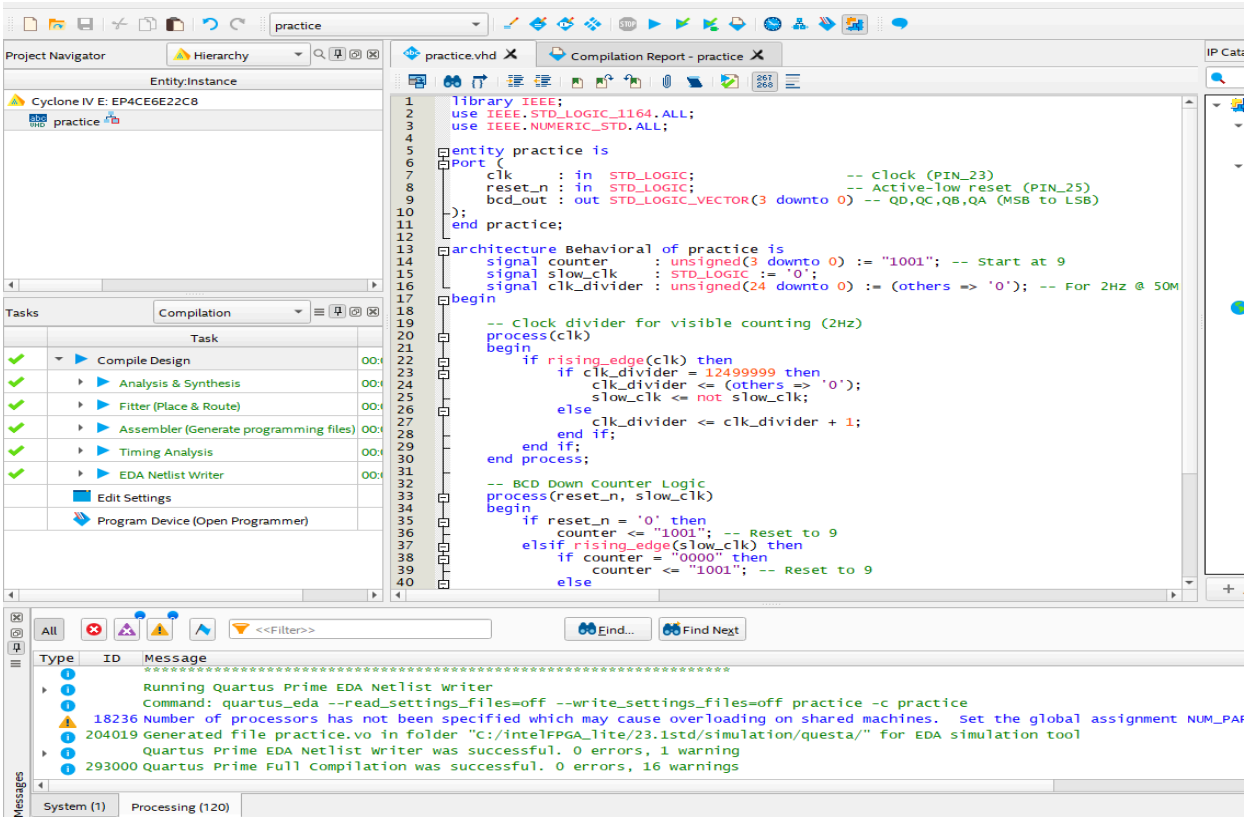
EMMAN ACE G. MENION
BSCPE 3A

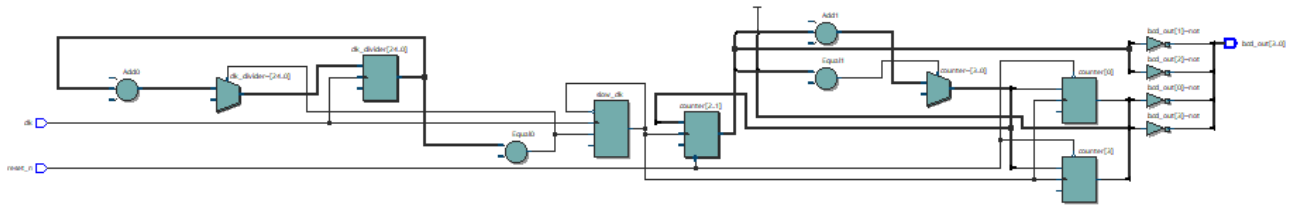
BCD DOWN COUNTER:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity practice is
Port (
   clk     : in  STD_LOGIC;               -- Clock (PIN_23)
   reset_n : in  STD_LOGIC;                -- Active-low reset (PIN_25)
   bcd_out : out STD_LOGIC_VECTOR(3 downto 0) -- QD,QC,QB,QA (MSB to LSB)
);
end practice;

architecture Behavioral of practice is
   signal counter     : unsigned(3 downto 0) := "1001"; -- Start at 9
   signal slow_clk    : STD_LOGIC := '0';
   signal clk_divider : unsigned(24 downto 0) := (others => '0'); -- For 2Hz @ 50MHz
begin

   -- Clock divider for visible counting (2Hz)
   process(clk)
   begin
      if rising_edge(clk) then
         if clk_divider = 12499999 then
            clk_divider <= (others => '0');
            slow_clk <= not slow_clk;
         else
            clk_divider <= clk_divider + 1;
         end if;
      end if;
   end process;

   -- BCD Down Counter Logic
   process(reset_n, slow_clk)
   begin
      if reset_n = '0' then
         counter <= "1001"; -- Reset to 9
      elsif rising_edge(slow_clk) then
         if counter = "0000" then
            counter <= "1001"; -- Reset to 9
         else
            counter <= counter - 1;
         end if;
      end if;
   end process;
```

```
    -- Active-low LED output (0=LED ON, 1=LED OFF)
    bcd_out <= not std_logic_vector(counter);

end Behavioral;
```




```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity practice is
Port (
    clk     : in  STD_LOGIC;                    -- Clock (PIN_23)
    reset_n : in  STD_LOGIC;                    -- Active-low reset (PIN_25)
    bcd_out : out STD_LOGIC_VECTOR(3 downto 0) -- QD,QC,QB,QA (MSB to LSB)
);
end practice;

architecture Behavioral of practice is
    signal counter     : unsigned(3 downto 0) := "1001"; -- Start at 9
    signal slow_clk    : STD_LOGIC := '0';
    signal clk_divider : unsigned(24 downto 0) := (others => '0'); -- For 2Hz @ 50M
begin

    -- Clock divider for visible counting (2Hz)
    process(clk)
    begin
        if rising_edge(clk) then
            if clk_divider = 12499999 then
                clk_divider <= (others => '0');
                slow_clk <= not slow_clk;
            else
                clk_divider <= clk_divider + 1;
            end if;
        end if;
    end process;

    -- BCD Down Counter Logic
    process(reset_n, slow_clk)
    begin
        if reset_n = '0' then
            counter <= "1001"; -- Reset to 9
        elsif rising_edge(slow_clk) then
            if counter = "0000" then
                counter <= "1001"; -- Reset to 9
            else
```



Clock	QD	QC	QB	QA
0	1	0	0	1
1	1	0	0	0
2	0	1	1	1
3	0	1	1	0
4	0	1	0	1
5	0	1	0	0
6	0	0	1	1
7	0	0	1	0
8	0	0	0	1
9	0	0	0	0