MuxDispDriver version 1 og 2 – og design af UR

Løsningsforslag

Lav en VHDL-komponent ”MuxDispDriver” med den viste ENTITY.
Den skal kunne styre et Mux-Display med fire cifre + dp (og eventuelt blanke dem)
Desuden skal kredsen kunne levere to frekvenser på 1 Hz (ca) og 1 kHz (sharp)

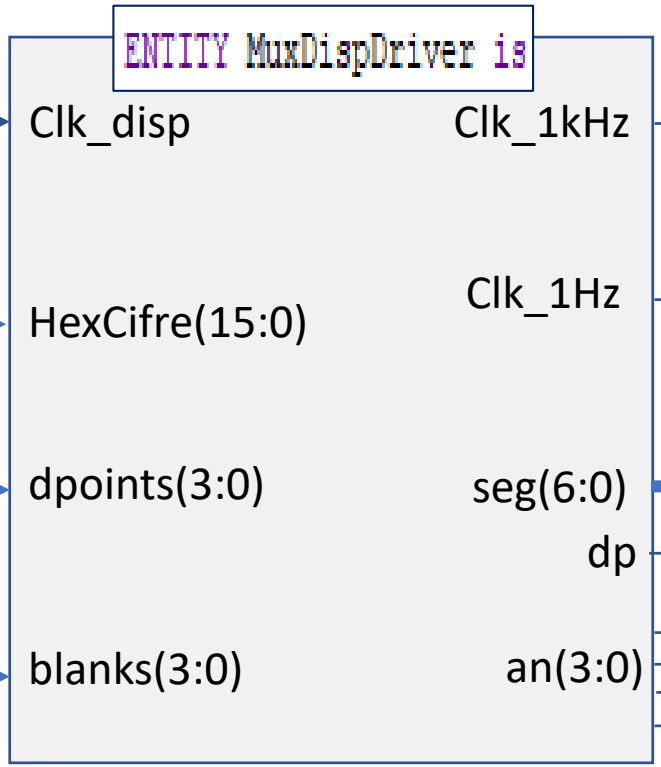
```
ENTITY MuxDisplay is
  Port (Clk_disp:  in  STD_LOGIC;
        HexCifre:  in  STD_LOGIC_VECTOR (15 downto 0);
        dpoints:   in  STD_LOGIC_VECTOR (3 downto 0);
        blanks:    in  STD_LOGIC_VECTOR (3 downto 0);
        Clk_1kHz:  out STD_LOGIC;
        Clk_1Hz:   out STD_LOGIC;
        an:        out STD_LOGIC_VECTOR (3 downto 0);
        seg:       out STD_LOGIC_VECTOR (6 downto 0);
        dp:        out STD_LOGIC);
end MuxDisplay;
```

Clock frekvens kan forventes at være = 100 MHz
(Men prøv andre frekvenser)

HexCifre er 4x4 bit der skal vises på displayet.
”0001 0110 1001 1110” =**169E**

dpoints er 4-bit som styrer de respektive dp.
”1001” =**1.69E.**

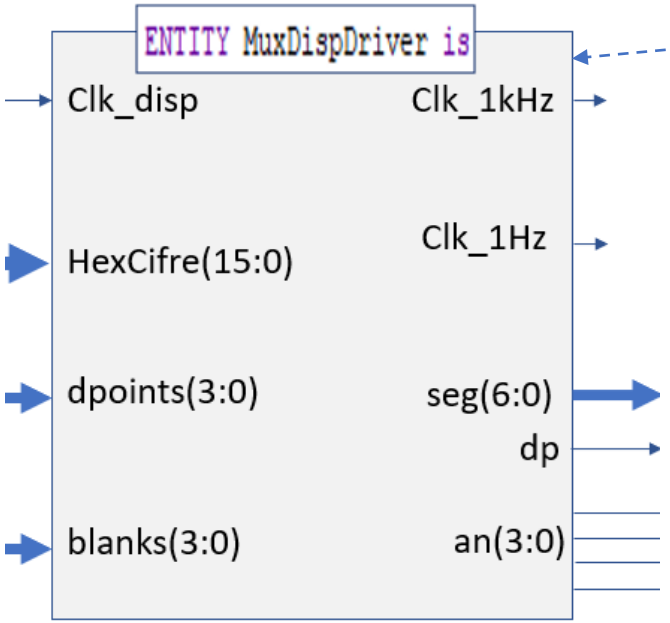
blanks er 4-bit der styrer om et display skal blankes
”0100” =**1.69E.**



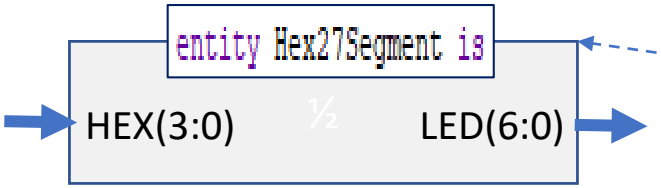
Denne udgang skal levere en frekvens på 1 kHz (duty cycle ?) som med fordel kan bruges til et ur .. Etc.

Denne udgang skal levere en frekvens på 1,024 Hz (50% dc)
Kan bruges til blink … etc.

Version1




```
ENTITY MuxDispDriver is
  Port (Clk_disp:  in  STD_LOGIC;
        HexCifre:  in  STD_LOGIC_VECTOR (15 downto 0);
        dpoints:   in  STD_LOGIC_VECTOR (3 downto 0);
        blanks:    in  STD_LOGIC_VECTOR (3 downto 0);
        Clk_1kHz:  out STD_LOGIC;
        Clk_1Hz:   out STD_LOGIC;
        an:        out STD_LOGIC_VECTOR (3 downto 0);
        seg:       out STD_LOGIC_VECTOR (6 downto 0);
        dp:        out STD_LOGIC);
end MuxDispDriver;
```




```
entity Hex27Segment is
    Port ( HEX : in STD_LOGIC_VECTOR (3 downto 0);
           LED : out STD_LOGIC_VECTOR (6 downto 0));
end Hex27Segment;
```

Bemærk forskellen på entity og component

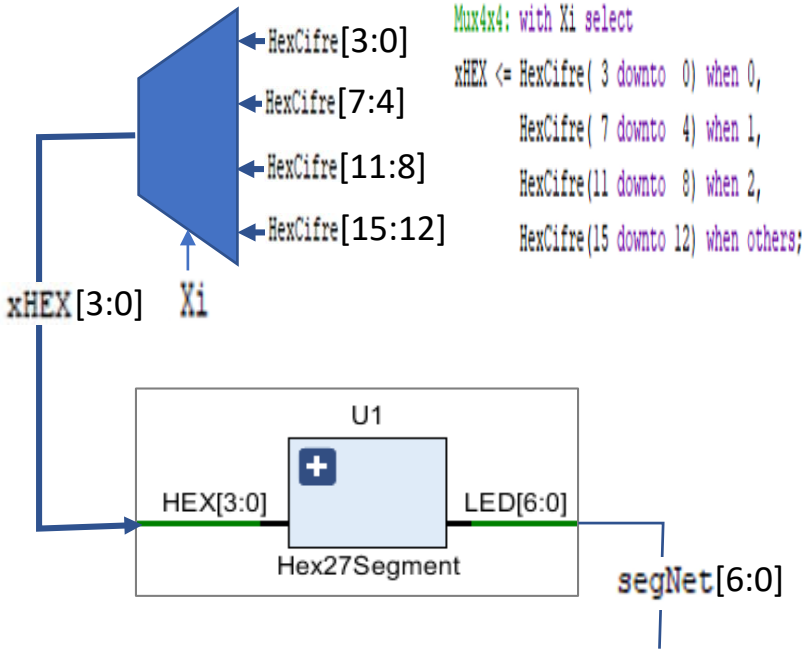
```
--#################################################################
--# Den architecture som ligger nederst bliver brugt ...
ARCHITECTURE Version1 of MuxDispDriver is
    component Hex27Segment
        port (HEX : in  std_logic_vector (3 downto 0);
              LED : out std_logic_vector (6 downto 0));
    end component;

    signal xHEX :  std_logic_vector (3 downto 0);
    signal segNet: std_logic_vector (6 downto 0);

    signal Scale1023:  std_logic_vector( 9 downto 0) := "0111111100";
    signal Sel :   std_logic_vector (1 downto 0) := "00";
    signal Xi:     integer range 0 to 3;
begin
```

Imellem ARCHITECTURE og begin kan/skal man erklære eksterne VHDL-komponenter og interne signaler som er nødvendige for at beskrive funktionaliteten.

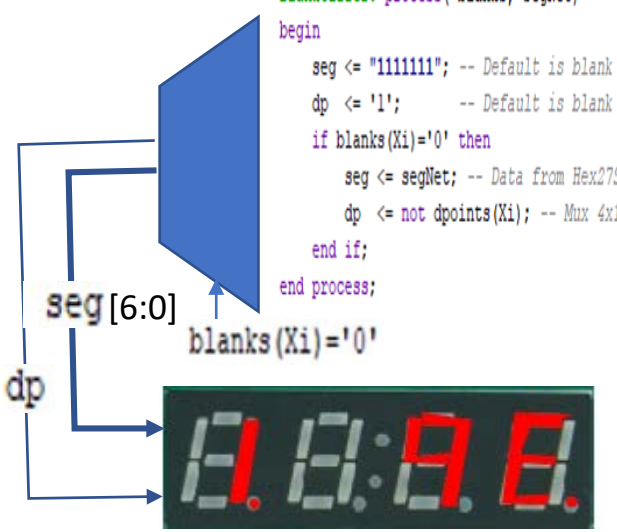
Multipleksere og Hex til 7-segment

Version1

```
HexCifre:  in  STD_LOGIC_VECTOR (15 downto 0);
```




```
Mux4x4: with Xi select
xHEX <= HexCifre( 3 downto  0) when 0,
        HexCifre( 7 downto  4) when 1,
        HexCifre(11 downto  8) when 2,
        HexCifre(15 downto 12) when others;
```

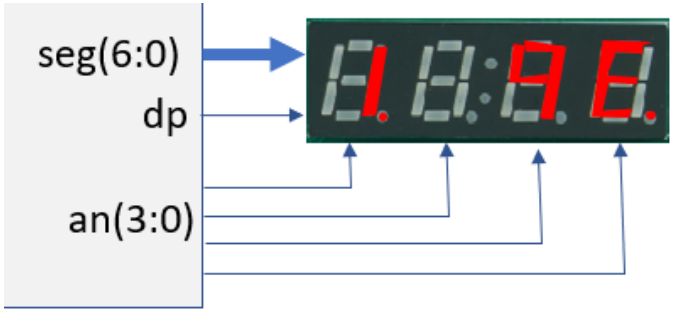


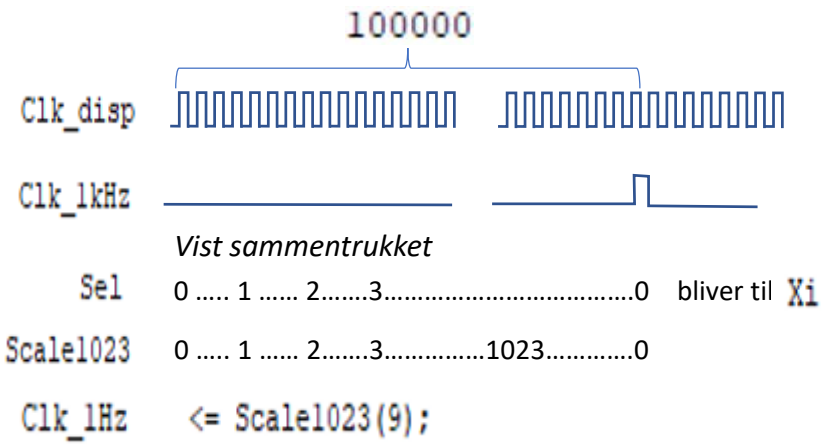

```
BlankCiffer: process( blanks, SegNet)
begin
    seg <= "1111111"; -- Default is blank ciffer
    dp  <= '1';       -- Default is blank dp
    if blanks(Xi)='0' then
        seg <= segNet; -- Data from Hex27Segment
        dp  <= not dpoints(Xi); -- Mux 4x1
    end if;
end process;
```

```
--#################################################################
--# Den architecture som ligger nederst bliver brugt ...
ARCHITECTURE Version1 of MuxDispDriver is
    component Hex27Segment
        port (HEX : in  std_logic_vector (3 downto 0);
              LED : out std_logic_vector (6 downto 0));
    end component;

    signal xHEX :  std_logic_vector (3 downto 0);
    signal segNet: std_logic_vector (6 downto 0);

    signal Scale1023:  std_logic_vector( 9 downto 0) := "0111111100";
    signal Sel :   std_logic_vector (1 downto 0) := "00";
    signal Xi:     integer range 0 to 3;
begin

    Mux4x4: with Xi select
    xHEX <= HexCifre( 3 downto  0) when 0,
            HexCifre( 7 downto  4) when 1,
            HexCifre(11 downto  8) when 2,
            HexCifre(15 downto 12) when others;

    U1: Hex27Segment  port map (HEX => xHEX, LED => segNet);

    BlankCiffer: process( blanks, SegNet)
    begin
        seg <= "1111111"; -- Default is blank ciffer
        dp  <= '1';       -- Default is blank dp
        if blanks(Xi)='0' then
            seg <= segNet; -- Data from Hex27Segment
            dp  <= not dpoints(Xi); -- Mux 4x1
        end if;
    end process;
```

Encodere og nedskalering af Clk

Version1






```
if rising_edge( Clk_disp) then
    Clk_1kHz    <= '0';
    if Scale100000 <100000 then
        Scale100000 := Scale100000+ 1;
    else
        Scale100000 := 1;
        Clk_1kHz    <= '1';
        Scale1023   <= Scale1023+1;
        Sel <= Sel+1;
    end if;
end if;
```

```
Encoder_1_of_4:
an <= "1110" when Xi=0 else
      "1101" when Xi=1 else
      "1011" when Xi=2 else
      "0111";

 Xi <= conv_integer( Sel); --Omsæt Sel(1:0) til integer Xi

ScaleCounter: process( Clk_disp)
    variable Scale100000: integer range 0 to 100000 := 1;
begin
   if rising_edge( Clk_disp) then
       Clk_1kHz   <= '0';          -- brug 10 til simulering
       if Scale100000 <100000 then -- Scale100000 <10 then
           Scale100000 := Scale100000+ 1;
        else
           Scale100000 := 1;
           Clk_1kHz    <= '1';
           Scale1023   <= Scale1023+1;
           Sel <= Sel+1;
        end if;
    end if;
end process;

    Clk_1Hz    <= Scale1023(9);    -- approx. 1 Hz 50% Duty cycle
end Version1;
```

Version2 er inspireret af Version1 og består af kun en Process med både sekventiel ogcombinatorisk logik

Version2

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

ENTITY MuxDispDriver is
  Port (Clk_disp:  in  STD_LOGIC;
        HexCifre:  in  STD_LOGIC_VECTOR (15 downto 0);
        dpoints:   in  STD_LOGIC_VECTOR (3 downto 0);
        blanks:    in  STD_LOGIC_VECTOR (3 downto 0);
        Clk_1kHz:  out STD_LOGIC;
        Clk_1Hz:   out STD_LOGIC;
        an:        out STD_LOGIC_VECTOR (3 downto 0);
        seg:       out STD_LOGIC_VECTOR (6 downto 0);
        dp:        out STD_LOGIC);
end MuxDispDriver;

ARCHITECTURE Version2 of MuxDispDriver is
BEGIN
    Mux_Display:
    process( Clk_disp, HexCifre, dpoints)
        variable Scale100000: integer range 0 to 100000      := 1;
        variable Scale1023:   std_logic_vector( 9 downto 0)  := "0111111100";
        variable Xi:          integer range 0 to 3           := 0;

        type ROM_array is array (0 to 15) of std_logic_vector (6 downto 0);
        constant Hex27Segm: ROM_array := (
        "1000000","1111001","0100100","0110000", -- 0123
        "0011001","0010010","0000010","1111000", -- 4567
        "0000000","0010000","0001000","0000011", -- 89Ab
        "1000110","0100001","0000110","0001110");-- CdEF
    begin
```

Version2

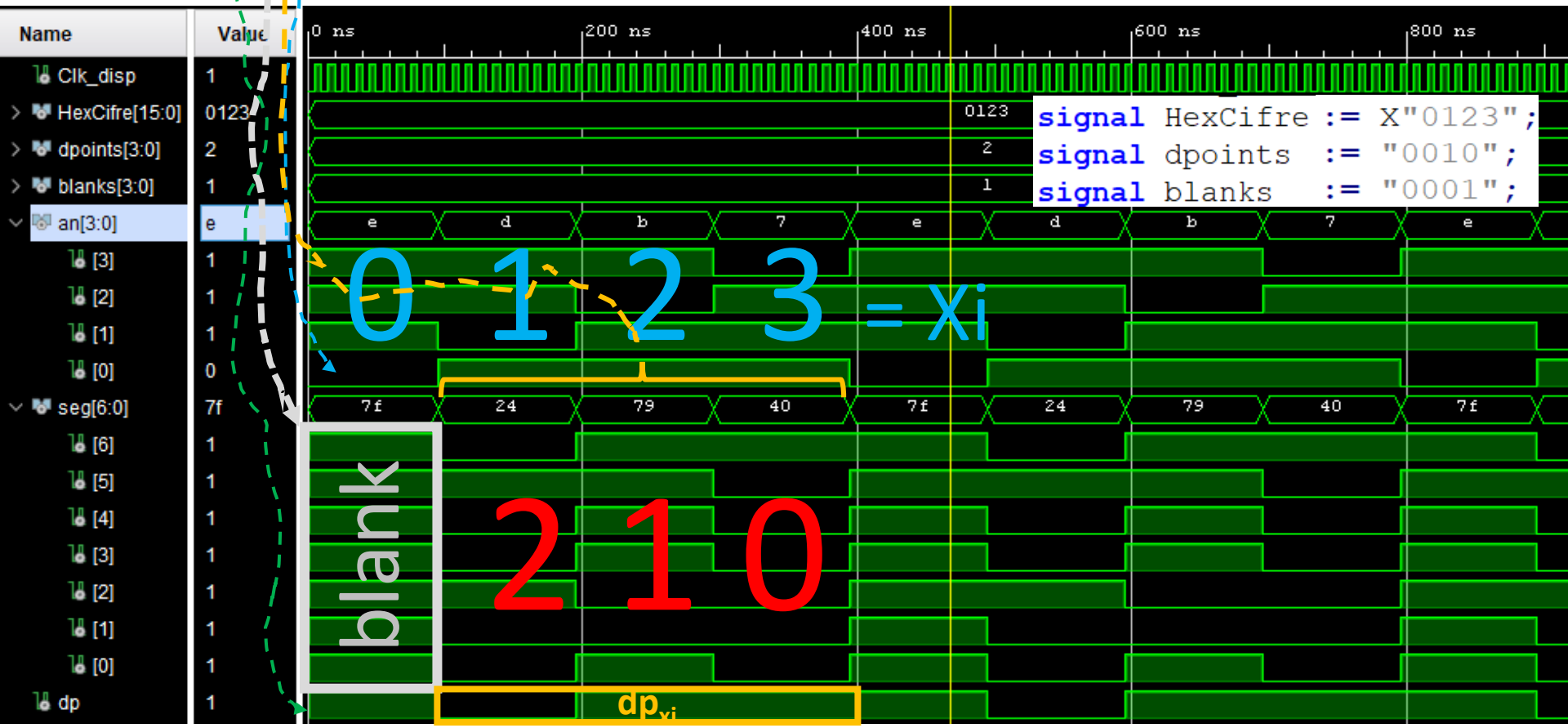
```
    begin
      ------------------------------------------------------ Sequential log
      if rising_edge( Clk_disp) then
          Clk_1kHz    <= '0';
       -- Clk_1Hz     <= Scale1023(9) or Scale1023(8); -- 75% Duty cycle
          Clk_1Hz     <= Scale1023(9);     -- approx. 1 Hz 50% Duty cycle
          if Scale100000 <10 then -- 0000 then
              Scale100000 := Scale100000+ 1;
           else
              Scale100000 := 1;
              Clk_1kHz    <= '1';
              Scale1023   := Scale1023+1;
              if Xi=3 then
                  Xi := 0;
              else
                  Xi := Xi+1;
              end if;
           end if;
       end if;
     ------------------------------------------------------ Combinatorial logic
     -- Note! HexCifre and dpoints must now be added to the sensivity list
       seg <= "1111111"; -- Default .. blank display Xi
       dp  <= '1';       -- Default .. blank dp Xi
       if Blanks(Xi)='0' then
           seg <= Hex27Segm( Conv_integer( HexCifre( Xi*4+3 downto Xi*4)) );
           dp  <= not dpoints(Xi);
       end if;
       an        <= "1111";
       an(Xi)    <= '0';
    end process;
end Version2;
```

TestBench til Muxdisp.

```vhdl
-- Testbench automatically generated online
-- at http://vhdl.lapinoo.net
-- Generation date : 18.9.2019 11:09:30 GMT

library ieee;
use ieee.std_logic_1164.all;

entity tb_MuxDispDriver is
end tb_MuxDispDriver;

architecture tb of tb_MuxDispDriver is

    component MuxDispDriver
        port (Clk_disp : in std_logic;
              HexCifre : in std_logic_vector (15 downto 0);
              dpoints  : in std_logic_vector (3 downto 0);
              blanks   : in std_logic_vector (3 downto 0);
              Clk_1kHz : out std_logic;
              Clk_1Hz  : out std_logic;
              an       : out std_logic_vector (3 downto 0);
              seg      : out std_logic_vector (6 downto 0);
              dp       : out std_logic);
    end component;

    signal Clk_disp : std_logic;
    signal HexCifre : std_logic_vector (15 downto 0) := X"0123";
    signal dpoints  : std_logic_vector (3 downto 0)  := "0010";
    signal blanks   : std_logic_vector (3 downto 0)  := "0001";
    signal Clk_1kHz : std_logic;
    signal Clk_1Hz  : std_logic;
    signal an       : std_logic_vector (3 downto 0);
    signal seg      : std_logic_vector (6 downto 0);
    signal dp       : std_logic;
    constant TbPeriod : time := 10 ns; -- EDIT
    signal TbClock : std_logic := '0';
    signal TbSimEnded : std_logic := '0';

begin
    dut : MuxDispDriver
    port map (Clk_disp => Clk_disp,
              HexCifre => HexCifre,
              dpoints  => dpoints,
              blanks   => blanks,
              Clk_1kHz => Clk_1kHz,
              Clk_1Hz  => Clk_1Hz,
              an       => an,
              seg      => seg,
              dp       => dp);

    -- Clock generation
    TbClock <= not TbClock after TbPeriod/2
               when TbSimEnded /= '1' else '0';

    -- EDIT: Check that Clk_disp is really your
    Clk_disp <= TbClock;

    stimuli : process
    begin
        wait;
    end process;
end tb;
```

Simulering af Version2

```
------------------------------------------------ Combinatorial logic
-- Note! HexCifre and dpoints must now be added to the sensivity list
   seg <= "1111111"; -- Default .. blank display Xi
   dp  <= '1';       -- Default .. blank dp Xi
   if Blanks(Xi)='0' then
      seg <= Hex27Segm( Conv_integer( HexCifre( Xi*4+3 downto Xi*4)) );
      dp  <= not dpoints(Xi);
   end if;
   an        <= "1111";
   an(Xi)    <= '0';
```



Version2

```
------------------------------------------------ Combinatorial logic
-- Note! HexCifre and dpoints must now be added to the sensivity list
   seg <= "1111111"; -- Default .. blank display Xi
   dp  <= '1';       -- Default .. blank dp Xi
   if Blanks(Xi)='0' then
       seg <= Hex27Segm( Conv_integer( HexCifre( Xi*4+3 downto Xi*4)) );
       dp  <= not dpoints(Xi);
   end if;
   an       <= "1111";
   an(Xi)   <= '0';
```

Den ”korte” version

```
seg <= "1111111"; -- Default .. blank display Xi
dp  <= '1';       -- Default .. blank dp Xi
if Blanks(Xi)='0' then
    HexCif := HexCifre( Xi*4+3 downto Xi*4);
    Hexi   := Conv_integer( HexCif);
    seg <= Hex27Segm( Hexi );
    dp  <= not dpoints(Xi);
end if;
an       <= "1111";
an(Xi)   <= '0';
```

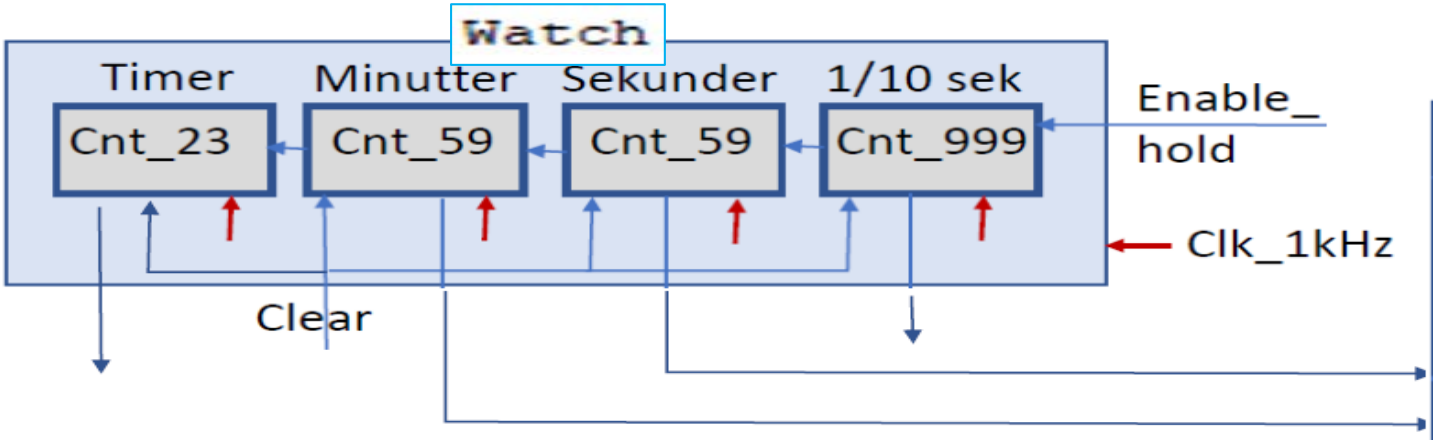
Den ”lange” version

```
type ROM_array is array (0 to 15) of std_logic_vector (6 downto 0);
constant Hex27Segm: ROM_array := (
"1000000","1111001","0100100","0110000", -- 0123
"0011001","0010010","0000010","1111000", -- 4567
"0000000","0010000","0001000","0000011", -- 89Ab
"1000110","0100001","0000110","0001110");-- CdEF
```

Der er brug for et ur (Watch) i forbindelse med stopuret.

Uret kan med fordel opbygges af flere ”standard” tællere … Cnt_9 eller Cnt_999, Cnt_59 og Cnt_23.
De skal kædes sammen med RCO (Ripple Carry Out) som føres til den næste tællers
Enable – Den første tæller styres med et Enable signal.

```
    signal RC : std_logic_vector( 4 downto 0);
begin
 ms1000:Cnt_9 port map(Clk=>Clk, Enable=>Enable,Clear=>Clear, Cif=> open,  RCO=>RC(0));
 ms100: Cnt_9 port map(Clk=>Clk, Enable=>RC(0), Clear=>Clear, Cif=> open,  RCO=>RC(1));
 ms10:  Cnt_9 port map(Clk=>Clk, Enable=>RC(1), Clear=>Clear, Cif=> S10th, RCO=>RC(2));
 Sec:  Cnt_59 port map(Clk=>Clk, Enable=>RC(2), Clear=>Clear, Cif59=> S,   RCO=>RC(3));
 Min:  Cnt_59 port map(Clk=>Clk, Enable=>RC(3), Clear=>Clear, Cif59=> M,   RCO=>RC(4));
--  Hour:  Cnt_23 port map(Clk=>Clk, Enable=>RC(4),  Clear=>Clear, Cif23=> H, RCO=>RCO);
```



Man kan med fordel skrive og teste (simulere)
Cnt_9, Cnt_59 og Cnt_23 hver for sig.
Dette er VHDL koden som laver uret (Cnt_23 mangler)

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

entity Watch is
    Port ( Clk :    in  STD_LOGIC; -- 1kHz clk
           Clear :  in  STD_LOGIC;
           Enable : in  STD_LOGIC;
           H :      out STD_LOGIC_VECTOR ( 7 downto 0); -- Hour
           M :      out STD_LOGIC_VECTOR ( 7 downto 0); -- Min
           S :      out STD_LOGIC_VECTOR ( 7 downto 0); -- Sec
           S10th :  out STD_LOGIC_VECTOR ( 3 downto 0); -- 1/10 sec
           RCO :    out STD_LOGIC);                     -- RCO=1 at 23:59:59:9
end Watch;

architecture Behavioral of Watch is
    component Cnt_9
        port (Clk    : in std_logic;
              Enable : in std_logic;
              Clear  : in std_logic;
              Cif    : out std_logic_vector (3 downto 0);
              RCO    : out std_logic);
    end component;
    component Cnt_59
        port (Clk    : in std_logic;
              Enable : in std_logic;
              Clear  : in std_logic;
              Cif59  : out std_logic_vector (7 downto 0);
              RCO    : out std_logic);
    end component;
    signal RC : std_logic_vector( 4 downto 0);
begin
 ms1000:Cnt_9 port map(Clk=>Clk, Enable=>Enable,Clear=>Clear, Cif=> open,  RCO=>RC(0));
 ms100: Cnt_9 port map(Clk=>Clk, Enable=>RC(0), Clear=>Clear, Cif=> open,  RCO=>RC(1));
 ms10:  Cnt_9 port map(Clk=>Clk, Enable=>RC(1), Clear=>Clear, Cif=> S10th, RCO=>RC(2));
 Sec:  Cnt_59 port map(Clk=>Clk, Enable=>RC(2), Clear=>Clear, Cif59=> S,   RCO=>RC(3));
 Min:  Cnt_59 port map(Clk=>Clk, Enable=>RC(3), Clear=>Clear, Cif59=> M,   RCO=>RC(4));
--  Hour:  Cnt_23 port map(Clk=>Clk, Enable=>RC(4),  Clear=>Clear, Cif23=> H, RCO=>RCO);

----------------- Nedenstående er en middertidig løsning ---------------------------------
    H <= "00100011";
end Behavioral;
```

Løsningsforslag til Cnt_9 (2 stk) - Inspiration til Cnt_59 og Cnt_23

Bemærk forskellen på brug af variable og signal samt Std_logic_vector og integer

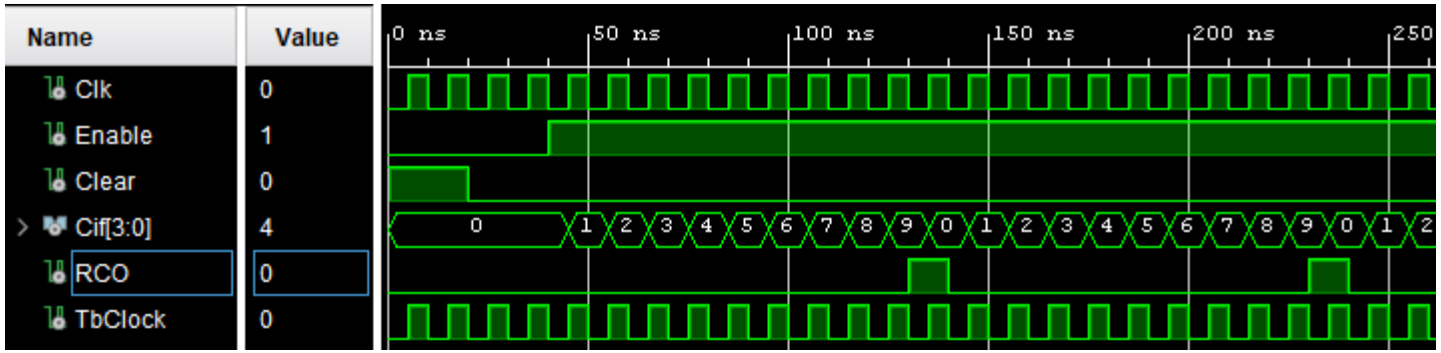
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Cnt_9 is
    Port ( Clk :    in STD_LOGIC;
           Enable : in STD_LOGIC;
           Clear :  in STD_LOGIC;
           Cif :    out STD_LOGIC_VECTOR (3 downto 0);
           RCO :    out STD_LOGIC);
end Cnt_9;
```

```
architecture Version1 of Cnt_9 is
    -- signal Count: STD_LOGIC_VECTOR (3 downto 0) := "0000";
begin
    process( Clk, Clear, Enable)
      variable Count9: integer range 0 to 9 := 1;
    begin
        if Clear='1' then
            Count9 := 0;
        elsif rising_edge( Clk) then
            if Enable='1' then
                if Count9<9 then
                   Count9 := Count9+1;
                else
                   Count9 := 0;
                end if;
            end if;
        end if;
        RCO <= '0';
        if Count9=9 and Enable='1' then
            RCO <= '1';
        end if;
        Cif <= conv_std_logic_vector( Count9, 4);
    end process;
end Version1;
```

```
architecture Version2 of Cnt_9 is
    signal Count9: STD_LOGIC_VECTOR (3 downto 0) := "0000";
begin
    Cif <= Count9;

    process( Clk, Clear, Count9, Enable)
    begin
        if Clear='1' then
            Count9 <= conv_std_logic_vector( 0,4); -- "0000"
        elsif rising_edge( Clk) then
            if Enable='1' then
                if Count9="1001" then
                   Count9 <= "0000";
                else
                   Count9 <= Count9+1;
                end if;
            end if;
        end if;
--        RCO <= '0';
--        if Count9="1001" and Enable='1' then
--            RCO <= '1';
--        end if;
    end process;

    RCO <= '1' when Count9="1001" and Enable='1' else '0';
end Version2;
```

Eksempel på TestBench til Cnt_9

```
library ieee;
use ieee.std_logic_1164.all;

entity tb_Cnt_9 is
end tb_Cnt_9;

architecture tb of tb_Cnt_9 is
    component Cnt_9
        port (Clk    : in std_logic;
              Enable : in std_logic;
              Clear  : in std_logic;
              Cif    : out std_logic_vector (3 downto 0);
              RCO    : out std_logic);
    end component;

    signal Clk    : std_logic;
    signal Enable : std_logic;
    signal Clear  : std_logic;
    signal Cif    : std_logic_vector (3 downto 0);
    signal RCO    : std_logic;

    constant TbPeriod : time := 10 ns; -- EDIT Put right p
    signal TbClock :    std_logic := '0';
    signal TbSimEnded : std_logic := '0';
begin
    dut : Cnt_9
    port map (Clk    => Clk,
              Enable => Enable,
              Clear  => Clear,
              Cif    => Cif,
              RCO    => RCO);

    -- Clock generation
    TbClock <= not TbClock after TbPeriod/2 when TbSimEnded /= '1' else '0';
    -- EDIT: Check that Clk is really your main clock signal
    Clk <= TbClock;

    stimuli : process
    begin
        Enable <= '0';
        Clear <= '1';
        wait for 2* TbPeriod;
        Clear <= '0';
        wait for 2* TbPeriod;
        Enable <= '1';
        wait for 120 * TbPeriod;
        Enable <= '1';
        wait for 4* TbPeriod;
        -- Stop the clock and hence terminate the simulation
        TbSimEnded <= '1';
        wait;
    end process;
end tb;
```



Start-hjælp til Cnt_59 og Cnt_23

http://jjmk.dk/MMMI/Logic_Problems/No04_StopWatch/index.htm

Der kan også findes inspiration i ovenstående link

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

entity Cnt_59 is
    Port ( Clk :    in STD_LOGIC;
           Enable : in STD_LOGIC;
           Clear :  in STD_LOGIC;
           Cif59 :  out STD_LOGIC_VECTOR (7 downto 0);
           RCO :    out STD_LOGIC);
end Cnt_59;

architecture Version of Cnt_59 is
    signal Count9: STD_LOGIC_VECTOR (3 downto 0) := "0000";
    signal Count5: STD_LOGIC_VECTOR (3 downto 0) := "0101";
begin
    Cif59 <= Count5 & Count9;

    -- Bemærk dette er ikke løsningen --- gør den selv færdig
    process( Clk, Clear, Count9, Enable)
    begin
        if Clear='1' then
            Count9 <= conv_std_logic_vector( 0,4); -- "0000"
            Count5 <= "0000";
        elsif rising_edge( Clk) then
            if Enable='1' then
                if Count9="1001" then
                   Count9 <= "0000";
                else
                   Count9 <= Count9+1;
                end if;
            end if;
        end if;
        RCO <= '0';
        if Count9="1001" and Enable='1' then
            RCO <= '1';
        end if;
    end process;
end Version;
```

Sources × Design Signals ?

0

- Design Sources (3)
  - Watch_demo_top(Behavioral) (Watch_demo_top.vhd) (2)
    - DISP : MuxDispDriver(Version1) (MuxDispDriver.vhd) (1)
      - U1 : Hex27Segment(Behavioral) (Hex27Segment.vhd)
    - UR : Watch(Behavioral) (Watch.vhd) (5)

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Watch_demo_top is
    Port ( clk : in STD_LOGIC;
           sw : in STD_LOGIC_VECTOR (15 downto 1);
           led : out STD_LOGIC_VECTOR (15 downto 0);
           btnU,btnL,btnC,btnR,btnD : in STD_LOGIC;
           seg : out STD_LOGIC_VECTOR (6 downto 0);
           dp : out STD_LOGIC;
           an : out STD_LOGIC_VECTOR (3 downto 0));
end Watch_demo_top;
```

```vhdl
architecture Behavioral of Watch_demo_top is
    component Watch
        port (Clk    : in std_logic;
              Clear  : in std_logic;
              Enable : in std_logic;
              H      : out std_logic_vector (7 downto 0);
              M      : out std_logic_vector (7 downto 0);
              S      : out std_logic_vector (7 downto 0);
              S10th  : out std_logic_vector (3 downto 0);
              RCO    : out std_logic);
    end component;

    signal H      : std_logic_vector (7 downto 0);
    signal M      : std_logic_vector (7 downto 0);
    signal S      : std_logic_vector (7 downto 0);
    signal S10th  : std_logic_vector (3 downto 0);
    signal RCO    : std_logic;
```

```vhdl
    component MuxDispDriver
        port (Clk_disp : in std_logic;
              HexCifre : in std_logic_vector (15 downto 0);
              dpoints  : in std_logic_vector (3 downto 0);
              blanks   : in std_logic_vector (3 downto 0);
              Clk_1kHz : out std_logic;
              Clk_1Hz  : out std_logic_vector (0 downto 0);
              an       : out std_logic_vector (3 downto 0);
              seg      : out std_logic_vector (6 downto 0);
              dp       : out std_logic);
    end component;

    signal HexCifre : std_logic_vector (15 downto 0);
    signal dpoints  : std_logic_vector (3 downto 0);
    signal blanks   : std_logic_vector (3 downto 0);
    signal Clk_1kHz : std_logic;
    signal Clk_1Hz  : std_logic_vector (0 downto 0);

begin
    DISP : MuxDispDriver
```

”Hjemmelavet” VHDL – Toplevel til test af UR

brug https://vhdl.lapinoo.net/testbench/ til at lave komponenter

```vhdl
entity Watch_demo_top is
    Port ( clk : in STD_LOGIC;
           sw : in STD_LOGIC_VECTOR (15 downto 1);
           led : out STD_LOGIC_VECTOR (15 downto 0);
           btnU,btnL,btnC,btnR,btnD : in STD_LOGIC;
           seg : out STD_LOGIC_VECTOR (6 downto 0);
           dp : out STD_LOGIC;
           an : out STD_LOGIC_VECTOR (3 downto 0));
end Watch_demo_top;
```

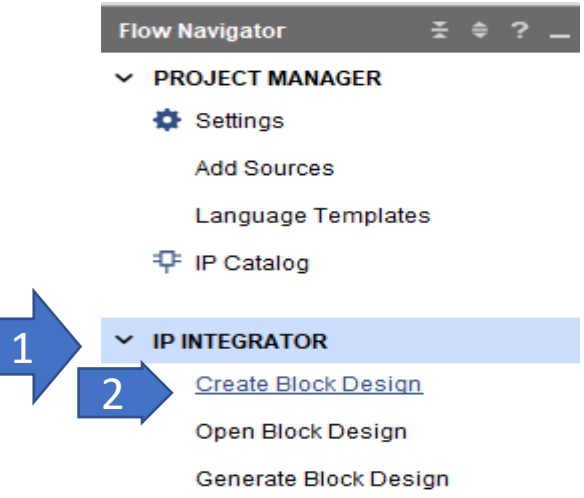
```vhdl
architecture Behavioral of Watch_demo_top is
    component Watch
        port (Clk    : in std_logic;
              Clear  : in std_logic;
              Enable : in std_logic;
              H      : out std_logic_vector (7 downto 0);
              M      : out std_logic_vector (7 downto 0);
              S      : out std_logic_vector (7 downto 0);
              S10th  : out std_logic_vector (3 downto 0);
              RCO    : out std_logic);
    end component;

    signal H      : std_logic_vector (7 downto 0);
    signal M      : std_logic_vector (7 downto 0);
    signal S      : std_logic_vector (7 downto 0);
    signal S10th  : std_logic_vector (3 downto 0);
    signal RCO    : std_logic;

    component MuxDispDriver
        port (Clk_disp : in std_logic;
              HexCifre : in std_logic_vector (15 downto 0);
              dpoints  : in std_logic_vector (3 downto 0);
              blanks   : in std_logic_vector (3 downto 0);
              Clk_1kHz : out std_logic;
              Clk_1Hz  : out std_logic_vector (0 downto 0);
              an       : out std_logic_vector (3 downto 0);
              seg      : out std_logic_vector (6 downto 0);
              dp       : out std_logic);
    end component;

    signal HexCifre : std_logic_vector (15 downto 0);
    signal dpoints  : std_logic_vector (3 downto 0);
    signal blanks   : std_logic_vector (3 downto 0);
    signal Clk_1kHz : std_logic;
    signal Clk_1Hz  : std_logic_vector (0 downto 0);
```

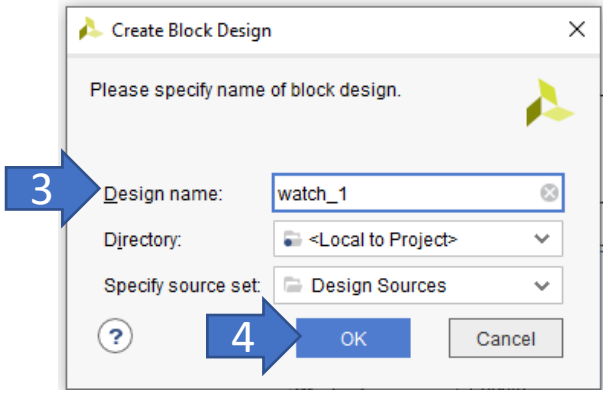
Se senere

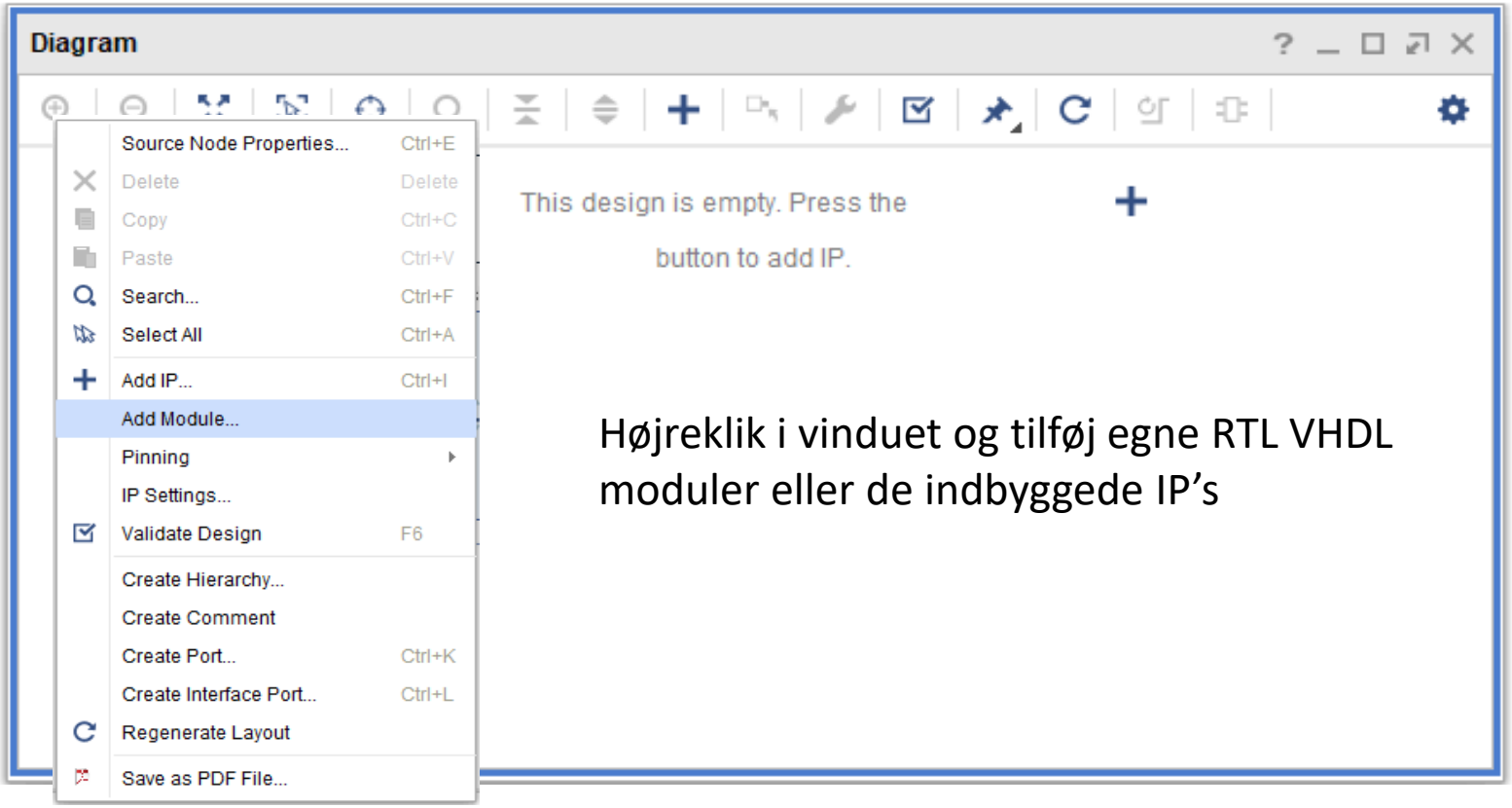
- **Watch_demo_top**(Behavioral) (Watch_demo_top.vhd) (2)
  - DISP : MuxDispDriver(Version1) (MuxDispDriver.vhd) (1)
  - UR : Watch(Behavioral) (Watch.vhd) (5)

```vhdl
begin
    DISP : MuxDispDriver
    port map (Clk_disp => clk,
              HexCifre => HexCifre,
              dpoints  => "0000",
              blanks   => "0000",
              Clk_1kHz => Clk_1kHz,
              Clk_1Hz  => open,
              an       => an,
              seg      => seg,
              dp       => dp);

    HexCifre          <= M & S;
    led(3 downto 0)   <= S10th;

    UR : Watch
    port map (Clk    => Clk_1kHz,
              Clear  => btnD,
              Enable => sw(1),
              H      => H,
              M      => M,
              S      => S,
              S10th  => S10th,
              RCO    => RCO);
end Behavioral;
```

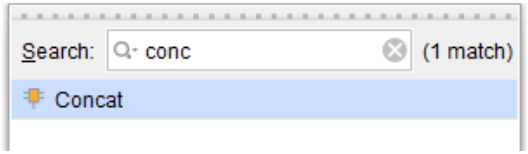
”Vivado” Toplevel med Block Design til test af UR

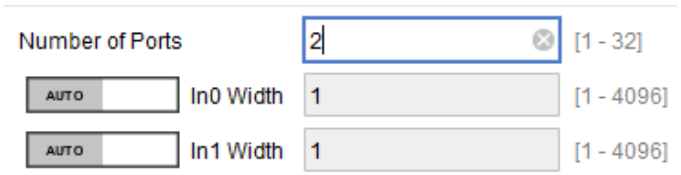




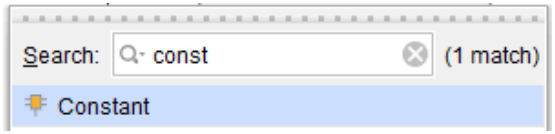


Højreklik i vinduet og tilføj egne RTL VHDL moduler eller de indbyggede IP’s

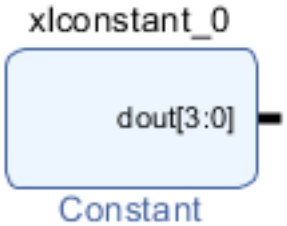

Search: conc
(1 match)
Concat

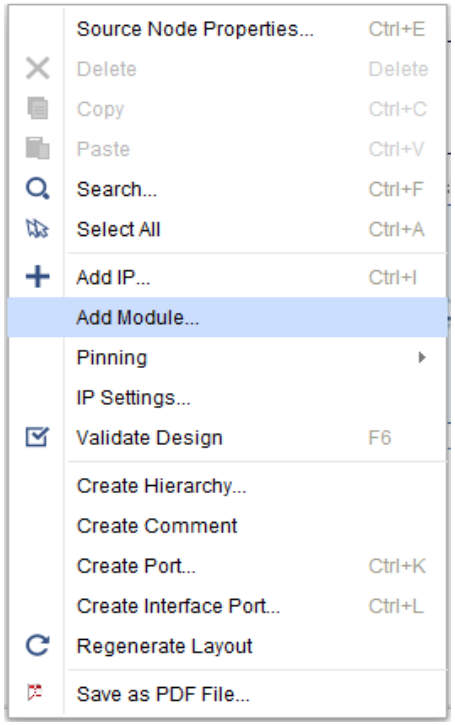

Number of Ports
2
[1 - 32]
AUTO
In0 Width
1
[1 - 4096]
AUTO
In1 Width
1
[1 - 4096]

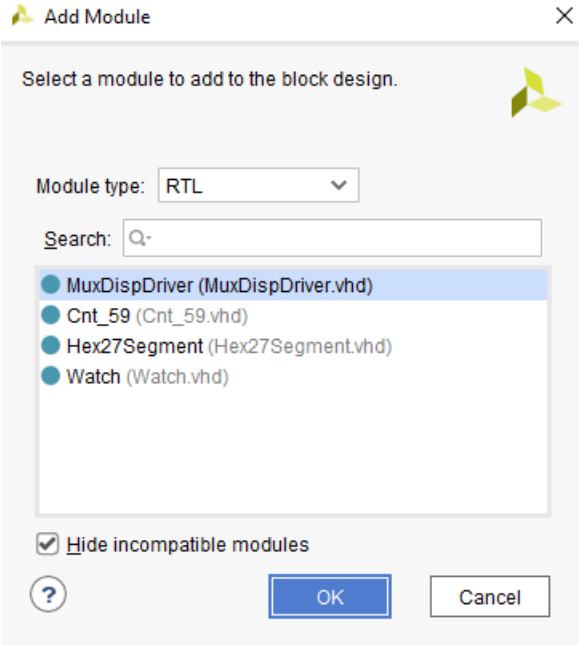

xlconcat_1
In0[0:0]
In1[0:0]
dout[1:0]
Concat

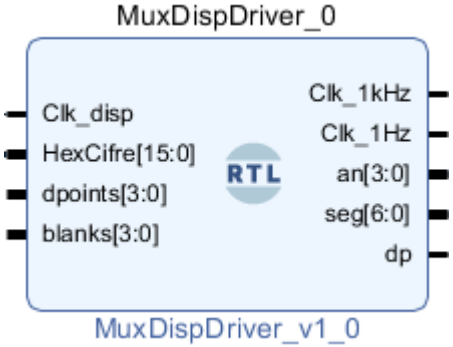

Search: const
(1 match)
Constant

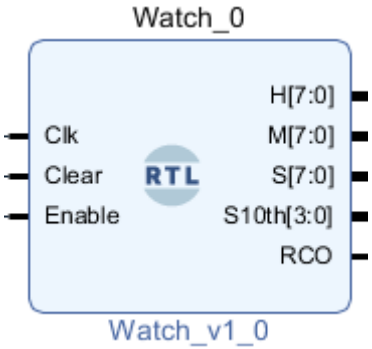

Const Width
4
[1 - 4096]
Const Val
0


xlconstant_0
dout[3:0]
Constant


Source Node Properties... Ctrl+E
Delete Delete
Copy Ctrl+C
Paste Ctrl+V
Search... Ctrl+F
Select All Ctrl+A
Add IP... Ctrl+I
Add Module...
Pinning
IP Settings...
Validate Design F6
Create Hierarchy...
Create Comment
Create Port... Ctrl+K
Create Interface Port... Ctrl+L
Regenerate Layout
Save as PDF File...

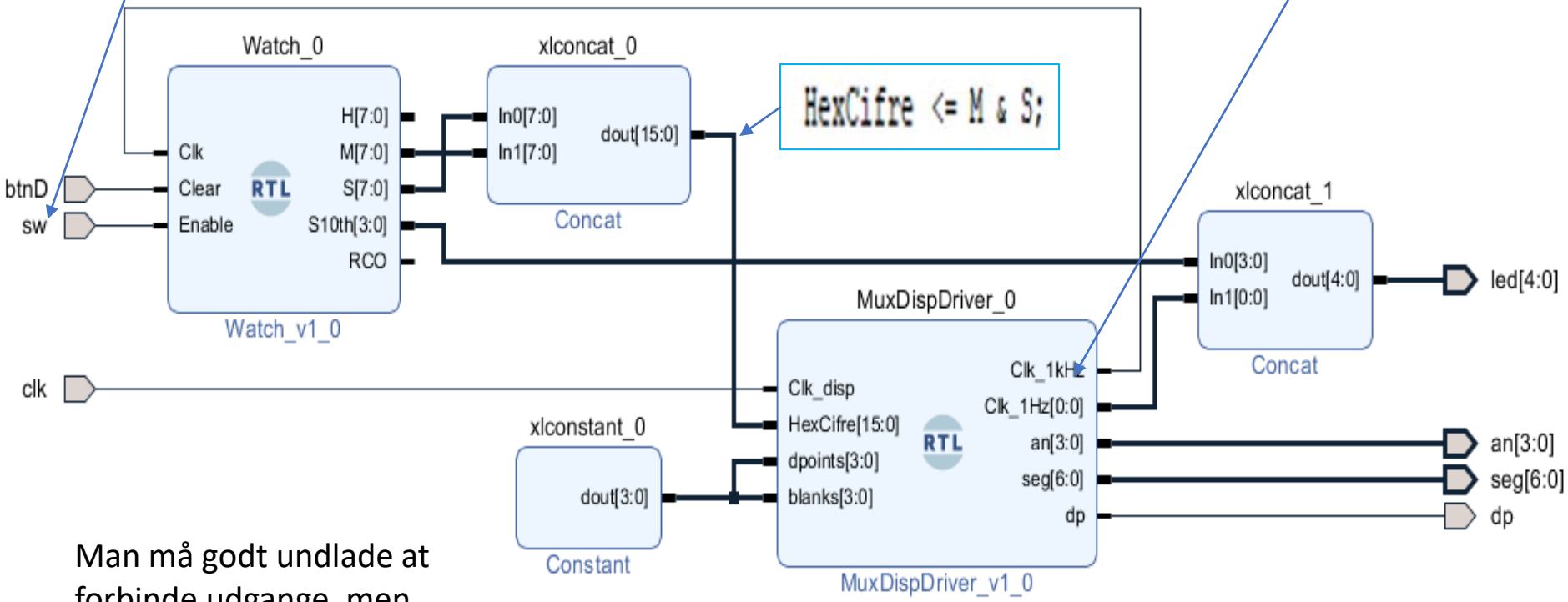

Add Module
Select a module to add to the block design.
Module type: RTL
Search:
MuxDispDriver (MuxDispDriver.vhd)
Cnt_59 (Cnt_59.vhd)
Hex27Segment (Hex27Segment.vhd)
Watch (Watch.vhd)
Hide incompatible modules
OK
Cancel


MuxDispDriver_0
Clk_disp
HexCifre[15:0]
dpoints[3:0]
blanks[3:0]
RTL
Clk_1kHz
Clk_1Hz
an[3:0]
seg[6:0]
dp
MuxDispDriver_v1_0

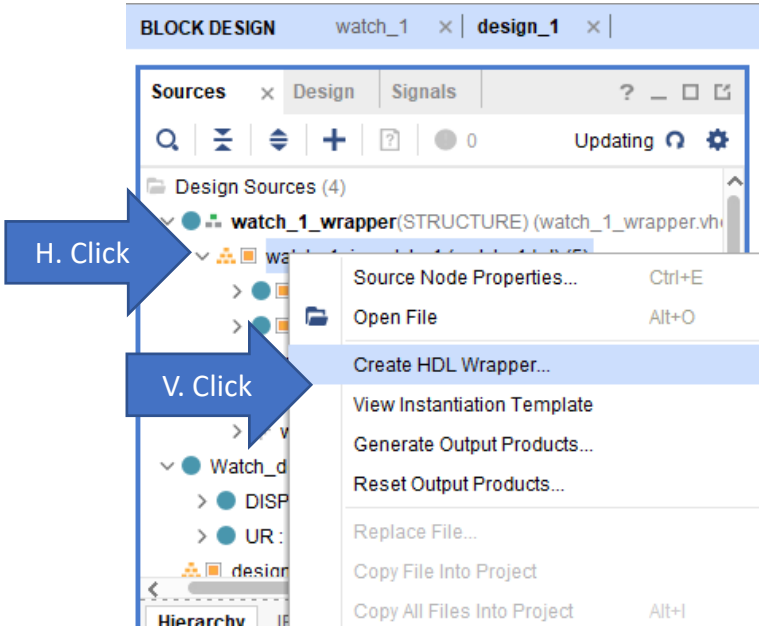

Watch_0
Clk
Clear
Enable
RTL
H[7:0]
M[7:0]
S[7:0]
S10th[3:0]
RCO
Watch_v1_0

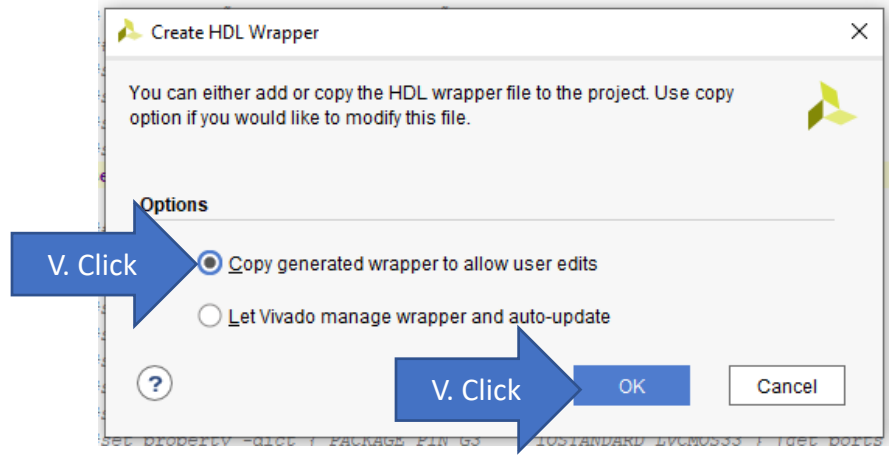
Senere ➡ Bemærk Clk_1Hz er blevet omdef. Til Clk_1kHz: Std_logic_vector (0 downto 0)
Ellers kan man ikke bruge Concat - komponenten

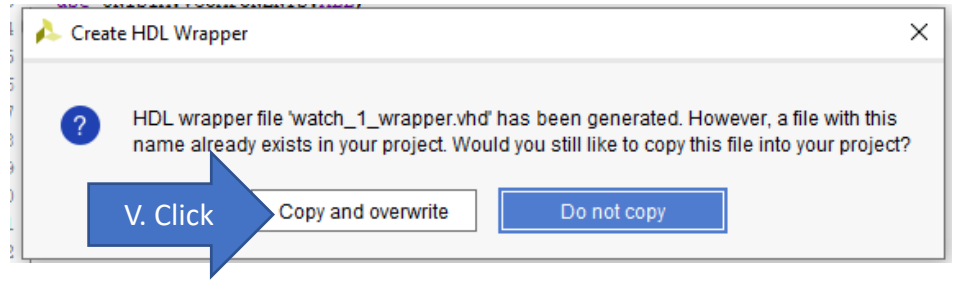
Bemærk at sw(0) er blevet omdøbt til sw i XDC filen

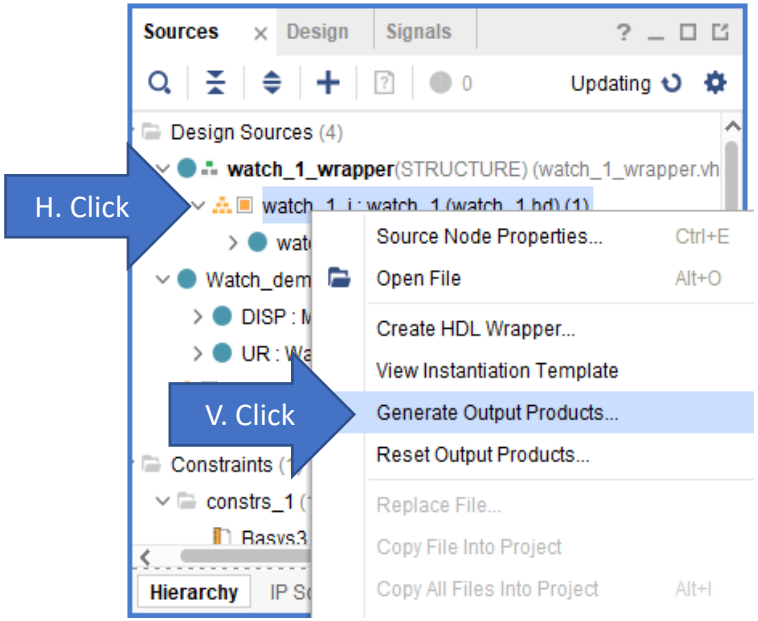


Man må godt undlade at forbinde udgange, men indgange skal forbindes
Brug evt en Constant









Når man har lavet sit Block Design skal der laves en VHDL_wrapper
Gentag eventuelt hver gang der er lavet rettelser (ikke nødvendigt altid)

Hvis der er problemer så prøv Generate Output Products