Digital Systems VLSI

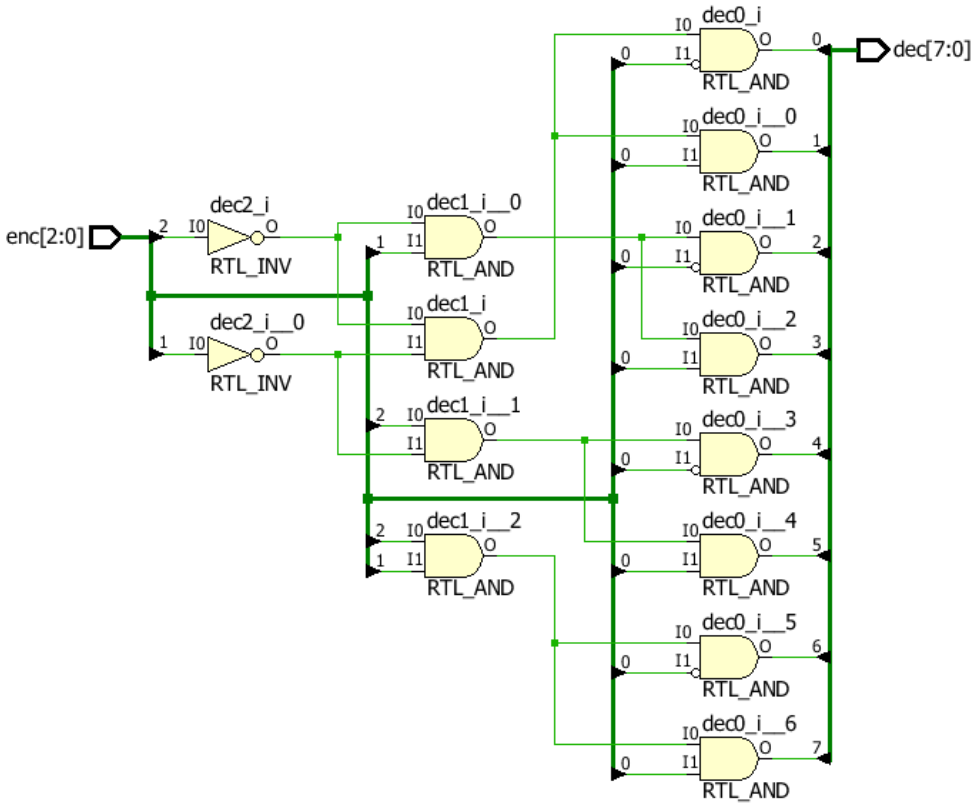
Nanos Georgios

Topic A2)
The following are the dataflow and behavioral descriptions of the 3-in-8 binary decoder:

Dataflow:

```
library ieee;
use ieee.std_logic_1164.all;
entity decoder3to8 is
   port(
      enc: in std_logic_vector(2 downto 0);
      dec: out std_logic_vector(7 downto 0)
   );
end entity;

architecture dataflow of  decoder3to8 is
   begin
      dec(7) <= enc(2) and enc(1) and enc(0);
      dec(6) <= enc(2) and enc(1) and not enc(0);
      dec(5) <= enc(2) and not enc(1) and enc(0);
      dec(4) <= enc(2) and not enc(1) and not enc(0);
      dec(3) <= not enc(2) and enc(1) and enc(0);
      dec(2) <= not enc(2) and enc(1) and not enc(0);
      dec(1) <= not enc(2) and not enc(1) and enc(0);
      dec(0) <= not enc(2) and not enc(1) and not enc(0);
 end dataflow;
```

The corresponding schematic diagram is as follows:

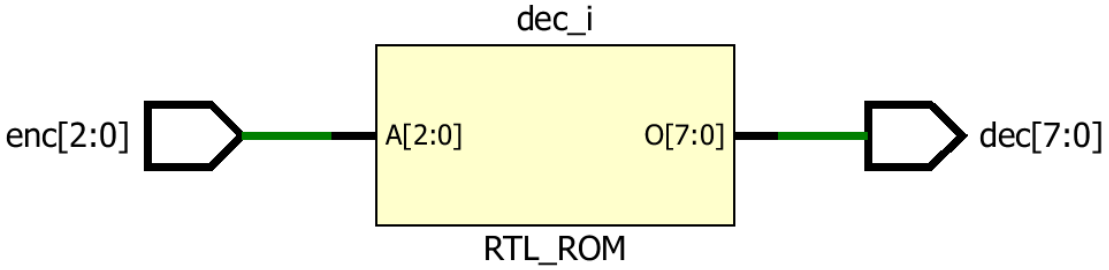


Behavioural:

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

entity decoder3to8 is
  port(
     enc: in std_logic_vector(2 downto 0);
     dec: out std_logic_vector(7 downto 0)
  );
end entity;

architecture behavioural of decoder3to8 is

begin

  process(enc)
  begin
    case enc is
      when "000" =>
        dec <= "00000001";
      when "001" =>
        dec <= "00000010";
      when "010" =>
        dec <= "00000100";
```

```
    when "011" =>
      dec <= "00001000";
    when "100" =>
      dec <= "00010000";
    when "101" =>
      dec <= "00100000";
    when "110" =>
      dec <= "01000000";
    when "111" =>
      dec <= "10000000";
    when others =>
      dec <=  (others  => '-');
  end case;
 end process;

end architecture;
```

The relevant RTL Schematic:



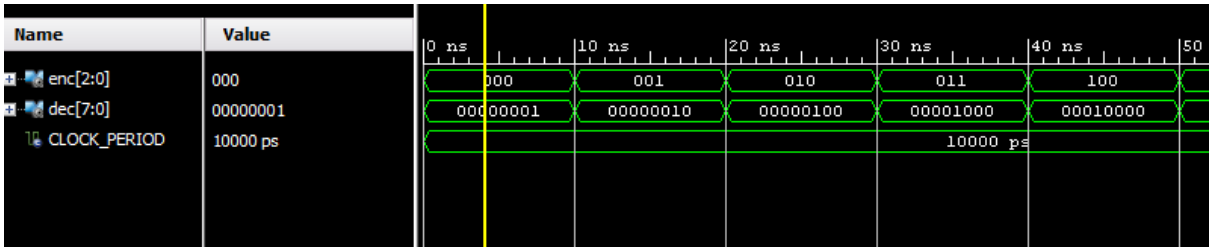
Simulation:

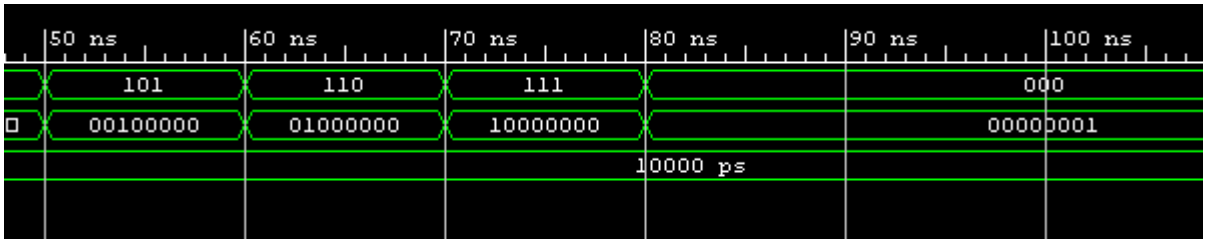
For both different architectures, the same testbench file was used, with the following code:

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

entity decoder3to8_tb is
end entity;
```

```
architecture bench of decoder3to8_tb is

  component decoder3to8 is
    port (
       enc: in std_logic_vector(2 downto 0);
       dec: out std_logic_vector(7 downto 0)
        );
  end component;

  signal enc  : std_logic_vector(2 downto 0) := (others => '0');
  signal dec  : std_logic_vector(7 downto 0);

  constant CLOCK_PERIOD : time := 10 ns;

begin

  test : decoder3to8
    port map (
             enc  => enc,
             dec => dec
            );

  stimulus : process
  begin
    ----- Stimulus example -----
    for i in 0 to 7 loop -- enc loop
       enc <= std_logic_vector(to_unsigned(i, 3));
       wait for CLOCK_PERIOD;
    end loop;
    enc <= (others => '0');
    wait;
  end process;


end architecture;
```

Finally, here is a snapshot of the simulation, in which the input variable is increased in order to constantly change the output value of the decoder:





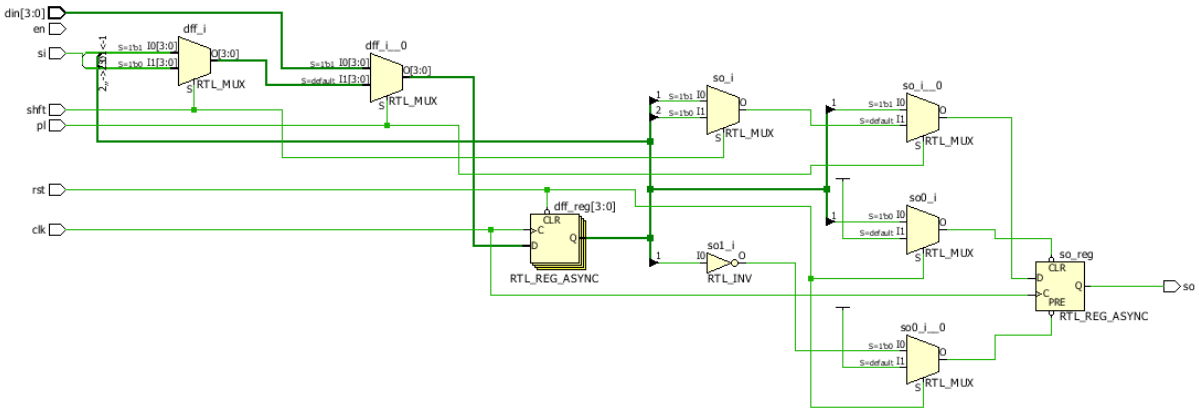
Topic B2)

The following is the description of the requested slider in Behavioural architecture:

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

entity shift_reg is
   port ( clk : in STD_LOGIC;
          rst : in STD_LOGIC;
          shft : in STD_LOGIC;
          si : in STD_LOGIC;
          en : in STD_LOGIC;
          pl : in STD_LOGIC;
          din : in STD_LOGIC_VECTOR (3 downto 0);
          so : out STD_LOGIC);
end shift_reg;

architecture behavioural of shift_reg is
   signal dff: std_logic_vector(3 downto 0);
begin
   edge: process (clk,rst)
   begin
      if rst='0' then
         dff<=(others=>'0');
         so <= dff(0);
      elsif clk'event and clk='1' then
         if pl='1' then
            dff<=din;
            so<=dff(0);
         elsif en='1' then
            case shft is
               when '1' =>
                  so<=dff(0);
                  dff<=si&dff(3 downto 1);
               when '0' =>
                  so<=dff(3);
                  dff<=dff(2 downto 0)&si;
               when others =>
```

```
                so<='-';
                dff<=(others=>'-');
            end case;
          else
            so<='-';
            dff<=(others=>'-');
          end if;
        end if;
    end process;
end behavioural;
```

The corresponding schematic:



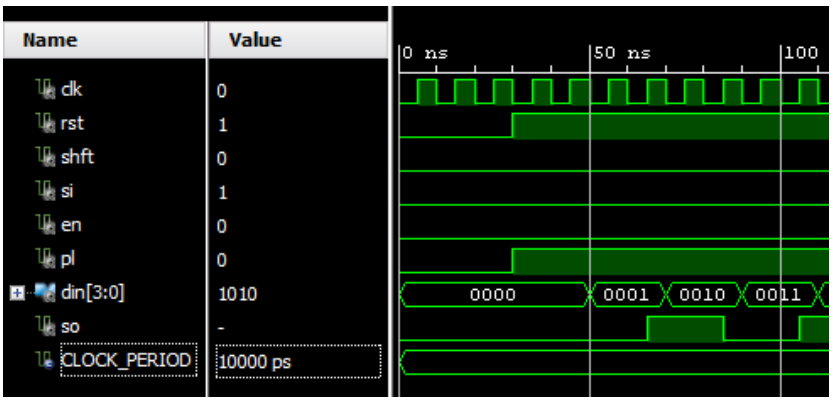
The Testbench file:

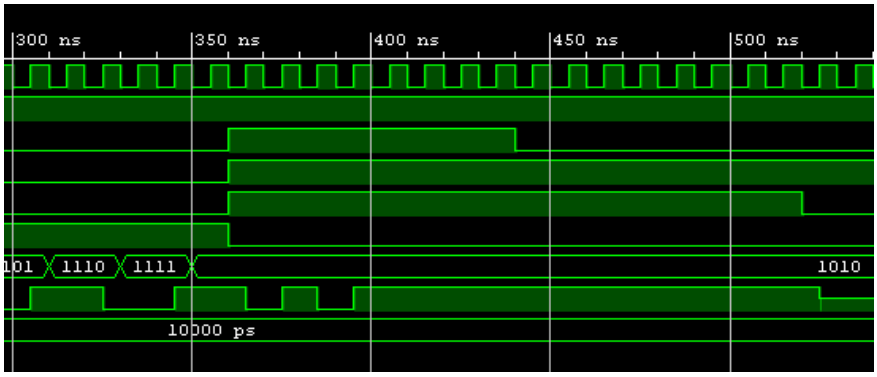
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity shift_reg_tb is
end entity;

architecture bench of shift_reg_tb is

  component shift_reg is
    port ( clk : in STD_LOGIC;
           rst : in STD_LOGIC;
           shft : in STD_LOGIC;
           si : in STD_LOGIC;
           en : in STD_LOGIC;
           pl : in STD_LOGIC;
           din : in STD_LOGIC_VECTOR (3 downto 0);
           so : out STD_LOGIC);
  end component;
```

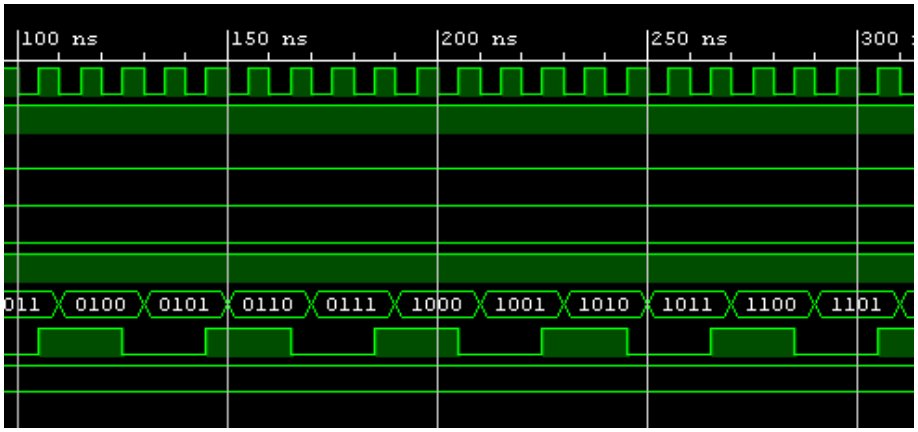
```
  signal clk  : std_logic;
  signal rst  : std_logic := '0';
  signal shft : std_logic := '0';
  signal si : std_logic := '0';
  signal en : std_logic := '0';
  signal pl : std_logic := '0';
  signal din : std_logic_vector(3 downto 0) := (others => '0');
  signal so : std_logic;

  constant CLOCK_PERIOD : time := 10 ns;

begin

  test : shift_reg
    port map (clk => clk,
          rst => rst,
          shft => shft,
          si => si,
          en => en,
          pl => pl,
          din => din,
          so => so
);

  stimulus : process
  begin
    ----- Stimulus example -----
  rst <= '0';
  din <= (others => '0');
  shft <= '0';
  si <= '0';
  en <= '0';
  pl <= '0';
  din <= (others => '0');
  wait for CLOCK_PERIOD*3;

  rst <= '1';

  pl <= '1';
  for i in 0 to 15 loop
            din <= std_logic_vector(to_unsigned(i, 4));
            wait for CLOCK_PERIOD*2;
  end loop;

  din <= (3 => '1',
          2 => '0',
          1 => '1',
          0 => '0');

  wait for CLOCK_PERIOD;

  pl <= '0';
  en <= '1';
```

```
  shft <= '1';
  si<='1';
  for i in 0 to 3 loop
          wait for CLOCK_PERIOD*2;
  end loop;

  shft <= '0';

  for i in 0 to 3 loop
          wait for CLOCK_PERIOD*2;
  end loop;

  en<='0';

  wait for CLOCK_PERIOD;

  wait;
  end process;

  generate_clock : process
  begin
    clk <= '0';
    wait for CLOCK_PERIOD/2;
    clk <= '1';
    wait for CLOCK_PERIOD/2;
  end process;

end architecture;
```

And finally, three snapshots of the simulation:







Topic B3)

Note: In the following descriptions, the output variable cout takes the value 1 only if the current output of the meter is equal to the maximum value of the meter. In all other cases it is 0.

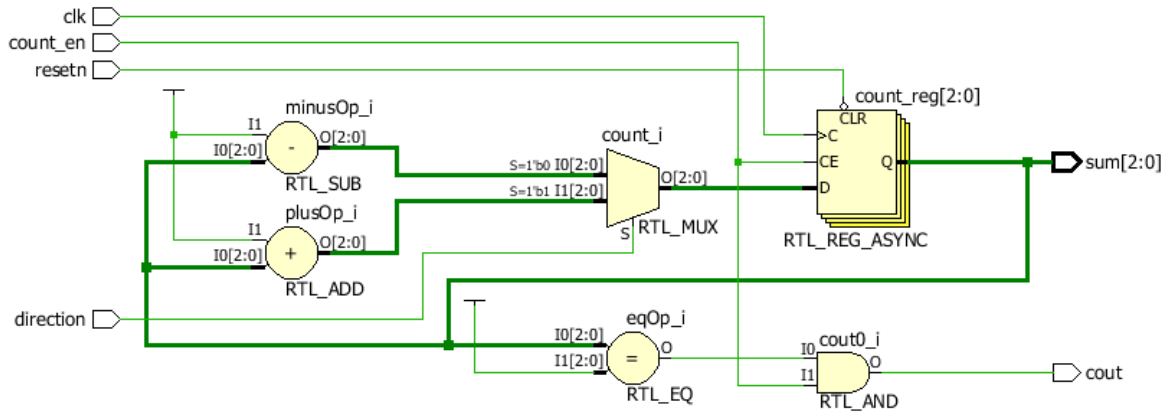
Up / Down Counter:

Behavioural Architecture:

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

entity ex3_count3updown is
    Port ( clk : in std_logic;
           count_en : in std_logic;
           resetn : in std_logic;
           direction : in std_logic;
           cout : out std_logic;
           sum : out std_logic_vector (2 downto 0));
end ex3_count3updown;

architecture behavioural of ex3_count3updown is
signal count: std_logic_vector(2 downto 0);
begin

    process(clk, resetn)
    begin
        if resetn='0' then
            -- Code for reset
            count <= (others=>'0');
        elsif clk'event and clk='1' then
            if count_en='1' then
```

```
            -- Count when count_en=1
            case direction is
            --0 for down, 1 for up
               when '0'=>  count<=count-1;
               when '1'=>  count<=count+1;
               when others=>   count <= (others=>'0');
            end case;
         end if;
      end if;
   end process;
   -- Output signals
   sum <= count;
   cout <= '1' when count=7 and count_en='1' else '0';

end behavioural;
```

The schematic:



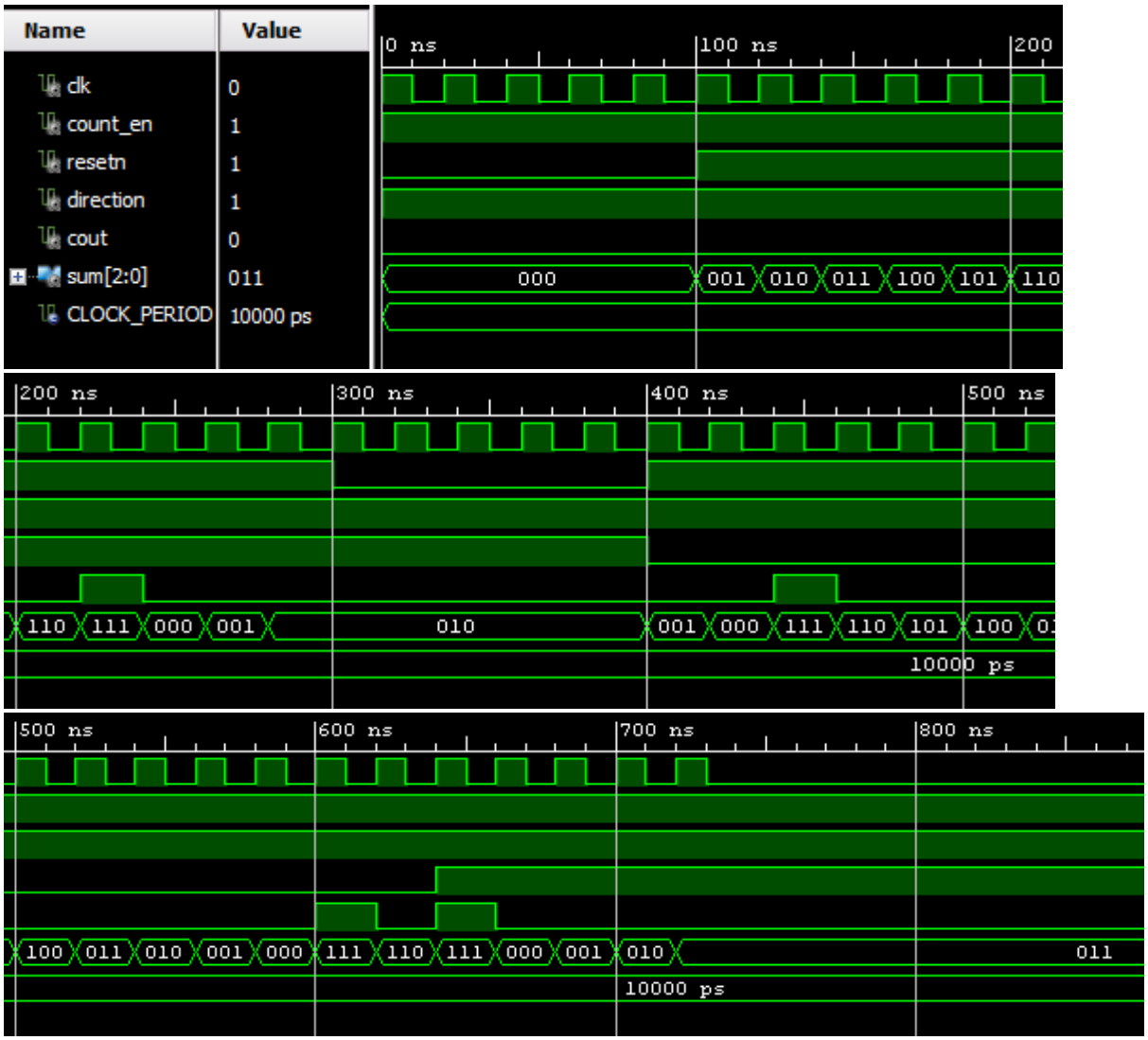
The ***Testbench***:

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

entity ex3_count3updown_tb is
end entity;

architecture bench of ex3_count3updown_tb is

   component ex3_count3updown is
      port ( clk : in std_logic;
             count_en : in std_logic;
             resetn : in std_logic;
             direction : in std_logic;
```

```
            cout : out std_logic;
            sum : out std_logic_vector (2 downto 0));
    end component;

    signal clk : std_logic := '0';
    signal count_en : std_logic := '0';
    signal resetn : std_logic := '0';
    signal direction : std_logic := '0';
    signal cout : std_logic;
    signal sum : std_logic_vector (2 downto 0);

    constant CLOCK_PERIOD : time := 10 ns;

begin

    test : ex3_count3updown
        port map ( clk => clk,
                   count_en => count_en,
                   resetn => resetn,
                   direction => direction,
                   cout => cout,
                   sum => sum
                   );

     stimulus : process
     begin
        --Test reset=0
        count_en <= '1';
        resetn <= '0';
        clk <='1';
        direction <='1';
        for i in 0 to 9 loop
          clk <= not clk;
          wait for CLOCK_PERIOD;
        end loop;

        resetn <= '1';
        --Test count up
        for i in 0 to 19 loop
           clk <= not clk;
           wait for CLOCK_PERIOD;
        end loop;

        count_en <= '0';
        --Test inactive count
        for i in 0 to 9 loop
           clk <= not clk;
           wait for CLOCK_PERIOD;
        end loop;

        --Test count down
        direction <='0';
        count_en <= '1';
```

```
        for i in 0 to 23 loop
          clk <= not clk;
          wait for CLOCK_PERIOD;
        end loop;

        direction <='1';

        for i in 0 to 9 loop
            clk <= not clk;
            wait for CLOCK_PERIOD;
        end loop;

        wait;
    end process;

end bench;
```

And the simulation:

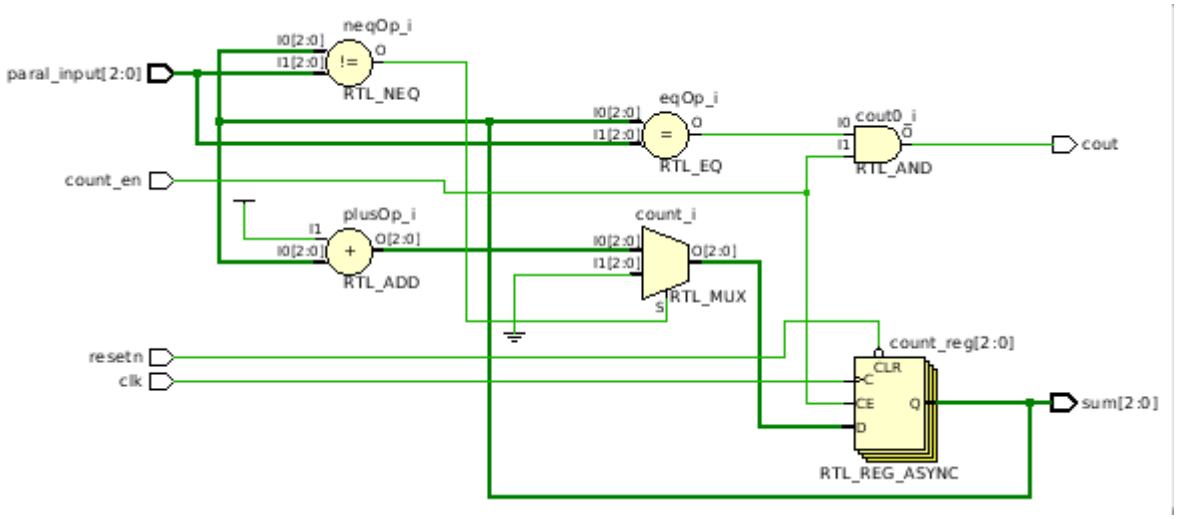


Meter with parallel measurement input:

Behavioural Architecture:

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

entity ex3_count3up_input is
    Port ( clk : in std_logic;
           count_en : in std_logic;
           resetn : in std_logic;
           paral_input : in std_logic_vector (2 downto 0);
           cout : out std_logic;
           sum : out std_logic_vector (2 downto 0));
end ex3_count3up_input;

architecture behavioural of ex3_count3up_input is
signal count: std_logic_vector(2 downto 0);
begin

    process(clk, resetn)
    begin
        if resetn='0' then
            -- Code for reset
            count <= (others=>'0');
        elsif clk'event and clk='1' then
            if count_en='1' then
                -- Count up when count_en = 1
                if count/=paral_input then
                -- Increase counter only when not equal to input
                count<=count+1;
                else
                --Else zero
                count<=(others=>'0');
                end if;
            end if;
        end if;
    end process;
    -- Output signals
    sum <= count;
    cout <= '1' when count=paral_input and count_en='1' else '0';

end behavioural;
```

The schematic:

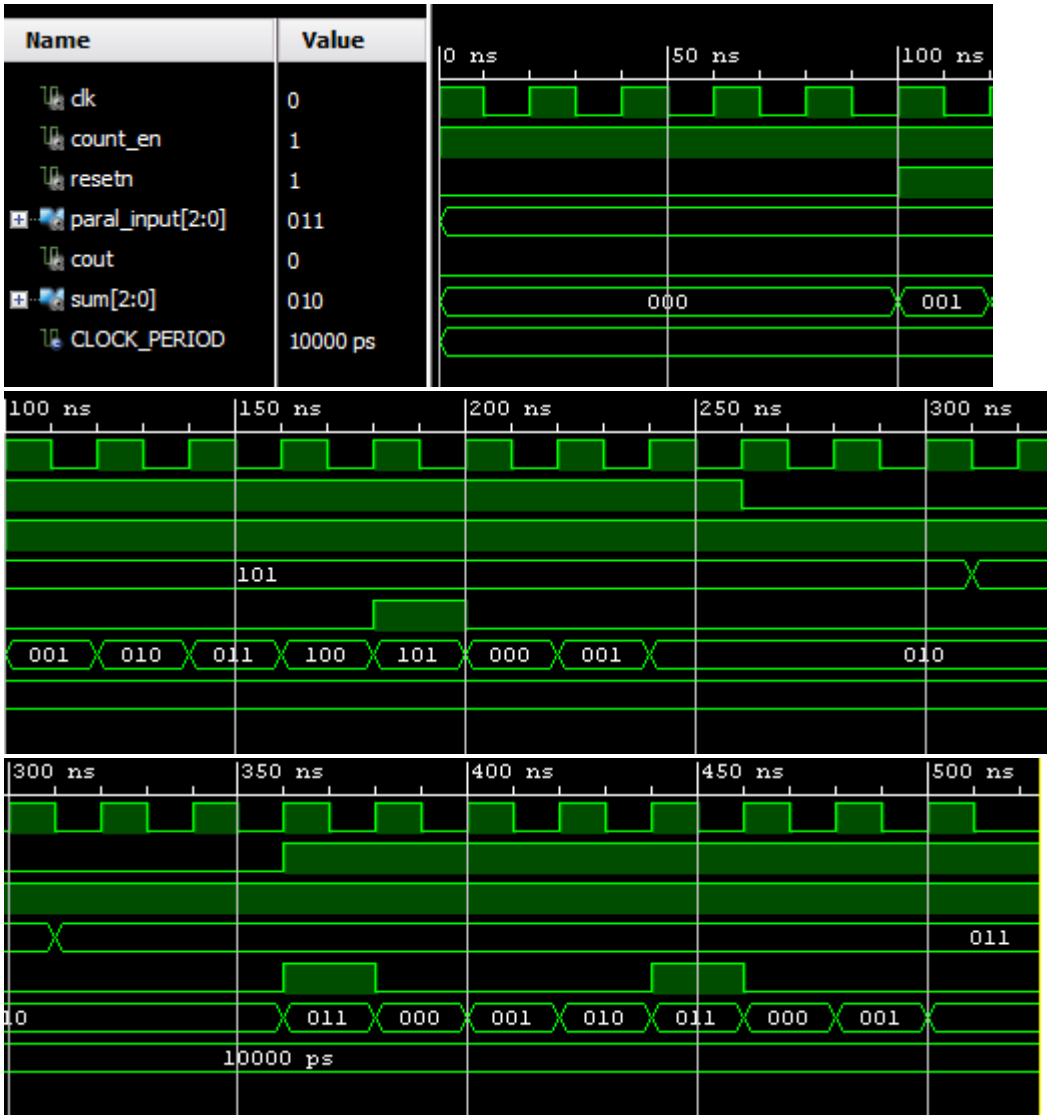


The ***testbench***:

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

entity ex3_count3up_input_tb is
end entity;

architecture bench of ex3_count3up_input_tb is

   component ex3_count3up_input is
      port ( clk : in std_logic;
             count_en : in std_logic;
             resetn : in std_logic;
             paral_input : in std_logic_vector (2 downto 0);
             cout : out std_logic;
             sum : out std_logic_vector (2 downto 0));
   end component;

   signal clk : std_logic := '0';
   signal count_en : std_logic := '0';
   signal resetn : std_logic := '0';
   signal paral_input : std_logic_vector (2 downto 0) := (others => '0');
   signal cout : std_logic;
   signal sum : std_logic_vector (2 downto 0);

   constant CLOCK_PERIOD : time := 10 ns;

begin

   test : ex3_count3up_input
      port map ( clk => clk,
                 count_en => count_en,
                 resetn => resetn,
                 paral_input => paral_input,
```

```
            cout => cout,
            sum => sum
            );

    stimulus : process
    begin
      --Test reset=0
      count_en <= '1';
      resetn <= '0';
      clk <='1';
      paral_input <= "101";
      for i in 0 to 9 loop
        clk <= not clk;
        wait for CLOCK_PERIOD;
      end loop;

      resetn <= '1';

      --Test count up with limit '5'
      for i in 0 to 15 loop
        clk <= not clk;
        wait for CLOCK_PERIOD;
      end loop;

      count_en <= '0';
      --Test inactive count
      for i in 0 to 4 loop
        clk <= not clk;
        wait for CLOCK_PERIOD;
      end loop;
      --Test 'change limit to 3'
      paral_input <= "011";

      for i in 0 to 4 loop
        clk <= not clk;
        wait for CLOCK_PERIOD;
      end loop;

      count_en <= '1';

      for i in 0 to 15 loop
        clk <= not clk;
        wait for CLOCK_PERIOD;
      end loop;

      wait;
    end process;

end bench;
```

The simulation:


Name
Value
clk
0
count_en
1
resetn
1
paral_input[2:0]
011
cout
0
sum[2:0]
010
CLOCK_PERIOD
10000 ps
0 ns
50 ns
100 ns
000
001
100 ns
150 ns
200 ns
250 ns
300 ns
101
001
010
011
100
101
000
001
010
300 ns
350 ns
400 ns
450 ns
500 ns
011
10
011
000
001
010
011
000
001
10000 ps