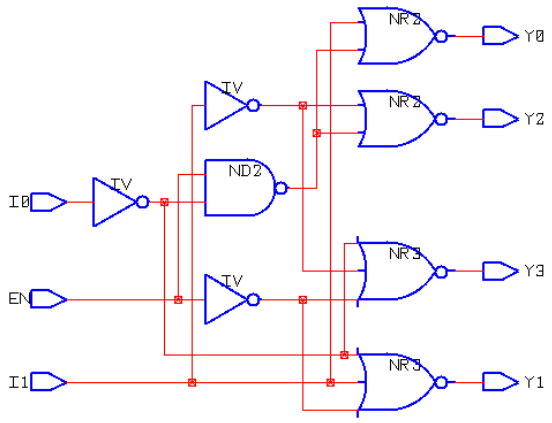
Ö4.1 Konstruera en 2-4 binäravkodare med *enable*-signal. Då enable-signalen (EN) är låg ska samtliga utgångar vara låga. I annat fall ska utgångarna följa funktionen för binär avkodning.

Ingångar: EN, I0, I1 (bit)

Utgångar: Y0, Y1, Y2, Y3 (bit)

```
architecture rtl of E4_1 in
begin
  p1 : process (I0, I1, EN)
    variable I10 : bit_vector(1 downto 0);  -- concatenation variable
    variable Y : bit_vector(3 downto 0);
  begin  -- process p1
    I10 := I1 & I0;
    if (EN = '1') then
      case I10 is
        when "00"   => Y := "0001";
        when "01"   => Y := "0010";
        when "10"   => Y := "0100";
        when others   => Y := "1000";
      end case;
    else
      Y := "0000";
    end if;
    Y3 <= Y(3); Y2 <= Y(2); Y1 <= Y(1); Y0 <= Y(0);
  end process;
end rtl;
```



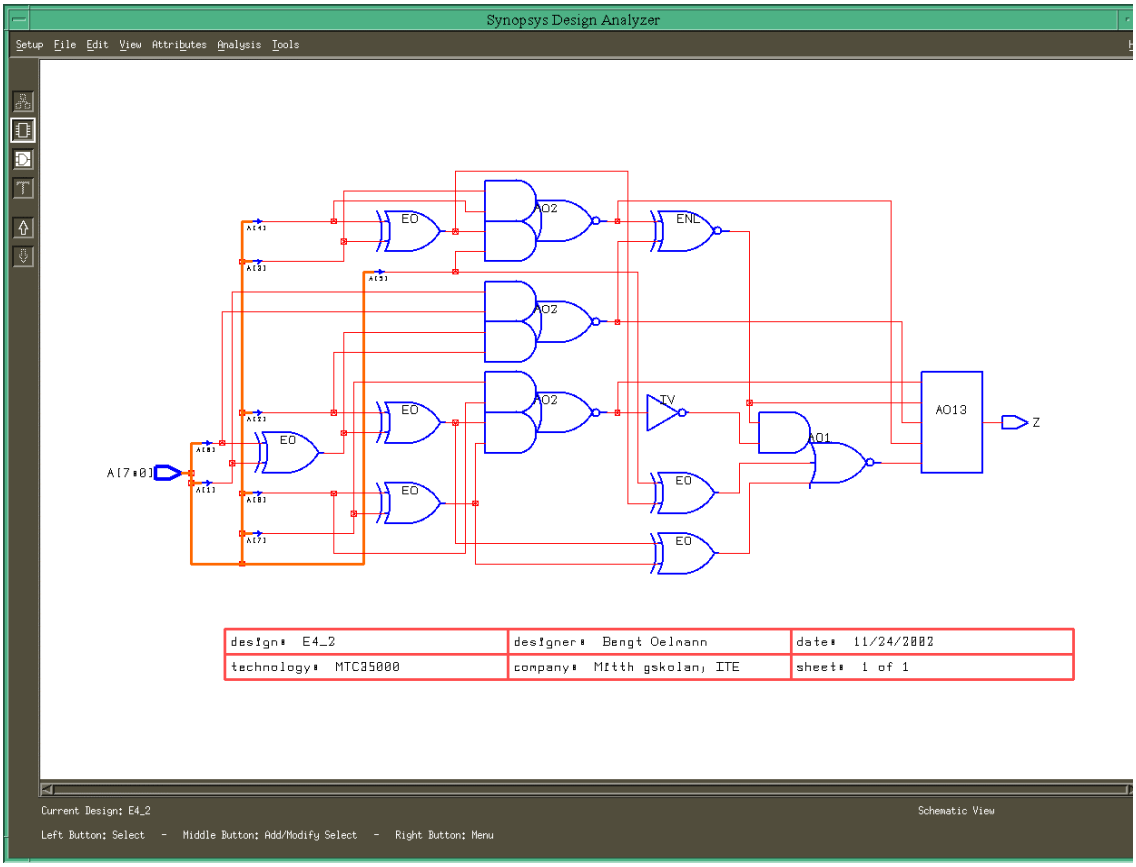
design: E4_1	designer: Bengt Oelmann	date: 11/24/2002
technology: MTC35000	company: Mitth gskolan, ITE	sheet: 1 of 1

Ö4.2 Konstruera en krets som ger ut ’1’ då fler än hälften av ingångarna till kretsen är ’1’.

Ingångar: A (std_logic_vector(7 downto 0)

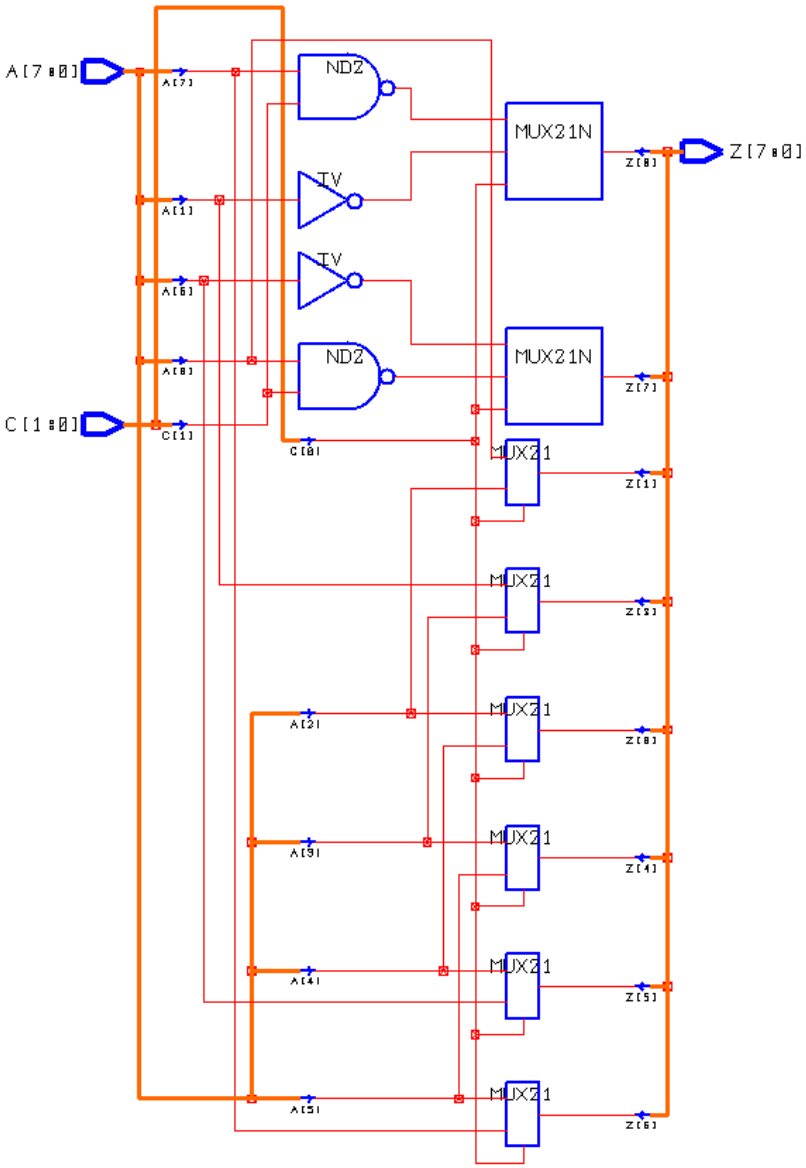
Utgång: Z (std_logic)

```
Library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity E4_2 is
  port (
    A: in std_logic_vector(7 downto 0);
    Z: out std_logic);
end E4_2;

architecture rtl of E4_2 is
begin
  p1 : process (A)
    variable count : integer;
  begin  -- process p1
    count := 0;
    for i in 7 downto 0 loop
      if A(i) = '1' then
        count := count +1;
      end if;
    end loop;  -- i

    if count > 4 then
      Z <= '1';
    else
      Z <= '0';
    end if;
  end process;
end rtl;
```



Ö4.3

```
Library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity E4_3 is
  port (
    A: in std_logic_vector(7 downto 0);
    C: in std_logic_vector(1 downto 0);
    Z: out std_logic_vector(7 downto 0));
end E4_3;

architecture rtl of E4_3 is
begin
  p1 : process (A, C)
begin
  case C is
    when "00" => Z <= A(6 downto 0)&'0';
    when "01" => Z <= '0'&A(7 downto 1);
    when "10" => Z <= A(6 downto 0)&A(7);
    when others => Z <= A(0)&A(7 downto 1);
  end case;
end process;
end rtl;
```



Ö4.4

```
Library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_ARITH.all;
use IEEE.STD_LOGIC_UNSIGNED.all;

entity E4_4 is
  port (
    A, B : in  std_logic_vector(7 downto 0);
    add_sub: in std_logic;
    SUM    : out std_logic_vector(7 downto 0);
    C : out std_logic);
end E4_4;

architecture rtl of E4_4 is
begin  -- rtl

  process (A, B, add_sub)
    variable sum_v : std_logic_vector(8 downto 0);

  begin
    if add_sub = '0' then
      sum_v := ('0' & A) + ('0' & B);
    else
      sum_v := ('0' & A) - ('0' & B);
    end if;
    SUM <= sum_v(7 downto 0);
    C <= sum_v(8);
  end process;
end rtl;
```