# Labprotokoll for Lab4, Digital klokke

|  |  |
| --- | --- |
| Innleveringsfrist: | Se Canvas |
| Innlevering | Laboratorieoppgave 4: Digital klokke |
| Navn: | Endre Røberg Løseth, Simen Fuglestad |
| Innlevert zip-fil | ERL\_SF\_lab4.zip |

## Oppg 1a: Skriv VHDL-kode for del 1a

|  |
| --- |
| **Sett inn bilde av VHDL koden for oppg1a her.** |
|  |

|  |
| --- |
| **Sett inn RTL-bilde:** (Tools, Netlist Viewer, RTL Viewer) |
|  |

|  |
| --- |
| **Sett inn simuleringsfilen her.** |
| constant periode : time := 10 us;  p\_clk : process -- prosess for å lage clk timing  begin clk <= '0';  loop  wait for periode/2;  clk <= not clk;  end loop;  wait;  end process p\_clk;  A<= '0';  B<= '0';  C<= '0';  wait for periode;  A<= '1';  B<= '0';  C<= '0';  wait for periode;  A<= '0';  B<= '1';  C<= '0';  wait for periode;  A<= '1';  B<= '1';  C<= '0';  wait for periode;  A<= '0';  B<= '0';  C<= '1';  wait for periode;  A<= '1';  B<= '0';  C<= '1';  wait for periode;  A<= '0';  B<= '1';  C<= '1';  wait for periode;  A<= '1';  B<= '1';  C<= '1';  wait for periode; |

|  |
| --- |
| **Sett inn bilde av simulering her. Analyser simuleringen.** |
|  |

## Oppg1b: variable

|  |
| --- |
| **Sett inn bilde av VHDL-koden her** |
|  |

|  |
| --- |
| **Sett inn RTL-bilde:** (Tools, Netlist Viewer, RTL Viewer) |
|  |

|  |
| --- |
| **Sett inn bilde av simulering her. Analyser simuleringen.** |
|  |

## Oppg1c

|  |
| --- |
| **Sett inn bilde av VHDL koden for oppg1c her.** |
|  |

## Oppgåve 2, binær-til-BCD-enkoder

|  |
| --- |
| **Sett inn bilde av VHDL-koden her** |
|  |

|  |
| --- |
| **Sett inn RTL-bilde:** (Tools, Netlist Viewer, RTL Viewer) |
|  |

|  |
| --- |
| **Sett inn bilde av simulering her. Analyser simuleringen.** |
| Her ser vi at når bin\_in er 1100000 som er 96 så vil bcd\_low være 6 og bcd\_high være 9. Da blir bcd\_out 10010110, dette stemmer med ønsket funksjon. |

## Oppgave 3a: blinkende lysdiode.

|  |
| --- |
| **Sett inn bilde av VHDL koden for oppgave 3a her.** |
|  |

|  |
| --- |
| **Sett inn signal-tap bilde som viser at blinkesignalet «hallo» skifter verdi etter 1 sekund.**  **Vis også verdien på Teller i «enable\_gen»** |
| A screenshot of a cell phone  Description automatically generated |

## Oppgave 3B teller

|  |
| --- |
| **Sett inn vhdl-filen for oppgave 3B her.** |
| library ieee;  use ieee.std\_logic\_1164.all;  use ieee.numeric\_std.all;  entity L4\_klokke\_ELR\_SF\_3b is  port (  CLOCK\_50 : in std\_logic;  KEY : in std\_logic\_vector(3 downto 0);  SW : in std\_logic\_vector(2 downto 0);  LEDR : out std\_logic\_vector(17 downto 0);  HEX0,  HEX1,  HEX2,  HEX3,  HEX4,  HEX5 : out std\_logic\_vector(6 downto 0)  );  end entity L4\_klokke\_ELR\_SF\_3b;  architecture RTL of L4\_klokke\_ELR\_SF\_3b is  component Enable\_gen is  port (  clock\_50 : in std\_logic;  resetn : in std\_logic;  velg\_enable : in std\_logic\_vector(2 downto 0);  Enable : out std\_logic  );  end component;  signal hallo : std\_logic;  signal clk\_50 : std\_logic;  signal enable\_in : std\_logic\_vector(2 downto 0);  signal enable\_temp : std\_logic\_vector(2 downto 0);  signal resetn : std\_logic;  signal t : integer range 0 to 50000000;  signal enable\_out : std\_logic;  signal sekund\_std : std\_logic\_vector(5 downto 0);  signal minutt\_std : std\_logic\_vector(5 downto 0);  signal timer\_std : std\_logic\_vector(4 downto 0);  begin  LEDR(17) <= hallo;  clk\_50 <= CLOCK\_50;  resetn <= KEY(0);  enable\_in <= SW(2 downto 0);  LEDR(5 downto 0) <= sekund\_std(5 downto 0);  LEDR(11 downto 6) <= minutt\_std(5 downto 0);  LEDR(16 downto 12) <= timer\_std(4 downto 0);  enable\_gen\_1 : Enable\_gen  port map(  clock\_50 => clk\_50,  resetn => resetn,  velg\_enable => enable\_in,  enable => enable\_out  );  tid : process(clk\_50)  variable s : integer := to\_integer(unsigned(sekund\_std));  variable m : integer := to\_integer(unsigned(minutt\_std));  variable t : integer := to\_integer(unsigned(timer\_std));  begin  if rising\_edge(clk\_50) then  if enable\_out = '1' then  if s < 59 then  s := s + 1;  else  s := 0;  m := m + 1;  if m = 60 then  m := 0;  t := t + 1;  if t = 24 then  t := 0;  end if;  end if;  end if;  end if;  if resetn = '0' then  s := 0;  m := 0;  t := 0;  end if;  sekund\_std <= std\_logic\_vector(to\_unsigned(s, sekund\_std'length));  minutt\_std <= std\_logic\_vector(to\_unsigned(m, minutt\_std'length));  timer\_std <= std\_logic\_vector(to\_unsigned(t, timer\_std'length));  end if;  end process tid;  blink : process(CLOCK\_50)  begin  if rising\_edge(CLOCK\_50) then  if enable\_out = '1' then  hallo <= not hallo;  end if;  end if;  end process blink;  end architecture RTL; |

|  |
| --- |
| **Sett inn bilde fra signaltap** |
|  |

## 

|  |
| --- |
| **Sett inn bilde av simulering her. Analyser simuleringen.** |
|  |

## 

## oppgave 3\_c: klokke med 7-segment-display

|  |
| --- |
| **Sett inn bilde av VHDL koden for oppgave 3c her.** |
| library ieee;  use ieee.std\_logic\_1164.all;  use ieee.numeric\_std.all;  entity L4\_klokke\_ELR\_SF\_3c is  port (  CLOCK\_50 : in std\_logic;  KEY : in std\_logic\_vector(3 downto 0);  SW : in std\_logic\_vector(2 downto 0);  LEDR : out std\_logic\_vector(17 downto 0);  HEX0,  HEX1,  HEX2,  HEX3,  HEX4,  HEX5 : out std\_logic\_vector(6 downto 0)  );  end entity L4\_klokke\_ELR\_SF\_3c;  architecture RTL of L4\_klokke\_ELR\_SF\_3c is  component Enable\_gen is  port (  clock\_50 : in std\_logic;  resetn : in std\_logic;  velg\_enable : in std\_logic\_vector(2 downto 0);  Enable : out std\_logic  );  end component;    component bin2bcd is  port(  bin\_in : in std\_logic\_vector(6 downto 0);  bcd\_out : out std\_logic\_vector(7 downto 0)  );  end component;    component ROM\_7\_seg is  port(  adresse : in std\_logic\_vector(3 downto 0);  HEX : out std\_logic\_vector(0 to 6)  );  end component;    signal hallo : std\_logic;  signal clk\_50 : std\_logic;  signal enable\_in : std\_logic\_vector(2 downto 0);  signal resetn : std\_logic;  signal enable\_out : std\_logic;  signal sekund\_std : std\_logic\_vector(5 downto 0);  signal minutt\_std : std\_logic\_vector(5 downto 0);  signal timer\_std : std\_logic\_vector(4 downto 0);  signal bcd\_sek,  bcd\_min,  bcd\_tim : std\_logic\_vector(7 downto 0);  begin  LEDR(17) <= hallo;  clk\_50 <= CLOCK\_50;  resetn <= KEY(0);  enable\_in <= SW(2 downto 0);    LEDR(5 downto 0) <= sekund\_std(5 downto 0);  LEDR(11 downto 6) <= minutt\_std(5 downto 0);  LEDR(16 downto 12) <= timer\_std(4 downto 0);    enable\_gen\_1 : Enable\_gen  port map(  clock\_50 => clk\_50,  resetn => resetn,  velg\_enable => enable\_in,  enable => enable\_out  );    bin2bcd\_0 : bin2bcd  port map(  bin\_in => '0' & sekund\_std,  bcd\_out => bcd\_sek  );    bin2bcd\_1 : bin2bcd  port map(  bin\_in => '0' & minutt\_std,  bcd\_out => bcd\_min  );    bin2bcd\_2 : bin2bcd  port map(  bin\_in => "00" & timer\_std,  bcd\_out => bcd\_tim  );    dekoder\_0 : ROM\_7\_seg  port map(  adresse => bcd\_sek(3 downto 0),  HEX => HEX0  );    dekoder\_1 : ROM\_7\_seg  port map(  adresse => bcd\_sek(7 downto 4),  HEX => HEX1  );    dekoder\_2 : ROM\_7\_seg  port map(  adresse => bcd\_min(3 downto 0),  HEX => HEX2  );    dekoder\_3 : ROM\_7\_seg  port map(  adresse => bcd\_min(7 downto 4),  HEX => HEX3  );    dekoder\_4 : ROM\_7\_seg  port map(  adresse => bcd\_tim(3 downto 0),  HEX => HEX4  );    dekoder\_5 : ROM\_7\_seg  port map(  adresse => bcd\_tim(7 downto 4),  HEX => HEX5  );    tid : process(clk\_50)  variable s : integer := to\_integer(unsigned(sekund\_std));  variable m : integer := to\_integer(unsigned(minutt\_std));  variable t : integer := to\_integer(unsigned(timer\_std));  begin  if rising\_edge(clk\_50) then  if enable\_out = '1' then  if s < 59 then  s := s + 1;  else  s := 0;  m := m + 1;  if m = 60 then  m := 0;  t := t + 1;  if t = 24 then  t := 0;  end if;  end if;  end if;  end if;  if resetn = '0' then  s := 0;  m := 0;  t := 0;  end if;  sekund\_std <= std\_logic\_vector(to\_unsigned(s, sekund\_std'length));  minutt\_std <= std\_logic\_vector(to\_unsigned(m, minutt\_std'length));  timer\_std <= std\_logic\_vector(to\_unsigned(t, timer\_std'length));  end if;  end process tid;      blink : process(CLOCK\_50)  begin  if rising\_edge(CLOCK\_50) then  if enable\_out = '1' then  hallo <= not hallo;  end if;  end if;  end process blink;  end architecture RTL; |

|  |
| --- |
| **Fungerende vhdl-design skal vises til faglærer** |
|  |
|  |

|  |
| --- |
| **Zip prosjektet og lever det inn** |
|  |