VHDL
sekvensnät

```
-- Sdet1_mo , tillståndsgraf på sid 217 i läroboken och på sid 407
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;
```

Sdet1_me

Sekvensnätet som ett Mealynät

```
 library IEEE;
use IEEE.std_logic_1164.all;
entity Sdet1_me is
   port(x,clock:in std_logic;
      u:out std_logic);
end entity Sdet1_me;

architecture beteende of Sdet1_me is
type state_type is (s0,s1,s2);
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<=s2;
                     end if;
      end case;
   end process;

   process(present_state,x)
      begin
            if present_state=s2 and x='1' then
               u<='1';
            else
               u<='0';
            end if;

            end process;
```

```
    process(present_state,x)
       begin
             if present_state=s2 and x='1' then
                u<='1';
             else
                u<='0';
             end if;

    end process;

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

end architecture beteende;
```

En enkel räknare

En binärräknare som räknar modulo 16

```
-- en 4-bitars binärräknare som räknar modulo-16

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;

entity cnt4b is
   port(clock:in std_logic;
      q:out std_logic_vector(3 downto 0));
end entity cnt4b;

architecture beteende of cnt4b is
subtype state_type is integer range 0 to 15;
signal present_state, next_state:state_type;
begin
   process(present_state)
   begin
         if present_state=15 then next_state<=0;
            else
               next_state<=present_state+1;
         end if;
   end process;
   q<=conv_std_logic_vector(present_state,4); -- omvandlar en integer till en 4-bitars vektor.

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

modulo -16 räknare

med enable

```vhdl
-- en 4-bitars binärräknare som räknar modulo-16 med enable

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;

entity cnt4b_e is
   port(clock, enable:in std_logic;
      q:out std_logic_vector(3 downto 0));
end entity cnt4b_e;

architecture beteende of cnt4b_e is
subtype state_type is integer range 0 to 15;
signal present_state, next_state:state_type;
begin
   process(present_state)
      begin
         if enable='0' then next_state<=present_state;
         elsif present_state=15 then next_state<=0;
         else
            next_state<=present_state+1;
         end if;
   end process;
q<=conv_std_logic_vector(present_state,4);

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

Modulo-16 räknare med reset

```vhdl
-- en 4-bitars binärräknare som räknar modulo-16 med synkron reset
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;

entity cnt4b_r is
   port(clock, reset:in std_logic;
      q:out std_logic_vector(3 downto 0));
end entity cnt4b_r;

architecture beteende of cnt4b_r is
subtype state_type is integer range 0 to 15;
signal present_state, next_state:state_type;
begin
   process(present_state,reset)
      begin
         if reset='1' then next_state<=0;
         elsif present_state>=15 then next_state<=0;
         else
             next_state<=present_state+1;
         end if;
   end process;
   q<=conv_std_logic_vector(present_state,4);

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

Räknare med bara en process

```vhdl
-- en 4-bitars binärräknare som räknar modulo-16 med synkron reset
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;

entity cnt4b_r_one is
   port(clock, reset:in std_logic;
      q:out std_logic_vector(3 downto 0));
end entity cnt4b_r_one;

architecture beteende of cnt4b_r_one is
subtype state_type is integer range 0 to 15;
signal state:state_type;
begin
   process(clock)
   begin
      if rising_edge(clock) then
         if reset='1' then state<=0;
         elsif state=15 then state<=0;
         else
            state<=state+1;
         end if;
      end if;
   end process;
q<=conv_std_logic_vector(state,4);

end architecture beteende;
```

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use work.Minfunktion.all;
entity varcounterII is
   port(clock: in std_logic;
      load: in std_logic_vector(5 downto 0);
      q: out  std_logic_vector(5 downto 0));
end entity varcounterII;

architecture beteende of varcounterII is
subtype state_type is integer range 0 to 63;
signal present_state, next_state:state_type;

begin
    process(present_state,load) is
  variable I:state_type ;
     begin
      I:=vector_int(load);
      if present_state=I then
        next_state<=0;
       else
        next_state<=present_state+1;
       end if;
  end process;
   q<=conv_std_logic_vector(present_state,6);

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

```
Library IEEE;
use ieee.std_logic_1164.all;

package Minfunktion is
function vector_int(vect:in std_logic_vector) return integer;
end Minfunktion;
package body Minfunktion is
function vector_int(vect:in std_logic_vector) return integer is
variable result:integer:=0;
begin
   If vect(0)='1' then result:=1;
   end if;
   if vect(1)='1' then result:=result+2;else result:=result;
   end if;
   if vect(2)='1' then result:=result+4;else result:=result;
   end if;
   if vect(3)='1' then result:=result+8;else result:=result;
   end if;
   if vect(4)='1' then result:=result+16;else result:=result;
   end if;
   if vect(5)='1' then result:=result+32;else result:=result;
   end if;
   return(result);
end vector_int;
end minfunktion;
```

Räknare Graykod

```
LIBRARY ieee;              ------------------------------------
USE ieee.std_logic_1164.ALL; -- State Machine Gray Code Counter --
                           ------------------------------------
ENTITY state_gray IS
   PORT(clock,reset: IN STD_LOGIC;
      q     : OUT STD_LOGIC_VECTOR(2 downto 0));
END state_gray ;

ARCHITECTURE arc OF state_gray IS
   TYPE state_type IS (s0, s1, s2, s3, s4, s5, s6, s7);
   SIGNAL state: state_type;
BEGIN
   PROCESS (clock)
   BEGIN
      if rising_edge(clock) THEN
         if reset='1' then state<=s0;
         else
         CASE state IS
            WHEN s0 => state <= s1; q<="000";
            WHEN s1 => state <= s2; q<="001";
            WHEN s2 => state <= s3; q<="011";
            WHEN s3 => state <= s4; q<="010";
            WHEN s4 => state <= s5; q<="110";
            WHEN s5 => state <= s6; q<="111";
            WHEN s6 => state <= s7; q<="101";
            WHEN s7 => state <= s0; q<="100";
         END CASE;
         end if;
       END IF;
   END PROCESS;

   --WITH state SELECT
   -- q  <= "000" WHEN  s0,
         -- "001" WHEN  s1,
         -- "011" WHEN  s2,
         -- "010" WHEN  s3,
         -- "110" WHEN  s4,
         -- "111" WHEN  s5,
         -- "101" WHEN  s6,
         -- "100" WHEN  s7;
END arc;
```

Johnsonräknare

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

entity Johnson is
   port(clock,reset:in std_logic;
         ut: out std_logic_vector(4 downto 0));
end entity Johnson;

architecture beteende of Johnson is
type state_type is (s0,s1,s2,s3,s4);
signal present_state, next_state: state_type;
begin
   process(present_state)
   begin
      case present_state is
         when s0 =>next_state<=s1;
         when s1 =>next_state<=s2;
         when s2 =>next_state<=s3;
         when s3 =>next_state<=s4;
         when s4 =>next_state<=s0;
      end case;
   end process;
   process(present_state)
   begin
      case present_state is
         when s0 =>ut<="10000";
         when s1 =>ut<="01000";
         when s2 =>ut<="00100";
         when s3 =>ut<="00010";
         when s4 => ut<="00001";
      end case;
   end process;
   process(clock)
   begin
      if rising_edge(clock) then
         if(reset='1') then
            present_state<=s0;
         else
            present_state<=next_state;
         end if;
      end if;
   end process;
end architecture beteende;
```