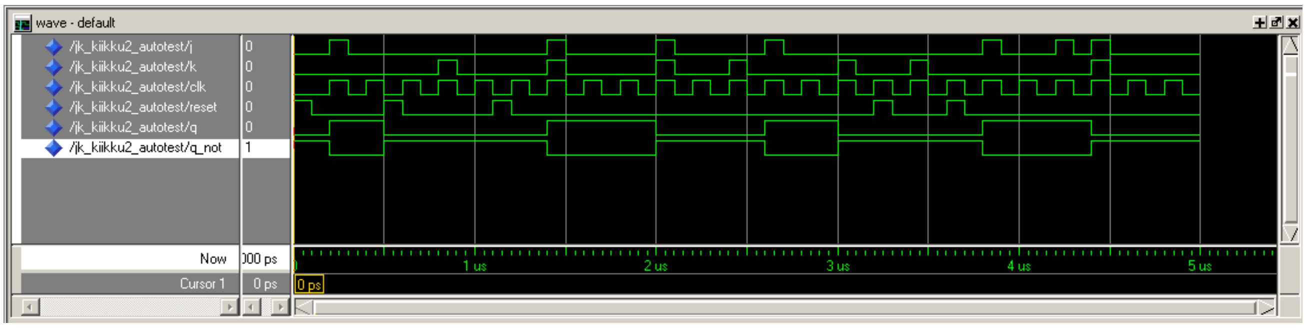
VHDL Tehtävä 1 : JK-Kiikku – toteutettu IF:llä

Koodi:

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

entity JK_kiikku2 is
   Port ( J : in std_logic;
          K : in std_logic;
          CLK : in std_logic;
          Reset : in std_logic;
          Q : out std_logic;
          Q_not : out std_logic);
end JK_kiikku2;

architecture Behavioral of JK_kiikku2 is

begin

process(J,K,CLK,Reset)

variable MP_Q : std_logic;
variable MP_Q_not : std_logic;

        begin

        if (Reset = '1') then
                MP_Q := '0';
                MP_Q_not := '1';
                Q <= '0';
                Q_not <= '1';

        elsif (CLK'event and CLK = '1') then
                if (J = '0' and K = '0') then
                        Q <= MP_Q;
                        Q_not <= MP_Q_not;
                elsif (J = '0' and K = '1') then
                        MP_Q := '0';
                        MP_Q_not := '1';
                        Q <= MP_Q;
                        Q_not <= MP_Q_not;
                elsif (J = '1' and K = '0') then
                        MP_Q := '1';
                        MP_Q_not := '0';
                        Q <= MP_Q;
```

```
            Q_not <= MP_Q_not;
        elsif (J = '1' and K = '1') then
            MP_Q := not MP_Q;
            MP_Q_not := not MP_Q_not;
            Q <= MP_Q;
            Q_not <= MP_Q_not;
        end if;
    end if;

    end process;

end Behavioral;
```

Simulointi:



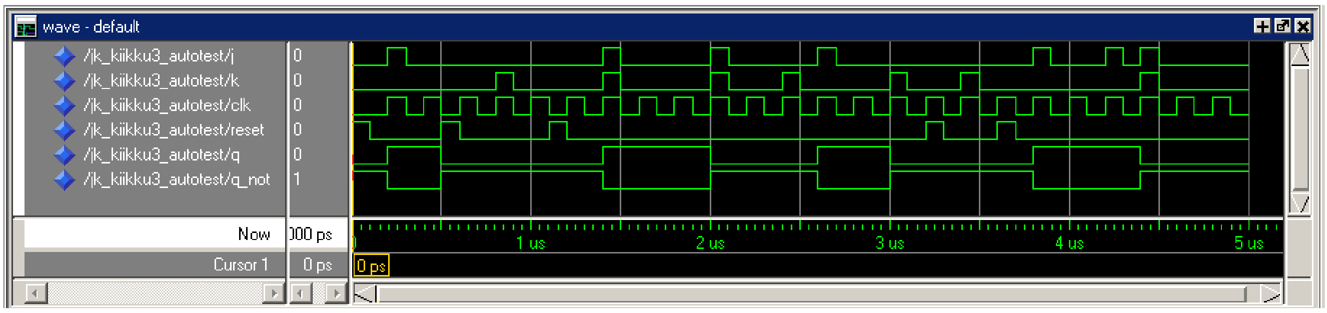
VHDL Tehtävä 1 : JK-Kiikku – toteutettu WHEN:llä

Koodi:

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

entity JK_kiikku3 is
   Port ( J : in std_logic;
          K : in std_logic;
          CLK : in std_logic;
          Reset : in std_logic;
          Q : out std_logic;
          Q_not : out std_logic);
end JK_kiikku3;

architecture Behavioral of JK_kiikku3 is

begin

process(J,K,CLK,Reset)

variable MP_Q : std_logic;
variable MP_Q_not : std_logic;
variable sel : std_logic_vector(1 downto 0);

         begin

         sel := J&K;

         if (Reset = '1') then
                  MP_Q := '0';
                  MP_Q_not := '1';
                  Q <= '0';
                  Q_not <= '1';

         elsif (CLK'event and CLK = '1') then
                  case (sel) is
                  when "00" =>
                           Q <= MP_Q;
                           Q_not <= MP_Q_not;
                  when "01" =>
                           MP_Q := '0';
                           MP_Q_not := '1';
                           Q <= MP_Q;
                           Q_not <= MP_Q_not;
```

```
                        when "10" =>
                                MP_Q := '1';
                                MP_Q_not := '0';
                                Q <= MP_Q;
                                Q_not <= MP_Q_not;
                        when "11" =>
                                MP_Q := not MP_Q;
                                MP_Q_not := not MP_Q_not;
                                Q <= MP_Q;
                                Q_not <= MP_Q_not;
                        when others =>
                        end case;
                end if;

                end process;

end Behavioral;
```

Simulointi:



VHDL Tehtävä 2 : My_and_or

Koodi:

my_and-behavioral

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

entity My_and is
    Port ( I1 : in std_logic;
           I2 : in std_logic;
           O1 : out std_logic);
end My_and;

architecture Behavioral of My_and is

begin

                    O1 <= I1 and I2;

end Behavioral;
```

---

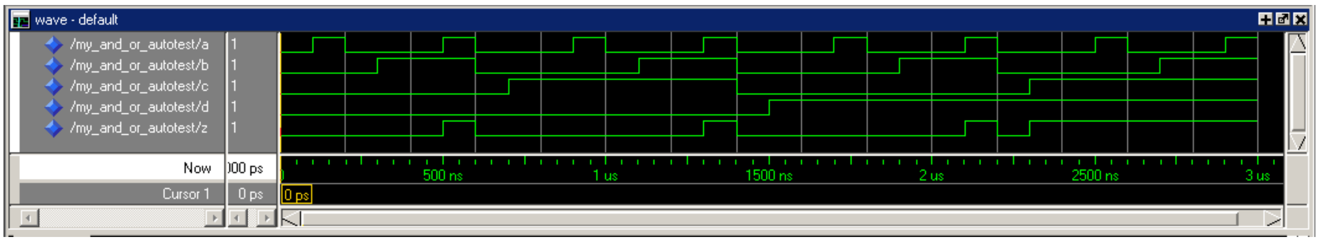
my_or-behavioral

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

entity my_or is
    Port ( I1 : in std_logic;
           I2 : in std_logic;
           O1 : out std_logic);
end my_or;

architecture Behavioral of my_or is

begin

          O1 <= I1 or I2;

end Behavioral;
```

my_and_or-behavioral

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

entity my_and_or is
        Port (  A : in std_logic;
                B : in std_logic;
                C : in std_logic;
                D : in std_logic;
                Z : out std_logic);
end my_and_or;

architecture Behavioral of my_and_or is

COMPONENT my_and
        Port (  I1 : in std_logic;
                I2 : in std_logic;
                O1 : out std_logic);
end component;

COMPONENT my_or is
   Port (       I1 : in std_logic;
                I2 : in std_logic;
                O1 : out std_logic);
end component;

signal SIG1 : std_logic;
signal SIG2 : std_logic;

begin
        my_and1 : my_and port map ( I1=>A, I2=>B, O1=>SIG1 );
        my_and2 : my_and port map ( I1=>C, I2=>D, O1=>SIG2 );
        my_or1 : my_or port map ( I1=>SIG1, I2=>SIG2, O1=>Z );
end Behavioral;
```

Simulointi:



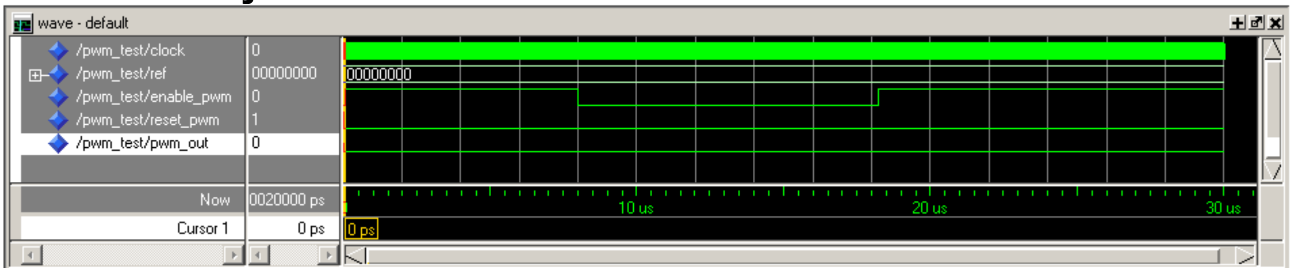
VHDL Tehtävä 3 : PWM-modulaattori

Koodi:

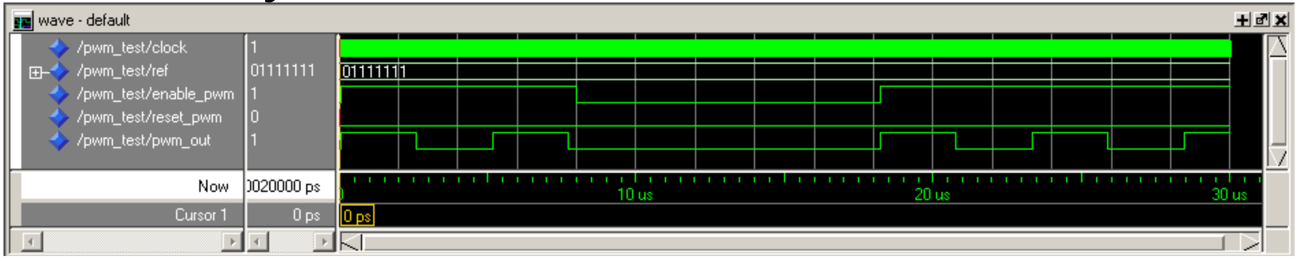
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity PWM is
   Port ( clock : in std_logic;
          ref : in std_logic_vector(7 downto 0);
          enable_PWM : in std_logic;
          reset_PWM : in std_logic;
          PWM_out : out std_logic
        );
end PWM;

architecture Behavioral of PWM is

signal count : std_logic_vector(7 downto 0);

begin

process (clock, reset_PWM, enable_PWM)

begin

        if reset_PWM = '1' or enable_PWM = '0' then
                count <= "00000000";
                PWM_out <= '0';

        elsif clock='0' and clock'event and enable_PWM='1' and reset_PWM = '0' then

                if count < "11111111" then

                        count <= count + '1';

                        if count > ref then
                                PWM_out <= '0';
                        else
                                PWM_out <= '1';
                        end if;

                else
                        count <= "00000000";
                end if;
        end if;
end process;
end Behavioral;
```

Simulointi:

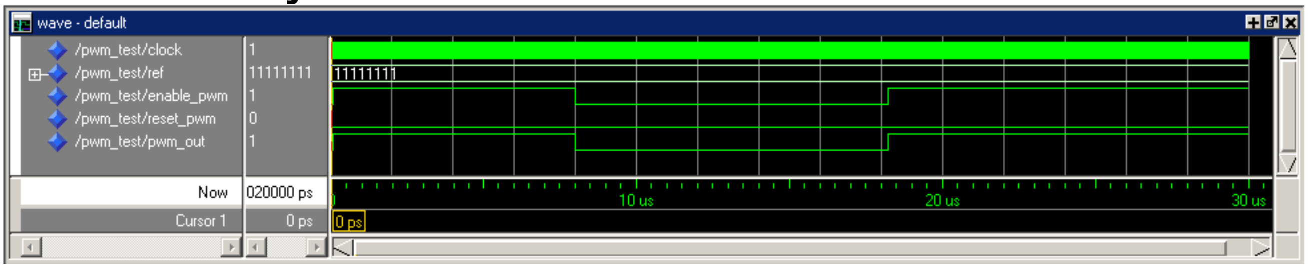
Lähtö 0% -ohjauksella



Lähtö 50% -ohjauksella



Lähtö 100% -ohjauksella

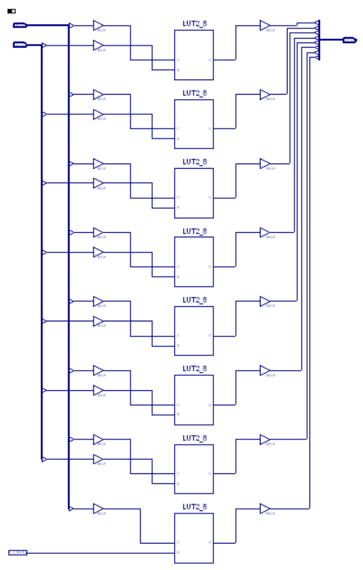


VHDL Tehtävä : FOR LOOP and

Koodi:

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

entity forluuppiandi is
    Port ( A : in std_logic_vector(7 downto 0);
           B : in std_logic_vector(7 downto 0);
           Z : out std_logic_vector(7 downto 0));
end forluuppiandi;

architecture Behavioral of forluuppiandi is

begin
        process (A,B)
                begin
                        for i in 7 downto 0 loop
                                Z(i) <= A(i) and B(i);
                        end loop;
        end process;
end Behavioral;
```

Teknologiatoteutus:

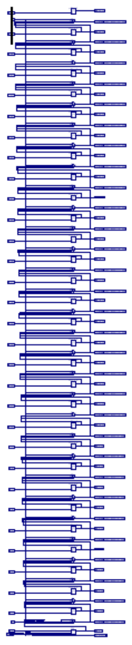


VHDL Tehtävä : FOR LOOP laskuri

Koodi:

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

entity forluuppilaskuri is
    Port ( out1 : out std_logic;
           clk : in std_logic;
           aa: out integer;
           reset : in std_logic);
end forluuppilaskuri;

architecture Behavioral of forluuppilaskuri is

begin

process (clk)
variable laskuri : integer;

begin

if reset = '1' then
         laskuri := 0;
         aa <= 0;

elsif clk='1' and clk'event then

                   for i in 19 downto 0 loop
                   laskuri := laskuri + 1;
                            if laskuri > 100 then
                                     out1 <= '0';
                            else
                                     out1 <= '1';
                            end if;
                            aa <=laskuri;
                   end loop;

end if;
end process;

end Behavioral;
```

Teknologiatoteutus:



41 sivua pelkkää p*skaa generoitu for-loopin ansiosta. Yllä oleva kuva esittää yhden sivun sisältöä.

VHDL Tehtävä : Generic ja Generate

Koodi:

generic_generate-behavioral

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

entity generic_generate is
        generic (cycles: positive:=10);
        Port(D,clr,clk : in bit;
                q : out bit);
end generic_generate;

architecture Behavioral of generic_generate is
component dff is
        Port(D,clr,clk : in bit;
                q : out bit);
end component;

signal s :bit_vEctor(cycles - 1 downto 1);

begin

cycleEQ1:if cycles = 1 generate
d1: dff port map (D,clr,clk,q);
end generate cycleEQ1;

cycleGT1: if cycles > 1 generate
first:dff port map (D,clr,clk,s(1));
inTheMiddle:For i in 2 to cycles-1 generate
di: dff port map (s(i-1),clr,clk,s(i));
end generate;

last: dff port map (s(cycles-1),clr,clk,q);
end generate cycleGT1;
end Behavioral;
```

dff-behavioral

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

entity dff is
    Port ( D : in std_logic;
           clr : in std_logic;
           clk : in std_logic;
           q : out std_logic);
end dff;

architecture Behavioral of dff is

begin

        process (D,clr,clk)
                begin
                        if (clr='1') then
                                q <= '0';

                        elsif (clk'event and clk='1' and clr='0') then
                                q <= D;
                        end if;

        end process;

end Behavioral;
```