數位系統設計實習

Lecture 3

Verilog HDL (Behavior Level)

指導老師：陳勇志 教授

實習課助教：蔡沛希

Outline

- Behavior Level
- 變數:wire和reg的分別
- Always的使用
- If...else if...else的使用
- case及for迴圈
- 例題:Decoder、Encoder、Multiplexer
- LAB 3-1 and 3-2

Chapter

- Behavior Level
- 變數:wire和reg的分別
- Always的使用
- If...else if...else的使用
- case及for迴圈
- 例題:Decoder、Encoder、Multiplexer
- LAB 3-1 and 3-2

Behavioral Level

- Verilog HDL中的的最抽象的層級
- 就像我們在寫C語言程序時，不必關心底層硬件的具體實現一樣，我們只需關注電路要執行的”功能”和”行為”。
- There are two structured procedure statements in Verilog
  - –Initial statement : 定義電路初始狀態
  - –Always statement : 對於不同輸入訊號的反應

Chapter

- Behavior Level
- 變數:wire和reg的分別
- Always的使用
- If...else if...else的使用
- case及for迴圈
- 例題:Decoder、Encoder、Multiplexer
- LAB 3-1 and 3-2

Verilog HDL

- Nets:
  - Nets是用來表示不同硬體元件之間連接的線。
  - Nets就像電線，能夠將信號從一個部件傳送到另一個部件。這些連接點上的信號是持續的。
  - Nets 主要通過關鍵字 wire 來定義。

All outputs of assign statement must be wires !

Verilog HDL

- Registers(暫存器):
  – Registers在Verilog中的概念類似於計算機的臨時記憶單元，它可以保存數據直到被更新為止。

  – In Verilog, 我們使用 register來存儲中間值或者暫存的結果。

  – Register 主要透過關鍵字 reg 來定義。

All outputs of always blocks must be reg!

Verilog HDL

- Dataflow Modeling：

  - 資料流敘述的前後次序不重要。

  *出現在 = 敘述左邊的信號必須是wire資料型態

  *例如 assign out = A & B;

- Behavior Modeling：

  - always方塊內敘述的前後次序很重要。

  但若其間沒有相關性,次序對調不影響結果

  *出現在 = 敘述左邊的信號必須宣告reg資料型態

  *例如 always@ (A,B)

  out = A & B;

Verilog HDL

- Gate-Level Modeling

```
ex:  and a1( a, b, c );
     nor a0(a, b, c );
```

- Data-Flow Modeling

```
ex:  assign a = ~b;
```

- Behavior Modeling

```
ex:  always @( load )
     begin
       if ( load )
         begin
           a = b;
           c = a;
         end
     end
```

Chapter

- Behavior Level
- 變數:wire和reg的分別
- Always的使用
- If...else if...else的使用
- case及for迴圈
- 例題:Decoder、Encoder、Multiplexer
- LAB 3-1 and 3-2

Always Statement

- 驅動某值至reg (等號的左式必為reg，右式可為net 或 reg)
- 事件與事件之間需用” or ”或” , ”分開
- 當”事件”有變化時(0→1 、1→0)，則會執行”敘述”
- 事件中可以使用正緣觸發posedge(0→1)和負緣觸發negedge(1→0)
- 句法 ：

```
always @( 事件1, 事件2, … )
    begin
    敘述1;
    敘述2;
    … … …
end
```

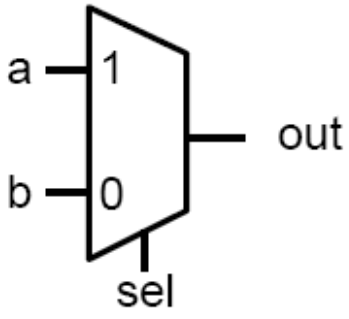
Always Block

Combinational

```
module combinational(a, b, sel,
                             out);
  input a, b;
  input sel;
  output out;
  reg out;

  always @ (a or b or sel)
  begin
    if (sel) out = a;
    else out = b;
  end

endmodule
```

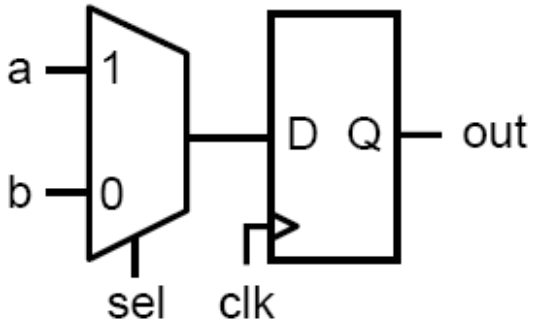


Sequential

```
module sequential(a, b, sel,
                          clk, out);
  input a, b;
  input sel, clk;
  output out;
  reg out;

  always @ (posedge clk)
  begin
    if (sel) out <= a;
    else out <= b;
  end

endmodule
```



Chapter

- Behavior Level
- 變數:wire和reg的分別
- Always的使用
- If...else if...else的使用
- case及for迴圈
- 例題:Decoder、Encoder、Multiplexer
- LAB 3-1 and 3-2

Conditional Statements
if … else…

```
if (condition 1)
   begin
        ..................
   end
else if (condition 2)
   begin
        ..................
   end
else
   begin
        ..................
   end
```

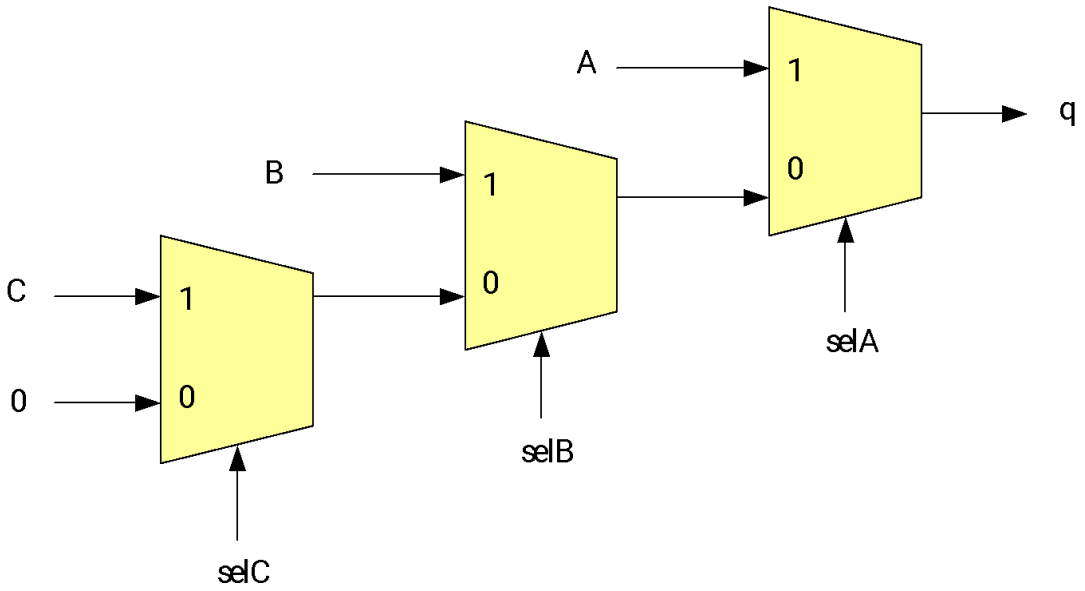
```
if (opcode == 2'b00)
    out = a + b;
else if(opcode == 2'b01)
    out = a - b;
else if(opcode == 2'b10)
    begin
        out = a * b;
    end
else
    out = a ^ b;
```

if … else statement

EX :

```
always @ ( selA or selB or selC or A or B or C
)
begin
if ( selA ) q = A;
      else if (selB )      q = B;
      else if (selC )      q = C;
      else                 q = 0;
end
```

if ... else statement



Chapter

- Behavior Level
- 變數:wire和reg的分別
- Always的使用
- If...else if...else的使用
- case及for迴圈
- 例題:Decoder、Encoder、Multiplexer
- LAB 3-1 and 3-2

Conditional Statements case

```
case (state)
    2'b00 : .....
    2'b01 : .....
    2'b10 : .....
    2'b11 : .....
    default : .....
endcase
```

```
case (opcode)
    2'b00 : out = a + b;
    2'b01 : out = a – b;
    2'b10 : out = a * b;
    2'b11 : out = a ^ b
    default :
endcase
```

case statement


```
module case_ex (selA, selB, selC, A, B, C, q );
    input …
    …
    always @ ( selA or selB or selC or A or B or C )
    begin
      case( { selA, selB, selC } )
        3'b100: q = A;
        3'b010: q = B;
        3'b001: q = C;
        default: q = 0;
      endcase
    end
endmodule
```

for loop

EX :

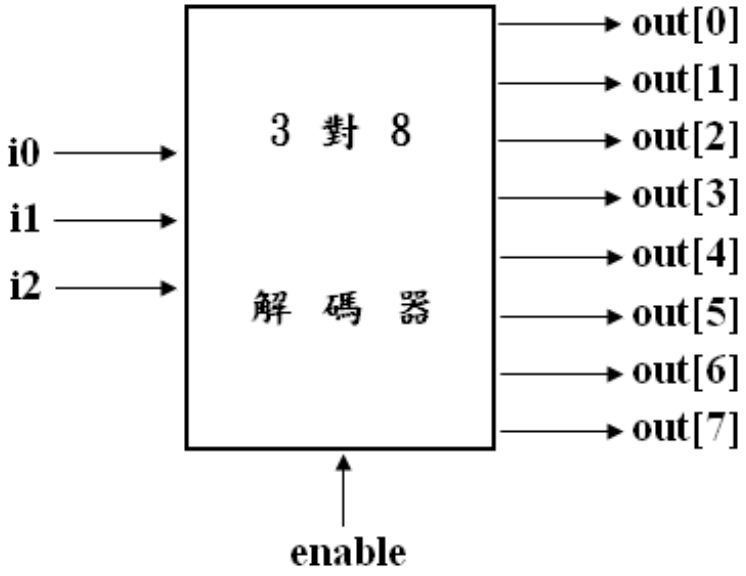
```
integer index;
for(index=low_range; index<high_range;
index=index+step)
        begin
                < statement >;
        end
```

Chapter

- Behavior Level
- 變數:wire和reg的分別
- Always的使用
- If...else if...else的使用
- case及for迴圈
- 例題:Decoder、Encoder、Multiplexer
- LAB 3-1 and 3-2

Decoder

- 具有Enable功能之3對8解碼器：



Input			Output	
i2	i1	i0	enable	out[7:0]
X	X	X	0	00000000
0	0	0	1	00000001
0	0	1	1	00000010
0	1	0	1	00000100
0	1	1	1	00001000
1	0	0	1	00010000
1	0	1	1	00100000
1	1	0	1	01000000
1	1	1	1	10000000

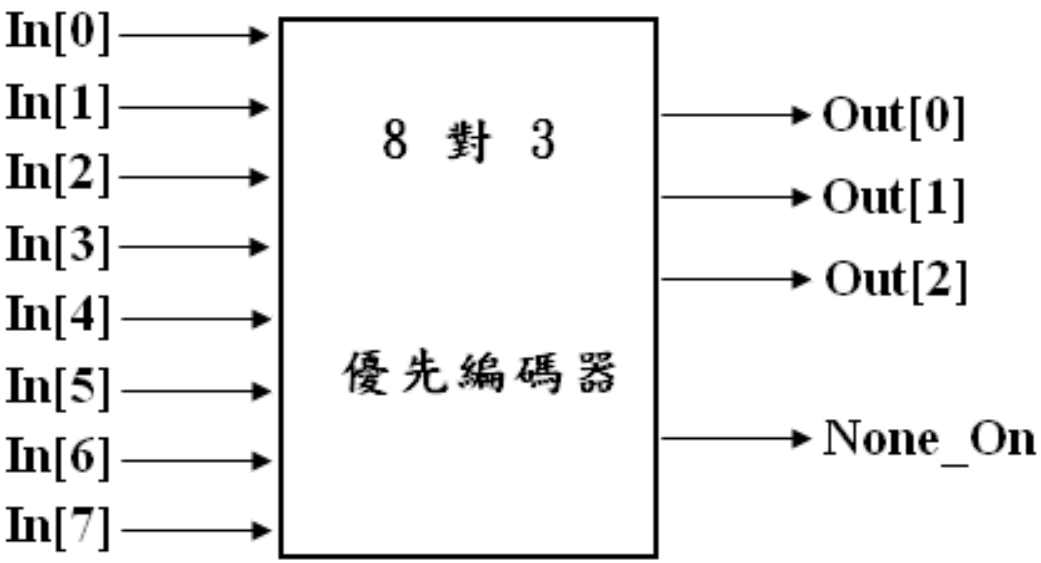
Decoder


```
module decoder_3x8(i0, i1, i2, en, out);
input i0, i1, i2, en;
output [7:0]out;
reg [7:0]out;
always @(*)
  begin
    if(en)
      begin
        case({i2, i1, i0})
                3'b000: out = 8'b00000001;
                3'b001: out = 8'b00000010;
                3'b010: out = 8'b00000100;
                3'b011: out = 8'b00001000;
                3'b100: out = 8'b00010000;
                3'b101: out = 8'b00100000;
                3'b110: out = 8'b01000000;
                3'b111: out = 8'b10000000;
          endcase
      end
  else
    begin
        out = 8'b00000000;
    end
end

endmodule
```

Encoder

- 8對3的優先編碼器（Priority Encoder）：



Encoder

Input								Output			
In[7]	In[6]	In[5]	In[4]	In[3]	In[2]	In[1]	In[0]	Out[2]	Out[1]	Out[0]	None_ON
0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	1	X	0	0	1	0
0	0	0	0	0	1	X	X	0	1	0	0
0	0	0	0	1	X	X	X	0	1	1	0
0	0	0	1	X	X	X	X	1	0	0	0
0	0	1	X	X	X	X	X	1	0	1	0
0	1	X	X	X	X	X	X	1	1	0	0
1	X	X	X	X	X	X	X	1	1	1	0

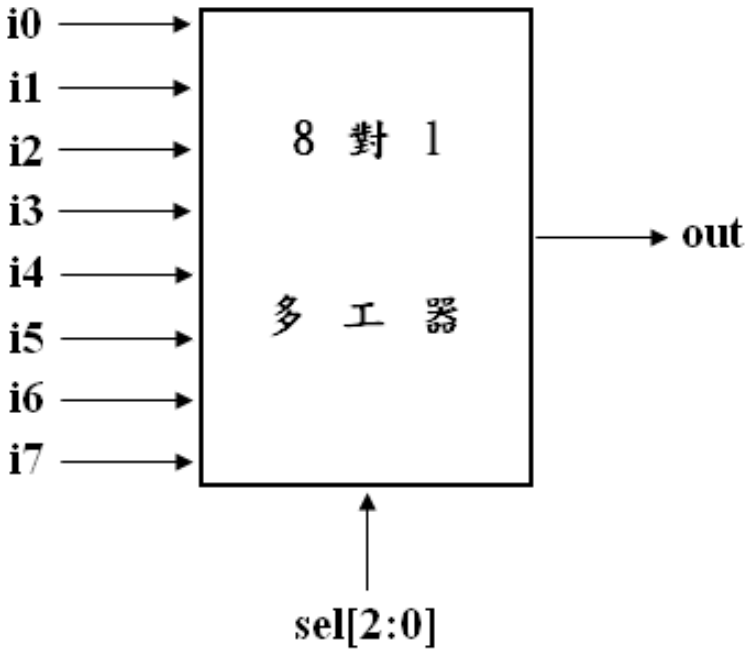
Encoder


```
module  Encoder_8X3 (In, Out,
None_ON);
input    [7:0] In;
output   [2:0] Out;
output       None_ON;
reg      [2:0] Out;
reg          None_ON;
```

```
always @(In)
   begin
          integer  i;
          Out = 0;
           None_ON = 1;
           for (i = 7; i >=0; i = i -1)
                    begin
                    if (In[i])
                              begin
                               Out = i;
                              None_ON = 0;
                              end
                    end
  end
endmodule
```

Multiplexer

- 8對1多工器：



Input			output
sel[2]	sel[1]	sel[0]	out
0	0	0	i0
0	0	1	i1
0	1	0	i2
0	1	1	i3
1	0	0	i4
1	0	1	i5
1	1	0	i6
1	1	1	i7

Multiplexer

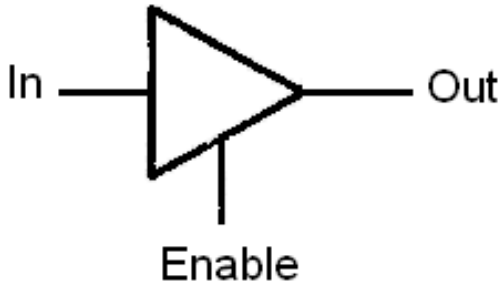

```
module MUX_8X1(sel, i0, i1, i2, i3, i4, i5, i6, i7, out);
input i0, i1, i2, i3, i4, i5, i6, i7;
input [2:0]sel;
output out;
reg out;
always @(sel or i0 or i1 or i2 or i3 or i4 or i5 or i6 or i7)
//always@(*)
        case(sel)
                3'b000: out = i0;
                3'b001: out = i1;
                3'b010: out = i2;
                3'b011: out = i3;
                3'b100: out = i4;
                3'b101: out = i5;
                3'b110: out = i6;
                3'b111: out = i7;
        endcase
endmodule
```

Chapter

- Behavior Level
- 變數:wire和reg的分別
- Always的使用
- If...else if...else的使用
- case及for迴圈
- 例題:Decoder、Encoder、Multiplexer
- LAB 3-1 and 3-2

補充：Verilog HDL

- 三態閘：

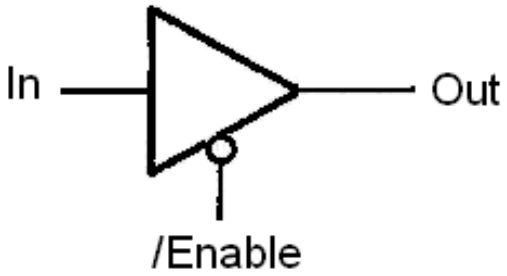


assign Out=(Enable)? In:1'bz;

Enable	In	Out
0	0	Z
0	1	Z
1	0	0
1	1	1

buf_name name(output, input, control);

bufif1
當 control = 1 則相當緩衝器
當 control = 0 則輸出呈現高阻抗



/Enable	In	Out
0	0	0
0	1	1
1	0	Z
1	1	Z

bufif0
當 control = 0 則相當緩衝器
當 control = 1 則輸出呈現高阻抗

Lab 3

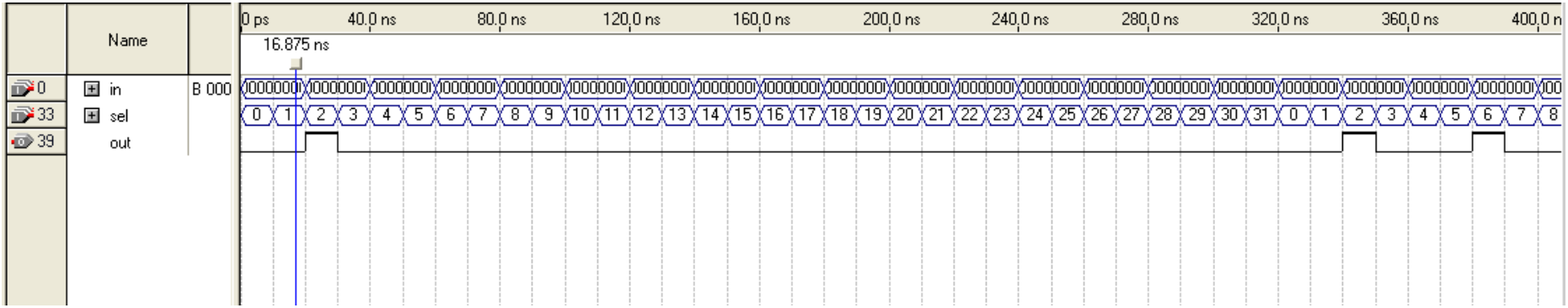
- LAB 3-1：

  完成一個32x1的多工器。

- LAB 3-2：

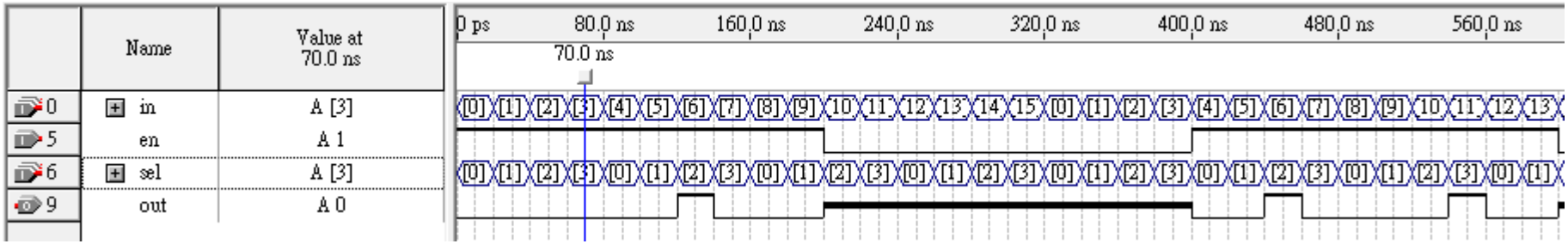
  請利用4個三態閘與1個2x4解碼器，或用Behavior的方式描述其功能，並設計一個有enable功能的4對1多工器。

LAB 3-1 :



輸入: [31:0]in, [4:0]sel;
輸出:out;

LAB 3-2 :



輸入: [3:0]in, [1:0]sel,
en;
輸出: out;

LAB 3

下課前繳交至moodle：

上傳verilog.v

波形截圖