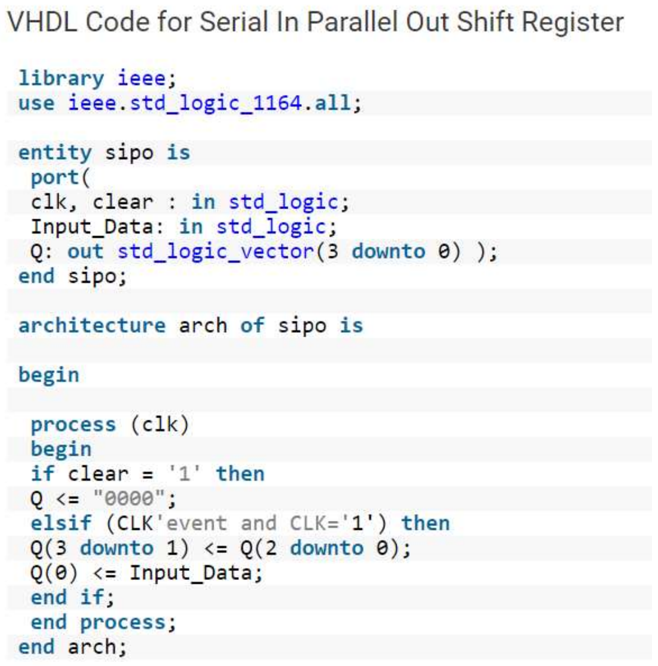
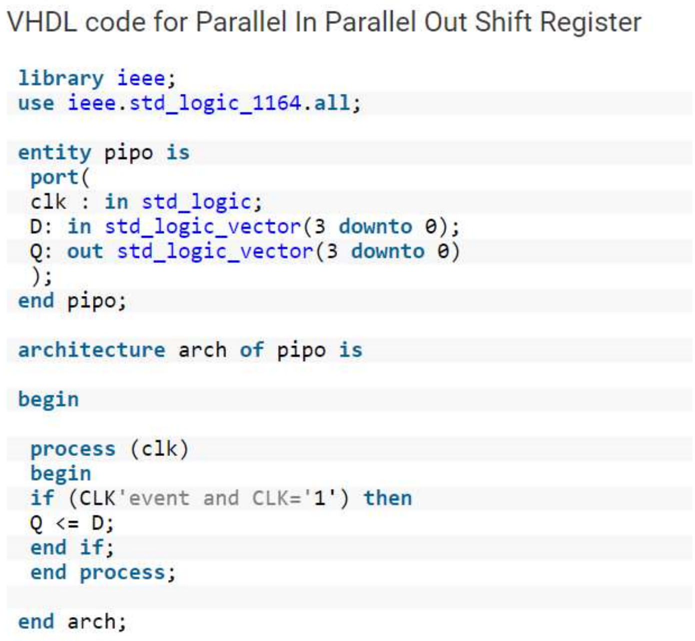
ESE VDHL CODES

02 May 2024 03:16 AM



VHDL Code for Serial In Parallel Out Shift Register

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

entity sipo is
 port(
 clk, clear : in std_logic;
 Input_Data: in std_logic;
 Q: out std_logic_vector(3 downto 0) );
end sipo;

architecture arch of sipo is

begin

 process (clk)
 begin
 if clear = '1' then
 Q <= "0000";
 elsif (CLK'event and CLK='1') then
 Q(3 downto 1) <= Q(2 downto 0);
 Q(0) <= Input_Data;
 end if;
 end process;
end arch;
```



VHDL code for Parallel In Parallel Out Shift Register

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

entity pipo is
 port(
 clk : in std_logic;
 D: in std_logic_vector(3 downto 0);
 Q: out std_logic_vector(3 downto 0)
 );
end pipo;

architecture arch of pipo is

begin

 process (clk)
 begin
 if (CLK'event and CLK='1') then
 Q <= D;
 end if;
 end process;

end arch;
```

------------------- D Flip Flop Design --------------------

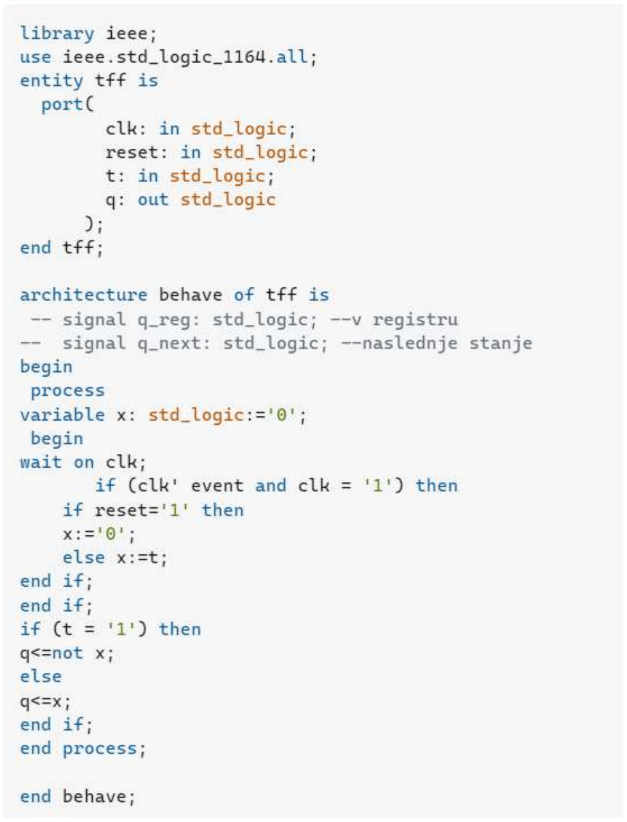
```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity d_flip_flop is
    port(
        clk : in STD_LOGIC;
        din : in STD_LOGIC;
        reset : in STD_LOGIC;
        dout : out STD_LOGIC
        );
end d_flip_flop;

architecture d_flip_flop_arc of d_flip_flop is

begin

    dff : process (din,clk,reset) is
    begin
        if (reset='1') then
            dout <= '0';
        elsif (rising_edge (clk)) then
            dout <= din;
        end if;
    end process dff;


end d_flip_flop_arc;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity siso is
    port(
        din : in STD_LOGIC;
        clk : in STD_LOGIC;
        reset : in STD_LOGIC;
        dout : out STD_LOGIC
        );
end siso;

architecture siso_arc of siso is

component d_flip_flop is
    port(
        clk : in STD_LOGIC;
        din : in STD_LOGIC;
        reset : in STD_LOGIC;
        dout : out STD_LOGIC
        );
end component d_flip_flop;

signal s : std_logic_vector(2 downto 0);

begin

    u0 : d_flip_flop port map (clk => clk,
    din => din,
    reset => reset,
    dout => s(0));

    u1 : d_flip_flop port map (clk => clk,
    din => s(0),
    reset => reset,
    dout => s(1));
    u2 : d_flip_flop port map (clk => clk,
    din => s(1),
    reset => reset,
    dout => s(2));

    u3 : d_flip_flop port map (clk => clk,
    din => s(2),
    reset => reset,
    dout => dout);


end siso_arc;
```

Full VHDL code for synchronous up-counter using the behavioral method

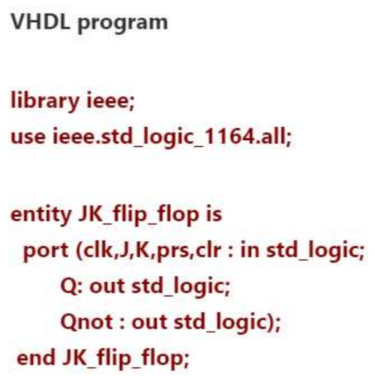
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity SOURCE is
    Port ( CLK,RST : in  STD_LOGIC;
             COUNT : inout  STD_LOGIC_VECTOR (3 downto 0));
end SOURCE;

architecture Behavioral of SOURCE is

begin
process (CLK,RST)
begin

if (RST = '1')then
COUNT <= "0000";
elsif(rising_edge(CLK))then
COUNT <= COUNT+1;

end if;
end process;
end Behavioral;
```

T-FLIP FLOP



```
library ieee;
use ieee.std_logic_1164.all;
entity tff is
  port(
        clk: in std_logic;
        reset: in std_logic;
        t: in std_logic;
        q: out std_logic
      );
end tff;

architecture behave of tff is
 -- signal q_reg: std_logic; --v registru
--  signal q_next: std_logic; --naslednje stanje
begin
 process
variable x: std_logic:='0';
 begin
wait on clk;
       if (clk' event and clk = '1') then
    if reset='1' then
    x:='0';
    else x:=t;
end if;
end if;
if (t = '1') then
q<=not x;
else
q<=x;
end if;
end process;

end behave;
```

Rising_edge (clk) = for edge triggering
Clock'event = for level triggering



VHDL program

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

entity JK_flip_flop is
 port (clk,J,K,prs,clr : in std_logic;
      Q: out std_logic;
      Qnot : out std_logic);
 end JK_flip_flop;
```

```
architecture JKFF_arch of JK_flip_flop is
  signal nxt_state,prv_state: std_logic;
  signal input: std_logic_vector(1 downto 0);
  begin
   input <= J & K;
    process(clk, prs,clr) is
     begin
      if (clr='1') then
        nxt_state <= '0';
      elsif (prs='1') then
        nxt_state <= '1';
      elsif (clk'event and clk='1') then
       case (input) is
        when "10" => nxt_state <= '1';
        when "01" => nxt_state <= '0';
        when "00" => nxt_state <= prv_state;
        when "11" => nxt_state <= not prv_state;
        when others => null;
       end case;
      end if;
     end process;
    Q <= nxt_state;
    Qnot <= not nxt_state;
    prv_state <= nxt_state;
end JKFF_arch;
```