```
::::::::::::::
adder4.v
::::::::::::::
module adder4(A, B, Ci, Co, S);
   input [3:0] A, B;
   input Ci;

   output Co;
   output [3:0] S;

   wire[3:1] C;

   fa  fa0 (A[0], B[0], Ci,   C[1], S[0]);
   fa  fa1 (A[1], B[1], C[1], C[2], S[1]);
   fa  fa2 (A[2], B[2], C[2], C[3], S[2]);
   fa  fa3 (A[3], B[3], C[3], Co,   S[3]);

endmodule


::::::::::::::
dff_en.v
::::::::::::::
module dff_en(clk, rst, d, en, q, qb);
  input d, en, clk, rst;
  output q, qb;

  assign d2 = (en) ? d : q;

  dff U1 (clk, rst, d2, q, qb);

  assign qb = ~q;
endmodule
::::::::::::::
dff_tb.v
::::::::::::::
`timescale 1ns/100ps

module dff_tb;

reg d, clk, rst;

dff U1 (d, clk, rst, q, qb);

always #10 clk = ~clk;


reg a, b;
initial
begin
        clk =0;
        d   =0;

        rst =1;
        rst =0;
   #20  rst = 1;

        a = 0;
        b = 1;

        #10
        a = b; // a=1
        b <= a; // b=1
```

```
   #10  d <=  1;
   #10  d <=  0;
   #10  d <=  1;
   #10  d <=  0;
   #10  d <=  1;
   #10  d <=  1;
  $finish;

end

initial
begin
    $dumpfile("test.vcd");
    $dumpvars(0, dff_tb);
end

endmodule
::::::::::::::
dff.v
::::::::::::::
module dff(clk, rst, d, q, qb);
  input d, clk, rst;
  output q, qb;
  reg    q;

  always @(posedge clk)
  begin
    if (~rst) q = 0;
    else      q = d;
  end

  assign qb = ~q;
endmodule
::::::::::::::
d_latch_tb.v
::::::::::::::
`timescale 1ns/100ps
`define D 10
`define GD 0
`define PD 10

module d_latch_tb;

reg d, c;

d_latch U1 (d, c, q, qb);



initial
begin
    $dumpfile("test.vcd");
    $dumpvars(0, d_latch_tb);
end

always #`PD c = ~c;

reg x, y;

initial
begin
      c=0;
      d=0;

   #`D d=1;
   #`D d=0;
   #`D d=1;
```

```
   #`D d=0;
   #`D d=0;
   #`D d=1;
   #`D d=1;

  $finish;
end // initial begin

endmodule
::::::::::::::
d_latch.v
::::::::::::::
module d_latch(d, c, q, qb);
  input d, c;
  output q, qb;

  nor #`GD U1 (q, r, qb);
  nor #`GD U2 (qb, s, q);

  not #`GD U3 (db, d);
  and #`GD U4 (r, c, db);
  and #`GD U5 (s, c, d);

endmodule
::::::::::::::
dpath.v
::::::::::::::
`timescale 1ns/100ps

module dpath;

reg clk, rst;
reg [3:0] Ain, Bin;
reg       en;
wire [3:0] A, B, S, C;

dreg_en UA (clk, rst, Ain, en, A);
dreg_en UB (clk, rst, Bin, en, B);

adder4  UU (A, B, 0, Co, S);

dreg_en UC (clk, rst, S, en, C);


always #10 clk = ~clk;

initial
begin
        clk =0;
        Ain =4'd0;
        Bin =4'd0;
        en = 0;

        rst =1;
        rst =0;
   #20  rst = 1;



   #20  Ain = 4'd1;  Bin = 4'd3;  en=1;
   #20  Ain = 4'd2;  Bin = 4'd3;  en=0;
   #20  Ain = 4'd3;  Bin = 4'd2;  en=1;
   #20  Ain = 4'd4;  Bin = 4'd2;  en=0;
   #20  Ain = 4'd5;  Bin = 4'd1;  en=1;
   #20  Ain = 4'd6;  Bin = 4'd1;  en=0;

  $finish;
```

```
end

initial
begin
    $dumpfile("test.vcd");
    $dumpvars(0, dpath);
end

endmodule
::::::::::::::
dreg_en_tb.v
::::::::::::::
`timescale 1ns/100ps

module dreg_en_tb;

reg clk, rst;
reg [3:0] din;
reg       en;
wire [3:0] dout;

dreg_en U1 (clk, rst, din, en, dout);


always #10 clk = ~clk;



initial
begin


        rst =0;
   #20  rst = 1;

    #35 rst = 0;
    #30 rst = 1;

end

initial
begin
        clk =0;
        din =4'd0;
        en = 0;



   #20  din =  4'd1;   en=1;
   #20  din =  4'd2;   en=0;
   #20  din =  4'd3;   en=1;
   #20  din =  4'd4;   en=1;
   #20  din =  4'd5;   en=0;
   #20  din =  4'd6;   en=1;
   #20  din =  4'd7;   en=0;
   #20  din =  4'd8;   en=1;
   #20  din =  4'd9;   en=0;
   #20  din =  4'd0;   en=1;

  $finish;

end

initial
begin
    $dumpfile("test.vcd");
```

```
    $dumpvars(0, dreg_en_tb);
end

endmodule
::::::::::::::
dreg_en.v
::::::::::::::
module dreg_en(clk, rst, din, en, dout);
input clk, rst;
input [3:0] din;
input       en;
output [3:0] dout;

  dff_en U3 (clk, rst, din[3], en, dout[3], qb3);
  dff_en U2 (clk, rst, din[2], en, dout[2], qb2);
  dff_en U1 (clk, rst, din[1], en, dout[1], qb1);
  dff_en U0 (clk, rst, din[0], en, dout[0], qb0);

endmodule
::::::::::::::
dreg_tb.v
::::::::::::::
`timescale 1ns/100ps

module dreg_tb;

reg clk, rst;
reg [3:0] din;
wire [3:0] dout, dout2;

dreg UDREG1 (clk, rst, din, dout);
dreg UDREG2 (clk, rst, dout, dout2);

always #10 clk = ~clk;

initial
begin
        clk =0;
        din =4'd0;

        rst =1;
        rst =0;
   #20  rst = 1;



   #20  din =  4'd1;
   #20  din =  4'd2;
   #20  din =  4'd3;
   #20  din =  4'd4;
   #20  din =  4'd5;
   #20  din =  4'd6;
   #20  din =  4'd7;
   #20  din =  4'd8;
   #20  din =  4'd9;
   #20  din =  4'd0;

  $finish;

end

initial
begin
    $dumpfile("test.vcd");
    $dumpvars(0, dreg_tb);
end
```

```
endmodule
::::::::::::::
dreg.v
::::::::::::::
module dreg(clk, rst, din, dout);
input clk, rst;
input [3:0] din;
output [3:0] dout;

  dff UDFF3 (clk, rst, din[3], dout[3], qb3);
  dff UDFF2 (clk, rst, din[2], dout[2], qb2);
  dff UDFF1 (clk, rst, din[1], dout[1], qb1);
  dff UDFF0 (clk, rst, din[0], dout[0], qb0);

endmodule
::::::::::::::
fa_gate.v
::::::::::::::
`timescale 1ns/100ps

module fa(a, b, c, Co, S);
  input a, b, c;
  output Co, S;

  wire P, S, G, PC;

  xor #1 U1 (P, a, b);
  xor #1 U2 (S, P, c);

  and #1 U3 (G, a, b);
  and #1 U4 (PC, P, c);

  or  #1 U5 (Co, G, PC);


endmodule
::::::::::::::
sr_nand_tb.v
::::::::::::::
`timescale 1ns/100ps
`define D 5
`define GD 1

module sr_nand_tb;

reg s, r;

sr_nand U1 (sb, rb, q, qb);

initial
begin
    $dumpfile("test.vcd");
    $dumpvars(0, sr_nand_tb);
end

assign sb = s;
assign rb = r;


initial
begin
       s<=0; r<=1;   // RESET
   #`D s<=0; r<=0;   // HOLD
   #`D s<=0; r<=1;   // RESET
   #`D s<=1; r<=0;   // SET
   #`D s<=0; r<=1;   // RESET
   #`D s<=0; r<=1;   // RESET
```

```
   #`D s<=1; r<=0;   // SET
   #`D s<=0; r<=0;   // HOLD
   #`D s<=1; r<=0;   // SET
   #`D s<=1; r<=0;   // SET
   #`D s<=1; r<=0;   // SET
   #`D s<=0; r<=1;   // RESET
  $finish;
end // initial begin

endmodule
::::::::::::::
sr_nand.v
::::::::::::::
module sr_nand(sb, rb, q, qb);
  input sb, rb;
  output q, qb;

  // nand #1 U1 (q, sb, qb);
  // nand #2 U2 (qb, rb, q);

  // assign #1 q = ~(sb & qb);
  // assign #2 qb = ~(rb & q);

  reg q, qb;
  always @(sb or qb)
    #1 q = ~(sb & qb);

  always @(rb or q)
    #1 qb = ~(rb & q);

endmodule
::::::::::::::
sr_nor_tb.v
::::::::::::::
`timescale 1ns/100ps
`define D 10
`define GD 1

module sr_nor_tb;

reg s, r;

sr_nor U1 (s, r, q, qb);

initial
begin
    $dumpfile("test.vcd");
    $dumpvars(0, sr_nor_tb);
end


initial
begin
       s=0; r=1;   // RESET
   #`D s=0; r=0;   // HOLD
   #`D s=0; r=1;   // RESET
   #`D s=1; r=0;   // SET
   #`D s=0; r=1;   // RESET
   #`D s=0; r=1;   // RESET
   #`D s=1; r=0;   // SET
   #`D s=0; r=0;   // HOLD
   #`D s=1; r=0;   // SET
   #`D s=1; r=0;   // SET
   #`D s=1; r=0;   // SET
   #`D s=0; r=1;   // RESET
  $finish;
end // initial begin
```

```
endmodule
::::::::::::::
sr_nor.v
::::::::::::::
module sr_nor(s, r, q, qb);
  input s, r;
  output q, qb;

  wire P, S, G, PC;

  nor #`GD U1 (q, r, qb);
  nor #`GD U2 (qb, s, q);


endmodule
```