (PCounter, A1, jump, PCadd);
 input    [11:0] PCounter, A1;
 input    jump;
 output   [11:0] PCadd;
 reg      [11:0] PCadd;
 always @(jump or A1)
  begin
        if (jump == 0)
          PCadd = PCounter + 1;
        else
          PCadd = PCounter + A1;
  end
endmodule
```

Adder

Schematic：





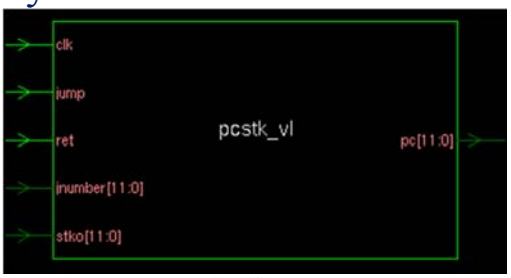
1. 八位元暫存器
2. 雙向輸入輸出腳暫存器
3. 資料選擇系統
4. 存入位址控制系統
5. 堆疊系統
6. 程式計數器系統之一
7. 程式計數器系統之二

程式計數器系統之二-1/3

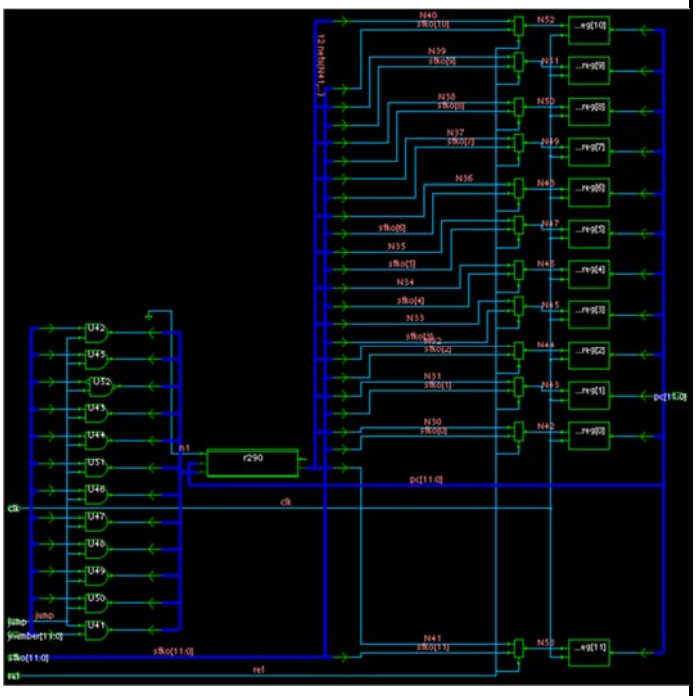
Verilog HDL Code：

```
module pcstk_vl (clk,jump,ret,jnumber,stko,pc);
 input   [11:0] jnumber, stko;
 input   clk, jump, ret;
 output  [11:0] pc;
 reg     [11:0] pc;
 always @(posedge clk)
  begin
        case (ret)
           0 : begin
                 if (jump == 0)
                    pc = pc + 1;
                 else
                    pc = pc + jnumber;
               end
           1 : pc = stko;
        endcase
  end
endmodule
```

Symbol view：



Schematic：



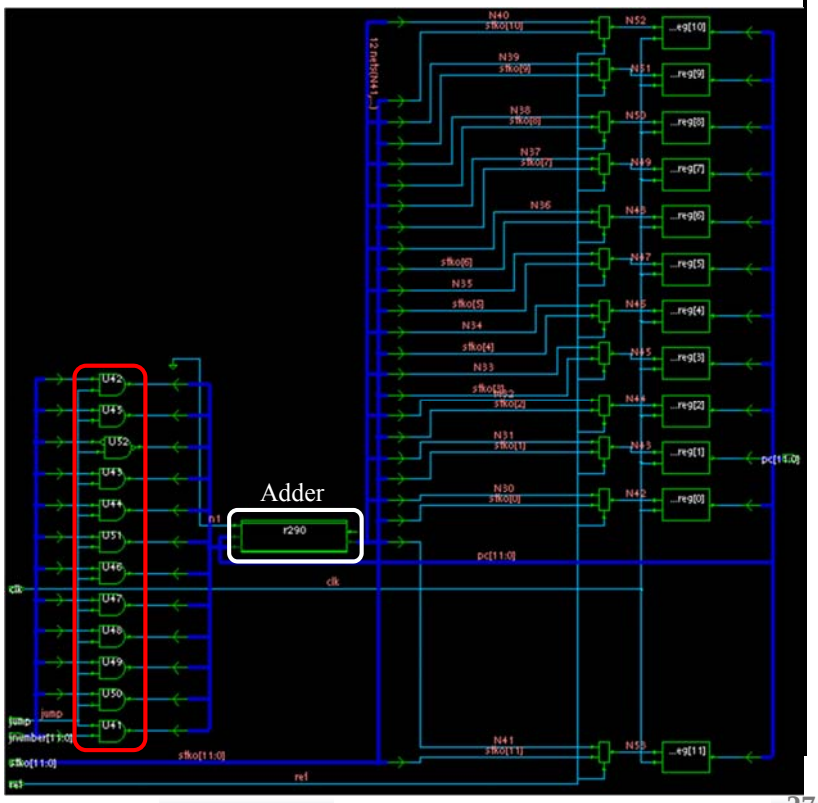
程式計數器系統之二-2/3

Verilog HDL Code：

```
module pcstk_v1 (clk,jump,ret,jnumber,stko,pc);
 input   [11:0] jnumber, stko;
 input   clk, jump, ret;
 output  [11:0] pc;
 reg     [11:0] pc;
 always @(posedge clk)
  begin
        case (ret)
           0 : begin
                 if (jump == 0)
                   pc = pc + 1;
                 else
                   pc = pc + jnumber;
               end
           1 : pc = stko;
        endcase
  end
endmodule
```

Adder

Schematic：

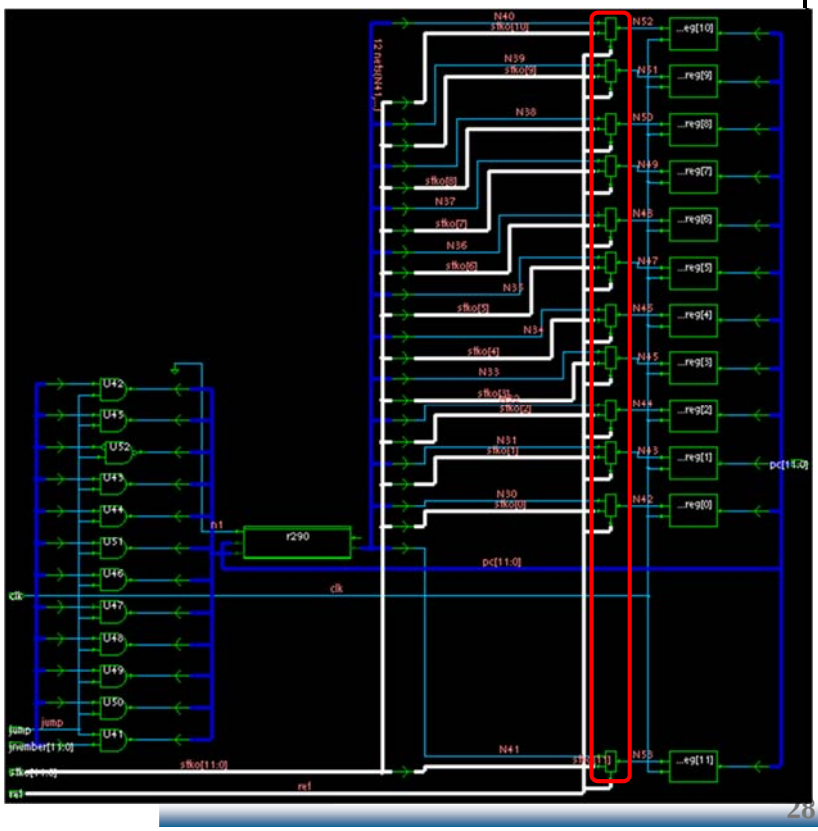


程式計數器系統之二-3/3

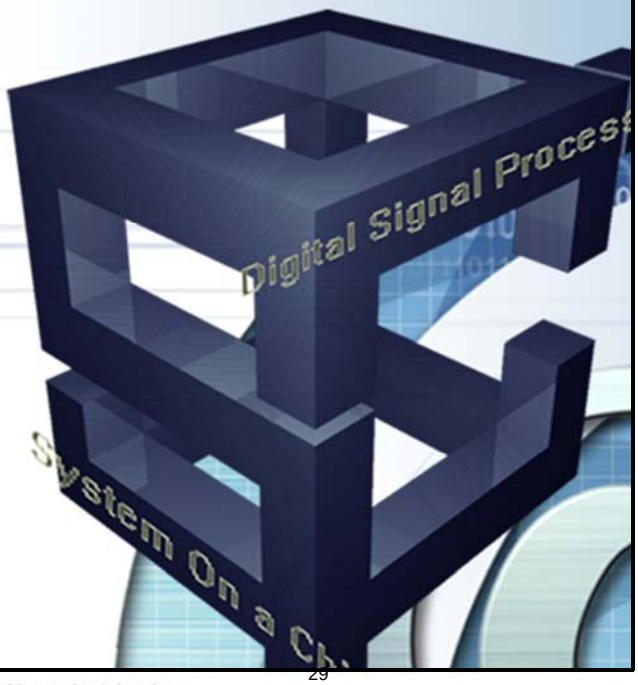
Verilog HDL Code：

```
module pcstk_v1 (clk,jump,ret,jnumber,stko,pc);
 input   [11:0] jnumber, stko;
 input   clk, jump, ret;
 output  [11:0] pc;
 reg     [11:0] pc;
 always @(posedge clk)
  begin
        case (ret)
           0 : begin
                 if (jump == 0)
                   pc = pc + 1;
                 else
                   pc = pc + jnumber;
               end
           1 : pc = stko;
        endcase
  end
endmodule
```

Schematic：



Verilog HDL設計範例



授課老師：黃穎聰 老師

TA：

徐偉傑 LAB723

Outline

1. 設計3x8解碼器電路
2. 設計ALU電路
3. 設計8位元平行多工暫存器電路
4. 設計16位元計數器電路
5. 設計管線處理電路

1. 設計3x8解碼器電路
2. 設計ALU電路
3. 設計8位元平行多工暫存器電路
4. 設計16位元計數器電路
5. 設計管線處理電路

設計3x8解碼器電路-1/3

Verilog HDL Code：
(?：條件運算子的應用)

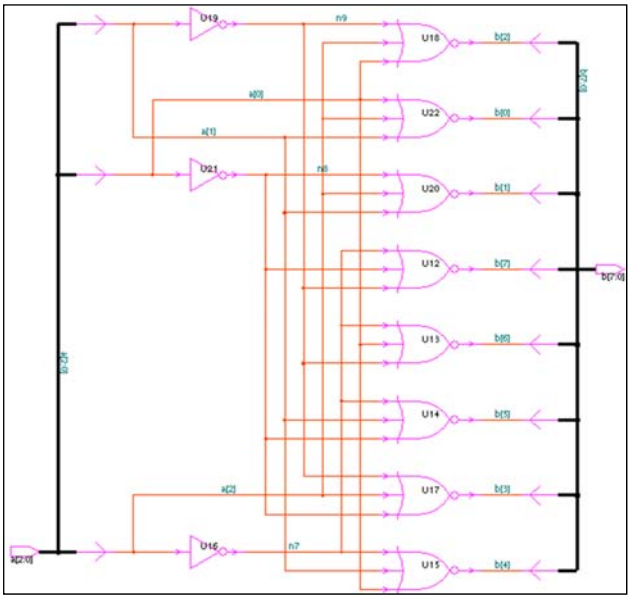
```
module EX_106_3x8_decoder(a, b);
input   [2:0]   a;
output  [7:0]   b;
wire    [7:0]   b;

assign  b[0] = ( a==3'b000 ) ? 1'b1 : 1'b0;
assign  b[1] = ( a==3'b001 ) ? 1'b1 : 1'b0;
assign  b[2] = ( a==3'b010 ) ? 1'b1 : 1'b0;
assign  b[3] = ( a==3'b011 ) ? 1'b1 : 1'b0;
assign  b[4] = ( a==3'b100 ) ? 1'b1 : 1'b0;
assign  b[5] = ( a==3'b101 ) ? 1'b1 : 1'b0;
assign  b[6] = ( a==3'b110 ) ? 1'b1 : 1'b0;
assign  b[7] = ( a==3'b111 ) ? 1'b1 : 1'b0;

endmodule
```

Verilog HDL Code：
(always行為模式及case循序指令)

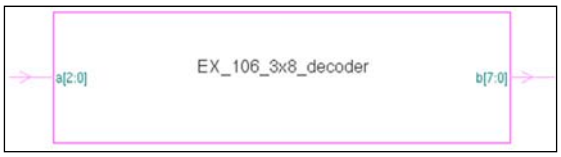
```
module EX_204_beh_3x8_decoder(a, b);
input   [2:0]   a;
output  [7:0]   b;
reg [7:0]   b;

always@( a )
begin
    case (a)
      3'b000  : b = 8'b0000_0001;
      3'b001  : b = 8'b0000_0010;
      3'b010  : b = 8'b0000_0100;
      3'b011  : b = 8'b0000_1000;
      3'b100  : b = 8'b0001_0000;
      3'b101  : b = 8'b0010_0000;
      3'b110  : b = 8'b0100_0000;
      default : b = 8'b1000_0000;
    endcase
end
endmodule
```

設計3x8解碼器電路-2/3

Schematic：



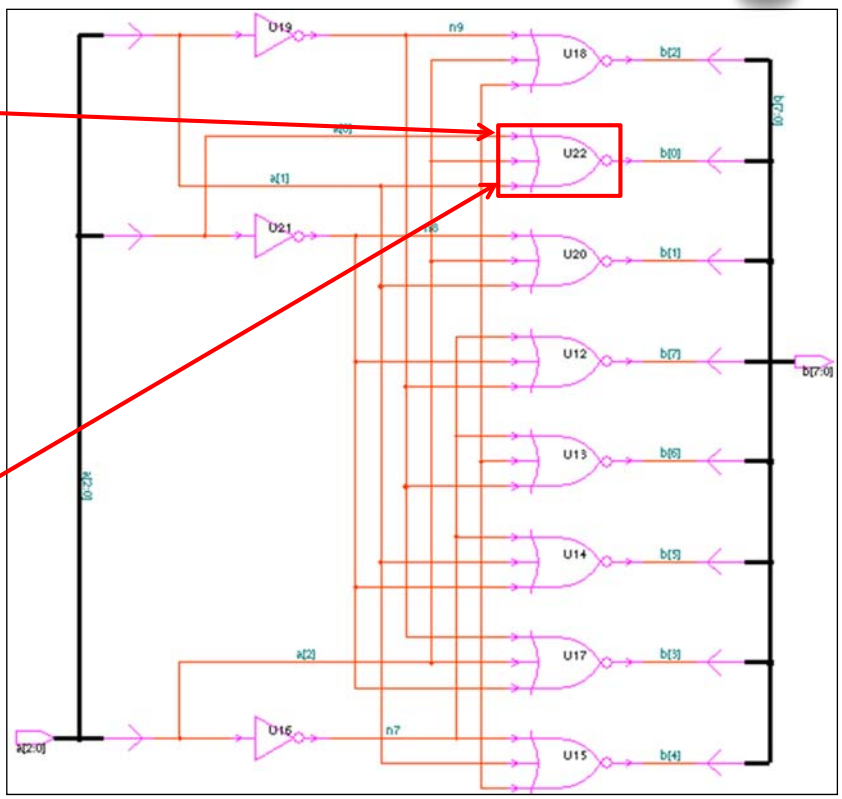
Symbol view：



設計3x8解碼器電路-3/3

```
module EX_106_3x8_decoder(a, b);
input   [2:0]   a;
output  [7:0]   b;
wire    [7:0]   b;

assign  b[0] = ( a==3'b000 ) ? 1'b1 : 1'b0;
assign  b[1] = ( a==3'b001 ) ? 1'b1 : 1'b0;
assign  b[2] = ( a==3'b010 ) ? 1'b1 : 1'b0;
assign  b[3] = ( a==3'b011 ) ? 1'b1 : 1'b0;
assign  b[4] = ( a==3'b100 ) ? 1'b1 : 1'b0;
assign  b[5] = ( a==3'b101 ) ? 1'b1 : 1'b0;
assign  b[6] = ( a==3'b110 ) ? 1'b1 : 1'b0;
assign  b[7] = ( a==3'b111 ) ? 1'b1 : 1'b0;

endmodule
```

```
module EX_204_beh_3x8_decoder(a, b);
input   [2:0]   a;
output  [7:0]   b;
reg [7:0]   b;

always@( a )
begin
    case (a)
      3'b000  : b = 8'b0000_0001;
      3'b001  : b = 8'b0000_0010;
      3'b010  : b = 8'b0000_0100;
      3'b011  : b = 8'b0000_1000;
      3'b100  : b = 8'b0001_0000;
      3'b101  : b = 8'b0010_0000;
      3'b110  : b = 8'b0100_0000;
      default : b = 8'b1000_0000;
    endcase
end
endmodule
```



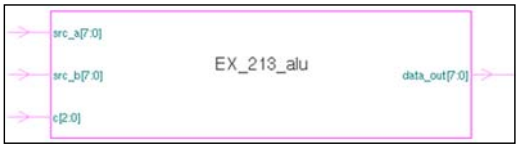
1. 設計3x8解碼器電路
2. 設計ALU電路
3. 設計8位元平行多工暫存器電路
4. 設計16位元計數器電路
5. 設計管線處理電路

設計ALU電路-1/7

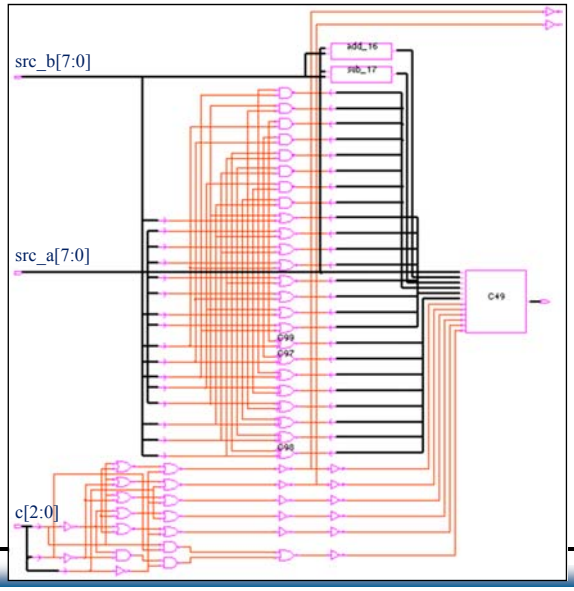
Verilog HDL Code：

```
module EX_213_alu (src_a, src_b, c, data_out);
input   [7:0]   src_a, src_b;
input   [2:0]   c;
output  [7:0]   data_out;
wire    [7:0]   data_out;

reg [7:0]   dest;

assign  data_out = dest;

always @(c or src_a or src_b)
begin
  case (c)
    3'b001 :  dest = src_a + src_b;
    3'b010 :  dest = src_a - src_b;
    3'b011 :  dest = src_a & src_b;
    3'b100 :  dest = src_a | src_b;
    3'b101 :  dest = src_a ^ src_b;
    default : dest = src_a;
  endcase
end

endmodule
```

Symbol view：

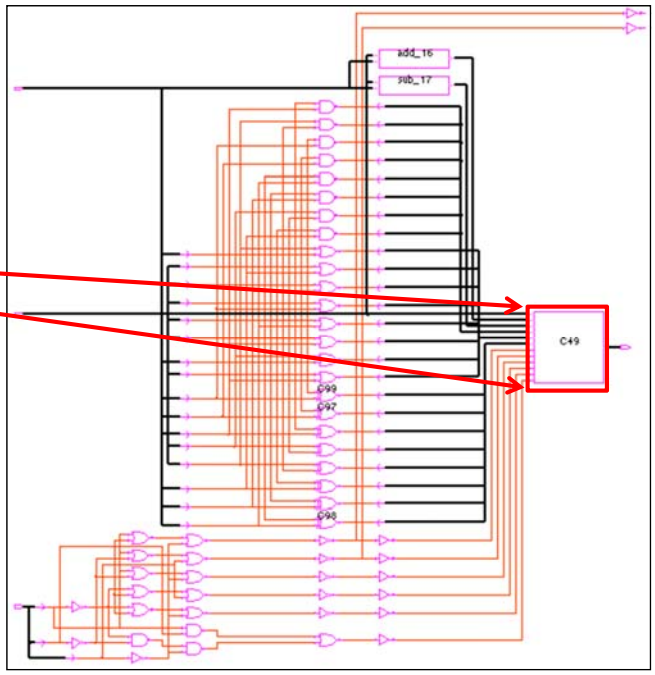


Schematic：



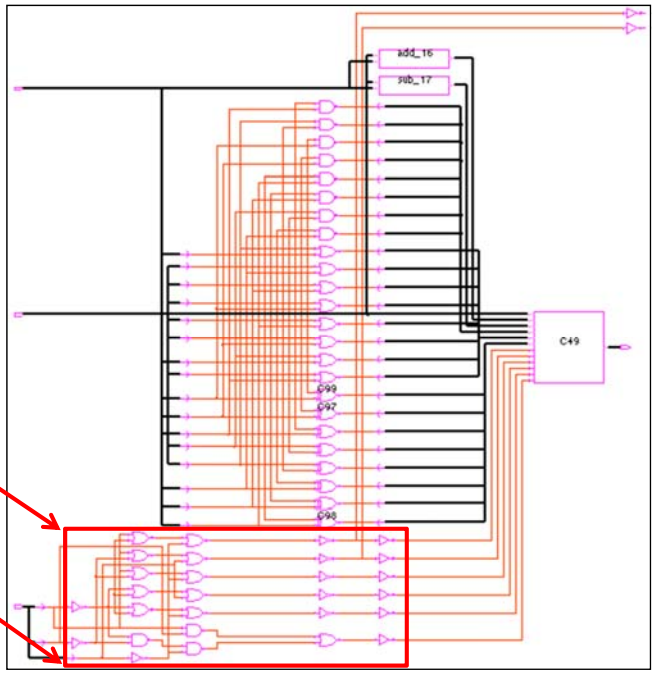
設計ALU電路-2/7

```
module EX_213_alu (src_a, src_b, c, data_out);
input   [7:0]   src_a, src_b;
input   [2:0]   c;
output  [7:0]   data_out;
wire    [7:0]   data_out;

reg [7:0]   dest;

assign  data_out = dest;

always @(c or src_a or src_b)
begin
  case (c)
    3'b001 :  dest = src_a + src_b;
    3'b010 :  dest = src_a - src_b;
    3'b011 :  dest = src_a & src_b;
    3'b100 :  dest = src_a | src_b;
    3'b101 :  dest = src_a ^ src_b;
    default : dest = src_a;
  endcase
end

endmodule
```



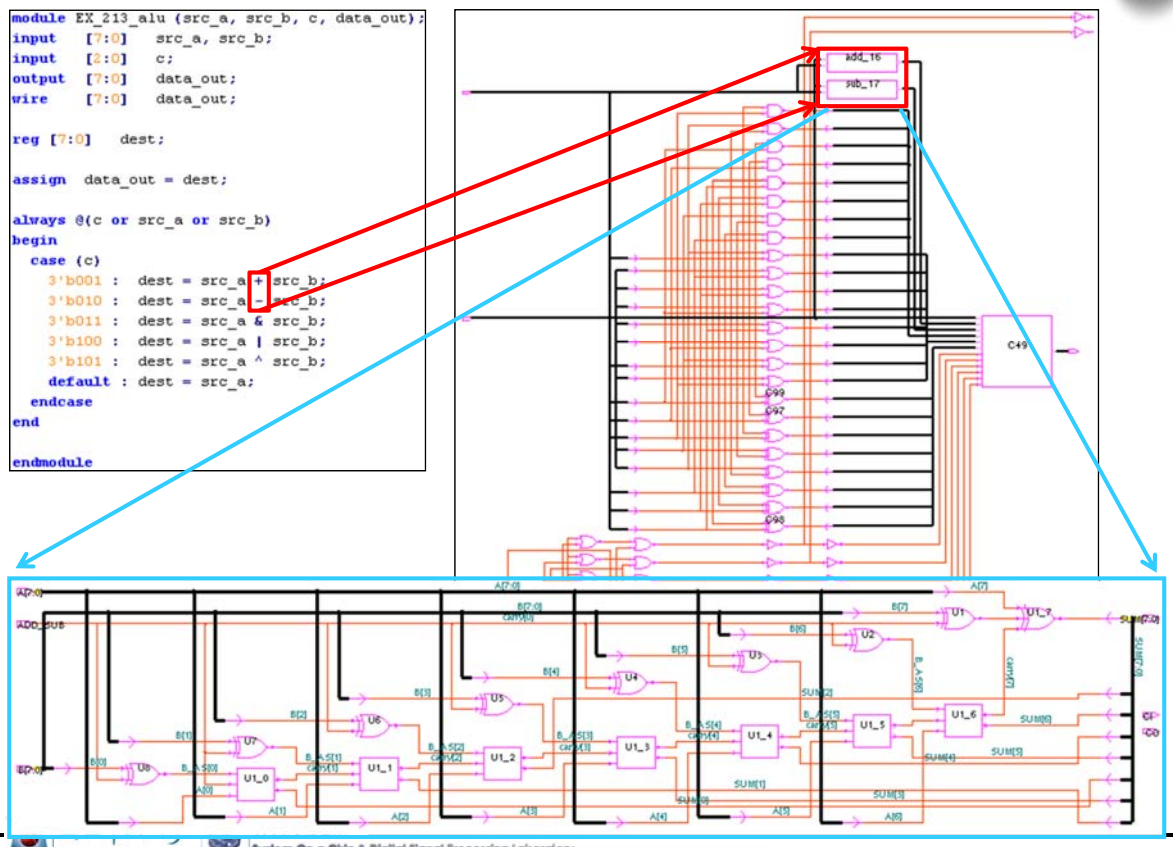
設計ALU電路-3/7

```
module EX_213_alu (src_a, src_b, c, data_out);
input   [7:0]   src_a, src_b;
input   [2:0]   c;
output  [7:0]   data_out;
wire    [7:0]   data_out;

reg [7:0]   dest;

assign  data_out = dest;

always @(c or src_a or src_b)
begin
  case (c)
    3'b001 :  dest = src_a + src_b;
    3'b010 :  dest = src_a - src_b;
    3'b011 :  dest = src_a & src_b;
    3'b100 :  dest = src_a | src_b;
    3'b101 :  dest = src_a ^ src_b;
    default : dest = src_a;
  endcase
end

endmodule
```



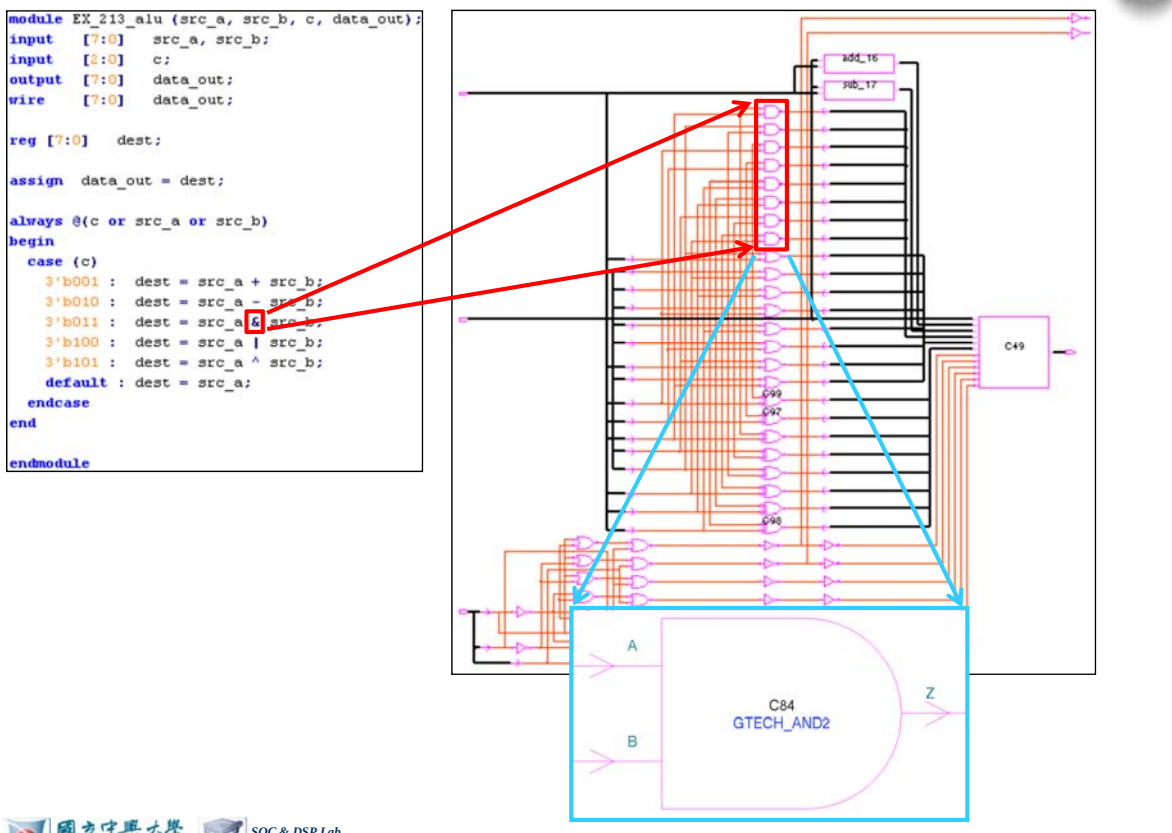
設計ALU電路-4/7

```
module EX_213_alu (src_a, src_b, c, data_out);
input   [7:0]   src_a, src_b;
input   [2:0]   c;
output  [7:0]   data_out;
wire    [7:0]   data_out;

reg [7:0]   dest;

assign  data_out = dest;

always @(c or src_a or src_b)
begin
  case (c)
    3'b001 :  dest = src_a + src_b;
    3'b010 :  dest = src_a - src_b;
    3'b011 :  dest = src_a & src_b;
    3'b100 :  dest = src_a | src_b;
    3'b101 :  dest = src_a ^ src_b;
    default : dest = src_a;
  endcase
end

endmodule
```



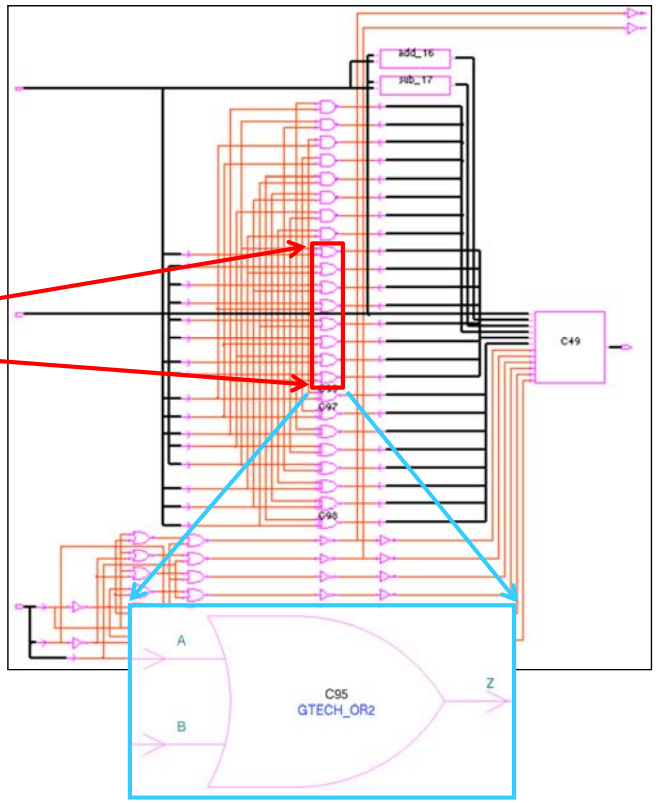
設計ALU電路-5/7

```
module EX_213_alu (src_a, src_b, c, data_out);
input   [7:0]   src_a, src_b;
input   [2:0]   c;
output  [7:0]   data_out;
wire    [7:0]   data_out;

reg [7:0]   dest;

assign  data_out = dest;

always @(c or src_a or src_b)
begin
  case (c)
    3'b001 :  dest = src_a + src_b;
    3'b010 :  dest = src_a - src_b;
    3'b011 :  dest = src_a & src_b;
    3'b100 :  dest = src_a | src_b;
    3'b101 :  dest = src_a ^ src_b;
    default : dest = src_a;
  endcase
end

endmodule
```



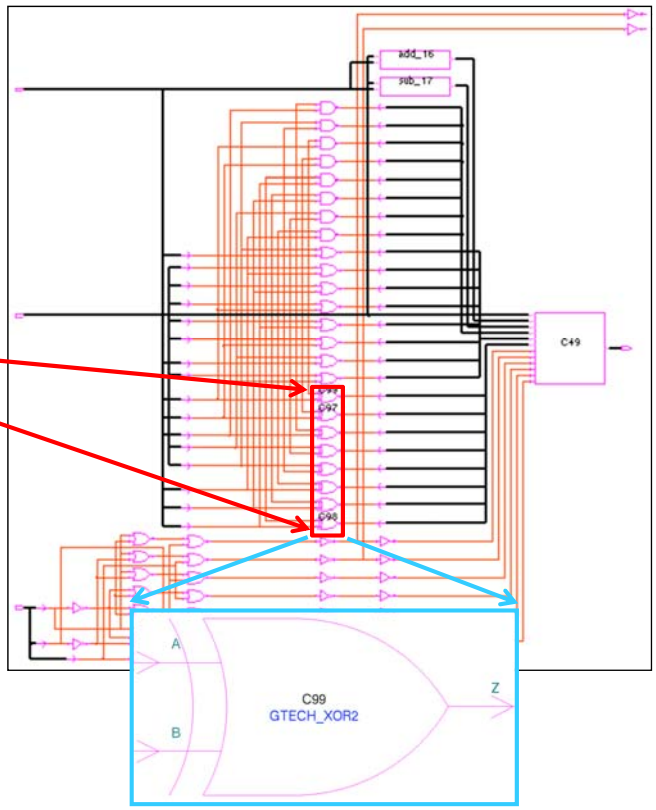
設計ALU電路-6/7

```
module EX_213_alu (src_a, src_b, c, data_out);
input   [7:0]   src_a, src_b;
input   [2:0]   c;
output  [7:0]   data_out;
wire    [7:0]   data_out;

reg [7:0]   dest;

assign  data_out = dest;

always @(c or src_a or src_b)
begin
  case (c)
    3'b001 :  dest = src_a + src_b;
    3'b010 :  dest = src_a - src_b;
    3'b011 :  dest = src_a & src_b;
    3'b100 :  dest = src_a | src_b;
    3'b101 :  dest = src_a ^ src_b;
    default : dest = src_a;
  endcase
end

endmodule
```



設計ALU電路-7/7

```
module EX_213_alu (src_a, src_b, c, data_out);
input   [7:0]   src_a, src_b;
input   [2:0]   c;
output  [7:0]   data_out;
wire    [7:0]   data_out;

reg [7:0]   dest;

assign  data_out = dest;

always @(c or src_a or src_b)
begin
  case (c)
    3'b001 :  dest = src_a + src_b;
    3'b010 :  dest = src_a - src_b;
    3'b011 :  dest = src_a & src_b;
    3'b100 :  dest = src_a | src_b;
    3'b101 :  dest = src_a ^ src_b;
    default : dest = src_a;
  endcase
end

endmodule
```



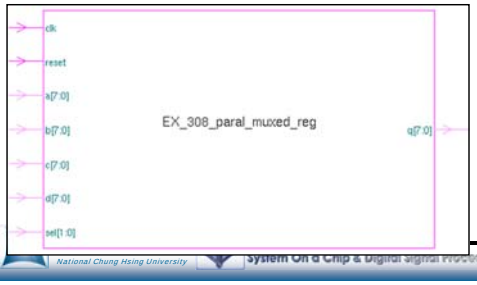
1. 設計3x8解碼器電路
2. 設計ALU電路
3. 設計8位元平行多工暫存器電路
4. 設計16位元計數器電路
5. 設計管線處理電路

設計8位元平行多工暫存器電路-1/4

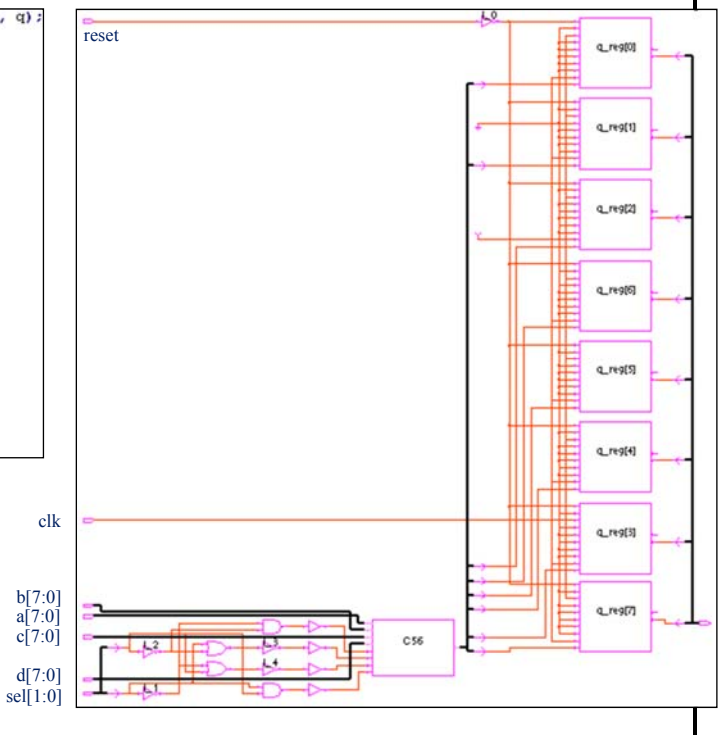
Verilog HDL Code：

```
module EX_308_paral_muxed_reg (clk, reset, a, b, c, d, sel, q);
input       clk, reset;
input   [7:0]   a, b, c, d;
input   [1:0]   sel;
output  [7:0]   q;
reg [7:0]   q;

always @ (posedge clk or negedge reset)
begin
    if (!reset)
      q <= 8'h00;
    else
      case (sel)
        2'b00 :   q <= a;
        2'b01 :   q <= b;
        2'b10 :   q <= c;
        2'b11 :   q <= d;
        default : q <= a;
      endcase
end

endmodule
```

Symbol view：

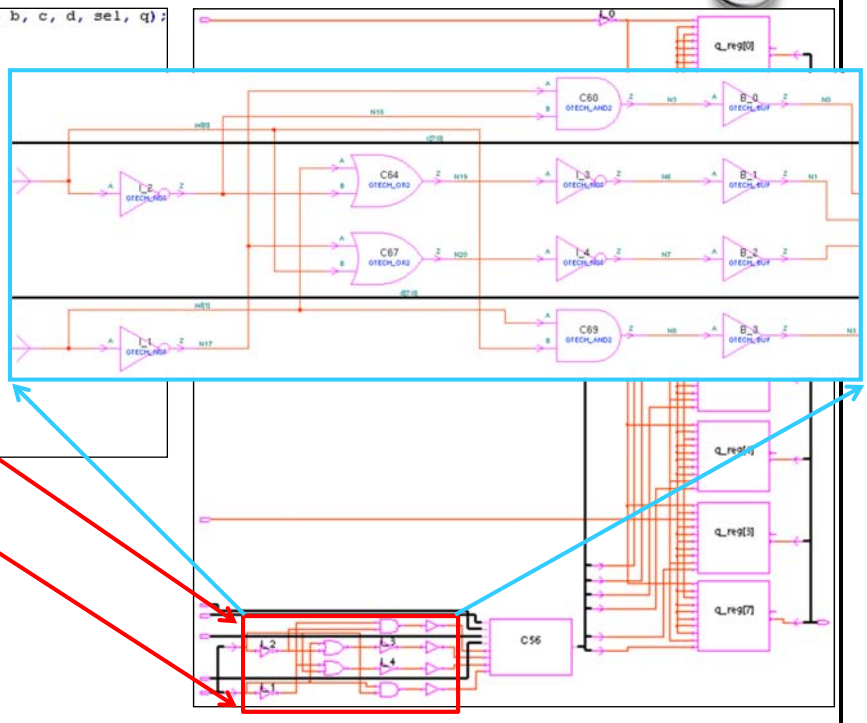


Schematic：



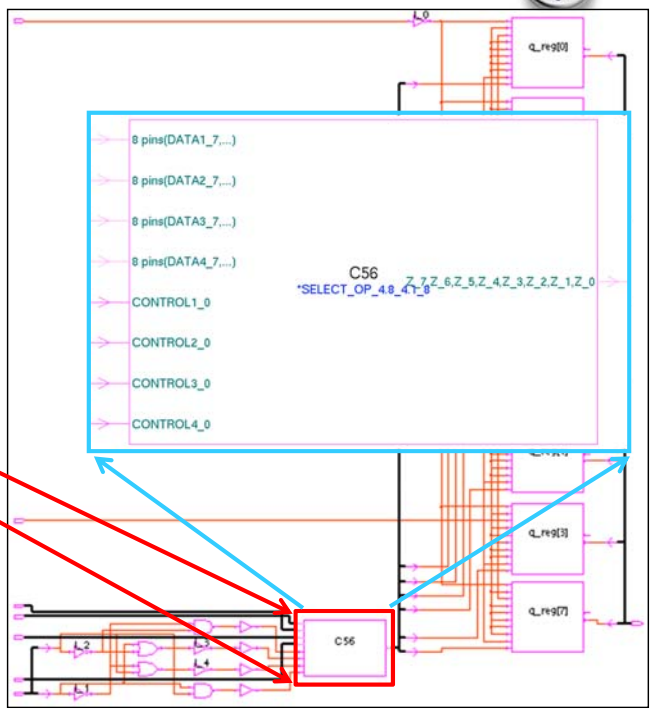
設計8位元平行多工暫存器電路-2/4

```
module EX_308_paral_muxed_reg (clk, reset, a, b, c, d, sel, q);
input       clk, reset;
input   [7:0]   a, b, c, d;
input   [1:0]   sel;
output  [7:0]   q;
reg [7:0]   q;

always @ (posedge clk or negedge reset)
begin
    if (!reset)
      q <= 8'h00;
    else
      case (sel)
        2'b00 :   q <= a;
        2'b01 :   q <= b;
        2'b10 :   q <= c;
        2'b11 :   q <= d;
       default : q <= a;
      endcase
end

endmodule
```



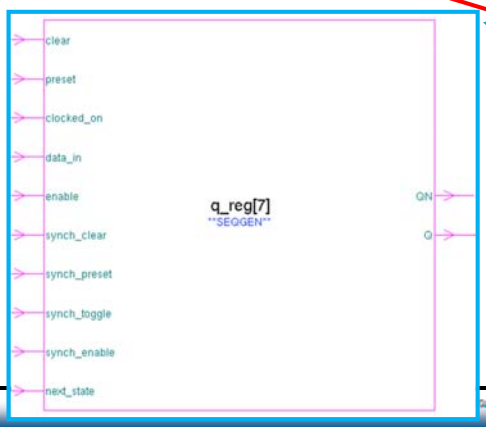
設計8位元平行多工暫存器電路-3/4

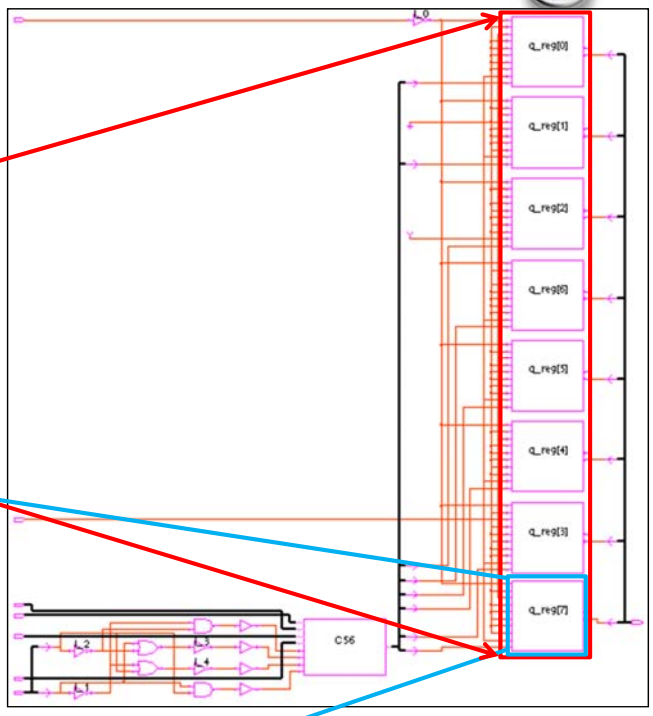
```
module EX_308_paral_muxed_reg (clk, reset, a, b, c, d, sel, q);
input       clk, reset;
input   [7:0]   a, b, c, d;
input   [1:0]   sel;
output  [7:0]   q;
reg [7:0]   q;

always @ (posedge clk or negedge reset)
begin
    if (!reset)
      q <= 8'h00;
    else
      case (sel)
        2'b00 :   q <= a;
        2'b01 :   q <= b;
        2'b10 :   q <= c;
        2'b11 :   q <= d;
       default : q <= a;
      endcase
end

endmodule
```



設計8位元平行多工暫存器電路-4/4

```
module EX_308_paral_muxed_reg (clk, reset, a, b, c, d, sel, q);
input       clk, reset;
input   [7:0]   a, b, c, d;
input   [1:0]   sel;
output  [7:0]   q;
reg [7:0]   q;

always @ (posedge clk or negedge reset)
begin
    if (!reset)
      q <= 8'h00;
    else
      case (sel)
        2'b00 :   q <= a;
        2'b01 :   q <= b;
        2'b10 :   q <= c;
        2'b11 :   q <= d;
       default : q <= a;
      endcase
end

endmodule
```





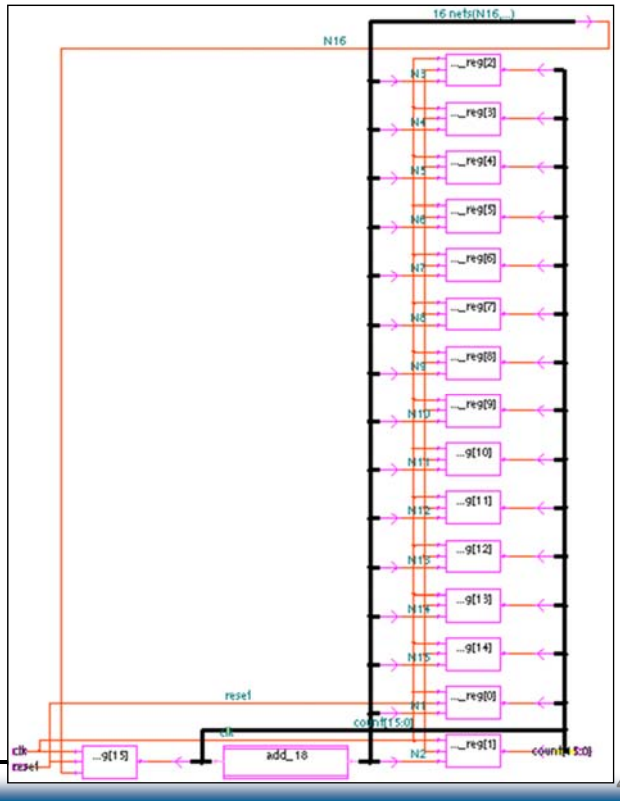
1. 設計3x8解碼器電路
2. 設計ALU電路
3. 設計8位元平行多工暫存器電路
4. 設計16位元計數器電路
5. 設計管線處理電路

設計16位元計數器電路-1/3

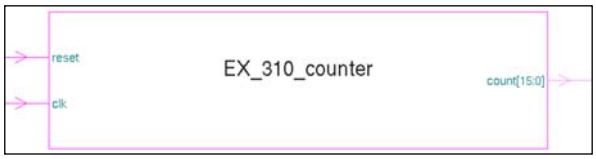
Verilog HDL Code：

```
module EX_310_counter ( reset, clk, count);
input       reset;
input       clk;
output  [15:0]  count;
wire    [15:0]  count;

reg [15:0]  cnt;

assign  count = cnt;

always@(posedge clk or negedge reset)
begin
  if(!reset)
    cnt <= 16'h0;
  else
    cnt <= cnt + 16'h0001;
end

endmodule
```

Schematic：

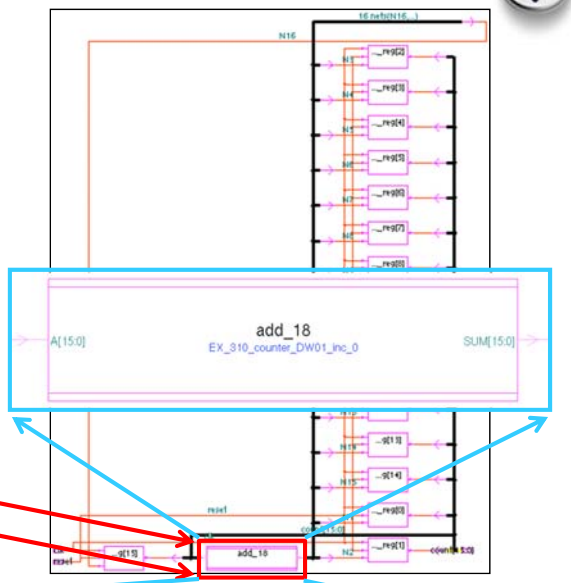


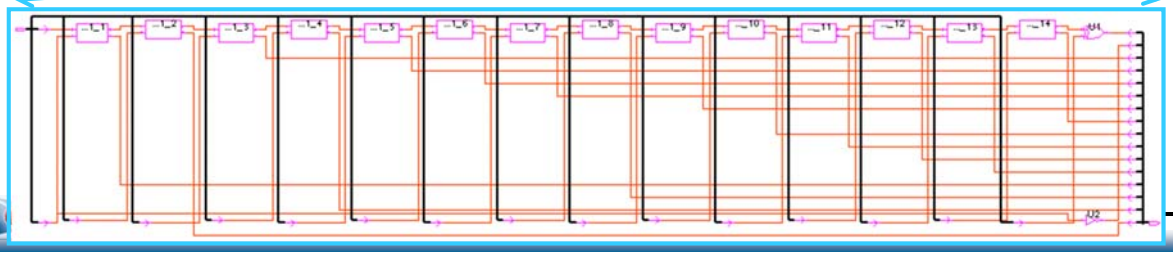
Symbol view：



設計16位元計數器電路-2/3

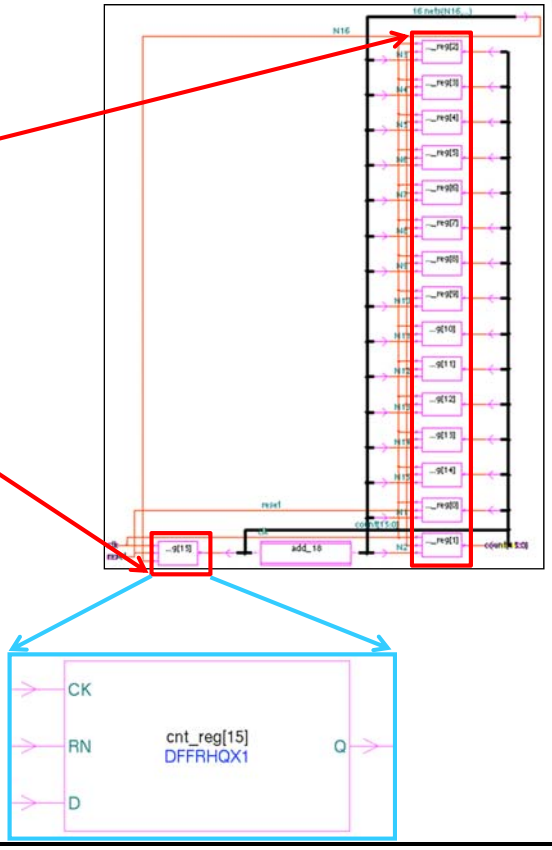
```
module EX_310_counter ( reset, clk, count);
input       reset;
input       clk;
output  [15:0]  count;
wire    [15:0]  count;

reg [15:0]  cnt;

assign  count = cnt;

always@(posedge clk or negedge reset)
begin
  if(!reset)
    cnt <= 16'h0;
  else
    cnt <= cnt + 16'h0001;
end

endmodule
```





設計16位元計數器電路-3/3

```
module EX_310_counter ( reset, clk, count);
input        reset;
input        clk;
output  [15:0]  count;
wire    [15:0]  count;

reg [15:0]  cnt;

assign  count = cnt;

always@(posedge clk or negedge reset)
begin
  if(!reset)
    cnt <= 16'h0;
  else
    cnt <= cnt + 16'h0001;
end

endmodule
```



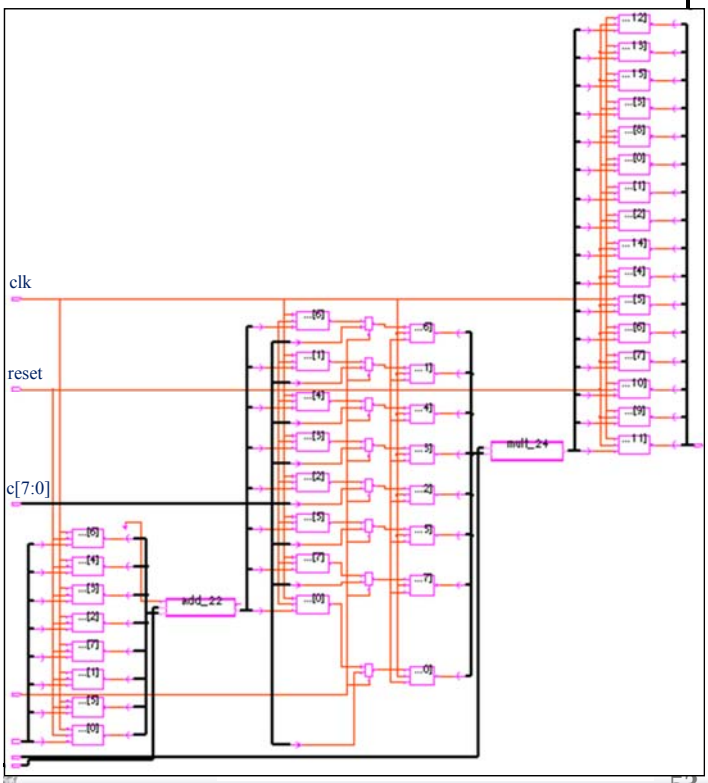
1. 設計3x8解碼器電路
2. 設計ALU電路
3. 設計8位元平行多工暫存器電路
4. 設計16位元計數器電路
5. 設計管線處理電路

設計管線處理電路-1/8

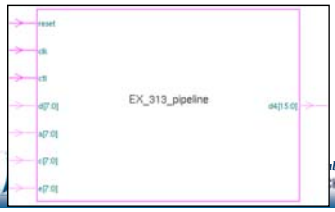
Verilog HDL Code：

```
module EX_313_pipeline(reset, clk, d, a, c, e, ctl, d4);
input   reset, clk;
input   [7:0]   d, a, c, e;
input   ctl;
output  [15:0]  d4;
reg     [15:0]  d4;

reg [7:0] d1, d2, d3;

always@(posedge clk or negedge reset)
begin
  if(!reset) begin
    d1 <= 8'h00;
    d2 <= 8'h00;
    d3 <= 8'h00;
    d4 <= 16'h0000;
  end
  else begin
    d1 <= d;
    d2 <= d1 + a;
    d3 <= (ctl==1'b1) ? c : d2;
    d4 <= d3 * e;
  end
end

endmodule
```

Schematic：

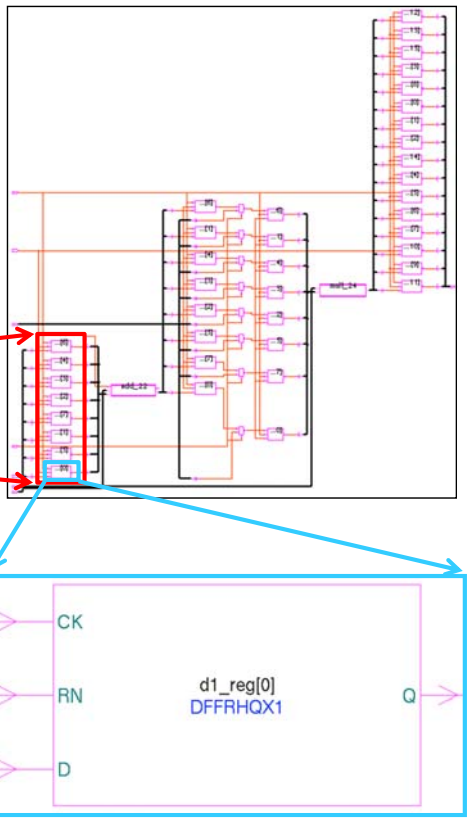


Symbol view：



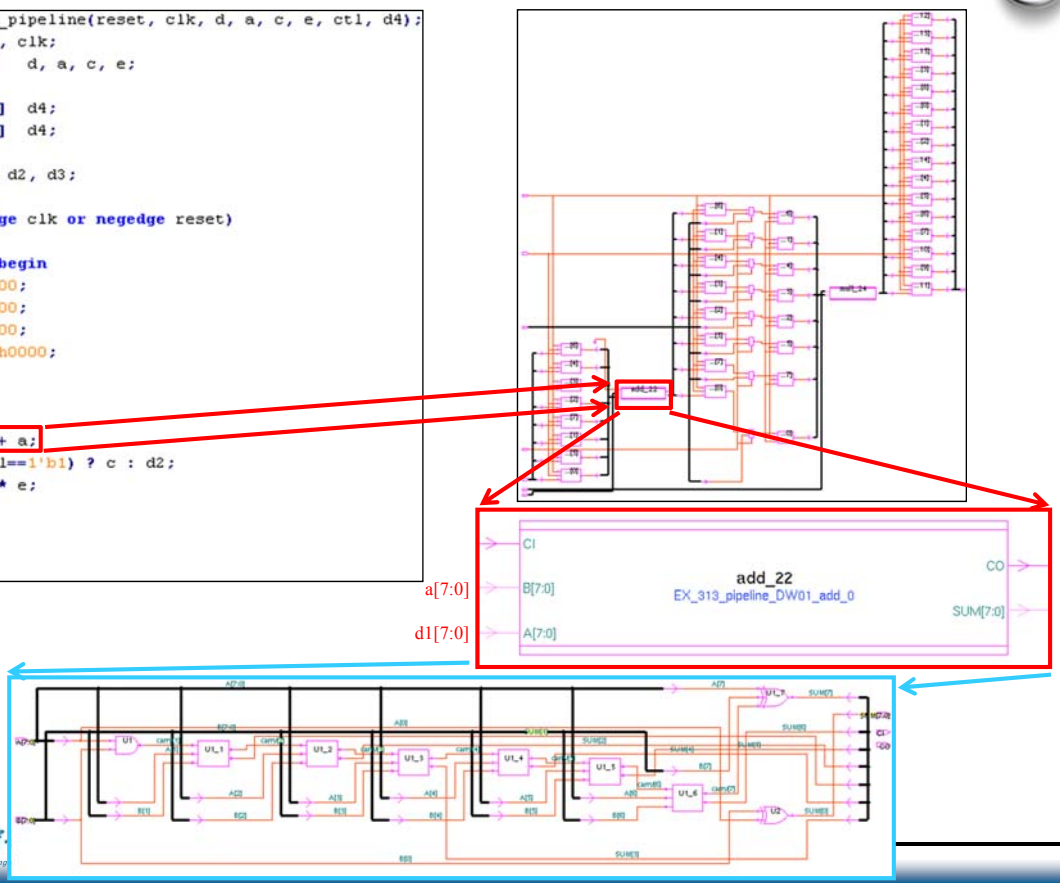
設計管線處理電路-2/8

```
module EX_313_pipeline(reset, clk, d, a, c, e, ctl, d4);
input   reset, clk;
input   [7:0]   d, a, c, e;
input   ctl;
output  [15:0]  d4;
reg     [15:0]  d4;

reg [7:0] d1, d2, d3;

always@(posedge clk or negedge reset)
begin
  if(!reset) begin
    d1 <= 8'h00;
    d2 <= 8'h00;
    d3 <= 8'h00;
    d4 <= 16'h0000;
  end
  else begin
    d1 <= d;
    d2 <= d1 + a;
    d3 <= (ctl==1'b1) ? c : d2;
    d4 <= d3 * e;
  end
end

endmodule
```



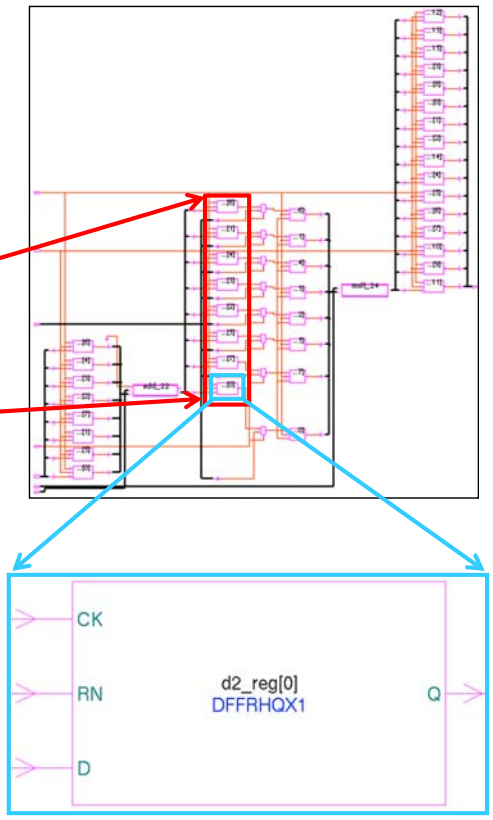
設計管線處理電路-3/8

```
module EX_313_pipeline(reset, clk, d, a, c, e, ctl, d4);
input   reset, clk;
input   [7:0]   d, a, c, e;
input   ctl;
output  [15:0]  d4;
reg     [15:0]  d4;

reg [7:0] d1, d2, d3;

always@(posedge clk or negedge reset)
begin
  if(!reset) begin
    d1 <= 8'h00;
    d2 <= 8'h00;
    d3 <= 8'h00;
    d4 <= 16'h0000;
  end
  else begin
    d1 <= d;
    d2 <= d1 + a;
    d3 <= (ctl==1'b1) ? c : d2;
    d4 <= d3 * e;
  end
end

endmodule
```



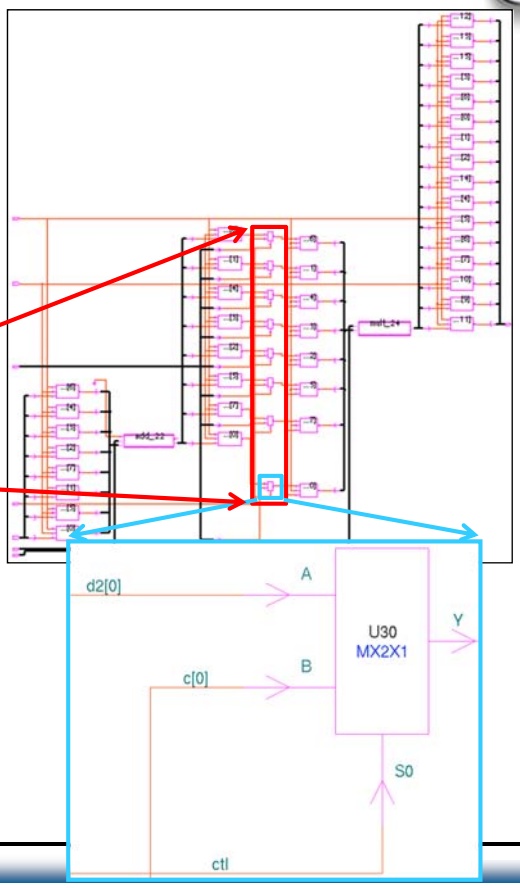
設計管線處理電路-4/8

```
module EX_313_pipeline(reset, clk, d, a, c, e, ctl, d4);
input   reset, clk;
input   [7:0]   d, a, c, e;
input   ctl;
output  [15:0]  d4;
reg     [15:0]  d4;

reg [7:0] d1, d2, d3;

always@(posedge clk or negedge reset)
begin
  if(!reset) begin
    d1 <= 8'h00;
    d2 <= 8'h00;
    d3 <= 8'h00;
    d4 <= 16'h0000;
  end
  else begin
    d1 <= d;
    d2 <= d1 + a;
    d3 <= (ctl==1'b1) ? c : d2;
    d4 <= d3 * e;
  end
end

endmodule
```



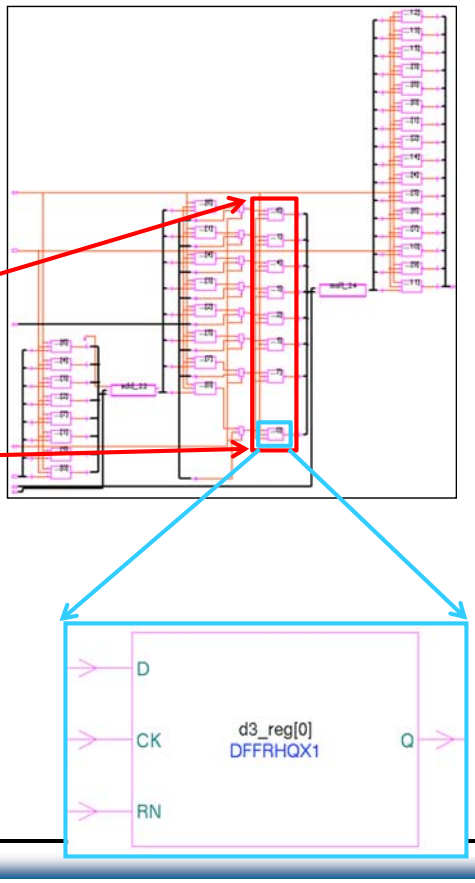
設計管線處理電路-5/8

```
module EX_313_pipeline(reset, clk, d, a, c, e, ctl, d4);
input   reset, clk;
input   [7:0]   d, a, c, e;
input   ctl;
output  [15:0]  d4;
reg     [15:0]  d4;

reg [7:0] d1, d2, d3;

always@(posedge clk or negedge reset)
begin
  if(!reset) begin
    d1 <= 8'h00;
    d2 <= 8'h00;
    d3 <= 8'h00;
    d4 <= 16'h0000;
  end
  else begin
    d1 <= d;
    d2 <= d1 + a;
    d3 <= (ctl==1'b1) ? c : d2;
    d4 <= d3 * e;
  end
end

endmodule
```



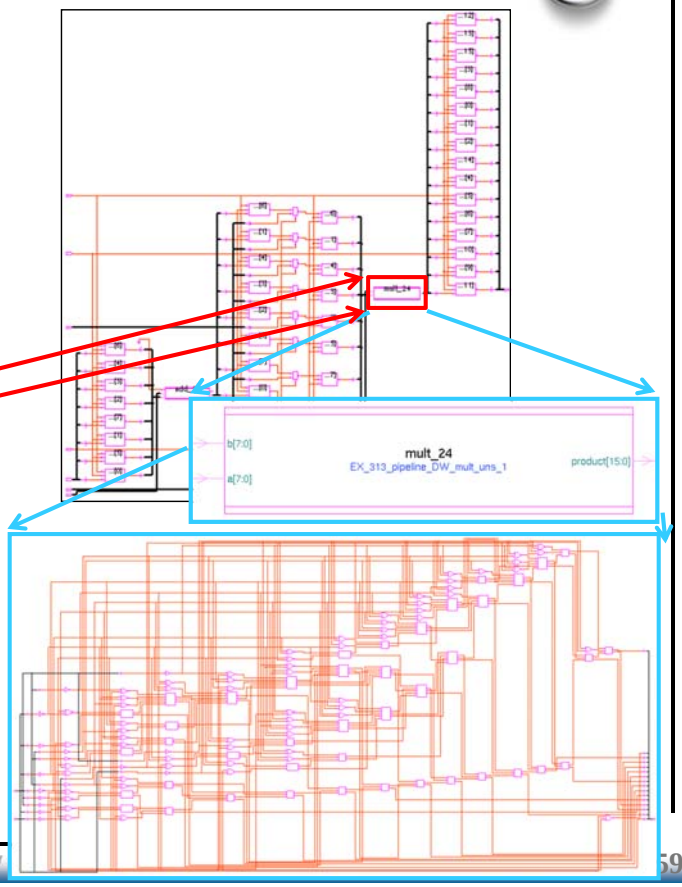
設計管線處理電路-6/8

```
module EX_313_pipeline(reset, clk, d, a, c, e, ctl, d4);
input   reset, clk;
input   [7:0]   d, a, c, e;
input   ctl;
output  [15:0]  d4;
reg     [15:0]  d4;

reg [7:0] d1, d2, d3;

always@(posedge clk or negedge reset)
begin
  if(!reset) begin
    d1 <= 8'h00;
    d2 <= 8'h00;
    d3 <= 8'h00;
    d4 <= 16'h0000;
  end
  else begin
    d1 <= d;
    d2 <= d1 + a;
    d3 <= (ctl==1'b1) ? c : d2;
    d4 <= d3 * e;
  end
end

endmodule
```



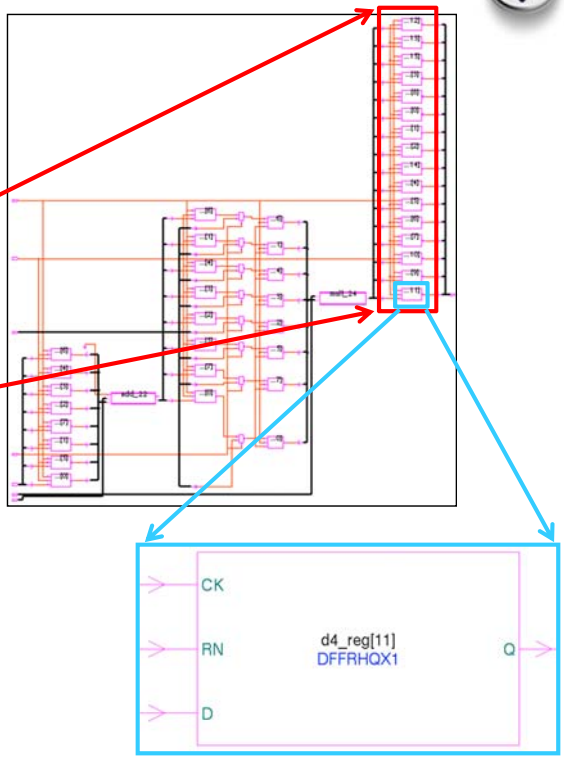
設計管線處理電路-7/8

```
module EX_313_pipeline(reset, clk, d, a, c, e, ctl, d4);
input   reset, clk;
input   [7:0]   d, a, c, e;
input   ctl;
output  [15:0]  d4;
reg     [15:0]  d4;

reg [7:0] d1, d2, d3;

always@(posedge clk or negedge reset)
begin
  if(!reset) begin
    d1 <= 8'h00;
    d2 <= 8'h00;
    d3 <= 8'h00;
    d4 <= 16'h0000;
  end
  else begin
    d1 <= d;
    d2 <= d1 + a;
    d3 <= (ctl==1'b1) ? c : d2;
    d4 <= d3 * e;
  end
end

endmodule
```

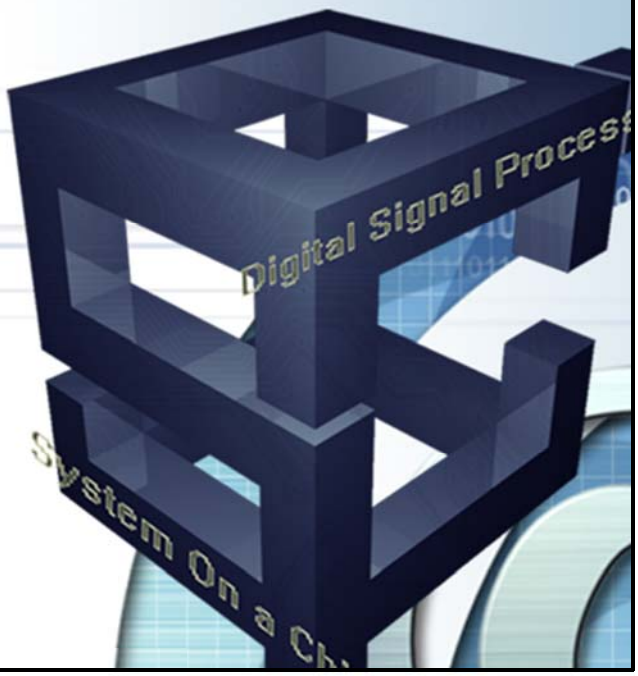


設計管線處理電路-8/8

```
module EX_313_pipeline(reset, clk, d, a, c, e, ctl, d4);
input   reset, clk;
input   [7:0]   d, a, c, e;
input   ctl;
output  [15:0]  d4;
reg     [15:0]  d4;

reg [7:0] d1, d2, d3;

always@(posedge clk or negedge reset)
begin
  if(!reset) begin
    d1 <= 8'h00;
    d2 <= 8'h00;
    d3 <= 8'h00;
    d4 <= 16'h0000;
  end
  else begin
    d1 <= d;
    d2 <= d1 + a;
    d3 <= (ctl==1'b1) ? c : d2;
    d4 <= d3 * e;
  end
end

endmodule
```



Verilog HDL 設計範例



授課教師：黃穎聰老師
助教：黃唯竣

Outline

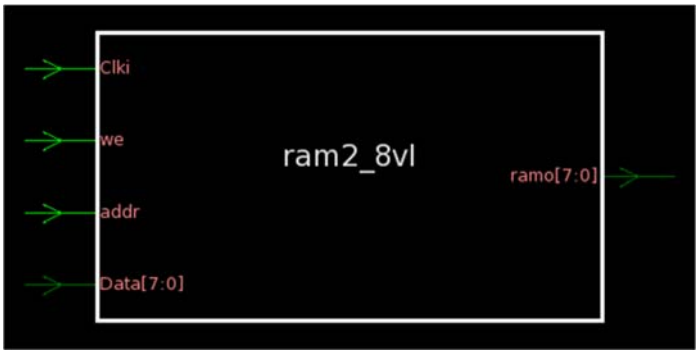
- 2x8隨機存取記憶體
- 堆疊指標系統
- 條件碼系統
- 控制系統一
- 控制系統二

2×8隨機存取記憶體

輸入				記憶體		輸出	說明
we	Clki	Data	addr	Q1	Q0	ramo	動作
1	↑	A	0	Q1(不變)	A	A	寫
1	↑	B	1	B	Q0(不變)	B	寫
0	↑	X	0	Q1(不變)	Q0(不變)	Q0	讀
0	↑	X	1	Q1(不變)	Q0(不變)	Q1	讀
1	其它	X	X	Q1(不變)	Q0(不變)	不變	不動作
0	其它	X	X	Q1(不變)	Q0(不變)	不變	不動作

2×8隨機存取記憶體

- Symbol View



2×8隨機存取記憶體

- Schematic

```
module ram2_8v1 (Data,Clki,we,addr,ramo);
   input   [7:0] Data;
   input   Clki, we, addr;
   output  [7:0] ramo;
   reg     [7:0] Q0, Q1;
   reg     [7:0] ramo;
   reg     addrtem;

   always @(posedge Clki )
    begin
          if (we)
           begin
             case (addr)
                  0 : Q0 = Data;
                  1 : Q1 = Data;
             endcase
           end
    end

always @(posedge Clki)
 begin
  addrtem = addr;
 end

always @(addrtem or Q0 or Q1)
 begin
   case (addrtem)
     0 :  ramo = Q0;
     1 :  ramo = Q1;
   endcase
 end
endmodule
```

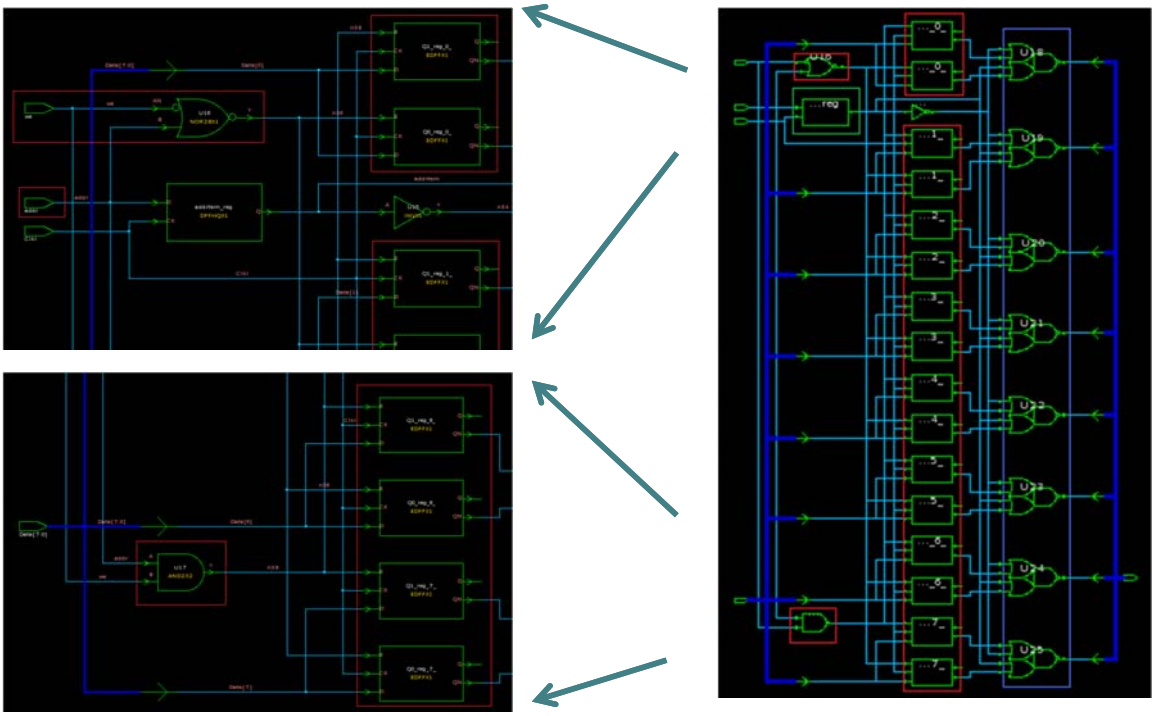

2×8隨機存取記憶體

we

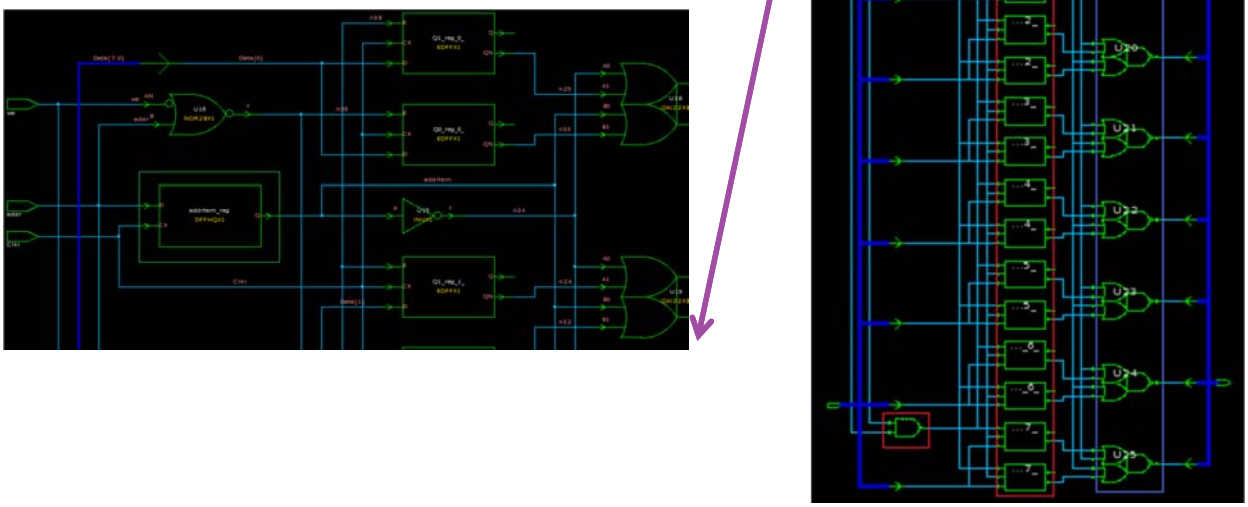
addr

clk

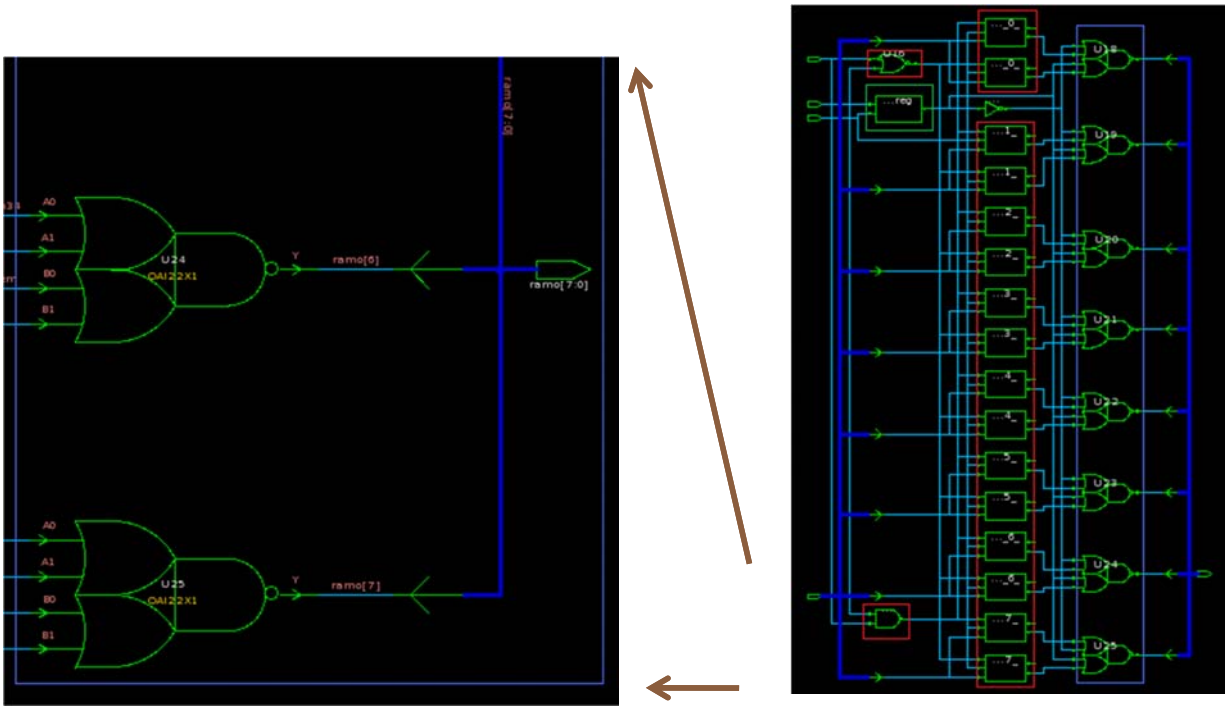
data



2×8隨機存取記憶體



2×8隨機存取記憶體



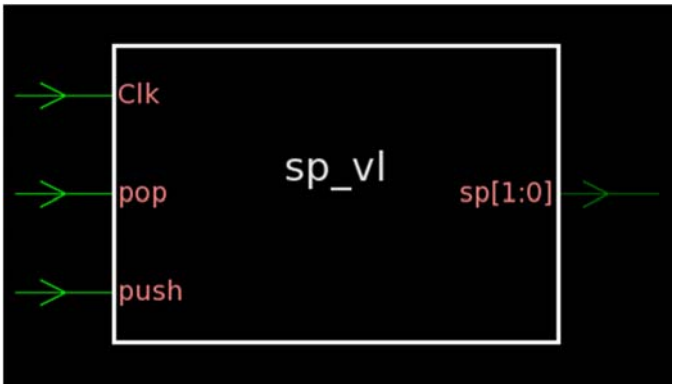
國立中興大學 National Chung Hsing University
SOC & DSP Lab.
System On a Chip & Digital Signal Processing Laboratory

堆疊指標系統

控制輸入			堆疊位址輸出
Clk	pop	push	sp
↑	1	X	sp+1
↑	0	1	sp-1
↑	0	0	不變

堆疊指標系統

- Symbol View



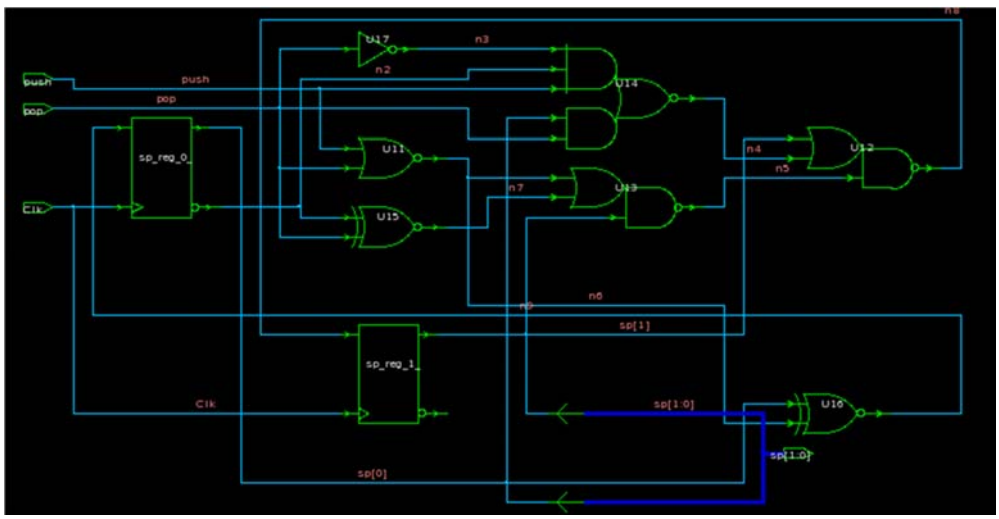
國立中興大學 National Chung Hsing University
SOC & DSP Lab. System On a Chip & Digital Signal Processing Laboratory

堆疊指標系統

```
module sp_v1 (Clk, pop, push, sp);
   input  Clk, pop, push;
   output [1:0] sp;
   reg    [1:0] sp;
   always @(posedge Clk)
    begin
        if (pop)
          sp = sp + 1;
        else if (push)
          sp = sp - 1;
    end
endmodule
```

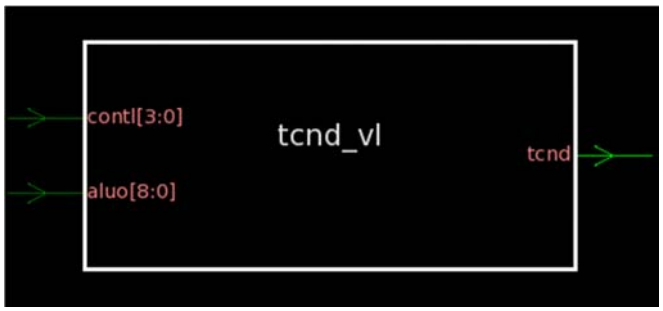

堆疊指標系統

- Schematic View



國立中興大學 National Chung Hsing University
SOC & DSP Lab.
System On a Chip & Digital Signal Processing Laboratory

- Symbol View



控制碼	指令	說明
0001	RET	回主程式
0010	JUMP	無條件跳躍 tcnd=1
0011	CALL	無條件呼叫 tcnd=1
0100	JZ	運算結果為零時(零旗號為1) tcnd =1 運算結果為非零時(零旗號為0) tcnd =0
0101	JNZ	運算結果為非零時(零旗號為0) tcnd =1 運算結果為零時(零旗號為1) tcnd =0
0110	JC	進位旗號為1時 tcnd =1 進位旗號為非1時 tcnd =0
0111	JNC	進位旗號為0時 tcnd =1 進位旗號為非0時 tcnd =0

條件碼系統

輸入			輸出	說明
contl[3:0]	aluo8	aluo[7:0]	tcnd	
0001	X	X	1	回主程式
0010	X	X	1	無條件跳躍
0011	X	X	1	無條件呼叫
0100	X	ABCDEFGH	NOT (A or B or C or D or E or F or G or H)	運算為零則跳躍
			aluo[7:0]零時 tcnd=1 aluo[7:0]為非零時 tcnd=0	

條件碼系統

0101	X	ABCDEFGH	(A or B or C or D or E or F or G or H)	運算非零則跳躍
			aluo[7:0]為非零時 tcnd=1 aluo[7:0]為零時 tcnd=0	
0110	A	X	A	進位旗號=1則跳躍
			aluo[7:0]非零時 tcnd=1 aluo[7:0]為非零時 tcnd=0	

條件碼系統

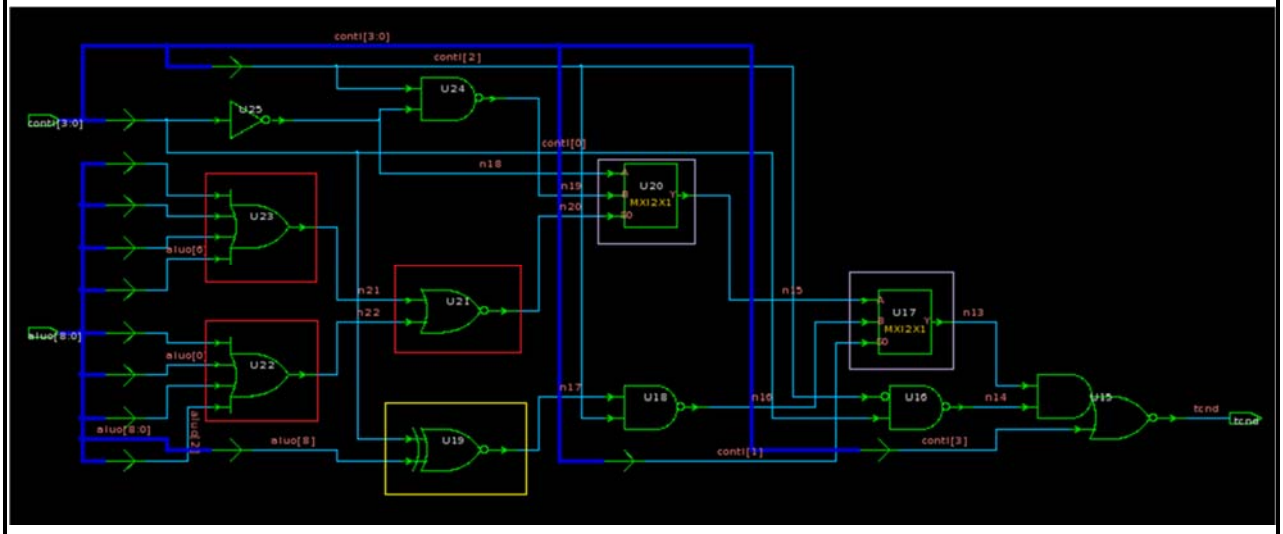
0111	A	X	A	進位旗號=0則跳躍
			aluo[7:0]非零時 tcnd=1 aluo[7:0]為非零時 tcnd=0	
其它	X	X	0	非跳躍指令

條件碼系統

```
module tcnd_v1 (cont1, aluo, tcnd);
   input   [3:0] cont1;
   input   [8:0] aluo;
   output  tcnd;
   reg     tcnd;
   always @(cont1 or aluo)
    begin
     case (cont1)
              1 :  tcnd = 1'b1;
              2 :  tcnd = 1'b1;
              3 :  tcnd = 1'b1;
              4 :  tcnd = ~|aluo[7:0];
              5 :  tcnd = |aluo[7:0];
              6 :  tcnd = aluo[8];
              7 :  tcnd = !aluo[8];
       default : tcnd = 1'b0;
     endcase
    end
endmodule
```

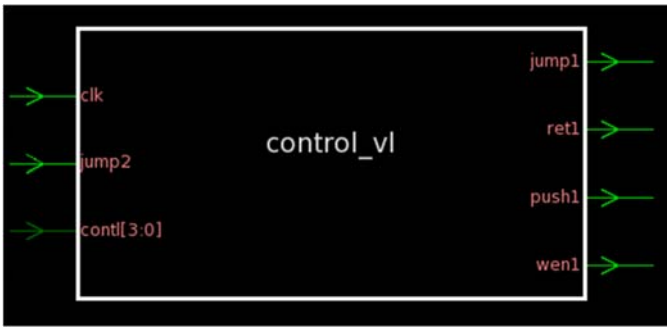
條件碼系統

- Schematic View



控制系統一

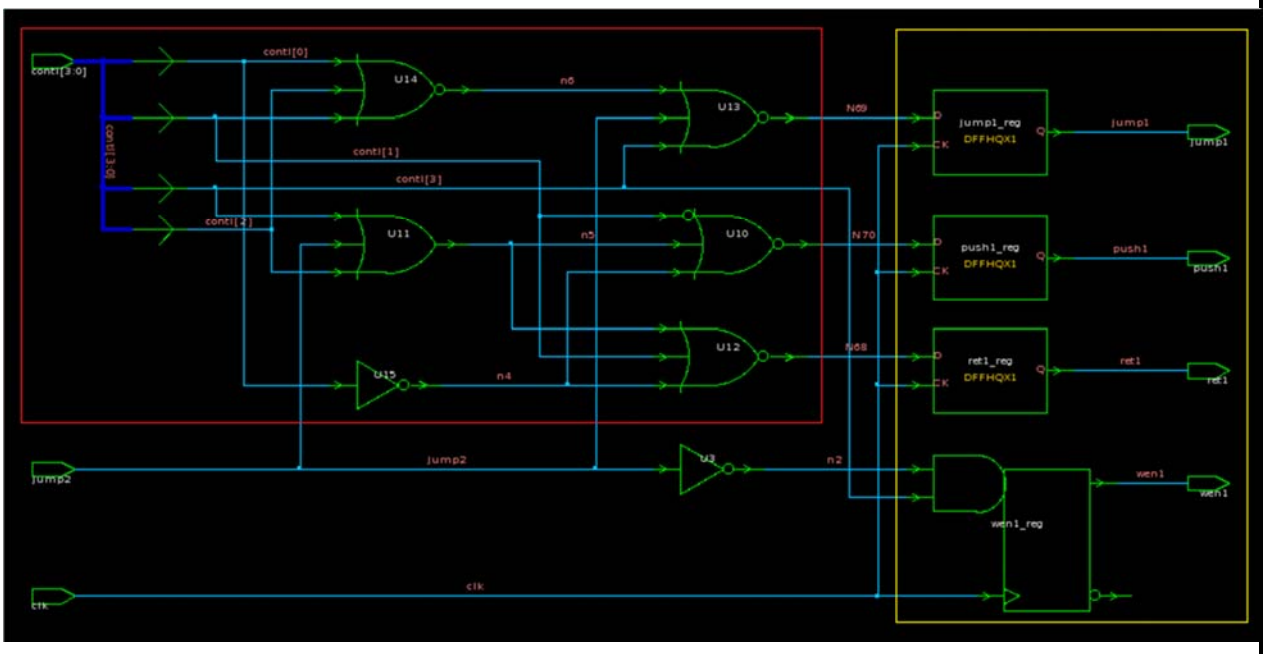
- Symbol View



- Schematic View

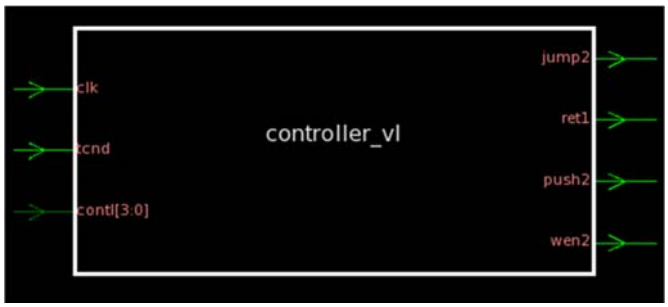
```
module control_v1 (cont1, clk, jump2, jump1, ret1, push1, wen1);
   input   [3:0] cont1;
   input   clk, jump2;
   output  jump1, ret1, push1, wen1;
   reg jump1, ret1, push1, wen1;
   always @(posedge clk)
    begin
       ret1 = 1'b0; wen1 = 1'b0; push1 = 1'b0; jump1 = 1'b0;
      case (cont1)
       1 : begin
            ret1 = !jump2; jump1 = !jump2;
           end
       2 : jump1 = !jump2;
       3 : begin
           push1 = !jump2; jump1 = !jump2;
           end
       4, 5, 6, 7 : jump1 = !jump2;
       8,9,10,11,12,13,14,15 : wen1 = !jump2;
      endcase
    end
endmodule
```


控制系統一



國立中興大學 National Chung Hsing University
SOC & DSP Lab.
System On a Chip & Digital Signal Processing Laboratory

- Symbol View



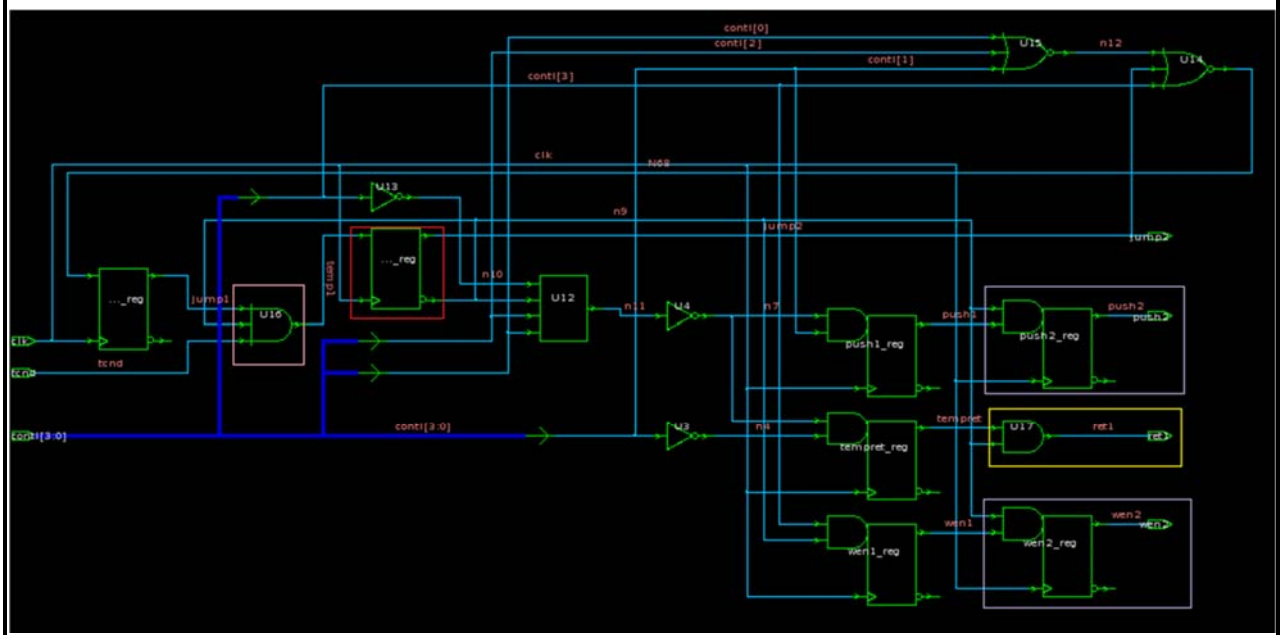
- Schematic View

```
module controller_vl (contl, clk, tcnd, jump2, ret1, push2, wen2);
   input   [3:0] contl;
   input   clk, tcnd;
   output  jump2, ret1, push2, wen2;
   reg     jump2, push2, wen2;
   wire    temp1;
   reg     jump1, tempret, push1, wen1;

   assign temp1 = jump1 & tcnd & !jump2;

   always @(posedge clk)
     begin
      jump2 = temp1;
     end
```

控制系統二

```
always @(posedge clk)
 begin
    tempret = 1'b0; wen1 = 1'b0; push1 = 1'b0; jump1 = 1'b0;
    case (cont1)
     1 : begin
          tempret = !jump2; jump1 = !jump2;
         end
     2 : jump1 = !jump2;
     3 : begin
         push1 = !jump2; jump1 = !jump2;
         end
     4, 5, 6, 7 : jump1 = !jump2;
     8,9,10,11,12,13,14,15 : wen1 = !jump2;
   endcase
 end

assign ret1 = tempret & !jump2;

always @(posedge clk)
 begin
    push2 = push1 & !jump2;
    wen2 = wen1 & !jump2;
 end
endmodule
```


控制系統二



國立中興大學 National Chung Hsing University
SOC & DSP Lab.
System On a Chip & Digital Signal Processing Laboratory