ARCHITETTURA DEI CALCOLATORI

[Esempi in Verilog]

A CURA DI ALESSANDRO PAGHI

PROFESSORE: Roberto Giorgi (http://www3.diism.unisi.it/people/person.php?id=73&aa=2015)

LINK AL CORSO ANNO 2015/2016:
http://www3.diism.unisi.it/FAC/index.php?bodyinc=didattica/inc.insegnamento.php&id=55204&aa=2015

FREQUENTAZIONE: Sconsigliata.

```
`timescale 1ns / 1ps

module nand_component (c,a,b);

    initial begin

        $display("NAND Usage");

    end

    input a,b;                                          //a e b sono gli input del
    modulo
    output c;                                           //c è l'output del modulo

    wire d;                                             //d serve per trasferire il
    risultato da and a not.

    and a1(d,a,b);
    not n1(c,d);

endmodule

module test_bench();

    reg A,B;
    wire C;

    nand_component nand1(C,A,B);

    initial begin

        A=1'b0 ; B=1'b0 ;
        #50 $display("A=%b , B=%b, NAND=%b\n", A,B,C);      //#50 significa che la
        simulazione con quei valori procede per 50ns
        A=1'b0 ; B=1'b1 ;
        #50 $display("A=%b , B=%b, NAND=%b\n", A,B,C);
        A=1'b1 ; B=1'b0 ;
        #50 $display("A=%b , B=%b, NAND=%b\n", A,B,C);
        A=1'b1 ; B=1'b1 ;
        #50 $display("A=%b , B=%b, NAND=%b\n", A,B,C);

    end

endmodule
```

```
`timescale 1ns / 1ps

module nand_component (c,a,b);

    initial begin

        $display("NAND Usage");

    end

    input a,b;                                          //a e b sono gli input del
    modulo
    output c;                                           //c è l'output del modulo

    assign c = !(a && b);

endmodule

module test_bench();

    reg A,B;
    wire C;

    nand_component nand1(C,A,B);

    initial begin

        A=1'b0 ; B=1'b0 ;
        #50 $display("A=%b , B=%b, NAND=%b\n", A,B,C);      //#50 significa che la
        simulazione con quei valori procede per 50ns
        A=1'b0 ; B=1'b1 ;
        #50 $display("A=%b , B=%b, NAND=%b\n", A,B,C);
        A=1'b1 ; B=1'b0 ;
        #50 $display("A=%b , B=%b, NAND=%b\n", A,B,C);
        A=1'b1 ; B=1'b1 ;
        #50 $display("A=%b , B=%b, NAND=%b\n", A,B,C);

    end

endmodule
```

```
`timescale 1ns / 1ps

module nor_component (c,a,b);

    initial begin

        $display("NOR Usage");

    end

    input a,b;                                          //a e b sono gli input del
    modulo
    output c;                                           //c è l'output del modulo

    wire d;                                             //d serve per trasferire il
    risultato da and a not.

    or o1(d,a,b);
    not n1(c,d);

endmodule

module test_bench();

    reg A,B;
    wire C;

    nor_component nor1(C,A,B);

    initial begin

        A=1'b0 ; B=1'b0 ;
        #50 $display("A=%b , B=%b, NAND=%b\n", A,B,C);      //#50 significa che la
        simulazione con quei valori procede per 50ns
        A=1'b0 ; B=1'b1 ;
        #50 $display("A=%b , B=%b, NAND=%b\n", A,B,C);
        A=1'b1 ; B=1'b0 ;
        #50 $display("A=%b , B=%b, NAND=%b\n", A,B,C);
        A=1'b1 ; B=1'b1 ;
        #50 $display("A=%b , B=%b, NAND=%b\n", A,B,C);

    end

endmodule
```

```
module test_bench();

    reg A,B;
    wire C;

    nor #(10,15) nor1(C,A,B);                          //rise=10, fall=11
                                                       //la transizione della nor
                                                       comincia con un ritardo di 10
                                                       ns e termina con un ritardo
                                                       di altri 5ns

    initial begin

        A=1'b0 ; B=1'b0 ;
        #50 $display("A=%b , B=%b, NAND=%b\n", A,B,C);     //#50 significa che la
        simulazione con quei valori procede per 50ns
        A=1'b0 ; B=1'b1 ;
        #50 $display("A=%b , B=%b, NAND=%b\n", A,B,C);
        A=1'b1 ; B=1'b0 ;
        #50 $display("A=%b , B=%b, NAND=%b\n", A,B,C);
        A=1'b1 ; B=1'b1 ;
        #50 $display("A=%b , B=%b, NAND=%b\n", A,B,C);

    end
endmodule
```

```
module latch_SR(Q,QN,S,R);

    input S,R;
    output Q,QN;

    assign Q=!(R|QN);
    assign QN=!(S|Q);

endmodule

module test_bench();

    reg s,r;
    wire q,qn;

    latch_SR l1(q,qn,s,r);

    initial begin

        s=1'b0 ; r=1'b1 ;
        #50 $display("S=%b , R=%b, Q=%b, Qn=%b\n", s,r,q,qn);
        s=1'b1 ; r=1'b0 ;
        #50 $display("S=%b , R=%b, Q=%b, Qn=%b\n", s,r,q,qn);
        s=1'b0 ; r=1'b0 ;
        #100 $display("S=%b , R=%b, Q=%b, Qn=%b\n", s,r,q,qn);
        s=1'b0 ; r=1'b1 ;
        #50 $display("S=%b , R=%b, Q=%b, Qn=%b\n", s,r,q,qn);

    end

endmodule
```

```
module latch_SR_CLK(Q,QN,S,R,CLK);

    input S,R,CLK;
    output Q,QN;

    assign Q=~((R&CLK)|QN);
    assign QN=~((S&CLK)|Q);

endmodule

module test_bench();

    reg s,r,clk;
    wire q,qn;

    latch_SR_CLK l1(q,qn,s,r,clk);

    initial begin

        s=1'b0 ; r=1'b1 ; clk=1'b1 ;
        #50 $display("S=%b , R=%b, CLK=%b, Q=%b, Qn=%b\n", s,r,clk,q,qn);
        s=1'b1 ; r=1'b0 ; clk=1'b1 ;
        #50 $display("S=%b , R=%b, CLK=%b, Q=%b, Qn=%b\n", s,r,clk,q,qn);
        s=1'b0 ; r=1'b0 ; clk=1'b1 ;
        #100 $display("S=%b , R=%b, CLK=%b, Q=%b, Qn=%b\n", s,r,clk,q,qn);
        s=1'b0 ; r=1'b1 ; clk=1'b1 ;
        #50 $display("S=%b , R=%b, CLK=%b, Q=%b, Qn=%b\n", s,r,clk,q,qn);
        s=1'b1 ; r=1'b0 ; clk=1'b0 ;
        #50 $display("S=%b , R=%b, CLK=%b, Q=%b, Qn=%b\n", s,r,clk,q,qn);

    end

endmodule
```

```
module latch_SR_EDGET(Q,QN,S,R,CLK);

    input S,R,CLK;
    output Q,QN;

    wire Q1,Q1N;

    assign Q1=~((R&CLK)|Q1N);
    assign Q1N=~((S&CLK)|Q1);

    assign Q=~((Q1N&(~CLK))|QN);
    assign QN=~((Q1&(~CLK))|Q);

endmodule

module test_bench();

    reg s,r,clk;
    wire q,qn;

    latch_SR_EDGET l1(q,qn,s,r,clk);

    initial begin

        clk=1'b1;
        s=1'b0 ;
        r=1'b1 ;
        #10 clk=1'b0;
        #50 $display("S=%b , R=%b, CLK=%b, Q=%b, Qn=%b\n", s,r,clk,q,qn);

        clk=1'b1;
        s=1'b1 ;
        r=1'b0 ;
        #10 clk=1'b0;
        #50 $display("S=%b , R=%b, CLK=%b, Q=%b, Qn=%b\n", s,r,clk,q,qn);

        clk=1'b1;
        s=1'b0 ;
        r=1'b0 ;
        #10 clk=1'b0;
        #50 $display("S=%b , R=%b, CLK=%b, Q=%b, Qn=%b\n", s,r,clk,q,qn);

        clk=1'b1;
        s=1'b1 ;
        r=1'b0 ;
        #50 $display("S=%b , R=%b, CLK=%b, Q=%b, Qn=%b\n", s,r,clk,q,qn);

    end

endmodule
```

```
module latch_D(Q,QN,D,CLK);

    input D,CLK;
    output Q,QN;

    assign Q=~(((~D)&CLK)|QN);
    assign QN=~((D&CLK)|Q);

endmodule

module test_bench();

    reg d,clk;
    wire q,qn;

    latch_D l1(q,qn,d,clk);

    initial begin

        d=1'b0 ; clk=1'b1 ;
        #50 $display("D=%b, CLK=%b, Q=%b, Qn=%b\n", d,clk,q,qn);
        d=1'b1 ; clk=1'b1 ;
        #50 $display("D=%b, CLK=%b, Q=%b, Qn=%b\n", d,clk,q,qn);
        d=1'b0 ; clk=1'b0 ;
        #100 $display("D=%b, CLK=%b, Q=%b, Qn=%b\n", d,clk,q,qn);

    end

endmodule
```

```
module latch_D_EDGET(Q,QN,D,CLK);

    input D,CLK;
    output Q,QN;

    wire Q1,Q1N;

    assign Q1=~(((~D)&CLK)|Q1N);
    assign Q1N=~((D&CLK)|Q1);

    assign Q=~((Q1N&(~CLK))|QN);
    assign QN=~((Q1&(~CLK))|Q);

endmodule

module test_bench();

    reg d,clk;
    wire q,qn;

    latch_D_EDGET l1(q,qn,d,clk);

    initial begin

        clk=1'b1;
        d=1'b0 ;
        #10 clk=1'b0;
        #50 $display("D=%b, CLK=%b, Q=%b, Qn=%b\n", d,clk,q,qn);

        clk=1'b1;
        d=1'b1 ;
        #10 clk=1'b0;
        #50 $display("D=%b, CLK=%b, Q=%b, Qn=%b\n", d,clk,q,qn);

        clk=1'b1;
        d=1'b1 ;
        #50 $display("D=%b, CLK=%b, Q=%b, Qn=%b\n", d,clk,q,qn);

    end

endmodule
```

```
module FlipFlop_D_Rise(Q,QN,D,CLK);

    input CLK, D;
    output Q,QN;
    reg Q;
    assign QN=~Q;
    always @(posedge CLK) begin
        Q <= D;
    end

endmodule

module FlipFlop_D_Fall(Q,QN,D,CLK);

    input CLK, D;
    output Q,QN;
    reg Q;
    assign QN=~Q;
    always @(negedge CLK)
        Q <= D;

endmodule

module test_bench();

    reg d,clk;
    wire q,qn;

    FlipFlop_D_Fall ff1(q,qn,d,clk);

    initial begin

        clk=1'b1;
        d=1'b0 ;
        #10 clk=1'b0;
        #50 $display("D=%b, CLK=%b, Q=%b, Qn=%b\n", d,clk,q,qn);

        clk=1'b1;
        d=1'b1 ;
        #10 clk=1'b0;
        #50 $display("D=%b, CLK=%b, Q=%b, Qn=%b\n", d,clk,q,qn);

    end

endmodule
```

```
module MUX21(IN0,IN1,SEL,OUT);

    input IN0,IN1,SEL;
    output OUT;
    wire OUT;

    assign OUT = (SEL) ? IN1 : IN0;

endmodule

module test_bench();

    reg in0,in1,sel;
    wire out;

    MUX21 m1(in0,in1,sel,out);

    initial begin

        in0=1'b1 ; in1=1'b0 ; sel=1'b0;
        #50;

        in0=1'b1 ; in1=1'b0 ; sel=1'b1;
        #50;

    end

endmodule
```

```
module MUX41(IN0,IN1,IN2,IN3,SEL0,SEL1,OUT);

    input IN0,IN1,IN2,IN3,SEL0,SEL1;
    output OUT;

    assign OUT = (IN0&(~SEL1)&(~SEL0))|(IN1&(~SEL1)&(SEL0))|(IN2&(SEL1)&(~SEL0))|(IN3&(
    SEL1)&(SEL0));

endmodule

module test_bench();

    reg in0,in1,in2,in3,sel0,sel1;
    wire out;

    MUX41 m1(in0,in1,in2,in3,sel0,sel1,out);

    initial begin

        in0=1'b1 ; in1=1'b0 ; in2=1'b1 ; in3=1'b0 ;
        sel0=1'b0 ; sel1=1'b0 ;
        #50;
        sel0=1'b1 ; sel1=1'b0 ;
        #50;
        sel0=1'b0 ; sel1=1'b1 ;
        #50;
        sel0=1'b1 ; sel1=1'b1 ;
        #50;

    end

endmodule
```

```
module DEMUX14(IN,SEL0,SEL1,OUT0,OUT1,OUT2,OUT3);

    input IN,SEL0,SEL1;
    output OUT0,OUT1,OUT2,OUT3;

    assign OUT0 = IN&(~SEL1)&(~SEL0);
    assign OUT1 = IN&(~SEL1)&(SEL0);
    assign OUT2 = IN&(SEL1)&(~SEL0);
    assign OUT3 = IN&(SEL1)&(SEL0);

endmodule

module test_bench();

    reg in,sel0,sel1;
    wire out0,out1,out2,out3;

    DEMUX14 d1(in,sel0,sel1,out0,out1,out2,out3);

    initial begin

        in=1'b1 ;
        sel0=1'b0 ; sel1=1'b0 ;
        #50;
        sel0=1'b1 ; sel1=1'b0 ;
        #50;
        sel0=1'b0 ; sel1=1'b1 ;
        #50;
        sel0=1'b1 ; sel1=1'b1 ;
        #50;

    end

endmodule
```

```
module DEC38(IN,OUT);

    input [2:0] IN;
    output [7:0] OUT;

    wire [7:0] OUT ;

    assign OUT = 8'b00000001 << IN;

endmodule

module test_bench();

    reg [2:0] in;
    wire [7:0] out;

    DEC38 d1(in,out);

    initial begin

        in=3'b000 ;
        #50;
        in=3'b001 ;
        #50;
        in=3'b010 ;
        #50;
        in=3'b011 ;
        #50;
        in=3'b100 ;
        #50;
        in=3'b101 ;
        #50;
        in=3'b110 ;
        #50;
        in=3'b111 ;
        #50;

    end

endmodule
```

```
module ENC83(IN,OUT);

    input [7:0] IN;
    output [2:0] OUT;

    reg [2:0] OUT;

    always @(IN) begin
        case (IN)
            8'b00000001 : OUT = 3'b000;
            8'b00000010 : OUT = 3'b001;
            8'b00000100 : OUT = 3'b010;
            8'b00001000 : OUT = 3'b011;
            8'b00010000 : OUT = 3'b100;
            8'b00100000 : OUT = 3'b101;
            8'b01000000 : OUT = 3'b110;
            8'b10000000 : OUT = 3'b111;
    endcase
    end

endmodule

module test_bench();

    reg [7:0] in;
    wire [2:0] out;

    ENC83 e1(in,out);

    initial begin

        in=8'b00000001 ;
        #50;
        in=8'b00000010 ;
        #50;
        in=8'b00000100 ;
        #50;
        in=8'b00001000 ;
        #50;
        in=8'b00010000 ;
        #50;
        in=8'b00100000 ;
        #50;
        in=8'b01000000 ;
        #50;
        in=8'b10000000 ;
        #50;

    end

endmodule
```

PRIORITY ENCODER 4:2

INPUTS				OUTPUTS		
D_3	D_2	D_1	D_0	A_1	A_0	V
0	0	0	0	X	X	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

A_0

D_2D_3 \ D_0D_1	00	01	11	10
00	X	1	1	0
01	1	1	1	1
11	1	1	1	1
10	0	0	0	0

$$A_0 = D_3 + D_1\overline{D_2}$$

A_1

D_2D_3 \ D_0D_1	00	01	11	10
00	X	0	0	0
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

$$A_1 = D_2 + D_3$$

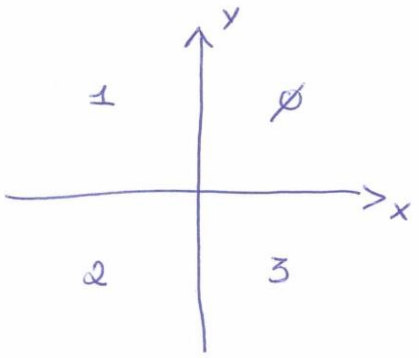
V

D_2D_3 \ D_0D_1	00	01	11	10
00	0	1	1	1
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

$$V = D_0 + D_1 + D_2 + D_3$$

```
module PENC42(IN,OUT,V);

    input [3:0] IN;
    output [1:0] OUT;
    output V;

    reg [1:0] OUT;
    reg V;

    always @(IN) begin
        OUT[0] = IN[3]|(IN[1]&(!IN[2]));
        OUT[1] = IN[2]|IN[3];
        V = IN[0]|IN[1]|IN[2]|IN[3];
    end

endmodule

module test_bench();

    reg [3:0] in;
    wire [1:0] out;
    wire v;

    PENC42 pe1(in,out,v);

    initial begin

        in=4'b0000 ;
        #50;
        in=4'b0001 ;
        #50;
        in=4'b0010 ;
        #50;
        in=4'b0100 ;
        #50;
        in=4'b1000 ;
        #50;

    end

endmodule
```

QUADRANTE

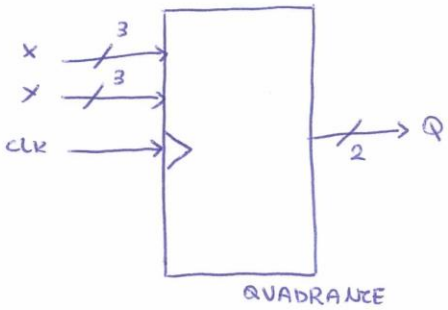


X	Y	X_{bim}	Y_{bim}	Q
+1	+1	001	001	0
-1	+1	111	001	1
-1	-1	111	111	2
+1	-1	001	111	3

Complemento a 2:

$-1_{10} \rightarrow +1_{10} = 001_2 \Rightarrow 110 + 1 = 111_2$

$-1_{10} = 111_2$



```
module Quadrante(X,Y,CLK,Q);

    input [2:0] X;
    input [2:0] Y;
    input CLK;
    output [1:0] Q;

    reg [1:0] Q;

    always @(CLK) begin

        if ((X[2]==0)&(Y[2]==0))
            assign Q = 2'b00;
        else if ((X[2]==1)&(Y[2]==0))
            assign Q = 2'b01;
        else if ((X[2]==1)&(Y[2]==1))
            assign Q = 2'b10;
        else if ((X[2]==0)&(Y[2]==1))
            assign Q = 2'b11;

    end

endmodule

module test_bench();

    reg [2:0] x;
    reg [2:0] y;
    reg clk;
    wire [1:0] q;

    Quadrante q1(x,y,clk,q);

    initial begin

        clk = 1'b1;
        x = 3'b001;
        y = 3'b001;
        #50;

        clk = 1'b0;
        #50;

        clk = 1'b1;
        x = 3'b111;
        y = 3'b001;
        #50;

        clk = 1'b0;
        #50;

        clk = 1'b1;
        x = 3'b111;
        y = 3'b111;
        #50;

        clk = 1'b0;
```

```
        #50;

        clk = 1'b1;
        x = 3'b001;
        y = 3'b111;
        #50;

    end

endmodule
```