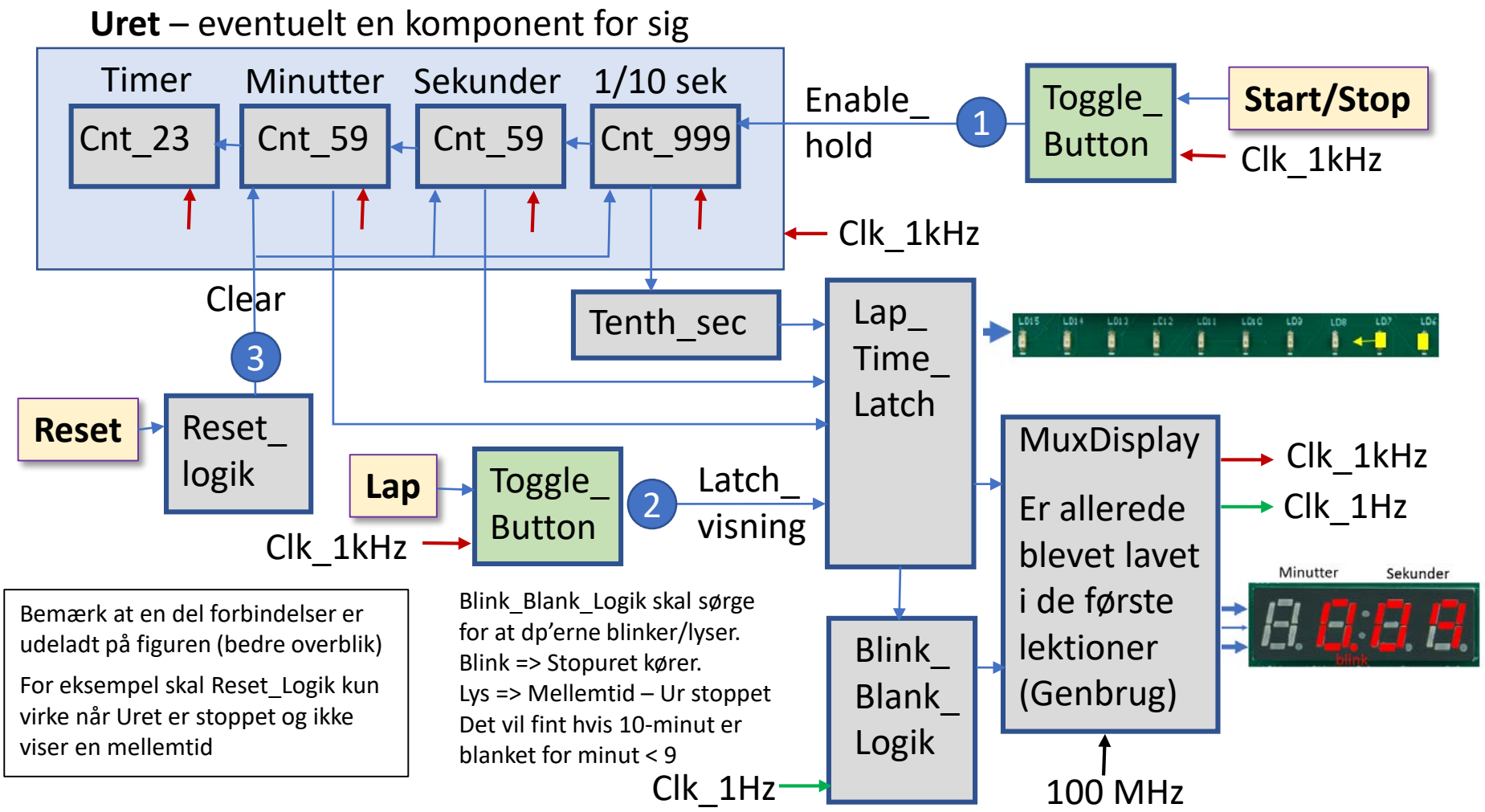
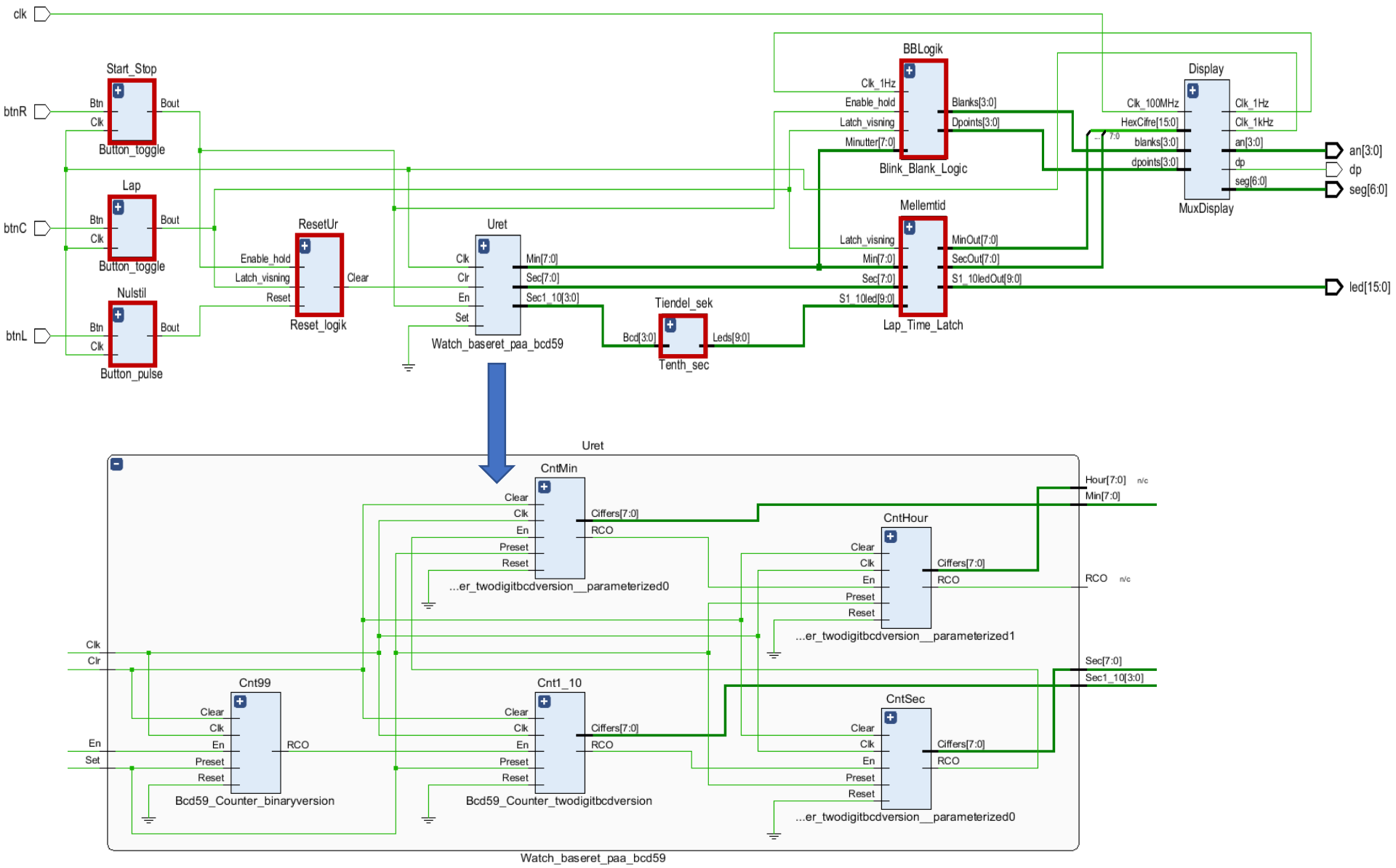
Blokdiagram – Opbygningen af Stopur V1

1 2 3 Er ”veldefinerede” styresignaler uden prel – vigtigt for (1) og (2)

MuxDisplay bruger en 100 MHz clk og leverer Clk_1kHz som bruges i resten af kredsløbet
Dette er ikke den helt optimale løsning, men kan godt accepteres i dette tilfælde.



Download og udpak projektet: Stopwatch_ver1.zip – så er du i gang



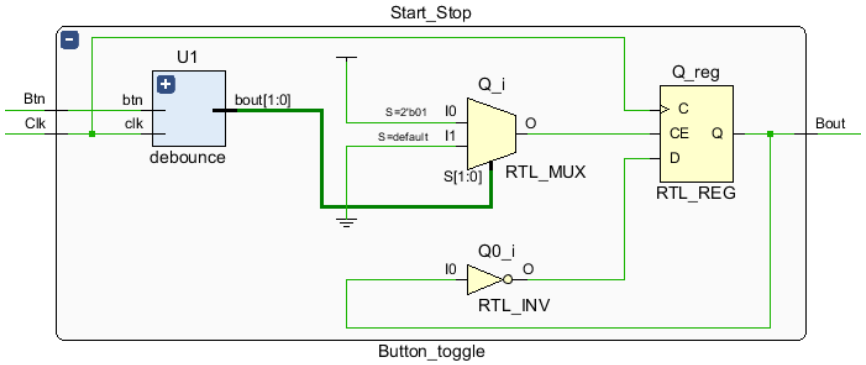
Opgaver

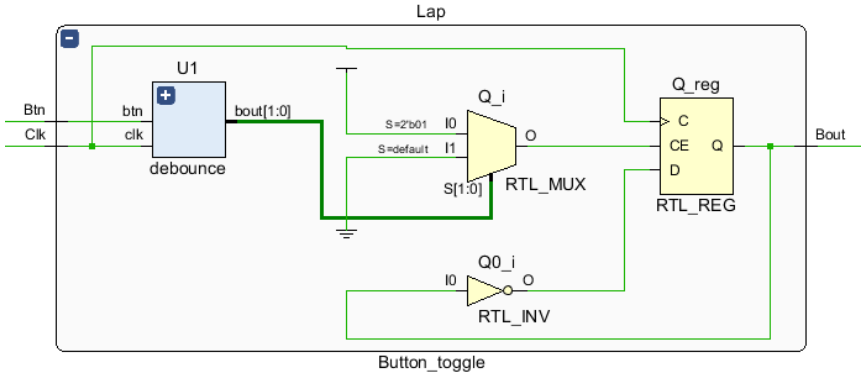
Download og udpak projektet: Stopwatch_ver1.zip – så er du igang

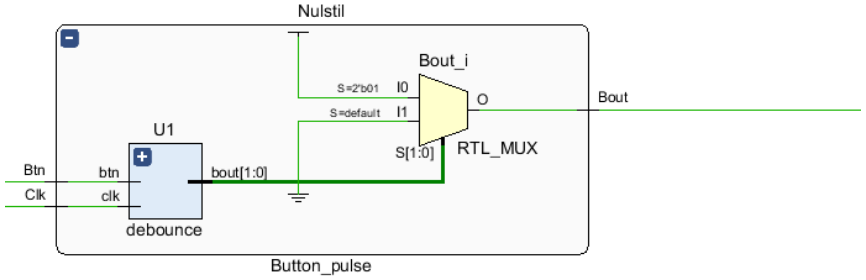
Din opgave er at implementere komponenterne markeret med ➡ og ▭

Bemærk at der allerede findes en ENTITY + en del af ARCHITECTURE + TestBench

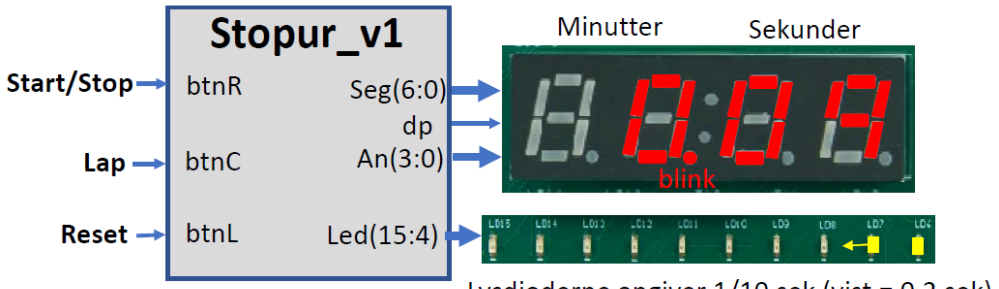
- Design Sources (3)
  - **TopLevel_StopWatch_v1** (Behavioral) (TopLevel_StopWatch_v1.vhd) (10)
    - ➡ Start_Stop : Button (Toggle) (Button.vhd) (1)
      - U1 : debounce (behavioral) (debounce.vhd)
    - ➡ Lap : Button (Toggle) (Button.vhd) (1)
      - U1 : debounce (behavioral) (debounce.vhd)
    - ➡ Nulstil : Button (Pulse) (Button.vhd) (1)
      - U1 : debounce (behavioral) (debounce.vhd)
    - ➡ ResetUr : Reset_logik (Behavioral) (Reset_logik.vhd)
    - Uret : Watch (Baseret_paa_Bcd59) (Watch.vhd) (5)
      - Cnt99 : Bcd59_Counter (BinaryVersion) (Bcd59_Counter.vhd)
      - Cnt1_10 : Bcd59_Counter (TwoDigitBcdVersion) (Bcd59_Counter.vhd)
      - CntSec : Bcd59_Counter (TwoDigitBcdVersion) (Bcd59_Counter.vhd)
      - CntMin : Bcd59_Counter (TwoDigitBcdVersion) (Bcd59_Counter.vhd)
      - CntHour : Bcd59_Counter (TwoDigitBcdVersion) (Bcd59_Counter.vhd)
    - ➡ Tiendel_sek : Tenth_sec (Behavioral) (Tenth_sec.vhd)
    - ➡ Mellemtid : Lap_Time_Latch (Behavioral) (Lap_Time_Latch.vhd)
    - ➡ BBLogik : Blink_Blank_Logic (Behavioral) (Blink_Blank_Logic.vhd)
    - Display : MuxDisplay (Version2) (MuxDisplay.vhd)







```
ENTITY TopLevel_StopWatch_v1 IS
    PORT ( clk: in  STD_LOGIC;
         --sw : in  STD_LOGIC_VECTOR (15 downto 0);
           led: out STD_LOGIC_VECTOR (15 downto 0);
           btnL, btnC, btnR: in  STD_LOGIC;
           seg: out STD_LOGIC_VECTOR (6 downto 0);
           dp:  out STD_LOGIC;
           an:  out STD_LOGIC_VECTOR (3 downto 0));
end TopLevel_StopWatch_v1;

ARCHITECTURE Behavioral of TopLevel_StopWatch_v1 is
-------------------------------------------------------------------------
    COMPONENT Button
        Generic( del: integer := 10);
        Port ( Clk :  in STD_LOGIC;
               Btn :  in STD_LOGIC;
               Bout : out STD_LOGIC);
    end COMPONENT;
    signal Enable_hold:   STD_LOGIC;
    signal Latch_visning: STD_LOGIC;
    signal NulstilUr:     STD_LOGIC;
-------------------------------------------------------------------------------
    FOR Nulstil:  Button use entity work.Button( Pulse);  -- BEMÆRK VALG AF .. Pulse
-------------------------------------------------------------------------------
    COMPONENT Watch
        Port ( Clk :     in STD_LOGIC;  -- Clock = 1kHz
               En :      in STD_LOGIC;  -- En='1' => uret går
               Clr :     in STD_LOGIC;  -- Nulstil uret Async
               Set :     in STD_LOGIC;  -- Set uret til 23:59:59:999 Async
               Hour :    out STD_LOGIC_VECTOR (7 downto 0);
               Min :     out STD_LOGIC_VECTOR (7 downto 0);
               Sec :     out STD_LOGIC_VECTOR (7 downto 0);
               Secl_10 : out STD_LOGIC_VECTOR (3 downto 0); --1/10 sek
               RCO:      out STD_LOGIC);  -- Tæl evt dage
    end COMPONENT;
    signal Secl_10: STD_LOGIC_VECTOR (3 downto 0);

    --Sådan man vælge en bestemt ARCHITECTURE til en given komponent - Uret
    FOR Uret:  Watch use entity work.Watch( Baseret_paa_Bcd59);
    --Bemærk at den ACHTECTURE som ligger sidst er default
 -------------------------------------------------------------------------------
    COMPONENT Blink_Blank_Logic
        Port ( Clk_1Hz :       in  STD_LOGIC;
               Latch_visning : in  STD_LOGIC;
               Enable_hold :   in  STD_LOGIC;
               Minutter :      in  STD_LOGIC_VECTOR (7 downto 0);
               Dpoints :       out STD_LOGIC_VECTOR (3 downto 0);
               Blanks :        out STD_LOGIC_VECTOR (3 downto 0));
    end COMPONENT;
```

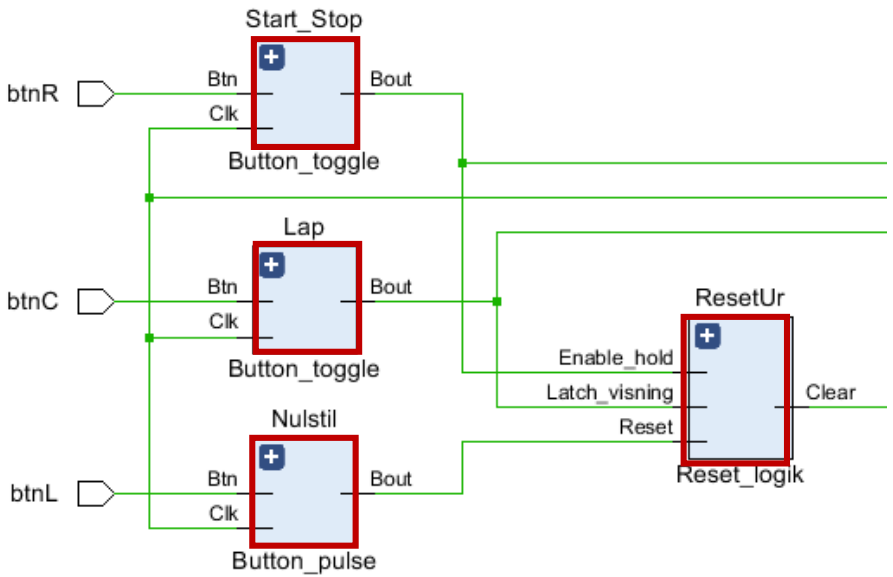


Lysdioderne angiver 1/10 sek (vist = 0.2 sek)

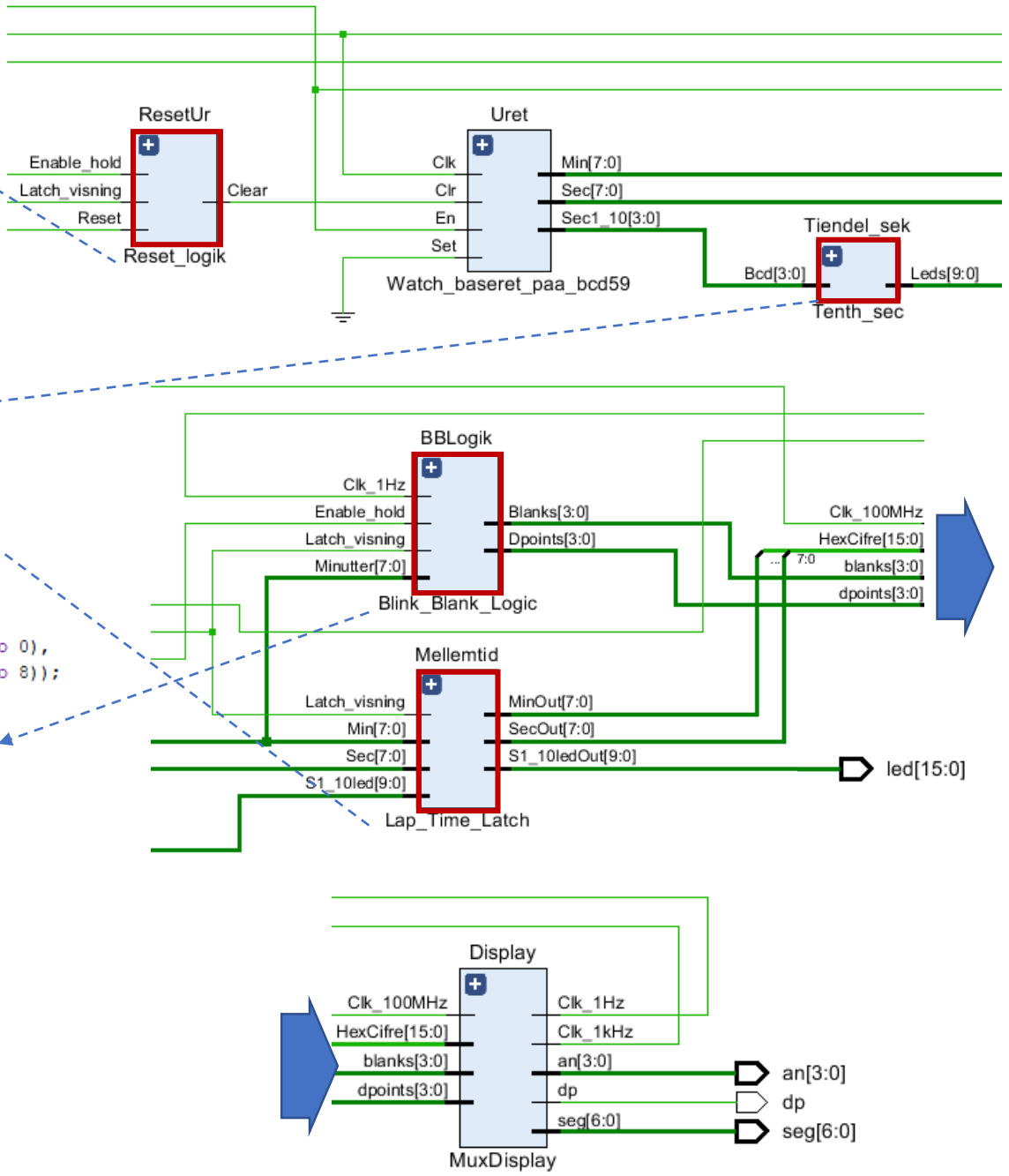
Default Architecture er Toggle

Default Architecture er Baseret_paa_Bcd9

```
57    COMPONENT Tenth_sec
58    Port ( Bcd :  in  STD_LOGIC_VECTOR (3 downto 0);
59           Leds : out STD_LOGIC_VECTOR (9 downto 0));
60    end COMPONENT;
61  ----------------------------------------------------------------
62    COMPONENT Lap_Time_Latch
63        Port ( Sl_10led : in STD_LOGIC_VECTOR (9 downto 0);
64               Sec :      in STD_LOGIC_VECTOR (7 downto 0);
65               Min :      in STD_LOGIC_VECTOR (7 downto 0);
66               Latch_visning : in STD_LOGIC;
67               Sl_10ledOut : out STD_LOGIC_VECTOR (9 downto 0);
68               SecOut :      out STD_LOGIC_VECTOR (7 downto 0);
69               MinOut :      out STD_LOGIC_VECTOR (7 downto 0));
70    end COMPONENT;
71    signal Sec:      STD_LOGIC_VECTOR (7 downto 0);
72    signal Min:      STD_LOGIC_VECTOR (7 downto 0);
73    signal Sl_10led: STD_LOGIC_VECTOR (9 downto 0);
74  ----------------------------------------------------------------
75    COMPONENT Reset_logik
76        Port ( Reset :         in STD_LOGIC;
77               Enable_hold :   in STD_LOGIC;
78               Latch_visning : in STD_LOGIC;
79               Clear :         out STD_LOGIC);
80    end COMPONENT;
81    signal Clear: STD_LOGIC;
82  ----------------------------------------------------------------
83    COMPONENT MuxDisplay
84      Port (Clk_100MHz: in  STD_LOGIC;
85            HexCifre: in  STD_LOGIC_VECTOR (15 downto 0);
86            dpoints:  in  STD_LOGIC_VECTOR (3 downto 0);
87            blanks:   in  STD_LOGIC_VECTOR (3 downto 0);
88            Clk_1kHz: out STD_LOGIC;
89            Clk_1Hz:  out STD_LOGIC;
90            an:       out STD_LOGIC_VECTOR (3 downto 0);
91            seg:      out STD_LOGIC_VECTOR (6 downto 0);
92            dp:       out STD_LOGIC);
93    end COMPONENT;
94    Signal Clk_1kHz:     STD_LOGIC;
95    Signal Clk_1Hz:      STD_LOGIC;
96    Signal HexCifre:     STD_LOGIC_VECTOR (15 downto 0);
97    Signal Dpoints:      STD_LOGIC_VECTOR (3 downto 0);
98    Signal Blanks:       STD_LOGIC_VECTOR (3 downto 0);
99  ----------------------------------------------------------------
100 BEGIN
101     Start_Stop: Button PORT MAP(
102           Clk => Clk_1kHz , Btn => btnR, Bout => Enable_hold);
103
104     Lap: Button PORT MAP(
105           Clk => Clk_1kHz , Btn => btnC, Bout => Latch_visning);
106
107     Nulstil: Button PORT MAP(
108           Clk => Clk_1kHz , Btn => btnL, Bout => NulstilUr);
```



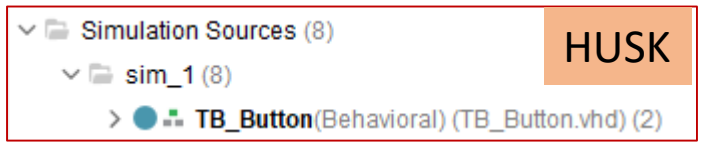
```
109
110     ResetUr: Reset_logik PORT MAP (
111            Reset        => NulstilUr,
112            Enable_hold  => Enable_hold,
113            Latch_visning => Latch_visning,
114            Clear        => Clear);
115
116    Uret: Watch PORT MAP (
117              Clk     => Clk_1kHz,
118              En      => Enable_hold,
119              Clr     => Clear,
120              Set     => '0',
121              Hour    => open,
122              Min     => Min,
123              Sec     => Sec,
124              Sec1_10 => Sec1_10,
125              RCO     => open);
126
127    Tiendel_sek:  Tenth_sec PORT MAP (
128              Bcd => Sec1_10, Leds => S1_10led);
129
130    Mellemtid: Lap_Time_Latch PORT MAP (
131              S1_10led      => S1_10led,
132              Sec           => Sec,
133              Min           => Min,
134              Latch_visning => Latch_visning,
135              S1_10ledOut   => led(15 downto 6),
136              SecOut        => HexCifre( 7 downto 0),
137              MinOut        => HexCifre(15 downto 8));
138
139    BBLogik: Blink_Blank_Logic PORT MAP (
140              Clk_1Hz       => Clk_1Hz,
141              Latch_visning => Latch_visning ,
142              Enable_hold   => Enable_hold ,
143              Minutter      => Min,
144              Dpoints       => Dpoints,
145              Blanks        => Blanks);
146
147    Display: MuxDisplay PORT MAP (
148              Clk_100MHz => Clk,
149              HexCifre   => HexCifre,
150              dpoints    => Dpoints,
151              blanks     => Blanks,
152              Clk_1kHz   => Clk_1kHz,
153              Clk_1Hz    => Clk_1Hz,
154              an         => an,
155              seg        => seg,
156              dp         => dp);
157 END Behavioral;
```

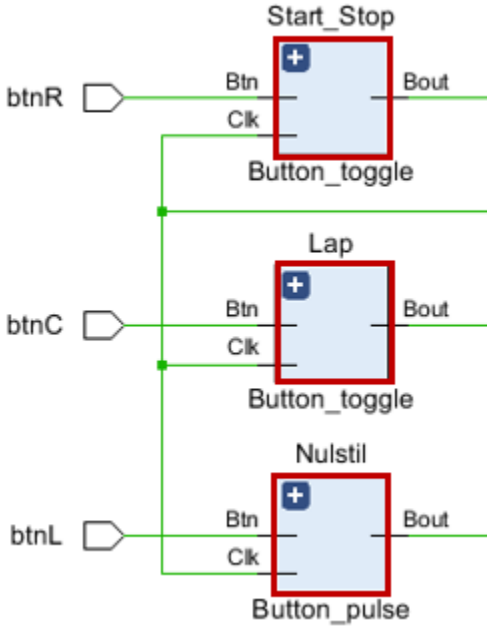


Button.vhd

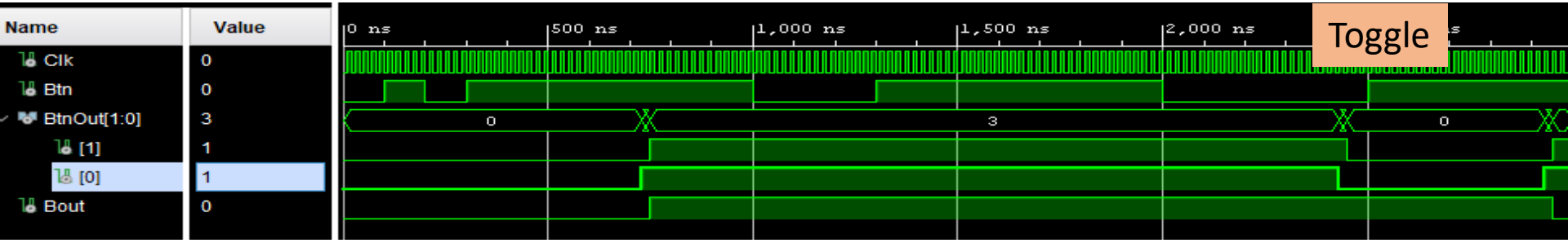
```
entity TB_Button is
end TB_Button;
--Husk at sætte denne TB_xxxxxx til Toplevel
architecture Behavioral of TB_Button is
    COMPONENT Button is
        Generic( del:    integer := 1000);
        Port ( Clk : in STD_LOGIC;
               Btn : in STD_LOGIC;
               Bout : out STD_LOGIC);
    end COMPONENT;
    signal Clk : STD_LOGIC := '0';
    signal Btn : STD_LOGIC := '0';
    signal Bout :STD_LOGIC;

    FOR UUT: Button use entity work.Button( Toggle);  --Prøv også Toggle og Pulse
begin
    UUT: Button  Generic map( del => 20)
                 Port map ( Clk  => Clk,
                            Btn  => Btn,
                            Bout => Bout);

    clk <= not clk after 10 ns;
    btn <= '1' after  100 ns,  '0' after  200 ns,
           '1' after  300 ns,  '0' after 1000 ns,
           '1' after 1300 ns,  '0' after 2000 ns,
           '1' after 2500 ns,  '0' after 3000 ns;
end Behavioral;
```

HUSK

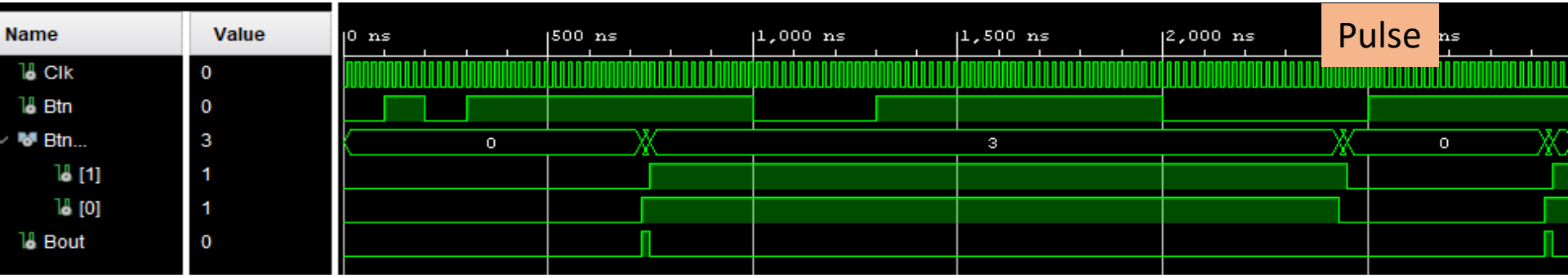




Toggle

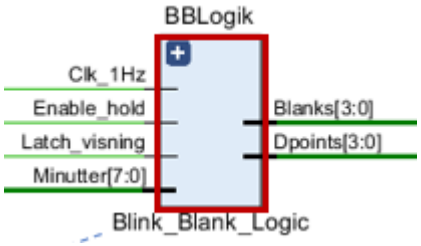


Pulse

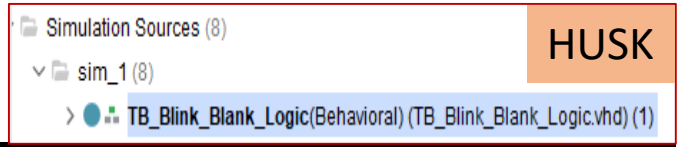


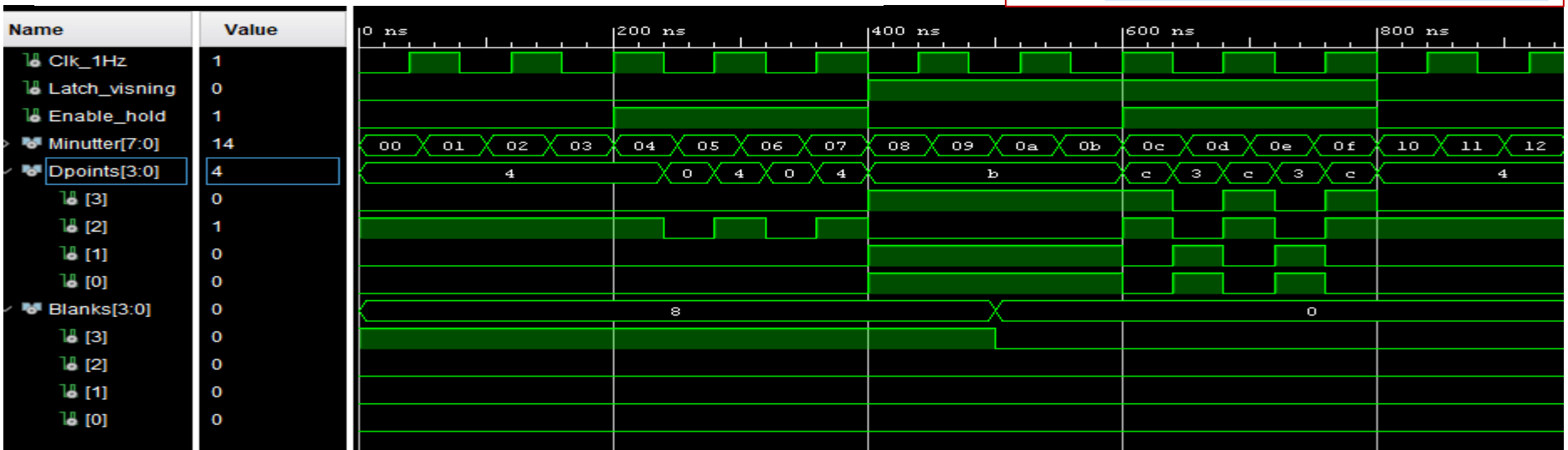
Blink_Blank_Logic.vhd

```
entity TB_Blink_Blank_Logic is
end TB_Blink_Blank_Logic;
architecture Behavioral of TB_Blink_Blank_Logic is
    COMPONENT Blink_Blank_Logic
        Port(  Clk_1Hz :       in STD_LOGIC;
               Latch_visning : in STD_LOGIC;
               Enable_hold :   in STD_LOGIC;
               Minutter :      in STD_LOGIC_VECTOR (7 downto 0);
               Dpoints :       out STD_LOGIC_VECTOR (3 downto 0);
               Blanks :        out STD_LOGIC_VECTOR (3 downto 0));
    end COMPONENT;
    Signal Clk_1Hz :       STD_LOGIC := '0';
    Signal Latch_visning : STD_LOGIC := '0';
    Signal Enable_hold :   STD_LOGIC := '0';
    Signal Minutter :      STD_LOGIC_VECTOR (7 downto 0) := "00000000";
    Signal Dpoints :       STD_LOGIC_VECTOR (3 downto 0);
    Signal Blanks :        STD_LOGIC_VECTOR (3 downto 0);
begin
    UUT: Blink_Blank_Logic Port map(
           Clk_1Hz       => Clk_1Hz,
           Latch_visning => Latch_visning,
           Enable_hold   => Enable_hold,
           Minutter      => Minutter,
           Dpoints       => Dpoints,
           Blanks        => Blanks);

    Clk_1Hz <= not Clk_1Hz after 40 ns; -- Ikke lige 1 Hz :)
    Enable_Hold <= not Enable_Hold after 200 ns;
    Latch_visning <= not Latch_visning after 400 ns;

    Minutter <= Minutter +1  after 50 ns;
end Behavioral;
```

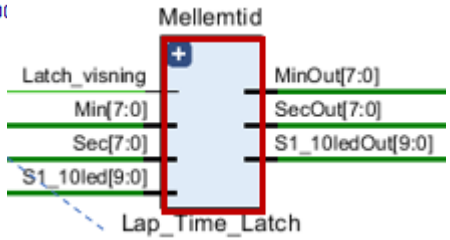


HUSK





```vhdl
entity TB_Mellemtid is
end TB_Mellemtid;
architecture Behavioral of TB_Mellemtid is
    COMPONENT Lap_Time_Latch
        Port ( S1_10led : in STD_LOGIC_VECTOR (9 downto 0);
               Sec :      in STD_LOGIC_VECTOR (7 downto 0);
               Min :      in STD_LOGIC_VECTOR (7 downto 0);
               Latch_visning : in STD_LOGIC;
               S1_10ledOut : out STD_LOGIC_VECTOR (9 downto 0);
               SecOut :      out STD_LOGIC_VECTOR (7 downto 0);
               MinOut :      out STD_LOGIC_VECTOR (7 downto 0));
    end COMPONENT;
    Signal S1_10led :      STD_LOGIC_VECTOR (9 downto 0) := (others=>'0');
    Signal Sec :           STD_LOGIC_VECTOR (7 downto 0) := (others=>'0');
    Signal Min :           STD_LOGIC_VECTOR (7 downto 0) := (others=>'0');
    Signal Latch_visning:  STD_LOGIC := '0';
    Signal S1_10ledOut :   STD_LOGIC_VECTOR (9 downto 0);
    Signal SecOut :        STD_LOGIC_VECTOR (7 downto 0);
    Signal MinOut :        STD_LOGIC_VECTOR (7 downto 0);
--Husk at sætte denne TB_xxxxx til Toplevel
begin
  UUT: Lap_Time_Latch  port map (
          S1_10led => S1_10led,
          Sec      =>  Sec,
          Min      =>  Min,
          Latch_visning => Latch_visning,
          S1_10ledOut   => S1_10ledOut,
          SecOut        => SecOut,
          MinOut       => MinOut);
     Test_Signaler: process
     begin
        S1_10led <= S1_10led(8 downto 0) &'1';
        Sec <= Sec+1;
        Min <= Min-1;
        wait for 30 ns;
        if S1_10led(9)='1' then
            S1_10led <= "000000(
            wait for 10 ns;
        end if;

     end process;

    Latch_signal: process
    begin
       wait for 150 ns;
       Latch_visning <= '1';
       wait for 250 ns;
       Latch_visning <= '0';
    end process;
end Behavioral;
```



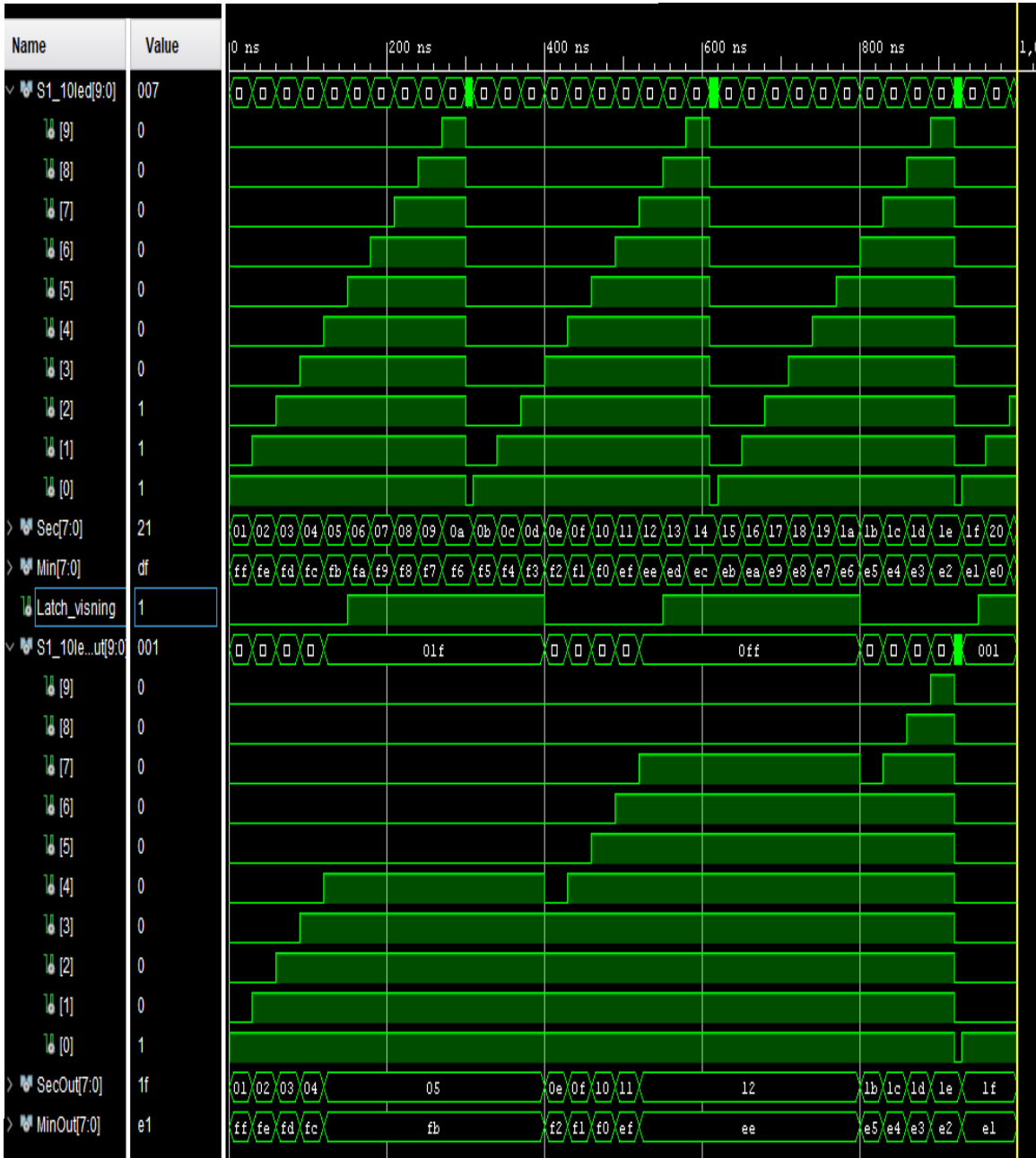
Simulation Sources (8)

sim_1 (8)

TB_Mellemtid(Behavioral) (TB_Mellemtid.vhd) (1)

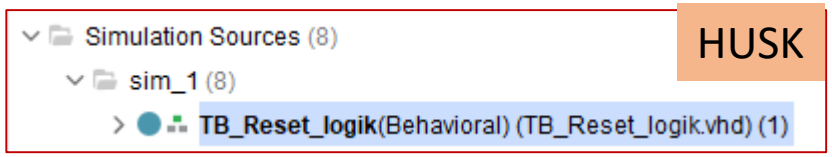
HUSK

Mellemtid.vhd

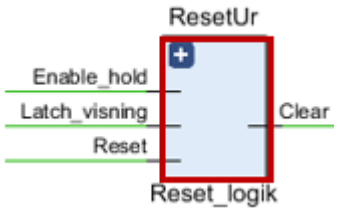


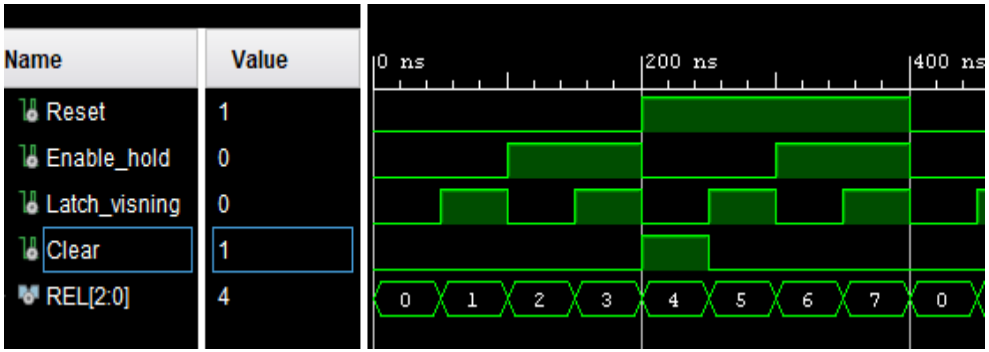
Reset_logik.vhd

HUSK




```
entity TB_Reset_logik is
end TB_Reset_logik;

architecture Behavioral of TB_Reset_logik is
    COMPONENT Reset_logik
        Port ( Reset :          in STD_LOGIC;
               Enable_hold :    in STD_LOGIC;
               Latch_visning :  in STD_LOGIC;
               Clear :          out STD_LOGIC);
    end COMPONENT;
    signal Reset :          STD_LOGIC;
    signal Enable_hold :    STD_LOGIC;
    signal Latch_visning :  STD_LOGIC;
    signal Clear :          STD_LOGIC;

    signal REL: STD_LOGIC_VECTOR( 2 downto 0) := "000";
begin
    UUT: Reset_logik PORT MAP(
            Reset => Reset,
            Enable_hold => Enable_hold,
            Latch_visning => Latch_visning,
            Clear => Clear);

    REL <= REL + 1 after 50 ns; --Lav alle kombinationer

    Reset         <= REL(2);
    Enable_hold   <= REL(1);
    Latch_visning <= REL(0);
end Behavioral;
```





Simulation Sources (8)
sim_1 (8)
TB_Tiendel_sek(Behavioral) (TB_Tiendel_sek.vhd) (1)

HUSK

Tiendel_sek.vhd

```
entity TB_Tiendel_sek is
end TB_Tiendel_sek;
--H u s k   a t   s æ t t e   d e n n e   TB_xxxxxx   t i l   T o p l e v e l
architecture Behavioral of TB_Tiendel_sek is
    COMPONENT Tenth_sec
    Port ( Bcd :  in  STD_LOGIC_VECTOR (3 downto 0);
           Leds : out STD_LOGIC_VECTOR (9 downto 0));
    end COMPONENT;
    Signal Bcd :   STD_LOGIC_VECTOR (3 downto 0) := "0000";
    Signal Leds :  STD_LOGIC_VECTOR (9 downto 0);
begin
    UUT: Tenth_sec Port Map
          ( Bcd => Bcd,
           Leds => Leds);
     process
        variable BCDx:  STD_LOGIC_VECTOR (3 downto 0) := "0000";
    begin
        for Testnr in 0 to 1 loop
            for i in 0 to 9 loop
                if BCDx < "1001" then
                    BCDx := BCDx+1;
                else
                    BCDx := "0000";
                end if;
                Bcd <= Bcdx;
                Wait for 20 ns;
            end loop; -- next i
        end loop; -- next Testnr
        ------------------Forever-------
        loop
           BCDx := BCDx+1;
           Bcd <= Bcdx;
           Wait for 20 ns;
        end loop;
        ---------------------------------
    end process;
end Behavioral;
```

Bemærk brugen af for … loop it TB