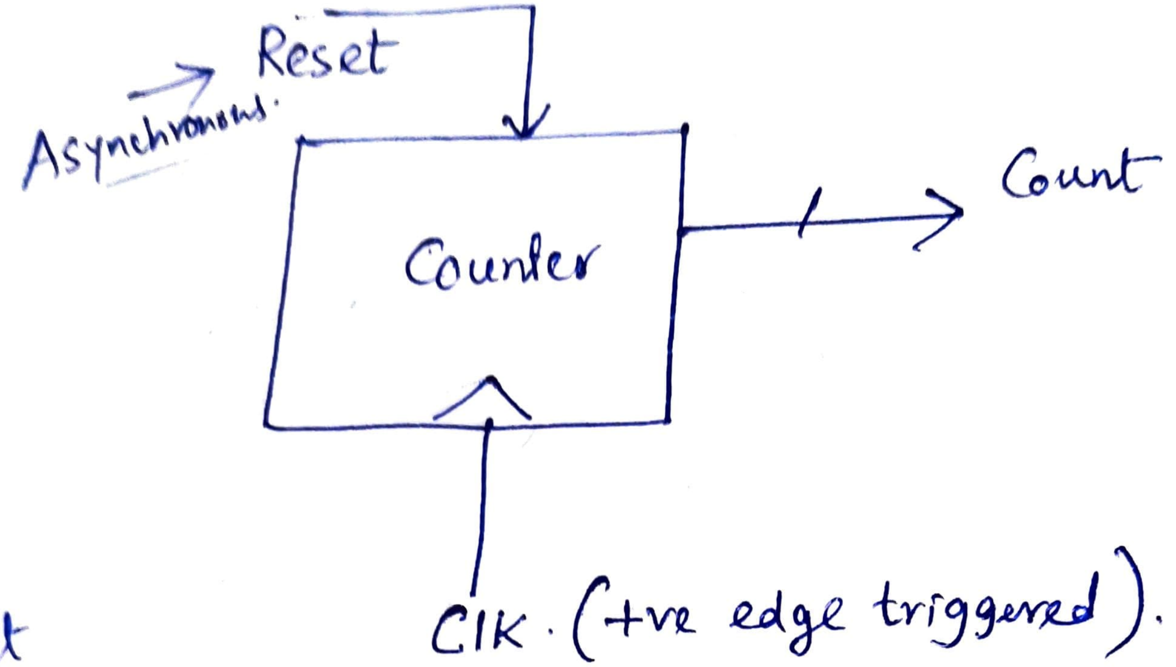
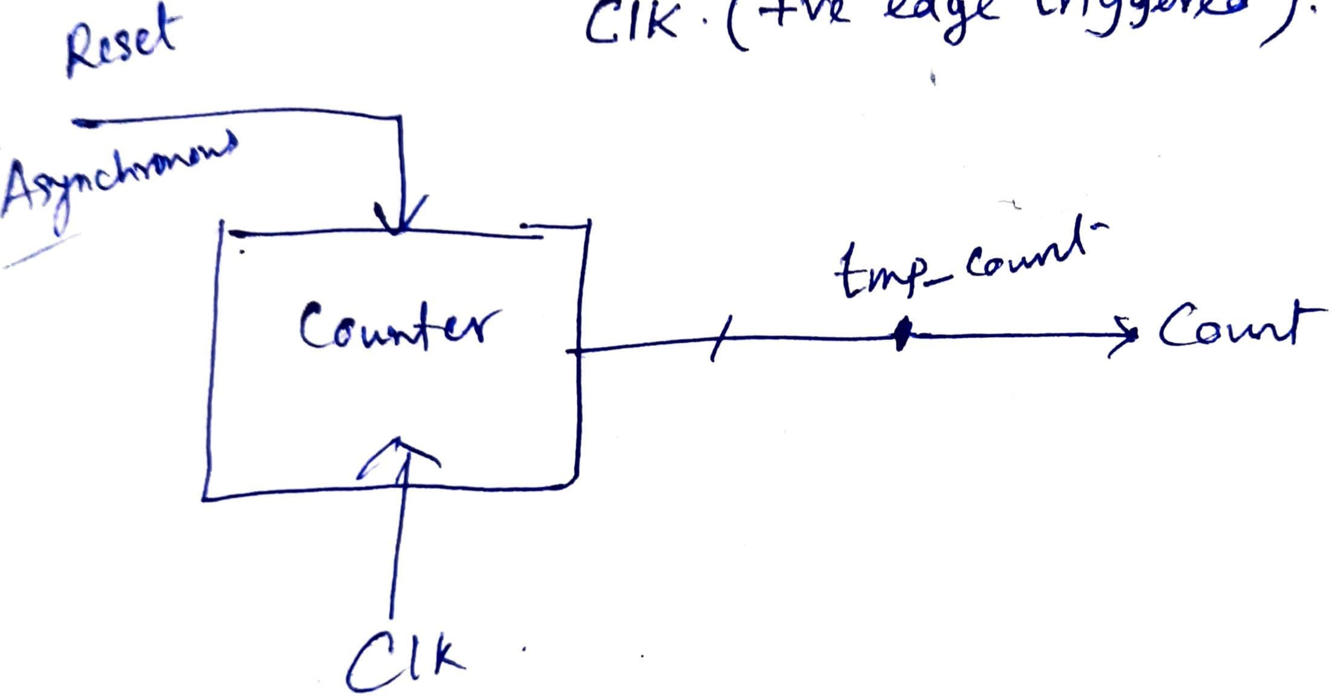
* How to design a 4-bit Counter (up-counter)?

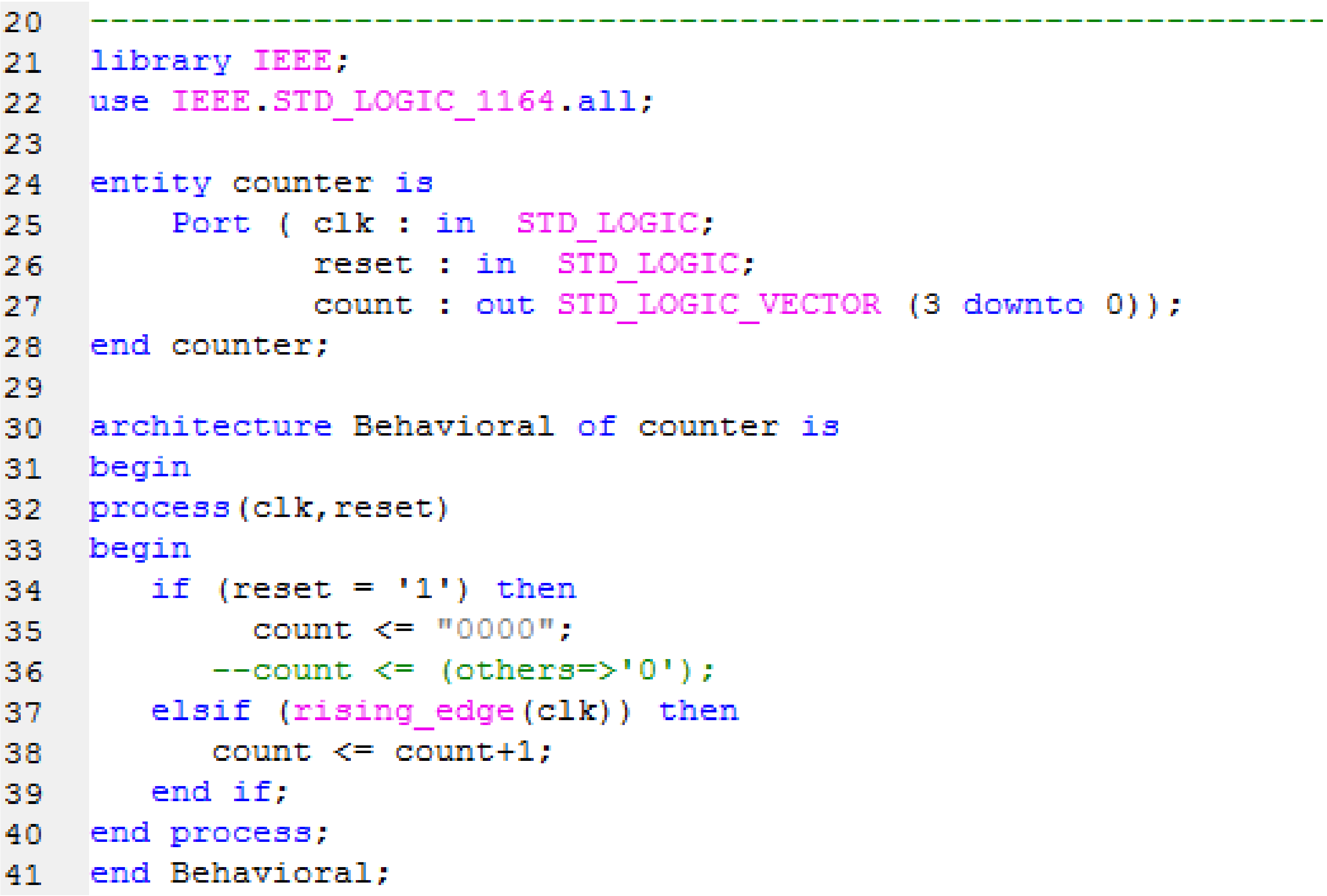




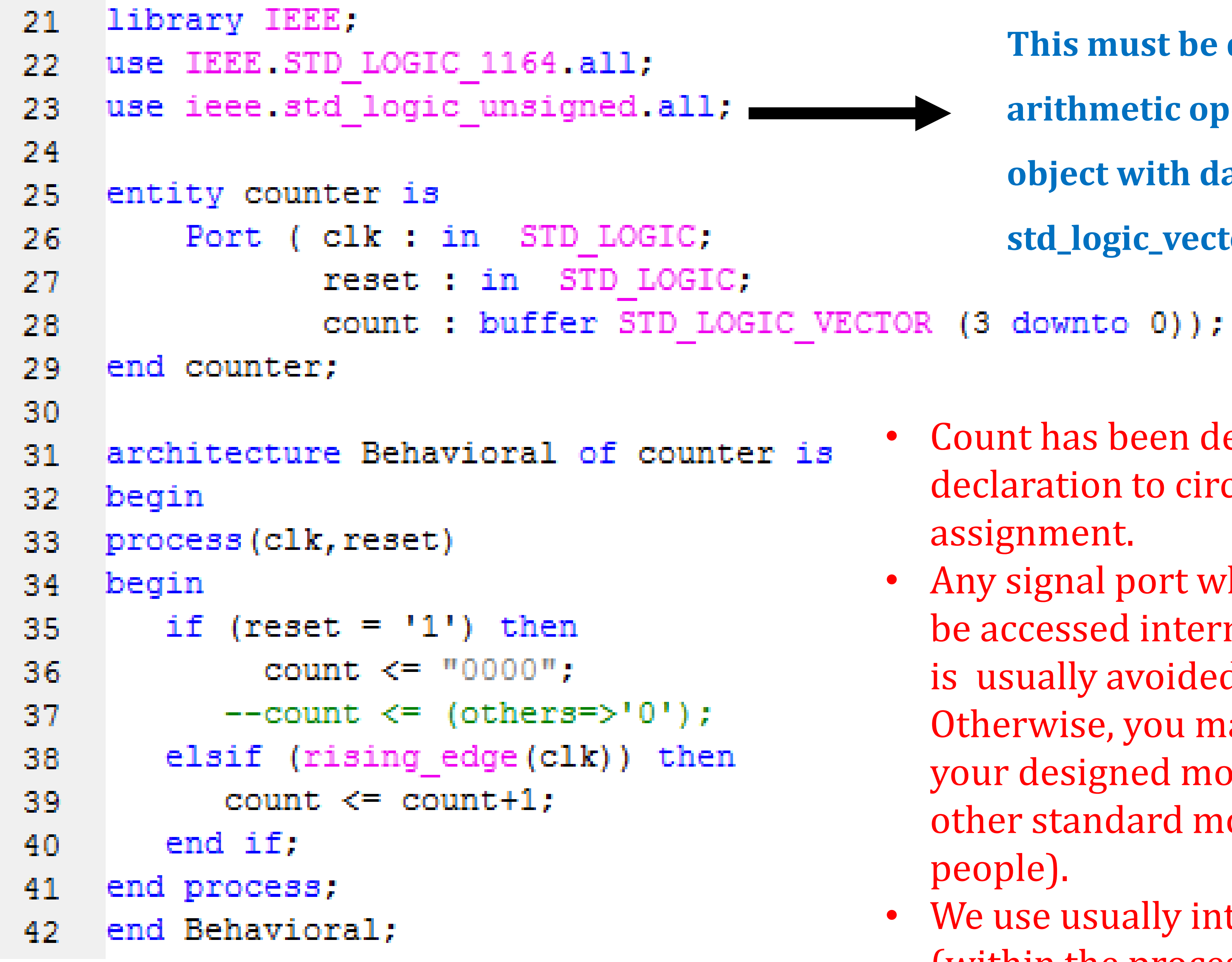
VHDL codes examples

Counter

Any problem with this code?



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

entity counter is
    Port ( clk : in  STD_LOGIC;
           reset : in  STD_LOGIC;
           count : out STD_LOGIC_VECTOR (3 downto 0));
end counter;

architecture Behavioral of counter is
begin
process(clk,reset)
begin
   if (reset = '1') then
        count <= "0000";
      --count <= (others=>'0');
   elsif (rising_edge(clk)) then
      count <= count+1;
   end if;
end process;
end Behavioral;
```

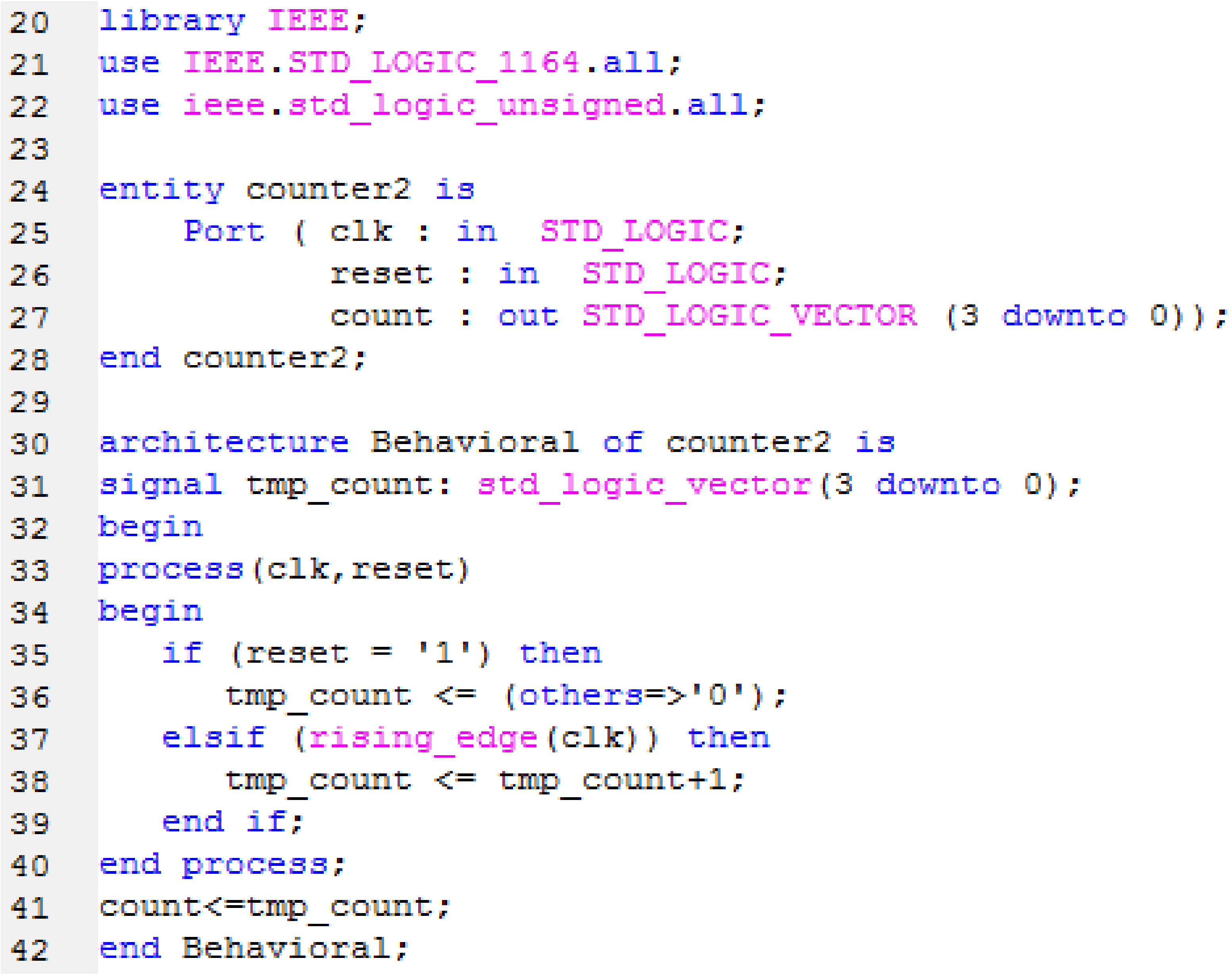


```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use ieee.std_logic_unsigned.all;

entity counter is
    Port ( clk : in  STD_LOGIC;
           reset : in  STD_LOGIC;
           count : buffer STD_LOGIC_VECTOR (3 downto 0));
end counter;

architecture Behavioral of counter is
begin
process(clk,reset)
begin
   if (reset = '1') then
        count <= "0000";
      --count <= (others=>'0');
   elsif (rising_edge(clk)) then
      count <= count+1;
   end if;
end process;
end Behavioral;
```

This must be declared in order to use arithmetic operator (+,*,-,etc.) with object with data type: std_logic and std_logic_vector.

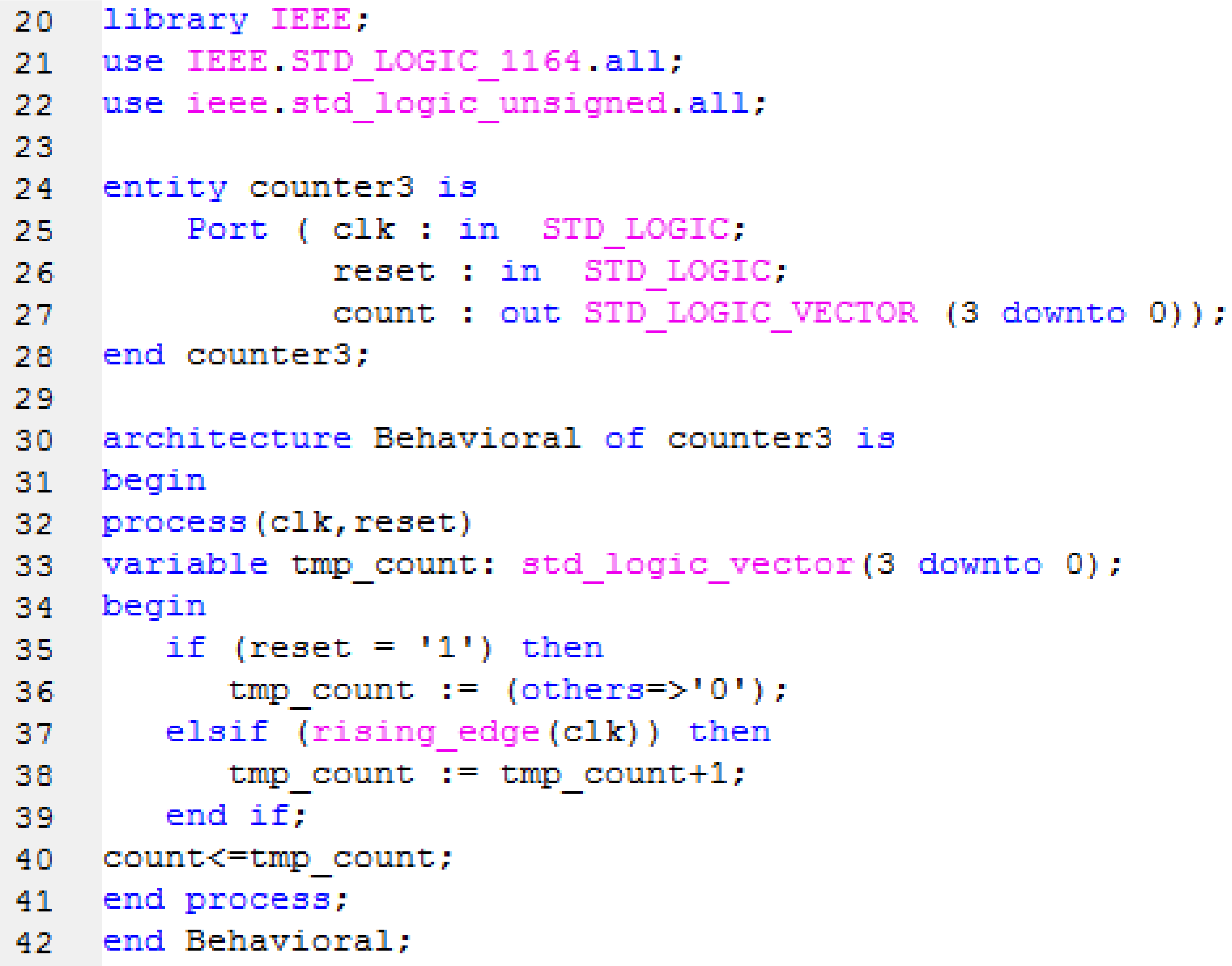
- Count has been declared as ‘buffer’ in port declaration to circumvent the problem of signal assignment.
- Any signal port which is declared as ‘buffer’ can be accessed internally. However, ‘buffer’ usage is usually avoided in a standard design practice. Otherwise, you may face problem in future if your designed module is interfaced with any other standard module (designed by some people).
- We use usually internal signals or variables (within the process) to avoid the usage of ‘buffer’ type port.

4-bit up counter by using signal

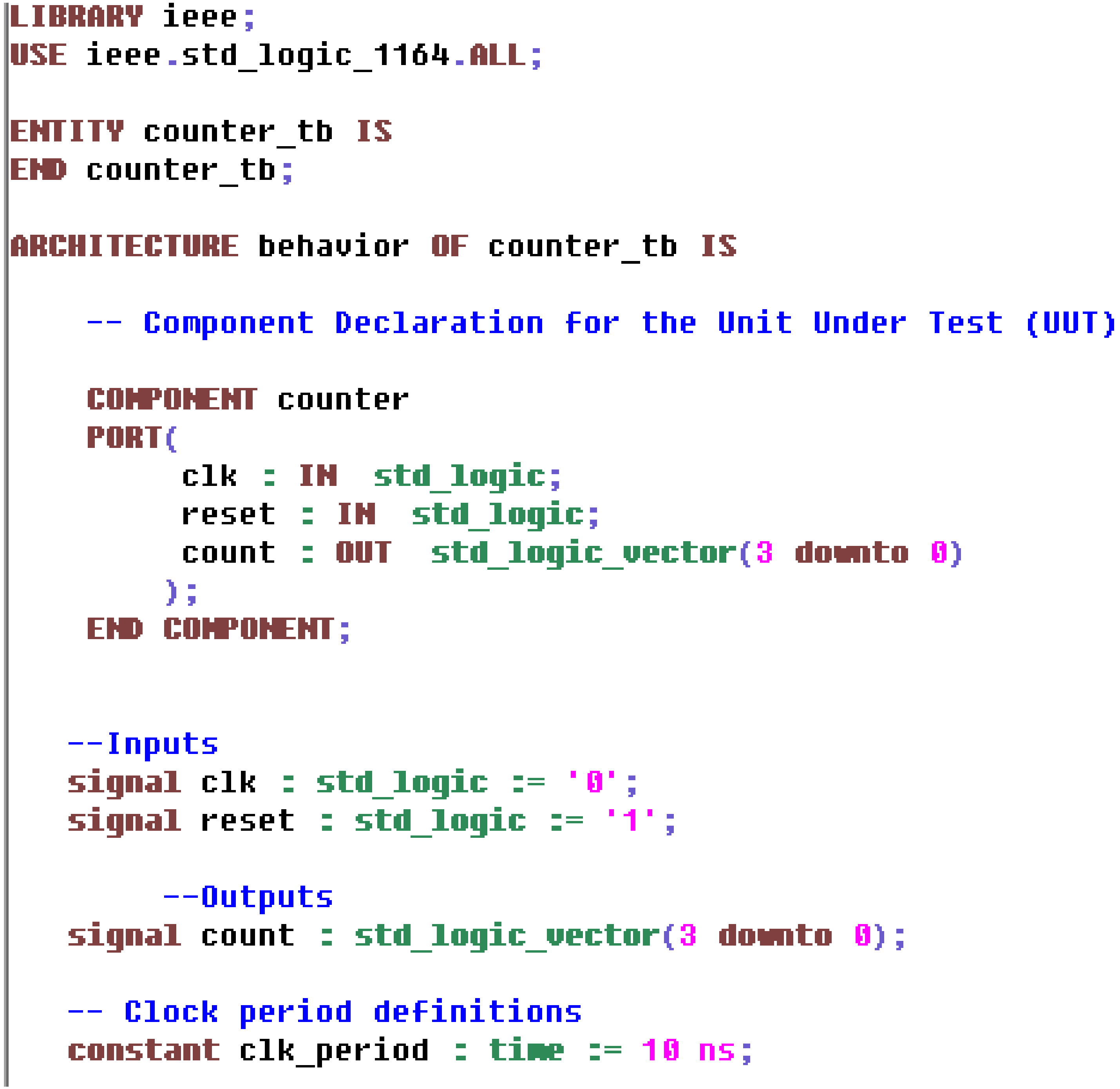


```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use ieee.std_logic_unsigned.all;

entity counter2 is
    Port ( clk : in  STD_LOGIC;
           reset : in  STD_LOGIC;
           count : out STD_LOGIC_VECTOR (3 downto 0));
end counter2;

architecture Behavioral of counter2 is
signal tmp_count: std_logic_vector(3 downto 0);
begin
process(clk,reset)
begin
   if (reset = '1') then
      tmp_count <= (others=>'0');
   elsif (rising_edge(clk)) then
      tmp_count <= tmp_count+1;
   end if;
end process;
count<=tmp_count;
end Behavioral;
```

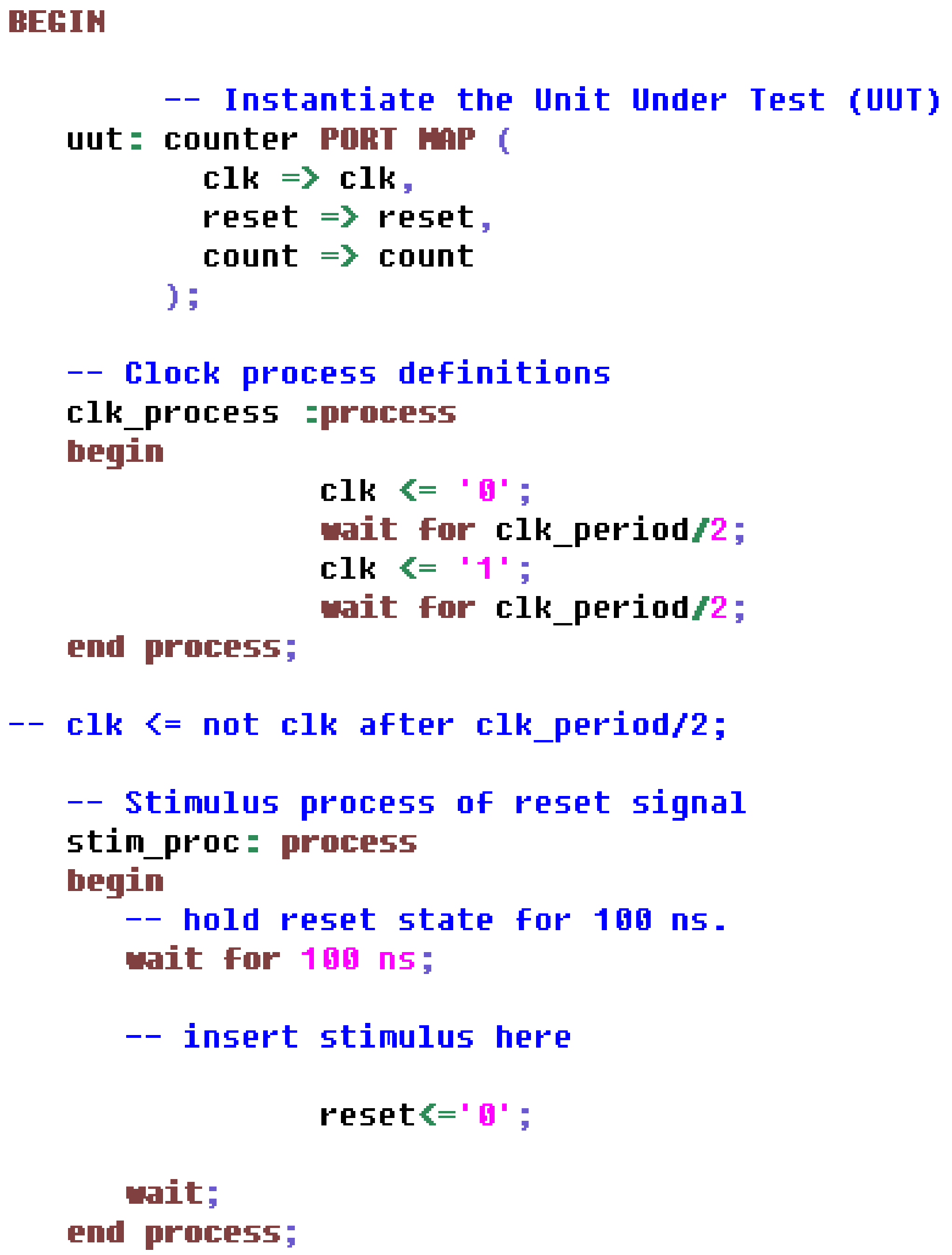
4-bit up counter by using variable



```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use ieee.std_logic_unsigned.all;

entity counter3 is
    Port ( clk : in  STD_LOGIC;
           reset : in  STD_LOGIC;
           count : out STD_LOGIC_VECTOR (3 downto 0));
end counter3;

architecture Behavioral of counter3 is
begin
process(clk,reset)
variable tmp_count: std_logic_vector(3 downto 0);
begin
   if (reset = '1') then
      tmp_count := (others=>'0');
   elsif (rising_edge(clk)) then
      tmp_count := tmp_count+1;
   end if;
count<=tmp_count;
end process;
end Behavioral;
```



```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY counter_tb IS
END counter_tb;

ARCHITECTURE behavior OF counter_tb IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT counter
    PORT(
         clk : IN  std_logic;
         reset : IN  std_logic;
         count : OUT  std_logic_vector(3 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal clk : std_logic := '0';
   signal reset : std_logic := '1';

        --Outputs
   signal count : std_logic_vector(3 downto 0);

   -- Clock period definitions
   constant clk_period : time := 10 ns;
```



```
BEGIN

        -- Instantiate the Unit Under Test (UUT)
   uut: counter PORT MAP (
          clk => clk,
          reset => reset,
          count => count
        );

   -- Clock process definitions
   clk_process :process
   begin
                clk <= '0';
                wait for clk_period/2;
                clk <= '1';
                wait for clk_period/2;
   end process;

-- clk <= not clk after clk_period/2;

   -- Stimulus process of reset signal
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
      wait for 100 ns;

      -- insert stimulus here

                reset<='0';

      wait;
   end process;
```