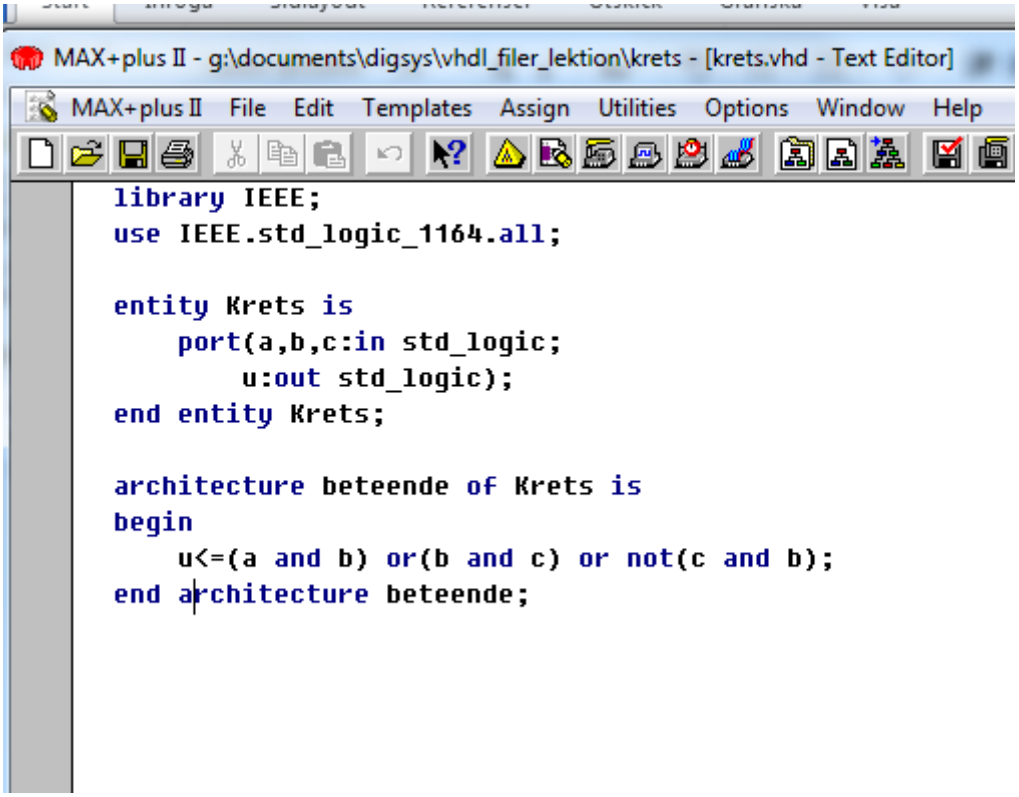
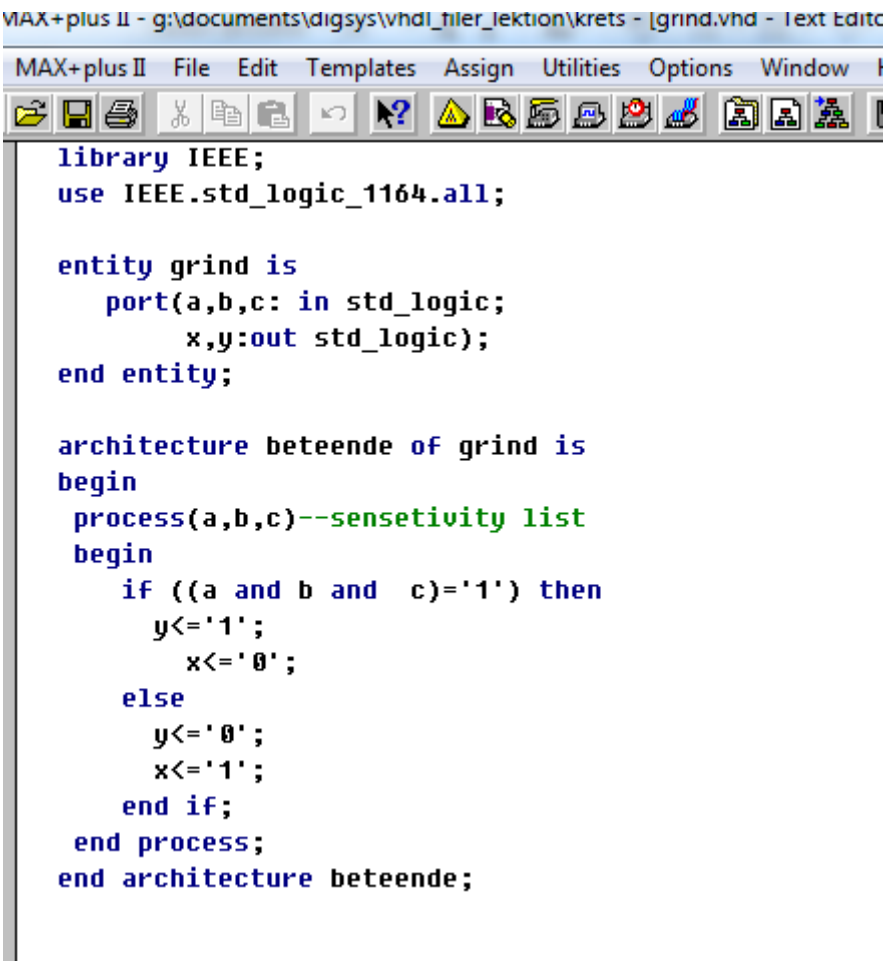

MAX+plus II - g:\documents\digsys\vhdl_filer_lektion\krets - [krets.vhd - Text Editor]
MAX+plus II File Edit Templates Assign Utilities Options Window Help
library IEEE;
use IEEE.std_logic_1164.all;
entity Krets is
port(a,b,c:in std_logic;
u:out std_logic);
end entity Krets;
architecture beteende of Krets is
begin
u<=(a and b) or(b and c) or not(c and b);
end architecture beteende;


MAX+plus II File Edit Templates Assign Utilities Options Window
library IEEE;
use IEEE.std_logic_1164.all;
entity grind is
port(a,b,c: in std_logic;
x,y:out std_logic);
end entity;
architecture beteende of grind is
begin
process(a,b,c)--sensetivity list
begin
if ((a and b and c)='1') then
y<='1';
x<='0';
else
y<='0';
x<='1';
end if;
end process;
end architecture beteende;

MAX+plus II File Edit Templates Assign Utilities Options Window Help

MAX+plus II - g:\documents\digsys\vhdl_filer_lektion\krets - [och_3.vhd - Text Editor]

MAX+plus II File Edit Templates Assign Utilities Options Window Help

```
--detta är en kommentar
--enkel 3-ingångars och funktion
--föreläsning ht-06
-- språket är inte casesenstivt(skiljer inte på stora och SMÅ bokstäver)
library IEEE;
use IEEE.std_logic_1164.all;

entity och_3 is
    port( a,b,c: in std_logic; ut:out std_logic);
end entity och_3;

architecture beteende of och_3 is
begin
     ut<=a and b and c;
end architecture beteende;
```

MAX+plus II - g:\documents\digsys\vhdl_filer_lektion\krets - [komparator.vhd - Text Editor]

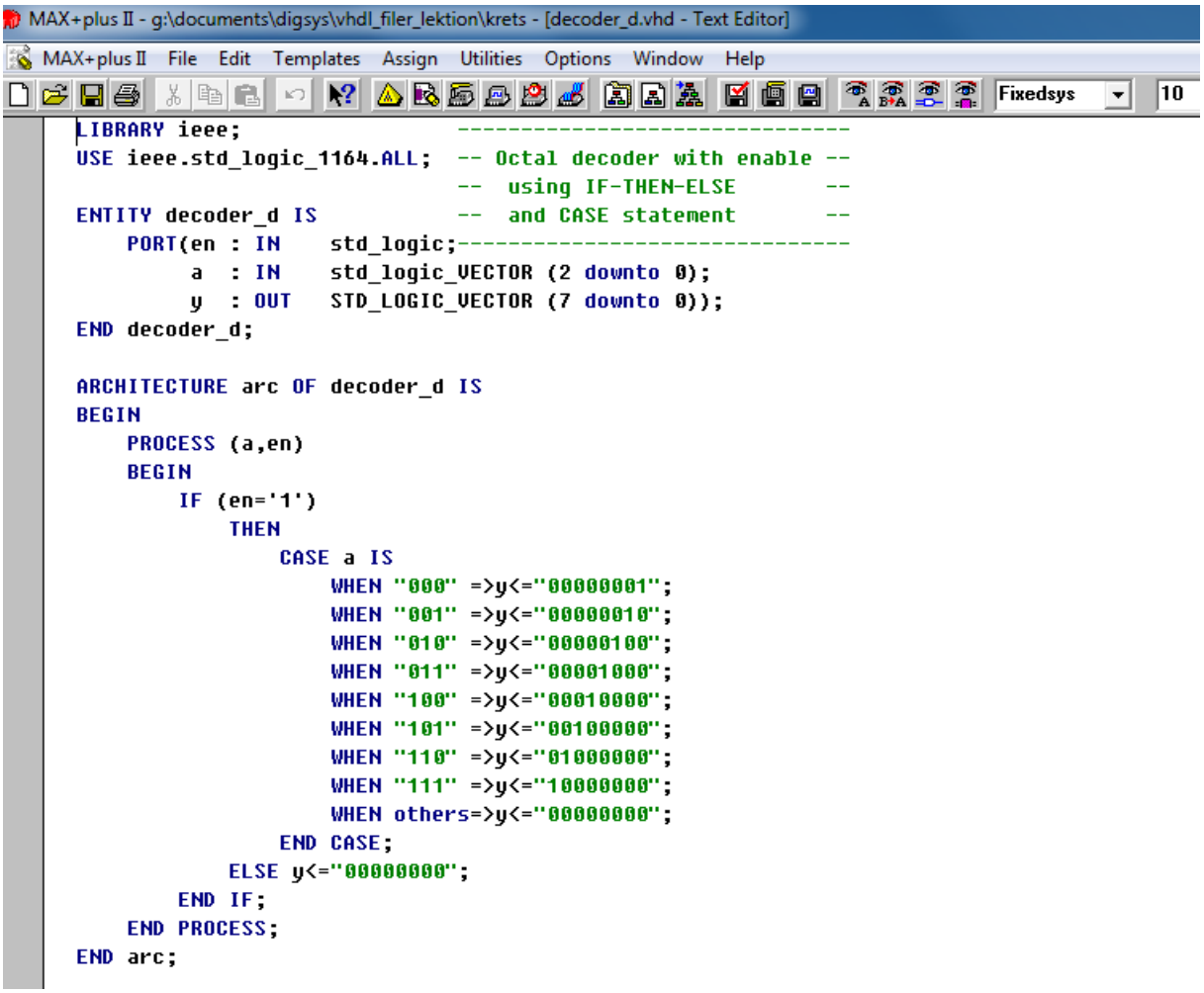
MAX+plus II File Edit Templates Assign Utilities Options Window Help

Fixedsys 10

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

entity Komparator is
    port (a, b:in std_logic_vector(7 downto 0);
        ut,ut2:out std_logic);
end entity Komparator;
architecture beteende of Komparator is
begin
ut2<=(a(0) xor b(0)) or (a(1) xor b(1)) or (a(2) xor b(2)) or (a(3) xor b(3)) or (a(4) xor b(4)) or (a(5) xor b(5)) or (a(6) xor b(6))

    process(a,b)
    begin
     if(a=b) then
        ut<='1';
     else
        ut<='0';
     end if;
    end process;
end architecture beteende;
```




```
LIBRARY ieee;                   ------------------------------
USE ieee.std_logic_1164.ALL;  -- Octal decoder with enable --
                              --  using IF-THEN-ELSE       --
ENTITY decoder_d IS           --  and CASE statement       --
    PORT(en : IN    std_logic;------------------------------
         a  : IN    std_logic_VECTOR (2 downto 0);
         y  : OUT   STD_LOGIC_VECTOR (7 downto 0));
END decoder_d;

ARCHITECTURE arc OF decoder_d IS
BEGIN
    PROCESS (a,en)
    BEGIN
        IF (en='1')
            THEN
                CASE a IS
                    WHEN "000" =>y<="00000001";
                    WHEN "001" =>y<="00000010";
                    WHEN "010" =>y<="00000100";
                    WHEN "011" =>y<="00001000";
                    WHEN "100" =>y<="00010000";
                    WHEN "101" =>y<="00100000";
                    WHEN "110" =>y<="01000000";
                    WHEN "111" =>y<="10000000";
                    WHEN others=>y<="00000000";
                END CASE;
            ELSE y<="00000000";
        END IF;
    END PROCESS;
END arc;
```

```
-- Sdet1_mo , tillståndsgraf på sid 217 i läroboken och på sid 407 samt sid 92
library IEEE;
use IEEE.std_logic_1164.all;

entity Sdet1_mo is
    port( x, clock: in std_logic;
        u:out std_logic);
end entity Sdet1_mo;

architecture beteende of Sdet1_mo is
type state_type is (s0,s1,s2,s3);
signal present_state, next_state: state_type;
begin
    process(present_state,x)
    begin
        case present_state is
            when s0 =>  if x='0' then
                            next_state<=s0;
                        else
                            next_state<=s1;
                        end if;
            when s1 =>  if x='0' then
                            next_state<=s0;
                        else
                            next_state<=s2;
                        end if;
            when s2 =>  if x='0' then
                            next_state<=s0;
                        else
                            next_state<=s3;
                        end if;
            when s3 =>  if x='0' then
                            next_state<=s0;
                        else
                            next_state<=s3;
                        end if;
        end case;
    end process;

    process(present_state)
        begin-- detta går minst lika bra med if ....  then.... else...
            case present_state is
                when    s0  =>u<='0';
                when    s1  =>u<='0';
                when    s2  =>u<='0';
                when    s3  =>u<='1';
            end case;
    end process;

    process(clock)
    begin
        if rising_edge(clock) then
            present_state<=next_state;
        end if;
    end process;

end architecture;
```