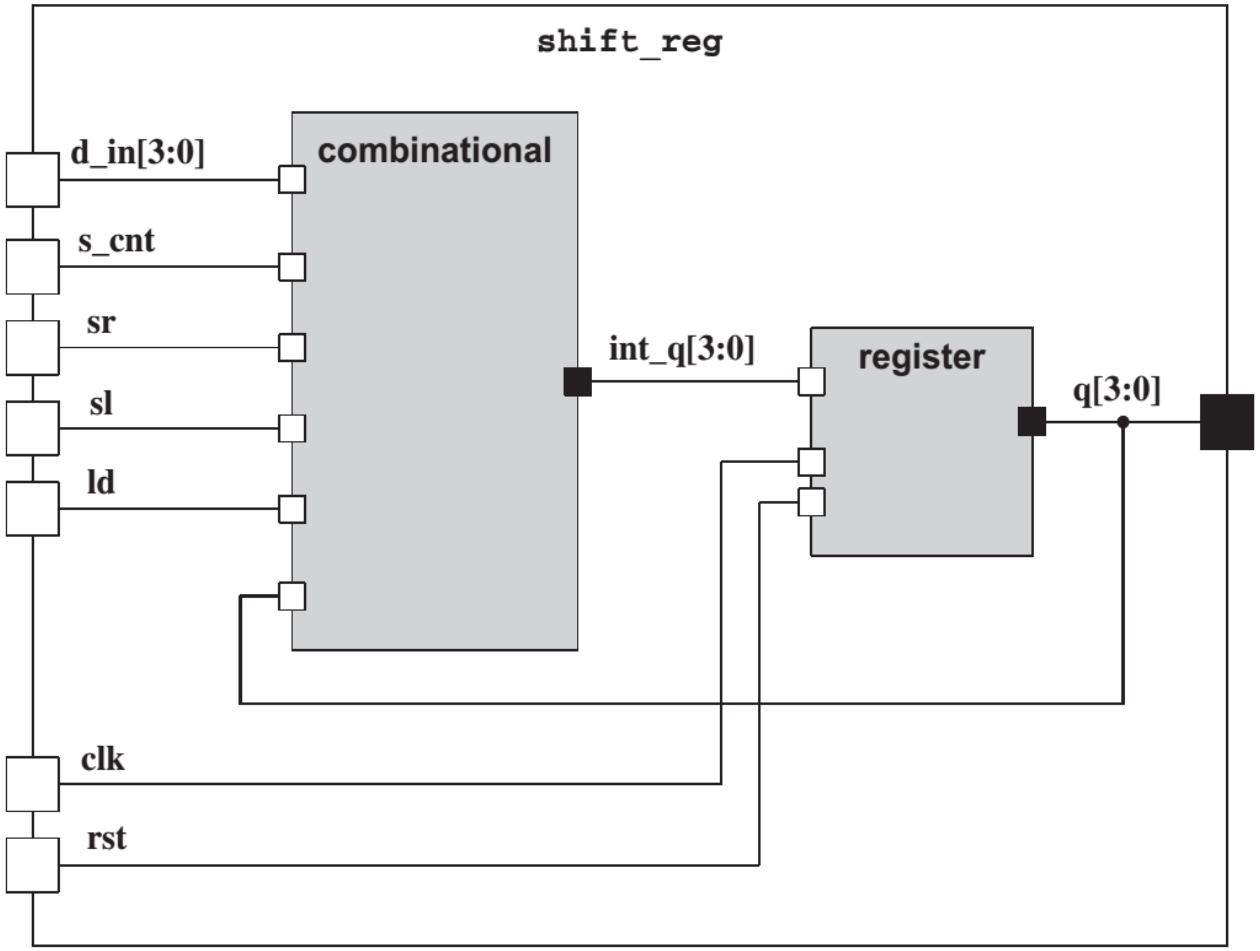
Sequential Circuit Description

Separate register&combinational blocks

Multi-bit Shifter with Separate Register Block



Sequential Circuit Description

Separate register&combinational blocks

Shifter Verilog Code

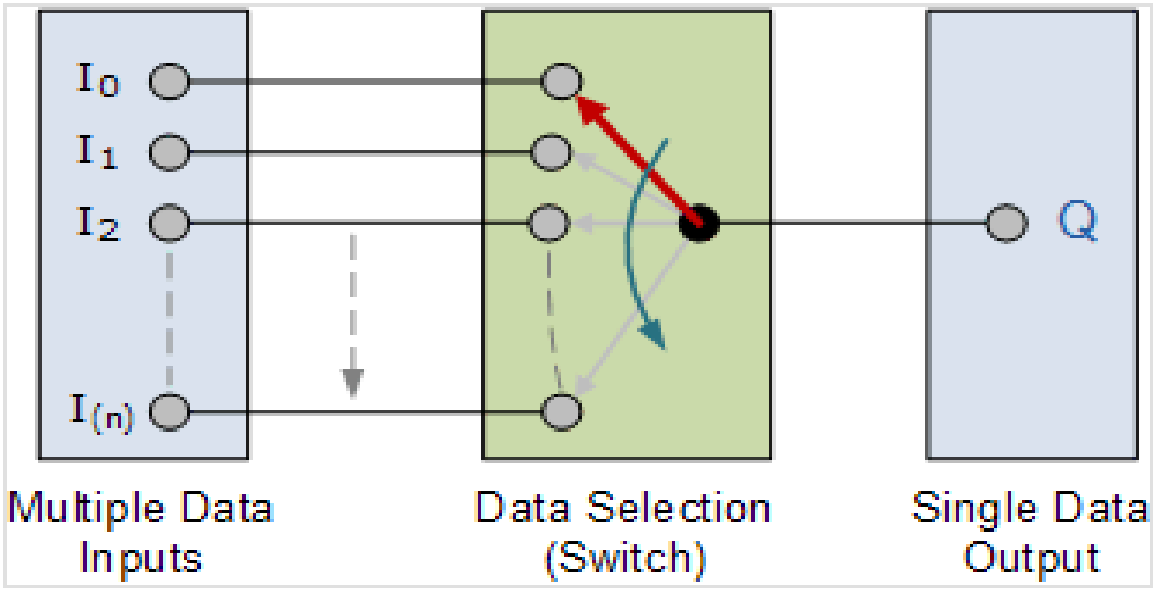
```
module shift_reg( input [3:0] d_in, input clk, sr, sl, ld, rst,
                  input [1:0] s_cnt, output reg [3:0] q);
   reg [3:0] int_q;
   always @( d_in, q, s_cnt, sr, sl, ld ) begin: combinational
      if( ld )
         int_q = d_in;
      else if( sr )
         int_q = q >> s_cnt;
      else if( sl )
         int_q = q << s_cnt;
      else int_q = q;
   end
   always @( posedge clk ) begin: register
      if (rst) q <= 0;
      else q <= int_q;
   end
endmodule
```

آزمایشگاه سیستم های دیجیتال2

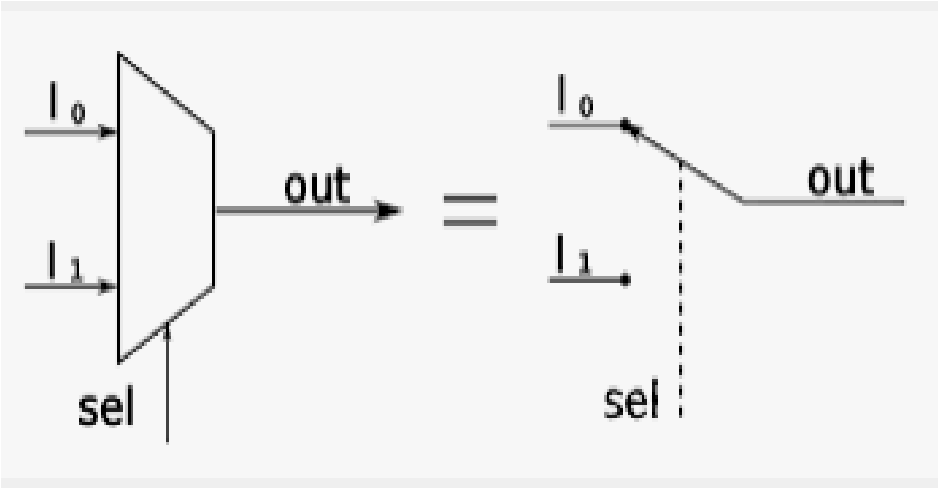
آزمایش 5

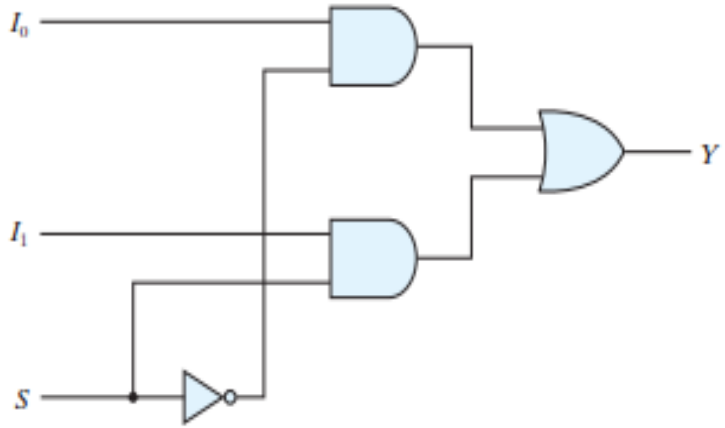
مالتی پلکسر - دی مالتی پلکسر

Multiplexer



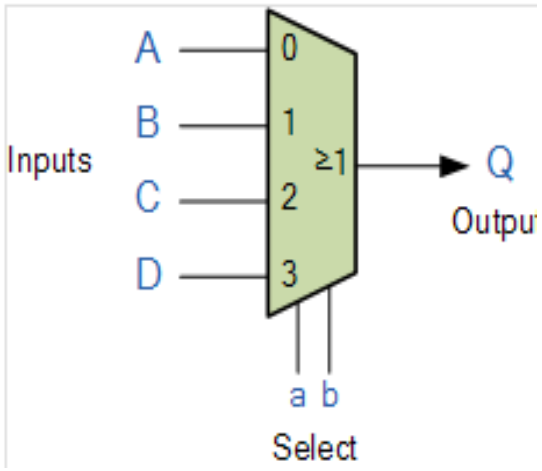
Continued

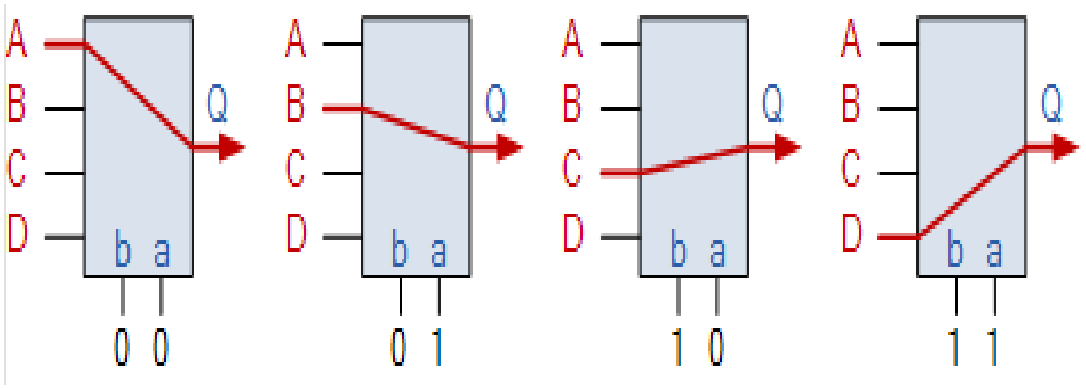


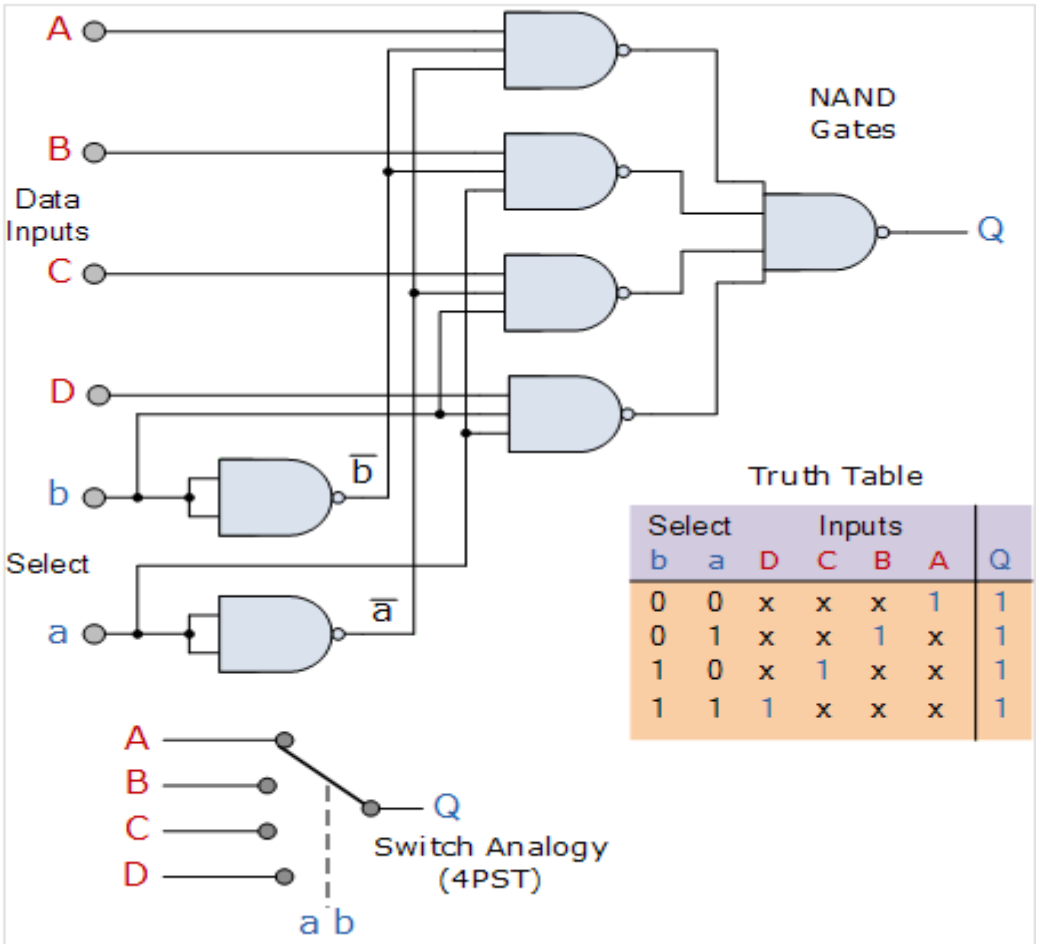


s	out
0	I0
1	I1

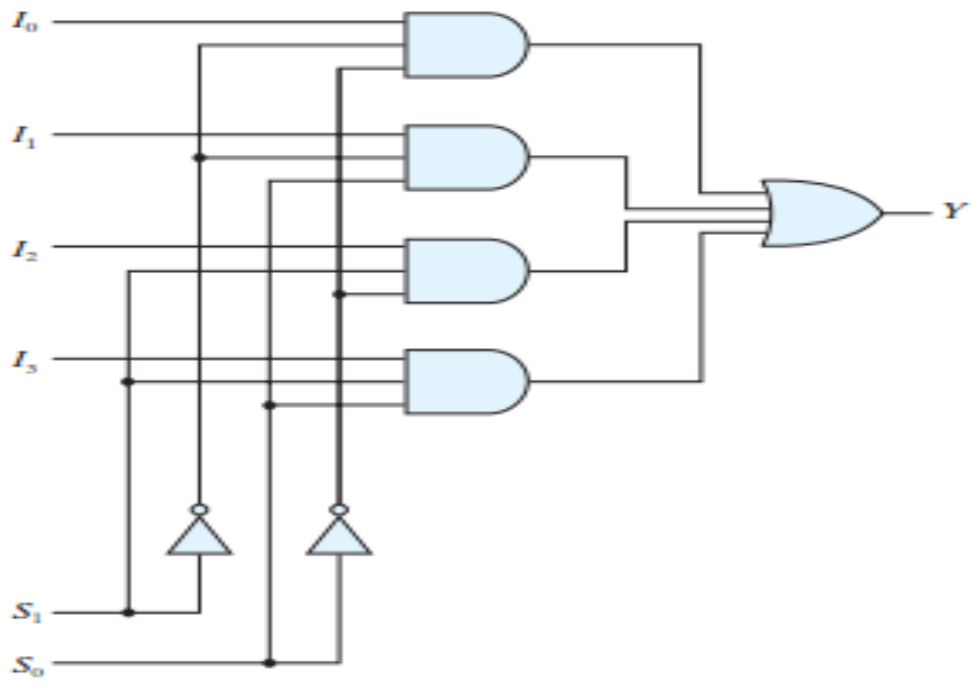
Continued


A
B
C
D
Inputs
0
1
2
3
≥1
Q
Outpu
a b
Select


A
B
C
D
Q
b a
0 0
0 1
1 0
1 1


A
B
Data
Inputs
C
D
b
Select
a
NAND
Gates
Q
b
a
Truth Table
Select
Inputs
b a D C B A Q
0 0 x x x 1 1
0 1 x x 1 x 1
1 0 x 1 x x 1
1 1 1 x x x 1
A
B
C
D
Q
Switch Analogy
(4PST)
a b

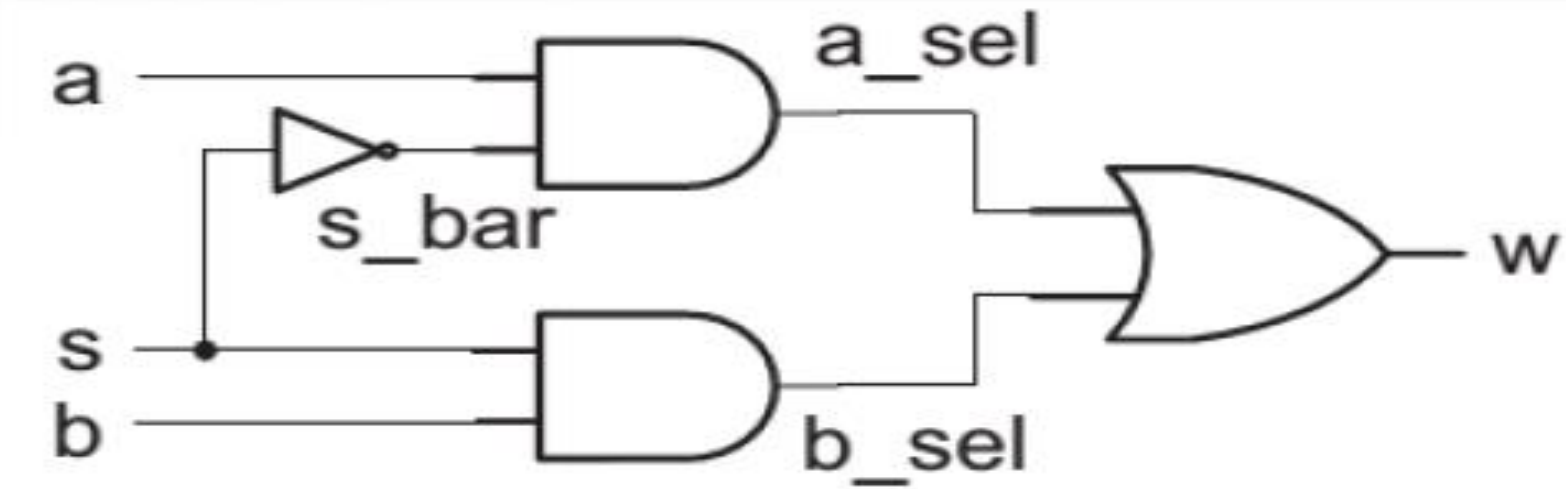
Continued



S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Register Transfer Level Design with Verilog

A Multiplexer Using Basic Gates



Register Transfer Level Design with Verilog

A Multiplexer Using Basic Gates

Primitive instantiations

```
module MultiplexerA (input a, b, s, output w);
    wire a_sel, b_sel, s_bar;
    not U1 (s_bar, s);
    and U2 (a_sel, a, s_bar);
    and U3 (b_sel, b, s);
    or  U4 (w, a_sel, b_sel);
endmodule
```

Register Transfer Level Design with Verilog

A Multiplexer

Assign Statement and Boolean

```
module MultiplexerB (input a, b, s, output w);
    assign w = (a & ~s) | (b & s);
endmodule
```

Register Transfer Level Design with Verilog

A Multiplexer

Assign Statement and Condition operator

```
module MultiplexerC (input a, b, s, output w);
    assign w = s ? b : a;
endmodule
```

Register Transfer Level Design with Verilog

A Multiplexer

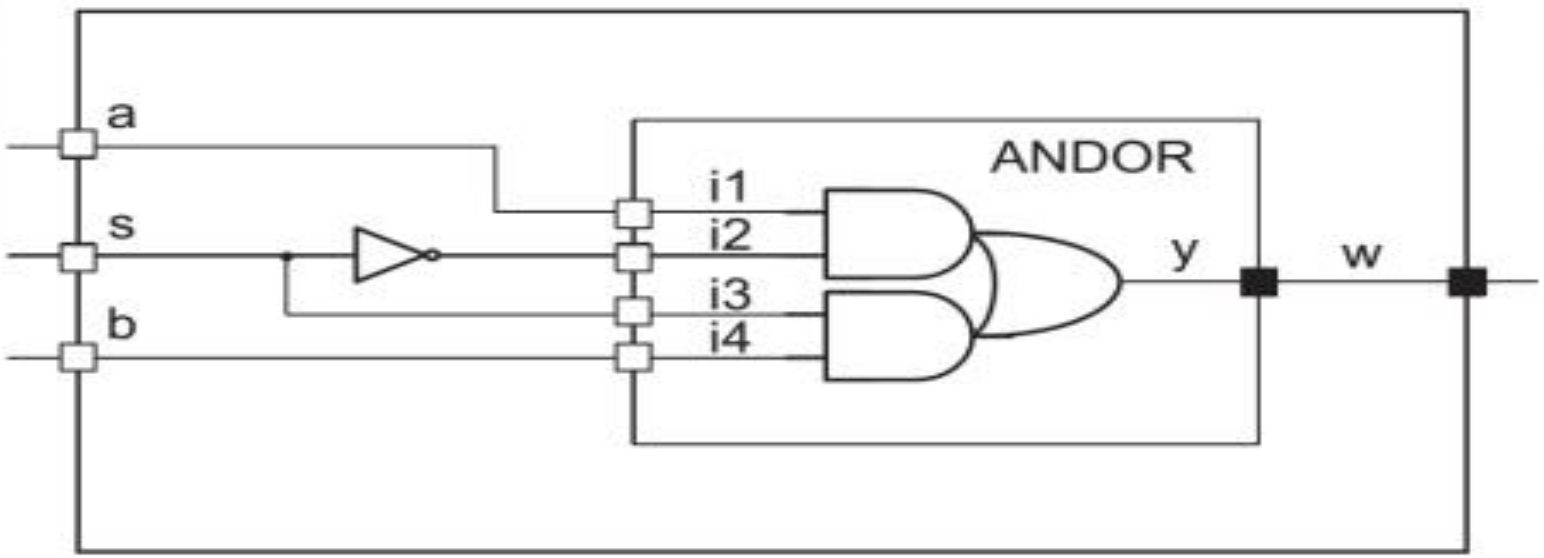
Procedural Statement

```
module MultiplexerD (input a, b, s, output w);
    reg w;
    always @ (a, b, s) begin
        if (s) w = b;
        else w = a;
    end
endmodule
```

Register Transfer Level Design with Verilog

A Multiplexer Using ANDOR

Module Instantiation



Register Transfer Level Design with Verilog

A Multiplexer Using ANDOR

Module Instantiation

```
module ANDOR (input i1, i2, i3, i4, output y);
    assign y = (i1 & i2) | (i3 & i4);
endmodule
//
module MultiplexerE (input a, b, s, output w);
    wire s_bar;
    not U1 (s_bar, s);
    ANDOR U2 (a, s_bar, s, b, w);
endmodule
```

Register Transfer Level Design with Verilog

A Multiplexer

Test bench for all Multiplexer

```
module test(output w);
      reg a,s=0,b;
      //wire w;
      MultiplexerA   ic1 (a,s,b,w);
      always
      begin
         #20; s = ~s;
      end

      initial
      begin
         #10; a = 1; b = 0;
         #50; a = 0; b = 1;
         #200; $stop;
         end

endmodule
```