Codici da sapere

giovedì 6 gennaio 2022 11:22

FULL ADDER

```
entity FA is
        generic (n : integer);
        port(
                A, B, Cin : in bit;
                S, Cout : out bit );
end FA;

architecture ex1 of FA is
signal p,g : bit

begin
        p <= A xor B;
        g <= A and B;

        Cout <=  (p and Cin) or g;
        S <= p xor Cin;
end ex1;
```

CARRY LOOK AHEAD senza overflow

```
entity RC is
        generic (n : integer);
        port(
                A, B : in std_logic_vector(n-1 downto 0);
                S: out  std_logic_vector(n downto 0);
                Cin : in std_logic;
end RC;

architecture ex1 of RC is
signal p,g : std_logic_vector(n-1 downto 0);
signal carry: std_logic_vector(n downto 0);

begin
        p <= A xor B;
        g <= A and B;

        carry(0) <= Cin;
        carry(n downto 1) <= p and carry(n-1 downto 0) or g;
        S <= p xor carry(n-1 downto 0);

end ex1;
```

CARRY LOOK AHEAD con segno

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity RC is
	generic (n : integer);
	port(
		A, B: in std_logic_vector(n-1 downto 0); -- 8bit
		S: out  std_logic_vector(n downto 0); --9 bit
		Cin : in std_logic);
end RC;

architecture ex1 of RC is
signal p,g : std_logic_vector(n downto 0);
signal carry: std_logic_vector(n+1 downto 0); -- 10 bit

begin
	p <= (A(n-1) xor B(n-1)) & (A xor B);  -- 9 bit
	g <= (A(n-1) and B(n-1)) & (A and B);  -- 9 bit

	carry(0) <= Cin;
	carry(n+1 downto 1) <= (p and carry(n downto 0)) or g;

	S <= p xor carry(n downto 0);

end ex1;
```

FLIP FLOP con clear e reset asincroni sensibile ai fronti di salita del clock

```
entity FF is
	generic (n:integer);
	port (
		D: in std_logic_vector(n-1 downto 0);
		pr: in std_logic;
		clr: in std_logic;
		clk: in std_logic;
		R: in std_logic_vector(n-1 downto 0)mm
	);
end FF;

architecture Behav of FF is

begin

	process(clk, pr, clr)
```

```
            if (clr ='1') then
                R <= (others => '0'));
            elsif (pr = '1') then
                R <= (others => '1'));
            else if (rising_edge(clk)) then
                R <= D;
        end process;

end Behav;
```

FLIP FLOP con clear e reset sincroni sensibile ai fronti di salita del clock

```
entity FF is
        generic (n:integer);
        port (
            D: in std_logic_vector(n-1 downto 0);
            pr: in std_logic;
            clr: in std_logic;
            clk: in std_logic;
            R: in std_logic_vector(n-1 downto 0)
        );
end FF;

architecture Behav of FF is

begin

        process(clk)
            if (rising_edge(clk))
                if (clr ='1') then
                    R <= (others => '0'));
                elsif (pr = '1') then
                    R <= (others => '1'));
                elsif (pr = '0' and clr = '0') then
                    R <= D;
        end process;

end Behav;
```

SOTTRATTORE

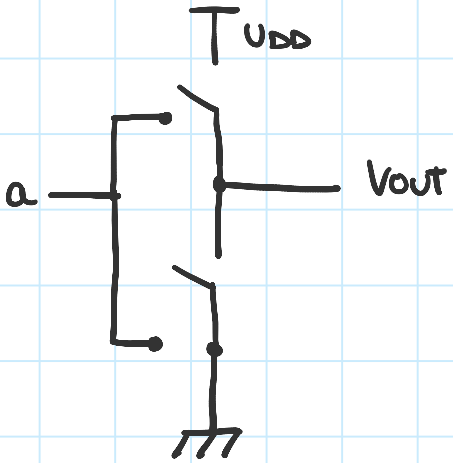
```
library IEEE;
use IEEE.std_logic_1164.all;

entity sub is
        generic (n:integer)
        port(
```

```
            A, B : in std_logic_vector(n-1 downto 0);
            sel: in std_logic;
            Ris: out std_logic_vector(n downto 0);
        );
end sub;

architecture behav of sub is
signal EA, EB, nB, p, g: std_logic_vector(n downto 0);
signal carry: std_logic_vector(n+1 downto 0);

begin
        EA <= A(n-1) & A;
        nB <= B(n-1) & B;
        EB <= nB when sel='0' else
                not nB when sel='1' else
                (others => 'x');
        carry(0) <= sel;
        carry(n+1 downto 1) <= (p and carry(n downto 0)) or g;
        p <= EA xor EB;
        g <= EA and EB;
        Ris <= p xor carry(n downto 0);


end behav;
```

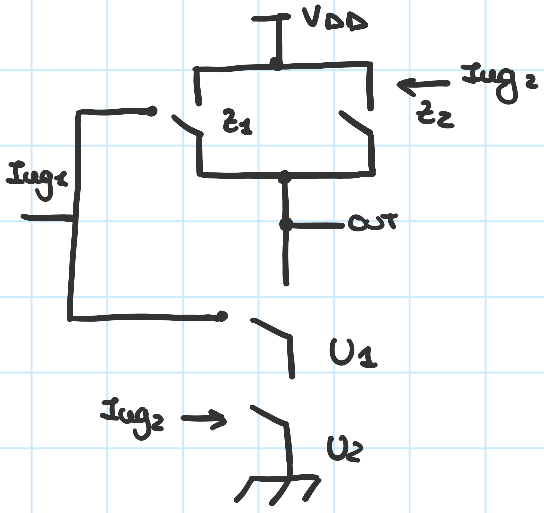
PACKAGE

```
library IEEE;
use IEEE.std_logic_1164.all;

package costanti is
        constant n: integer := 8;
end package costanti;
```

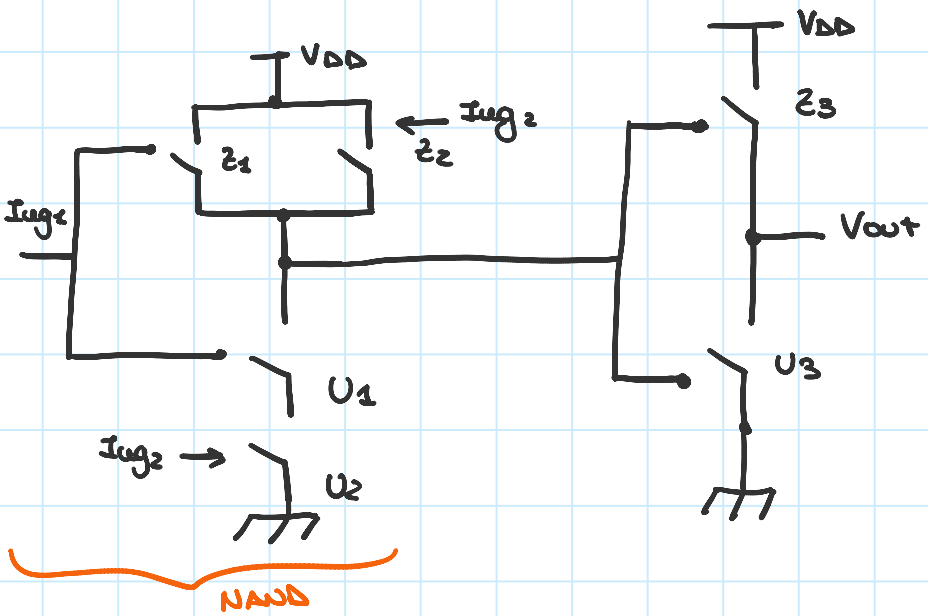
INVERTER



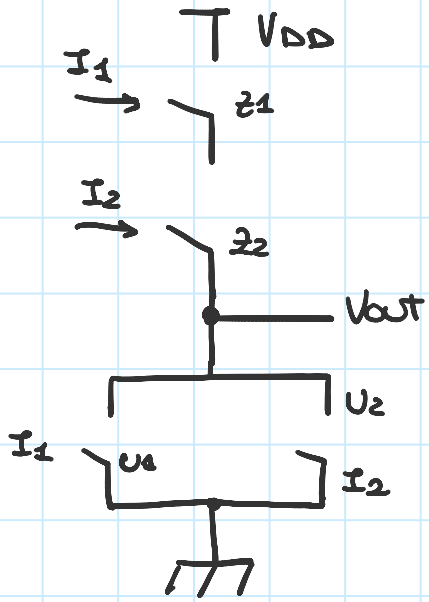
NAND



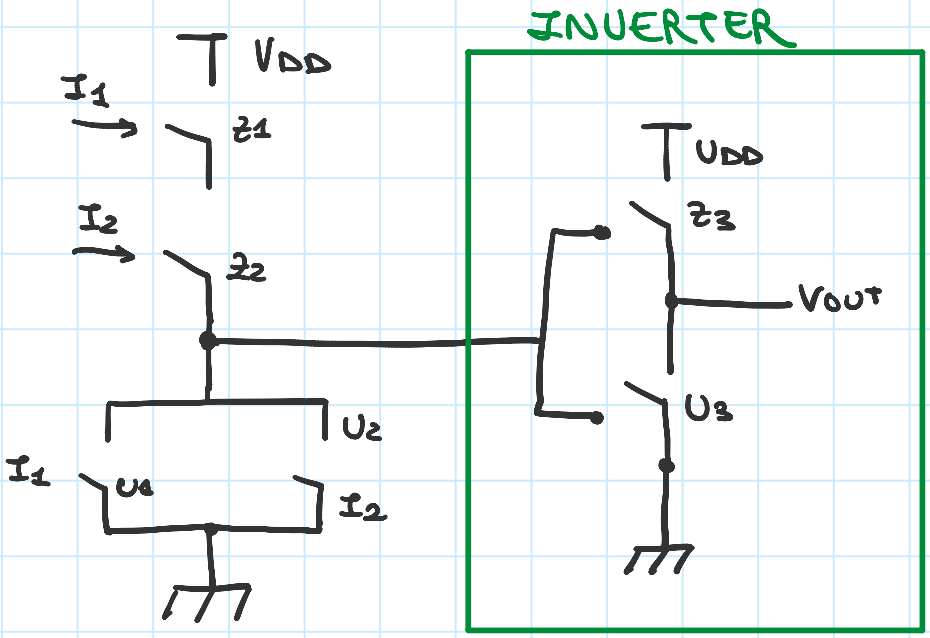
AND



NOR



OR



XNOR