```
module mux2_1 (
   input wire sel,
   input wire in0,
   input wire in1,
   output wire out
);
   assign out = sel ? in1 : in0;
endmodule

// Testbench
module tb_mux2_1;
   reg sel, in0, in1;
   wire out;

   mux2_1 uut (.sel(sel), .in0(in0), .in1(in1), .out(out));

   initial begin
      $monitor("sel=%b in0=%b in1=%b => out=%b", sel, in0, in1, out);
      in0=0; in1=1;
      sel=0; #10;
      sel=1; #10;
      $finish;
   end
endmodule
```

```
module mux4_1 (
   input wire [1:0] sel,
   input wire in0,
   input wire in1,
   input wire in2,
   input wire in3,
   output wire out
);
   assign out = (sel == 2'b00) ? in0 :
                (sel == 2'b01) ? in1 :
                (sel == 2'b10) ? in2 : in3;
endmodule

// Testbench
module tb_mux4_1;
   reg [1:0] sel;
   reg in0, in1, in2, in3;
   wire out;
   mux4_1 uut (.sel(sel), .in0(in0), .in1(in1), .in2(in2), .in3(in3), .out(out));
   initial begin
      $monitor("sel=%b in0=%b in1=%b in2=%b in3=%b => out=%b", sel,in0,in1,in2,in3,out);
      in0=0; in1=1; in2=0; in3=1;
      sel=2'b00; #10;
      sel=2'b01; #10;
      sel=2'b10; #10;
      sel=2'b11; #10;
      $finish;
   end
endmodule
```

```
module mux4_1_from_mux2 (
    input wire [1:0] sel,
    input wire in0,
    input wire in1,
    input wire in2,
    input wire in3,
    output wire out
);
    wire mux_a_out, mux_b_out;
    mux2_1 mux_a (.sel(sel[0]), .in0(in0), .in1(in1), .out(mux_a_out));
    mux2_1 mux_b (.sel(sel[0]), .in0(in2), .in1(in3), .out(mux_b_out));
    mux2_1 mux_final (.sel(sel[1]), .in0(mux_a_out), .in1(mux_b_out), .out(out));
endmodule
// Testbench
module tb_mux4_1_from_mux2;
    reg [1:0] sel;
    reg in0, in1, in2, in3;
    wire out;
    mux4_1_from_mux2 uut (.sel(sel), .in0(in0), .in1(in1), .in2(in2), .in3(in3), .out(out));
    initial begin
        $monitor("sel=%b in0=%b in1=%b in2=%b in3=%b => out=%b", sel,in0,in1,in2,in3,out);
        in0=0; in1=1; in2=0; in3=1;
        sel=2'b00; #10;
        sel=2'b01; #10;
        sel=2'b10; #10;
        sel=2'b11; #10;
        $finish;
    end
endmodule
```

```
module shift_register_4bit (
    input wire clk,
    input wire rst,
    input wire [1:0] mode,
    input wire serial_in,
    input wire [3:0] parallel_in,
    output reg [3:0] Q
);
    always @(posedge clk or posedge rst) begin
        if (rst)
            Q <= 4'b0000;
        else begin
            case (mode)
                2'b00: Q <= Q; // giữ nguyên
                2'b01: Q <= {Q[2:0], serial_in}; // dịch trái
                2'b10: Q <= {serial_in, Q[3:1]}; // dịch phải
                2'b11: Q <= parallel_in; // load song song
            endcase
        end
    end
endmodule

// Testbench
module tb_shift_register_4bit;
    reg clk, rst;
    reg [1:0] mode;
    reg serial_in;
    reg [3:0] parallel_in;
    wire [3:0] Q;
```

```
   shift_register_4bit uut (.clk(clk), .rst(rst), .mode(mode), .serial_in(serial_in), .parallel_in(parallel_in), .Q(Q));


   initial begin
      clk = 0;
      forever #5 clk = ~clk; // clock 10ns
   end


   initial begin
      $monitor("time=%0t rst=%b mode=%b serial_in=%b parallel_in=%b => Q=%b", $time, rst, mode, serial_in, parallel_in, Q);
      rst=1; #10; rst=0;
      mode=2'b11; parallel_in=4'b1010; #10; // load song song
      mode=2'b01; serial_in=1; #20; // dịch trái
      mode=2'b10; serial_in=0; #20; // dịch phải
      mode=2'b00; #20; // giữ nguyên
      $finish;
   end
endmodule
```