Soft Computing And Next Generation Network Lab.
Hanyang Univ.

# Verilog HDL
# 조합회로

# “begin-end” vs “fork-join”

## begin ~ end

- 이 안에 있는 명령들은 순차적으로 실행
  앞에서 실행된 것이 뒤의 결과에 영향을 줌
- 한두 개가 아닌 여러 명령들이
  일정 delay 후에 시작되어야 한다면,
  begin ~ end 구문 사용이 적합

## fork ~ join

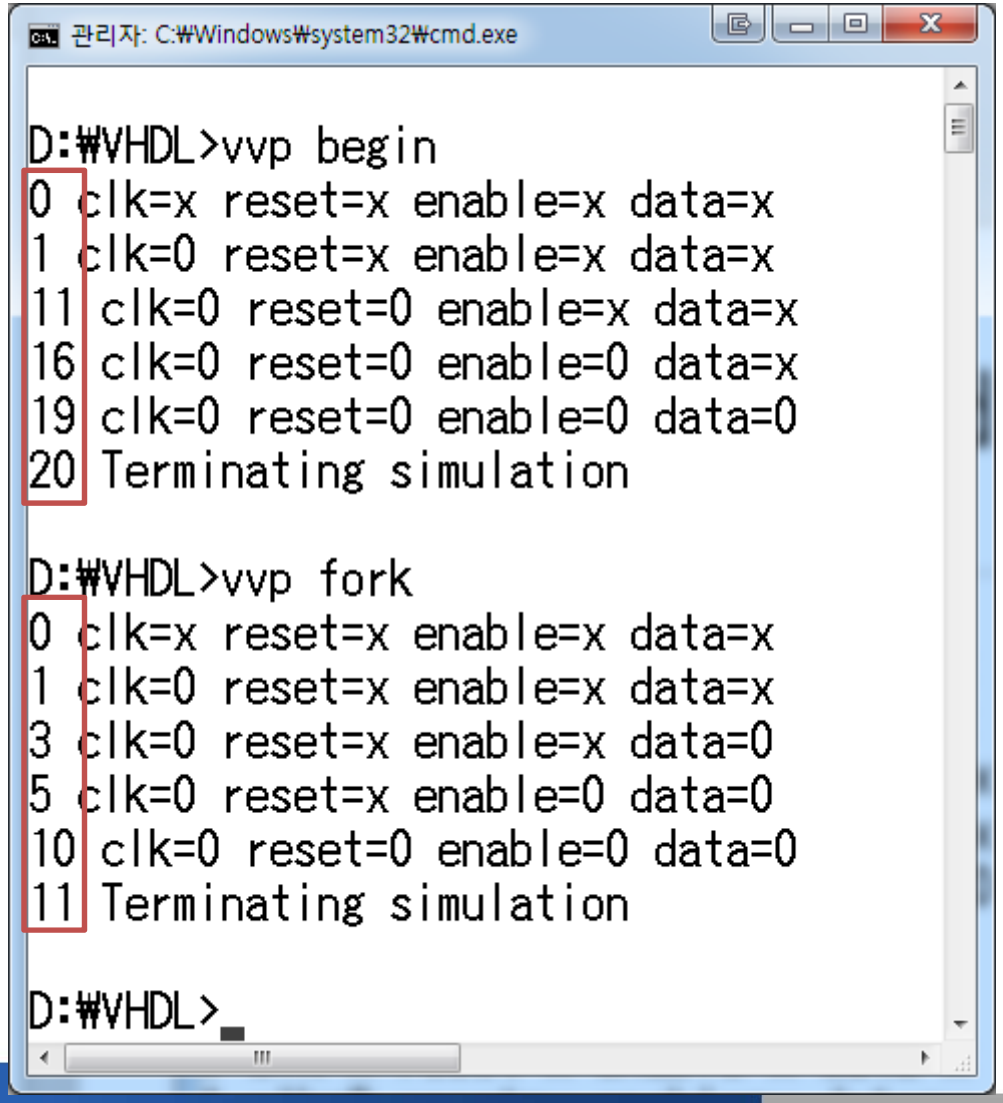
- 이 안에 있는 명령들은 병렬적으로 실행
- clock에 따라 동기화되어 작동하는
  synchronous system을 표현하기에 적합

# “begin-end” vs “fork-join”

```
module initial_begin_end();
reg clk,reset,enable,data;

initial begin
 $monitor("%g clk=%b reset=%b enable=%b
data=%b",
    $time, clk, reset, enable, data);

 #1  clk = 0;
 #10 reset = 0;
 #5  enable = 0;
 #3  data = 0;

 #1 $display ("%g Terminating simulation", $time);
 $finish;
end

endmodule
```

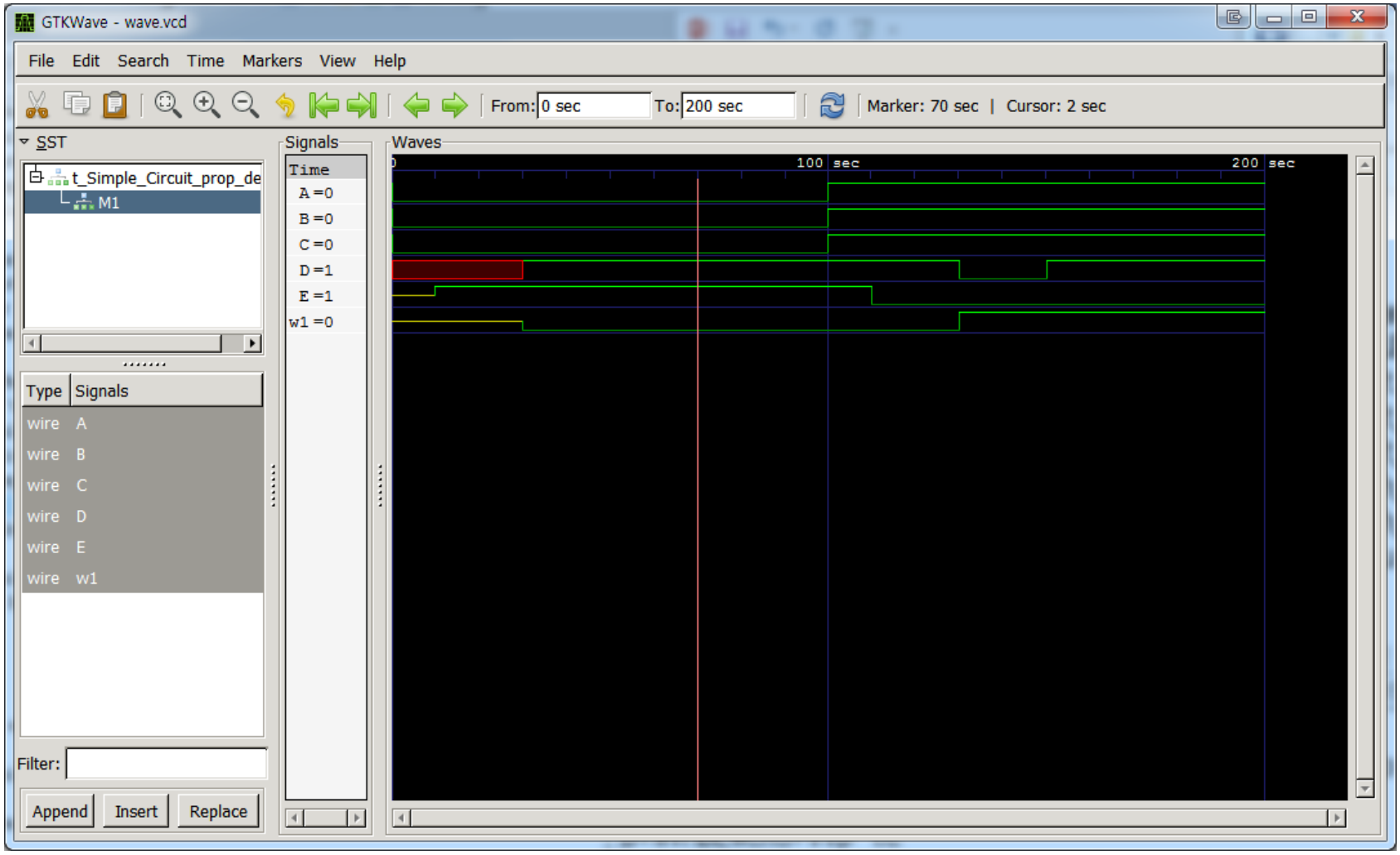
```
module initial_fork_join();
reg clk,reset,enable,data;

initial begin
 $monitor("%g clk=%b reset=%b enable=%b
data=%b",
    $time, clk, reset, enable, data);
 fork
    #1  clk = 0;
    #10 reset = 0;
    #5  enable = 0;
    #3  data = 0;
 join
 #1 $display ("%g Terminating simulation", $time);
 $finish;
end

endmodule
```

# “begin-end” vs “fork-join”



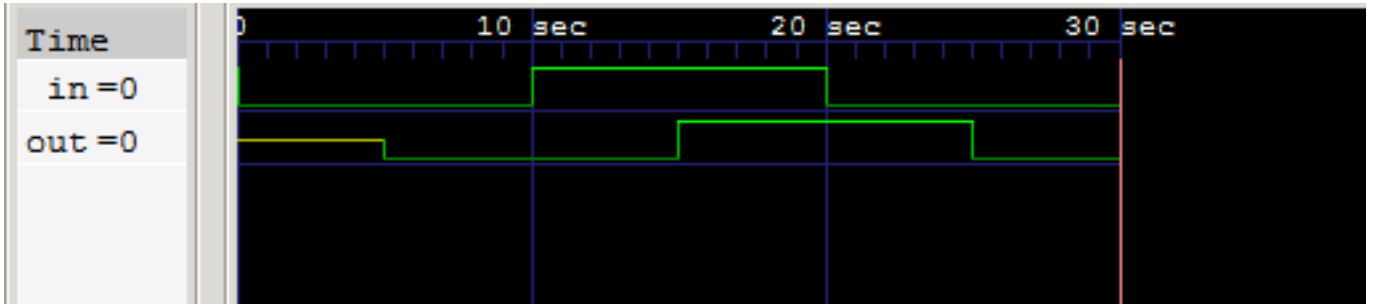
# 게이트 지연

```
1  // Verilog model of simple circuit with propagation delay
2  module Simple_Circuit_prop_delay (A, B, C, D, E);
3    output  D, E;
4    input   A, B, C;
5    wire    w1;
6
7    and   #(30) G1 (w1, A, B);
8    not   #(10) G2 (E, C);
9    or    #(20) G3 (D, w1, E);
10 endmodule
11
12 // Testbench for Simple_Circuit_prop_delay
13 // `timescale 1 ns / 100 ps
14 module t_Simple_Circuit_prop_delay;
15   wire  D, E;
16   reg A, B, C;
17
18   Simple_Circuit_prop_delay M1 (A, B, C, D, E);  // Instance name required
19
20   initial
21     begin
22       $dumpfile("wave.vcd");
23       $dumpvars(0, t_Simple_Circuit_prop_delay);
24       A = 1'b0; B = 1'b0; C = 1'b0;
25       #100 A = 1'b1; B = 1'b1; C = 1'b1;
26     end
27
28   initial #200 $finish;
29
30 endmodule
```


GTKWave - wave.vcd
File Edit Search Time Markers View Help
From: 0 sec
To: 200 sec
Marker: 70 sec | Cursor: 2 sec
SST
t_Simple_Circuit_prop_de
M1
Type Signals
wire A
wire B
wire C
wire D
wire E
wire w1
Filter:
Append Insert Replace
Signals
Time
A =0
B =0
C =0
D =1
E =1
w1 =0
Waves
100 sec
200 sec

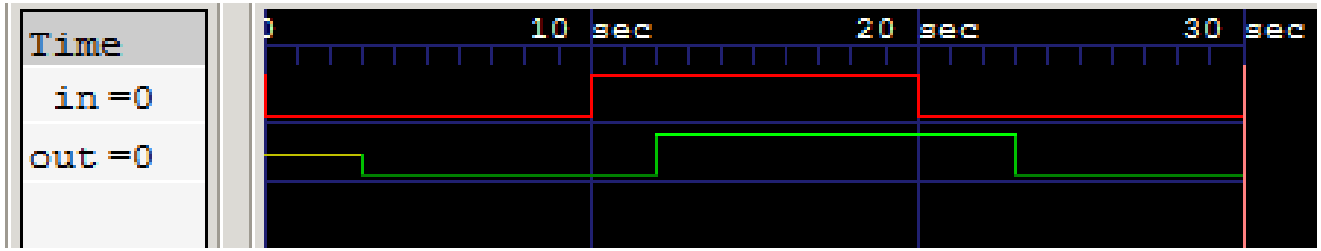
# single Delay

```
module  buf_gate ();
reg in;
wire out;

buf #(5) (out,in);

initial begin
  $monitor ("Time = %g in = %b out=%b",
$time, in, out);
  in = 0;
  #10 in = 1;
  #10 in = 0;
  #10 $finish;
end

endmodule
```



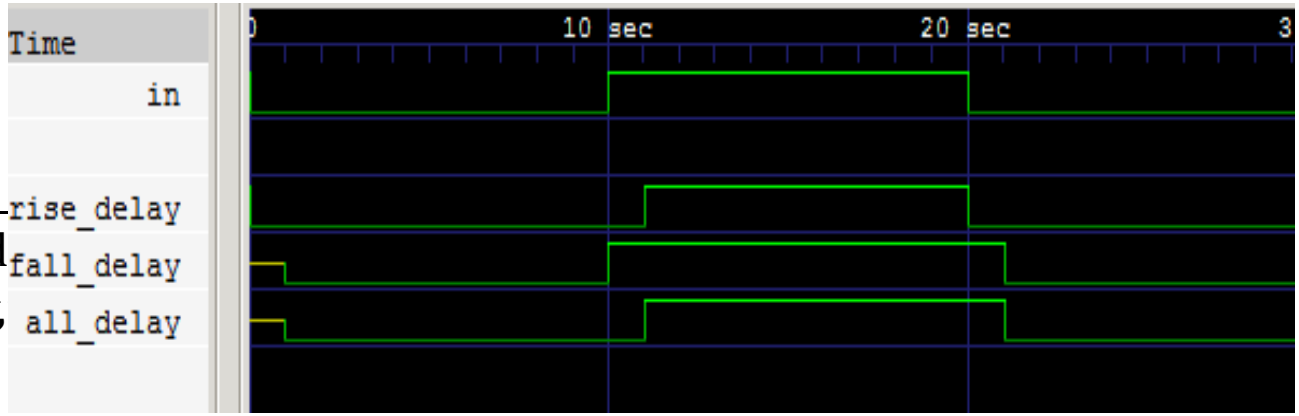
# Two Delays

```
module  buf_gate1 ();
reg in;
wire out;

buf #(2,3) (out,in);

initial begin
  $dumpfile("two_delay.vcd");
  $dumpvars;
  $monitor ("Time = %g in = %b out=%b", $time, in, out);
  in = 0;
  #10 in = 1;
  #10 in = 0;
  #10 $finish;
end

endmodule
```



# All delay

```
module delay();
 reg in;
 wire rise_delay, fall_delay, all_delay;

 initial begin
  $monitor (
    "Time=%g in=%b rise_delay=%b fall_delay=%b all_delay=%b",
    $time, in, rise_delay, fall_delay, all_delay);
  in = 0;
  #10 in = 1;
  #10 in = 0;
  #20 $finish;
 end

 buf #(1,0)U_rise (rise_
 buf #(0,1)U_fall (fall_d
 buf #1 U_all (all_delay,

endmodule
```



```
// This code shows how UDP body
primitive udp_body (
a, // Port a
b, // Port b
c  // Port c
);
output a;
input b,c;

// UDP function code here
// A = B | C;
table
 // B  C    : A
    ?  1    : 1;
    1  ?    : 1;
    0  0    : 0;
endtable

endprimitive
```

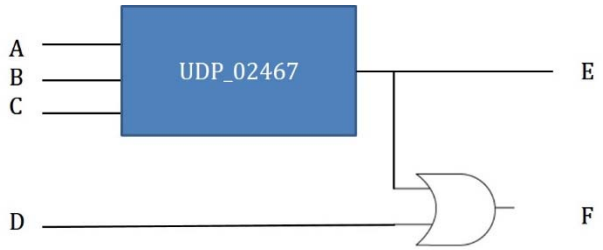
```
module udp_body_tb();

reg b,c;
wire a;

udp_body udp (a,b,c);

initial begin
  $monitor(" B = %b C = %b  A = %b",b,c,a);
  b = 0;
  c = 0;
  #1 b = 1;
  #1 b = 0;
  #1 c = 1;
  #1 b = 1'bx;
  #1 c = 0;
  #1 b = 1;
  #1 c = 1'bx;
  #1 b = 0;
  #1 $finish;
end

endmodule
```

| | | |
|---|---|---|
| ? | 0 or 1 or X | ? means the variable can be 0 or 1 or x |
| b | 0 or 1 | Same as ?, but x is not included |
| f | (10) | Falling edge on an input |
| r | (01) | Rising edge on an input |
| p | (01) or (0x) or (x1) or (1 z) or (z1) | Rising edge including x and z |
| n | (10) or (1x) or (x0) or (0 z) or (z0) | Falling edge including x and z |
| * | (??) | All transitions |
| - | no change | No Change |

# User-defined Primitive

```
primitive UDP_02467 (D, A, B, C);
  output D;
  input    A, B, C;
// Truth table for D = f (A, B, C) = ??(0, 2, 4, 6, 7);
  table
//          A          B          C          :          D          // Column header comment
            0          0          0          :          1;
            0          0          1          :          0;
            0          1          0          :          1;
            0          1          1          :          0;
            1          0          0          :          1;
            1          0          1          :          0;
            1          1          0          :          1;
            1          1          1          :          1;
  endtable
endprimitive
```

# User-defined Primitive




```
// Verilog model: Circuit instantiation of Circuit_UDP_02467
module Circuit_with_UDP_02467 (e, f, a, b, c, d);
  output   e, f;
  input                a, b, c, d;

 UDP_02467          M0 (e, a, b, c);
  and                    (f, e, d);          //Option gate instance name omitted
endmodule
```

# User-defined Primitive

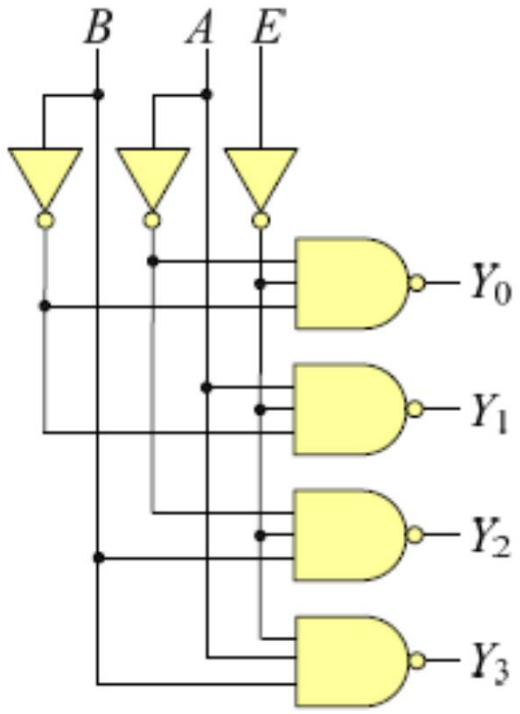
```
23 // Verilog model: User-defined Primitive
24 primitive UDP_02467 (D, A, B, C);
25   output D;
26   input A, B, C;
27 // Truth table for D = f (A, B, C) = ??(0, 2, 4, 6, 7);
28   table
29 //  A B C : D   // Column header comment
30   0 0 0 : 1;
31   0 0 1 : 0;
32   0 1 0 : 1;
33   0 1 1 : 0;
34   1 0 0 : 1;
35   1 0 1 : 0;
36   1 1 0 : 1;
37   1 1 1 : 1;
38   endtable
39 endprimitive
40 // Verilog model: Circuit instantiation of Circuit_UDP_02467
41 module Circuit_with_UDP_02467 (e, f, a, b, c, d);
42   output  e, f;
43   input   a, b, c, d;
44
45  UDP_02467  M0 (e, a, b, c);
46   and          (f, e, d);    //Option gate instance name omitted
47 endmodule
```

# t_Circuit_with_UDP_02467.v

```
module t_Circuit_with_UDP_02467;
 wire          E, F;
 reg           A, B, C, D;
 Circuit_with_UDP_02467 m0 (E, F, A, B, C, D);

 initial #100 $finish;
 initial begin
                $dumpfile("udp_wave.vcd");
                 $dumpvars(0, t_Circuit_with_UDP_02467);
 fork
  A = 0; B = 0; C = 0; D = 0;
  #40 A = 1;
  #20 B = 1;
  #40 B = 0;
  #60 B = 1;
  #10 C = 1; #20 C = 0; #30 C = 1; #40 C = 0; #50 C = 1; #60 C = 0; #70 C = 1;
  #20 D = 1;
 join
end

endmodule
```

# 인에이블 입력을 갖는 2 to 4 라인 디코더

| 입력 | | | 출력 | | | |
|---|---|---|---|---|---|---|
| $E$ | $B$ | $A$ | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ |
| 1 | × | × | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 |



# decoder_2x4_gates.v

testbench 에

```
#10 enable = 0;  A = 0; B = 0;
#10 enable = 0;  A = 0; B = 1;
#10 enable = 0;  A = 1; B = 0;
#10 enable = 0;  A = 1; B = 1;
#10 enable = 1;  A = 0; B = 0;
#10 enable = 1;  A = 0; B = 1;
#10 enable = 1;  A = 1; B = 0;
#10 enable = 1;  A = 1; B = 1;
```

wave form으로 출력

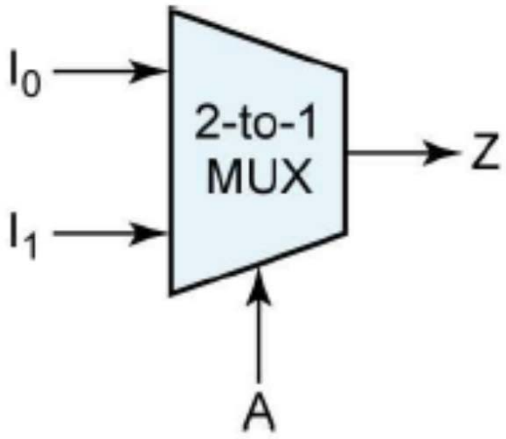
# decoder_2x4_gates.v

```
1  // Gate-level description of 2-to-4-line decoder
2  // See Fig. 4-19. Note: The figure uses symbol E, but the
3  // Verilog model uses enable to clearly indicate functionality.
4
5  module decoder_2x4_gates (D, A, B, enable);
6    output  [0: 3]  D;
7    input   A, B;
8    input   enable;
9    wire    A_not, B_not, enable_not;
10   not
11     G1   (A_not, A),
12     G2   (B_not, B),
13     G3   (enable_not, enable);
14   nand
15     G4   (D[0], A_not, B_not, enable_not),
16     G5   (D[1], A_not, B, enable_not),
17     G6   (D[2], A, B_not, enable_not),
18     G7   (D[3], A, B, enable_not);
19 endmodule
```

testbench 예

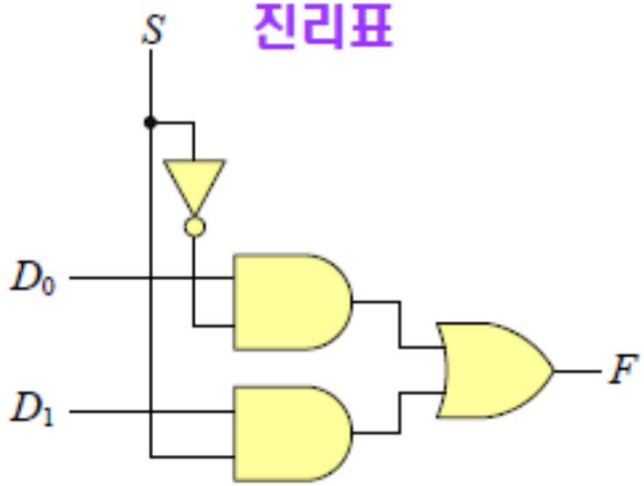
```
#10 enable = 0;  A = 0; B = 0;
#10 enable = 0;  A = 0; B = 1;
#10 enable = 0;  A = 1; B = 0;
#10 enable = 0;  A = 1; B = 1;
#10 enable = 1;  A = 0; B = 0;
#10 enable = 1;  A = 0; B = 1;
#10 enable = 1;  A = 1; B = 0;
#10 enable = 1;  A = 1; B = 1;
```

# 2x1 mux




| 선택선 | 출력 |
|---|---|
| $S$ | $F$ |
| 0 | $D_0$ |
| 1 | $D_1$ |

진리표




회로도

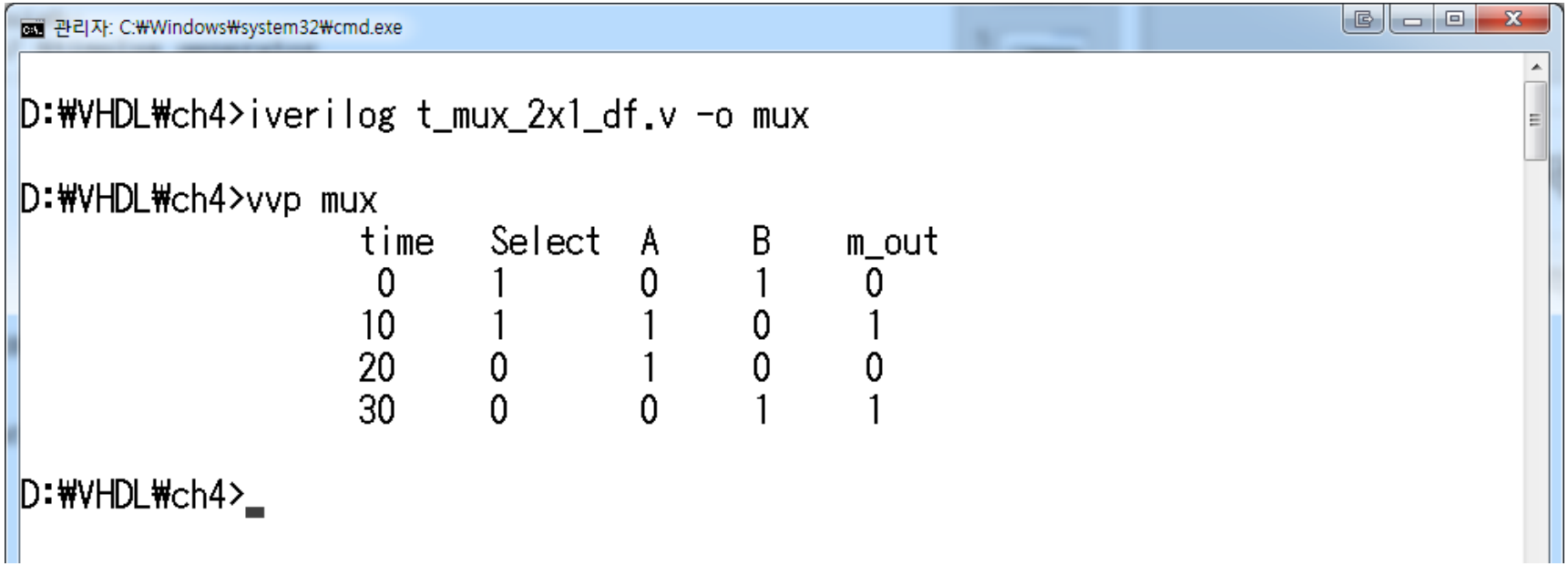
# mux_2x1_df.v

```
// Dataflow description of 2-to-1 line multiplexer
// from Example 4-6
module mux_2x1_df (
  output      m_out,
  input       A, B,
  input       select
 );
  assign m_out = (select)? A : B;
 endmodule
```

```
module mux_2x1_df (m_out, A, B, select);
 output m_out,
 input   A, B;
 input   select;
```

testbench 에

```
t_select = 1; t_A = 0; t_B = 1;
   #10     t_A = 1; t_B = 0;
   #10     t_select = 0;
   #10     t_A = 0; t_B = 1;
```


관리자: C:₩Windows₩system32₩cmd.exe
D:₩VHDL₩ch4>iverilog t_mux_2x1_df.v -o mux
D:₩VHDL₩ch4>vvp mux
time Select A B m_out
0 1 0 1 0
10 1 1 0 1
20 0 1 0 0
30 0 0 1 1
D:₩VHDL₩ch4>