Md Maniruzzaman
ID: 155009

EE461-HW#03

Q1: Answer

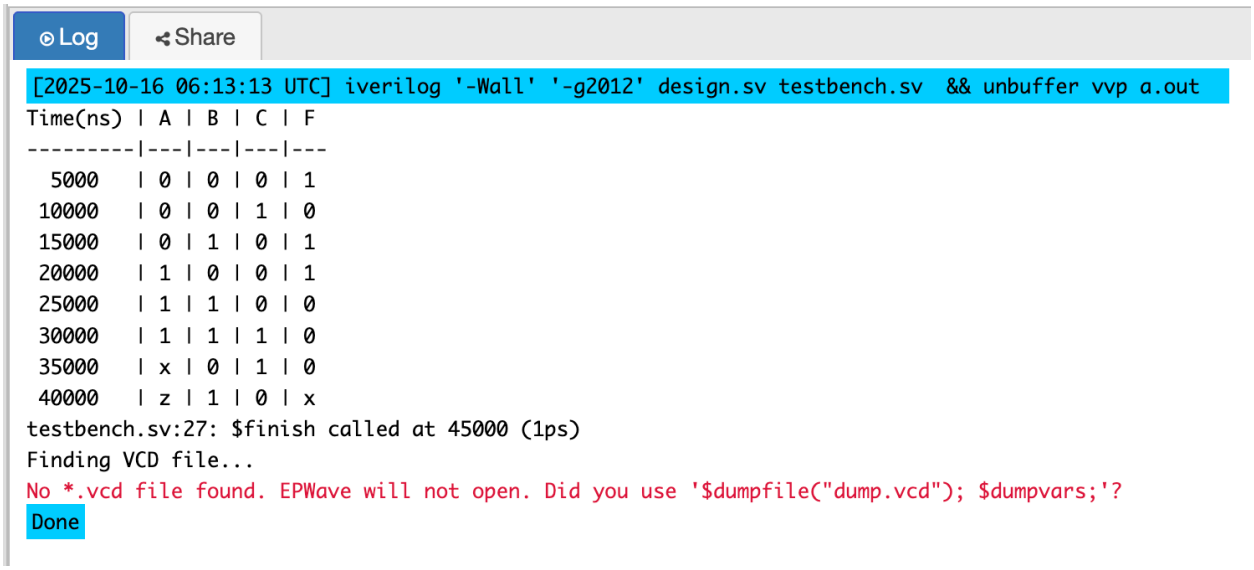
testbench

```
`timescale 1ns/1ps
module testbench;
   reg a_r;
   reg b_r;
   reg c_r;
   wire f_w;

   designModule u(
      .a_i(a_r),
      .ab_i(b_r),
      .c_i(c_r),
      .f_o(f_w)
   );

   initial begin
     $display("Time(ns) | A | B | C | F");
     $display("---------|---|---|---|---");

     a_r = 0; b_r = 0; c_r = 0; #5;  $display("%6t   | %b | %b | %b | %b", $time, a_r, b_r, c_r, f_w);
     a_r = 0; b_r = 0; c_r = 1; #5;  $display("%6t   | %b | %b | %b | %b", $time, a_r, b_r, c_r, f_w);
     a_r = 0; b_r = 1; c_r = 0; #5;  $display("%6t   | %b | %b | %b | %b", $time, a_r, b_r, c_r, f_w);
     a_r = 1; b_r = 0; c_r = 0; #5;  $display("%6t   | %b | %b | %b | %b", $time, a_r, b_r, c_r, f_w);
     a_r = 1; b_r = 1; c_r = 0; #5;  $display("%6t   | %b | %b | %b | %b", $time, a_r, b_r, c_r, f_w);
     a_r = 1; b_r = 1; c_r = 1; #5;  $display("%6t   | %b | %b | %b | %b", $time, a_r, b_r, c_r, f_w);
     a_r = 1'bx; b_r = 1'b0; c_r = 1'b1; #5;      $display("%6t   | %b | %b | %b | %b", $time, a_r, b_r, c_r, f_w);
     a_r = 1'bz; b_r = 1'b1; c_r = 1'b0; #5;      $display("%6t   | %b | %b | %b | %b", $time, a_r, b_r, c_r, f_w);
     #5 $finish;
   end
endmodule
```

design

```
`timescale 1ns/1ps
module designModule (
   input  wire a_i,
   input  wire ab_i,
   input  wire c_i,
   output wire f_o
);
   supply1 Vdd;
   supply0 Gnd;

   wire n_ab;
   wire p_ab;

   // PMOS Pull-Up
   pmos (p_ab, Vdd, a_i);
   pmos (p_ab, Vdd, ab_i);
   pmos (f_o, p_ab, c_i);

   // NMOS Pull-Down
   nmos (n_ab, Gnd, a_i);
   nmos (n_ab, Gnd, ab_i);
   nmos (f_o, n_ab, c_i);

   pulldown(f_o);
endmodule
```

Output:

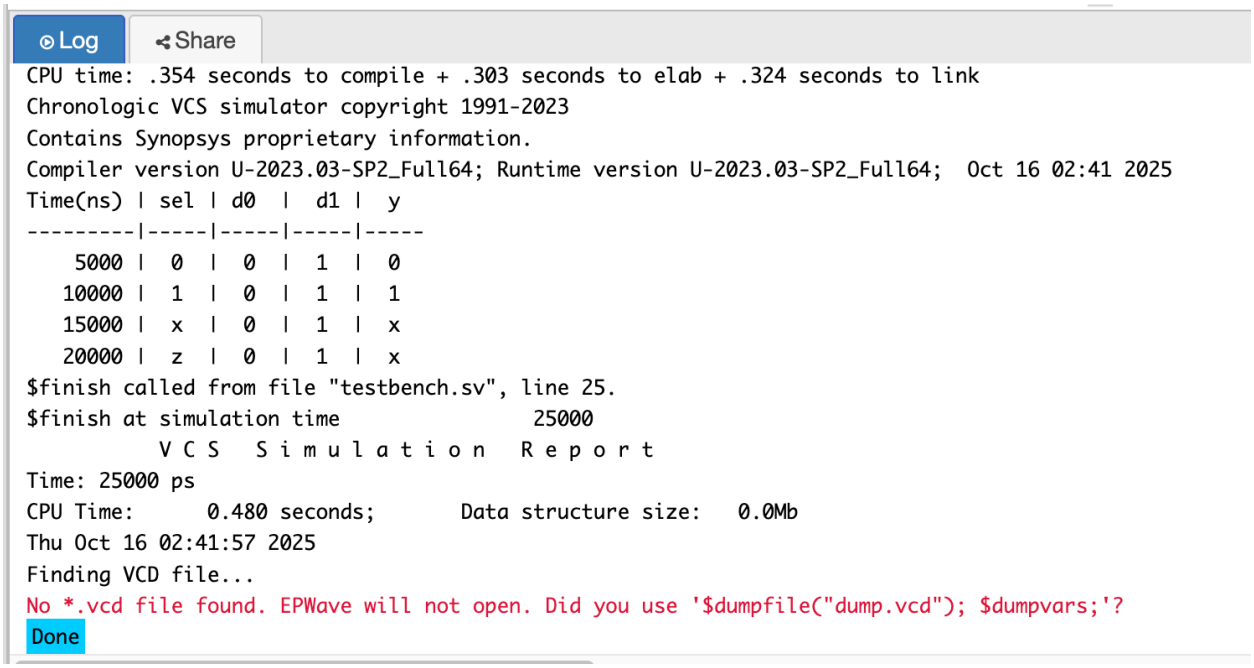


Q2: Answer

testbench	design	udp

`timescale 1ns/1ps
module testbench;
reg sel;
reg d0, d1;
wire y;
 	
mux2to1_top dut (
.sel(sel),
.d0(d0),
.d1(d1),
.y(y)
);
 	
initial begin
$display("Time(ns) | sel | d0 | d1 | y");
$display("---------|-----|-----|-----|-----");
 	
d0 = 0; d1 = 1;
 	
sel = 0; #5; $display("%8t | %b | %b | %b | %b", $time, sel, d0, d1, y);
sel = 1; #5; $display("%8t | %b | %b | %b | %b", $time, sel, d0, d1, y);
sel = 1'bx; #5; $display("%8t | %b | %b | %b | %b", $time, sel, d0, d1, y);
sel = 1'bz; #5; $display("%8t | %b | %b | %b | %b", $time, sel, d0, d1, y);
 	
#5 $finish;
end
endmodule
	`include "udp.sv"
	
	module mux2to1_top (
	input wire sel,
	input wire d0,
	input wire d1,
	output wire y
	);
	
	mux2to1_udp U1 (y, sel, d0, d1);
	endmodule
		primitive mux2to1_udp (out, sel, d0, d1);
		output out;
		input sel, d0, d1;
		
		table
		// sel d0 d1 : out
		0 0 ? : 0;
		0 1 ? : 1;
		1 ? 0 : 0;
		1 ? 1 : 1;
		x ? ? : x;
		z ? ? : x;
		endtable
		endprimitive

Output:



```
Log  Share
CPU time: .354 seconds to compile + .303 seconds to elab + .324 seconds to link
Chronologic VCS simulator copyright 1991-2023
Contains Synopsys proprietary information.
Compiler version U-2023.03-SP2_Full64; Runtime version U-2023.03-SP2_Full64;  Oct 16 02:41 2025
Time(ns) | sel | d0  |  d1 |  y
---------|-----|-----|-----|-----
    5000 |  0  |  0  |  1  |  0
   10000 |  1  |  0  |  1  |  1
   15000 |  x  |  0  |  1  |  x
   20000 |  z  |  0  |  1  |  x
$finish called from file "testbench.sv", line 25.
$finish at simulation time                25000
          V C S   S i m u l a t i o n   R e p o r t
Time: 25000 ps
CPU Time:      0.480 seconds;       Data structure size:   0.0Mb
Thu Oct 16 02:41:57 2025
Finding VCD file...
No *.vcd file found. EPWave will not open. Did you use '$dumpfile("dump.vcd"); $dumpvars;'?
Done
```

Q3: Answer

Longest data-path calculation (general case)

Path label	Data path (from input to out)	Gate delays along path	Total delay
P1 (longest)	A/B => OR => MUX1 => OR => MUX2	5 + 8 + 5 + 8	26
P2	C => NOT =>MUX1 => OR => MUX2	3 + 8 + 5 + 8	24
P3	C => NOT => MUX1 => NOT => MUX2	3 + 8 + 3 + 8	22
P4	(C direct into OR2 only) => OR => MUX2	5 + 8	13
P5	D => NOT => MUX2	3 + 8	11

The longest input => output delay = 26 time units (via P1).

If D is held constant

D fixed	Active longest data path	Gate delays	Longest delay
D = 0	A/B => OR => MUX1 => OR => MUX2	5 + 8 + 5 + 8	26
D = 1	D => NOT => MUX2 (A/B/C paths are masked at MUX2)	3 + 8	11

testbench

```
module tb_top_module;
   reg A, B, C, D;
   wire out;

   top_module uut (out, A, B, C, D);

   initial begin
      $dumpfile("mux_circuit.vcd");
      $dumpvars(0, tb_top_module);

      $display("Time\t| A | B | C | D |
OUT");
      $display("--------|---|---|---|---|----");
           $monitor("%0t\t\t| %b | %b | %b |
%b | %b", $time, A, B, C, D, out);

      A=0; B=0; C=0; D=0;
      #50;

      A=1; B=0; C=1; D=0;
      #50;

      A=0; B=1; C=1; D=1;
      #50;

      A=1; B=1; C=0; D=1;
      #50;

      A=0; B=0; C=1; D=0;
      #50;

      $finish;
   end
endmodule
```

design

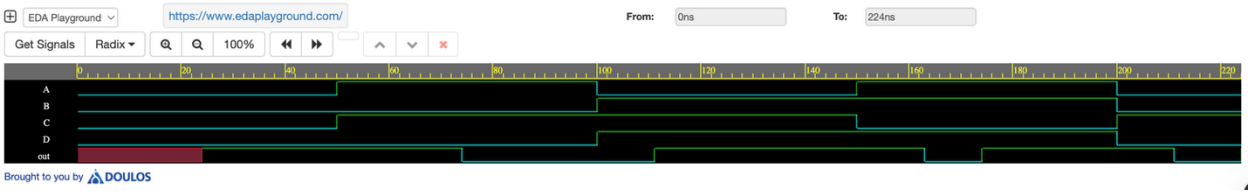
```
`include "udp.sv"
module top_module (output out, input
A, B, C, D);
   wire signal_AB, signal_notC,
signal_MUX1_out;
   wire signal_OR2, signal_NOT2,
signal_notD;

   or #5 or_gate1(signal_AB, A, B);
   not #3 not_gate1(signal_notC, C);
   mux2to1 #8
mux1(signal_MUX1_out, signal_AB,
signal_notC, D);
   or #5 or_gate2(signal_OR2,
signal_MUX1_out, C);
   not #3 not_gate2(signal_NOT2,
signal_MUX1_out);
   not #3 not_gate3(signal_notD, D);
   mux2to1 #8 mux2(out, signal_OR2,
signal_NOT2, signal_notD);

endmodule
```

udp

```
primitive mux2to1 (out,
in0, in1, sel);
   output out;
   input in0, in1, sel;

   table
   // in0  in1  sel : out
      0    0    ? : 0;
      1    1    ? : 1;
      0    ?    0 : 0;
      1    ?    0 : 1;
      ?    0    1 : 0;
      ?    1    1 : 1;
      x    x    0 : x;
      x    x    1 : x;
   endtable
endprimitive
```

Output:

```
Log  Share
CPU time: .373 seconds to compile + .337 seconds to elab + .378 seconds to link
Chronologic VCS simulator copyright 1991-2023
Contains Synopsys proprietary information.
Compiler version U-2023.03-SP2_Full64; Runtime version U-2023.03-SP2_Full64;  Oct 17 04:09 2025
Time    | A | B | C | D | OUT
--------|---|---|---|---|----
0       | 0 | 0 | 0 | 0 | x
24      | 0 | 0 | 0 | 0 | 1
50      | 1 | 0 | 1 | 0 | 1
74      | 1 | 0 | 1 | 0 | 0
100     | 0 | 1 | 1 | 1 | 0
111     | 0 | 1 | 1 | 1 | 1
150     | 1 | 1 | 0 | 1 | 1
163     | 1 | 1 | 0 | 1 | 0
174     | 1 | 1 | 0 | 1 | 1
200     | 0 | 0 | 1 | 0 | 1
211     | 0 | 0 | 1 | 0 | 0
224     | 0 | 0 | 1 | 0 | 1
$finish called from file "testbench.sv", line 30.
$finish at simulation time                  250
          V C S   S i m u l a t i o n   R e p o r t
Time: 250 ns
CPU Time:      0.330 seconds;       Data structure size:   0.0Mb
Fri Oct 17 04:09:47 2025
Finding VCD file...
./mux_circuit.vcd
[2025-10-17 08:09:47 UTC] Opening EPWave...
Done
```



Q4: Answer

```
`include "basicPrimitive.v"

primitive example(a0, a1, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r0, r1);
   output a0, a1, r0, r1;
   input  b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q;

   reg a0, a1, r0, r1;

   table
   // a0 a1  c  r0 r1  b  d  e  f  g  h  i  j  k  l  m  n  o  p  q
     0  0   1  0  1  0  1  0  1  0  1  0  1  0  1  0  1  0  1  0;
     1  0   1  1  1  0  1  0  1  0  1  0  1  0  1  0  1  0  1  ?;
     0  0   0  0  1  0  1  0  1  0  1  0  1  0  1  0  1  0  1  0;
     // more rows as needed
   endtable
endprimitive
```

Explanation of Fixes

Issue	Problem in Original	Fix Applied	Why
Include directive	Used #include	Changed to `include	Verilog uses backtick, not hash, for include directives.
Multi-bit outputs	Used [1:0] a, r inside a UDP	Split into single bit a0, a1, r0, r1	UDPs support scalar outputs only, not vectors.
In-out port	c declared as in-out	Changed to input c	UDPs cannot have in-out ports - only input and output.
Table format	Had two colons (:) per row	Rewritten with one colon separator between input and output fields (semicolon to end each row)	UDP syntax allows only one colon to separate the input pattern from the output.
Nested instantiation	Tried to instantiate another primitive (basicPrimitive u0) inside a UDP	Removed — not allowed inside UDPs	UDPs are atomic; cannot instantiate other modules or primitives.

Q5: Answer

```
`timescale 1ns/1ps
// UDP: Positive-Edge-Triggered D Flip-Flop with Asynchronous Reset
primitive DFF (q, d, clk, rst);
  output q;
  reg q;
  input d, clk, rst;

  table
  //  clk    d   rst :  q :  q+
      ?      ?    1  :  ? :  0;   // Async reset (active-high)
      (01)   1    0  :  ? :  1;   // Rising-edge clock, capture 1
      (01)   0    0  :  ? :  0;   // Rising-edge clock, capture 0
      (0?)   ?    0  :  ? :  -;   // Ignore falling edge
      (1?)   ?    0  :  ? :  -;   // Ignore glitches
      ?      ?    0  :  ? :  -;   // Hold otherwise
  endtable
endprimitive

// 3-Bit Ripple Counter: 3 DFF + 2 XOR + 1 NAND + 1 NOT

module rippleCounter(input clk, input rst, output [2:0] out);
  wire rst_bar;
  wire n1, n2, n3;

  not (rst_bar, rst);
  nand (n1, out[0], out[1]);
  xor  (n2, n1, ~out[2]);
  xor  (n3, out[0], out[1]);

  DFF dff0(out[0], n3, clk,   rst_bar);  // LSB toggles every clk
  DFF dff1(out[1], ~out[0], out[0], rst_bar); // middle bit
  DFF dff2(out[2], n2,    out[1], rst_bar);   // MSB
endmodule

// TESTBENCH

module tb_rippleCounter;
  reg clk, rst;
  wire [2:0] out;

  rippleCounter uut (.clk(clk), .rst(rst), .out(out));

  // Generate 10 ns clock
  initial begin
    clk = 0;
```

```
    forever #5 clk = ~clk;
  end

  initial begin
    $dumpfile("ripple_counter.vcd");
    $dumpvars(0, tb_rippleCounter);
    $display("Time(ns)\tclk\trst\tout[2:0]");
    $display("---------------------------------");
    $monitor("%0t\t%b\t%b\t%b", $time, clk, rst, out);

    rst = 1;   // assert reset
    #10 rst = 0; // de-assert reset
    #200 $finish;
  end
endmodule
```
