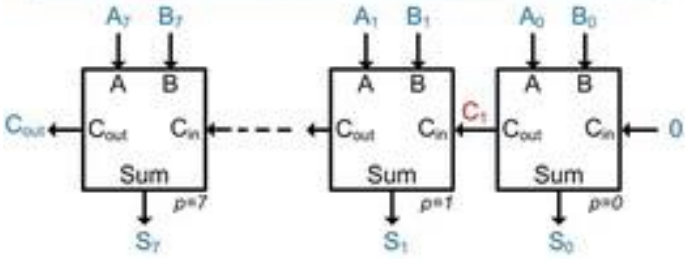
7. Аритметичка логичка кола

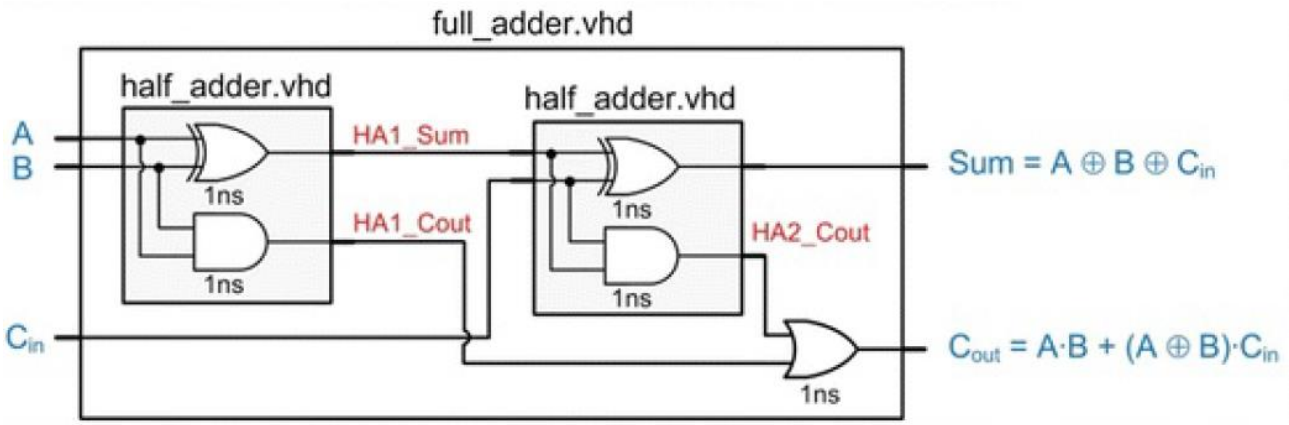
Задатак 12.1.5
Написати VHDL модел понашања за 8-битни Ripple-Carry сабирач (RCA) користећи структурни приступ пројектовања. Ово укључује стварање полусабирача (half_adder.vhd), потпуни сабирач (full_adder.vhd), а затим на крају највиши ниво сабирач (rca.vhd) инстанцирањем осам компоненти потпуних сабирача. Модел таласног кашњења реализовати уметањем 1ns кашњења за кола XOR, AND и OR. Коришћењем доделе сигнала са кашњењем. Општа топологија и дефиниција ентитета за дизајн су приказане на слици 12.4. Направити тест бенч за исцрпну проверу овог модела за све улазе. Тест бенч треба да мења вредности улаза на сваких 30 ns да би било довољно времена да сигнали прођу кроз сабирач.

```
entity rca_8bit is
  port (A, B : in  std_logic_vector(7 downto 0);
        S    : out std_logic_vector(7 downto 0);
        Cout : out std_logic);
end entity;
```



Слика. 12.4 Ентитет 4-битног RCA сабирача

Решење задатка 12.1.5



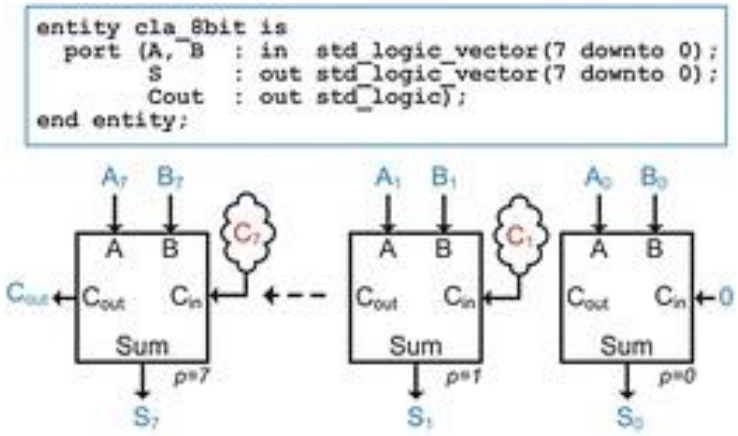

```
-- half_adder.vhd
library ieee;
use ieee.std_logic_1164.all;

entity half_adder is
      port (A, B      : in std_logic;
            Sum, Cout : out std_logic);
end entity;

architecture half_adder_arch of half_adder is
begin
      Sum <= A xor B after 1 ns;
      Cout <= A and B after 1 ns;
end architecture;
```

```
-- full_adder.vhd
library ieee;
use ieee.std_logic_1164.all;

entity full_adder is
      port (A, B, Cin : in std_logic;
            Sum, Cout : out std_logic);
end entity;

architecture full_adder_arch of full_adder is
      component half_adder
            port (A, B : in std_logic;
            Sum, Cout  : out std_logic);
      end component;
      signal HA1_Sum, HA1_Cout, HA2_Cout : std_logic;
begin
      HA1 : half_adder port map (A, B, HA1_Sum, HA1_Cout);
      HA2 : half_adder port map (HA1_Sum, Cin, Sum, HA2_Cout);
      Cout <= HA1_Cout or HA2_Cout after 1 ns;
end architecture;

-- rca.vhd
library ieee;
use ieee.std_logic_1164.all;

entity rca_8bit is
      port (A, B : in std_logic_vector(7 downto 0);
            Sum  : out std_logic_vector(7 downto 0);
            Cout : out std_logic);
end entity;

architecture rca_8bit_arch of rca_8bit is
      component full_adder
            port (A, B, Cin : in std_logic;
                  Sum, Cout : out std_logic);
      end component;

      signal C1, C2, C3, C4, C5, C6, C7 : std_logic;
begin
      A0 : full_adder port map (A(0), B(0), '0', Sum(0), C1);
      A1 : full_adder port map (A(1), B(1), C1, Sum(1), C2);
      A2 : full_adder port map (A(2), B(2), C2, Sum(2), C3);
      A3 : full_adder port map (A(3), B(3), C3, Sum(3), C4);
      A4 : full_adder port map (A(4), B(4), C4, Sum(4), C5);
      A5 : full_adder port map (A(5), B(5), C5, Sum(5), C6);
      A6 : full_adder port map (A(6), B(6), C6, Sum(6), C7);
      A7 : full_adder port map (A(7), B(7), C7, Sum(7), Cout);
end architecture;

-- rca_tb.vhd
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity rca_8bit_TB is
end entity;

architecture rca_8bit_TB_arch of rca_8bit_TB is
      component rca_8bit
            port (A, B : in std_logic_vector(7 downto 0);
                  Sum  : out std_logic_vector(7 downto 0);
                  Cout : out std_logic);
      end component;
      signal A_TB, B_TB, Sum_TB : std_logic_vector(7 downto 0);
      signal Cout_TB            : std_logic;
```

```
begin
      DUT : rca_8bit port map (A_TB, B_TB, Sum_TB, Cout_TB);
      STIM : process
      begin
            for i in 0 to 255 loop
                  for j in 0 to 255 loop
                        A_TB <= std_logic_vector(to_unsigned(i,8));
                        B_TB <= std_logic_vector(to_unsigned(j,8));
                        wait for 30 ns;
                  end loop;
            end loop;
      end process;
end architecture;
```

Задатак 12.1.8

Написати VHDL модел понашања за 8-битни Carry-Look-Ahead сабирач (cla.vhd). Модел треба да инстанцира 8 модификованих потпуних сабирача (mod_full_adder.vhd). Логику за Carry-Look-Ahead сабирач треба да буде имплементирана коришћењем конкурентне доделе сигнала са логичким операторима. Моделовати сваки ниво кашњења кола са 1 ns коришћењем доделе сигнала са кашњењем. Општа топологија и дефиниција ентитета за дизајн су приказане на слици 12.6. Направити тест бенч за исцрпну проверу овог модела за све улазе. Тест бенч треба да мења вредности улаза на сваких 30 ns да би било довољно времена да сигнали прођу кроз сабирач.



Слика. 12.6 Ентитет 8-битног CLA сабирача

Решење задатка 12.1.8

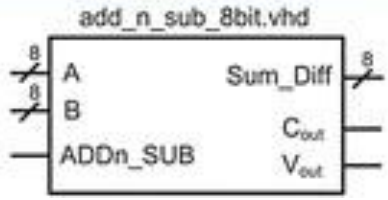
```
-- mod_full_adder.vhd
library ieee;
use ieee.std_logic_1164.all;

entity mod_full_adder is
      port (A, B, Cin : in std_logic;
            Sum, p, g : out std_logic);
end entity;

architecture mod_full_adder_arch of mod_full_adder is
begin
      Sum <= (A xor B xor Cin) after 2 ns;
      p   <= (A or B)          after 1 ns;
      q   <= (A and B)         after 1 ns;
end architecture;

-- cla.vhd
library ieee;
use ieee.std_logic_1164.all;
```

```
entity cla_8bit is
      port (A, B : in std_logic_vector(7 downto 0);
            Sum  : out std_logic_vector(7 downto 0);
            Cout : out std_logic);
end entity;

architecture cla_8bit_arch of cla_8bit is
      component mod_full_adder
            port (A, B, Cin : in std_logic;
                  Sum, p, q : out std_logic);
      end component;

      signal C0, C1, C2, C3, C4, C5, C6, C7 : std_logic;
      signal p, q                           : std_logic_vector(7 downto 0);
begin
      C0   <= '0';
      C1   <= g(0) or (p(0) and C0) after 2 ns;
      C2   <= g(1) or (p(1) and C1) after 2 ns;
      C3   <= g(2) or (p(2) and C2) after 2 ns;
      C4   <= g(3) or (p(3) and C3) after 2 ns;
      C5   <= g(4) or (p(4) and C4) after 2 ns;
      C6   <= g(5) or (p(5) and C5) after 2 ns;
      C7   <= g(6) or (p(6) and C6) after 2 ns;
      Cout <= g(7) or (p(7) and C7) after 2 ns;

      A0 : mod_full_adder port map (A(0), B(0), C0, Sum(0), p(0), g(0));
      A1 : mod_full_adder port map (A(1), B(1), C1, Sum(1), p(1), g(1));
      A2 : mod_full_adder port map (A(2), B(2), C2, Sum(2), p(2), g(2));
      A3 : mod_full_adder port map (A(3), B(3), C3, Sum(3), p(3), g(3));
      A4 : mod_full_adder port map (A(4), B(4), C4, Sum(4), p(4), g(4));
      A5 : mod_full_adder port map (A(5), B(5), C5, Sum(5), p(5), g(5));
      A6 : mod_full_adder port map (A(6), B(6), C6, Sum(6), p(6), g(6));
      A7 : mod_full_adder port map (A(7), B(7), C7, Sum(7), p(7), g(7));
end architecture;

-- cla_tb.vhd
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity cla_8bit_TB is
end entity;

architecture cla_8bit_TB_arch of cla_8bit_TB is
      component cla_8bit
            port (A, B : in std_logic_vector(7 downto 0);
                  Sum  : out std_logic_vector(7 downto 0);
                  Cout : out std_logic);
      end component;
      signal A_TB, B_TB, Sum_TB : std_logic_vector(7 downto 0);
      signal Cout_TB            : std_logic;
begin
      DUT : cla_8bit port map (A_TB, B_TB, Sum_TB, Cout_TB);
      STIM : process
      begin
            for i in 0 to 255 loop
                  for j in 0 to 255 loop
                        A_TB <= std_logic_vector(to_unsigned(i,8));
                        B_TB <= std_logic_vector(to_unsigned(j,8));
                        wait for 30 ns;
                  end loop;
            end loop;
      end process;
end architecture;
```

Задатак 12.2.4

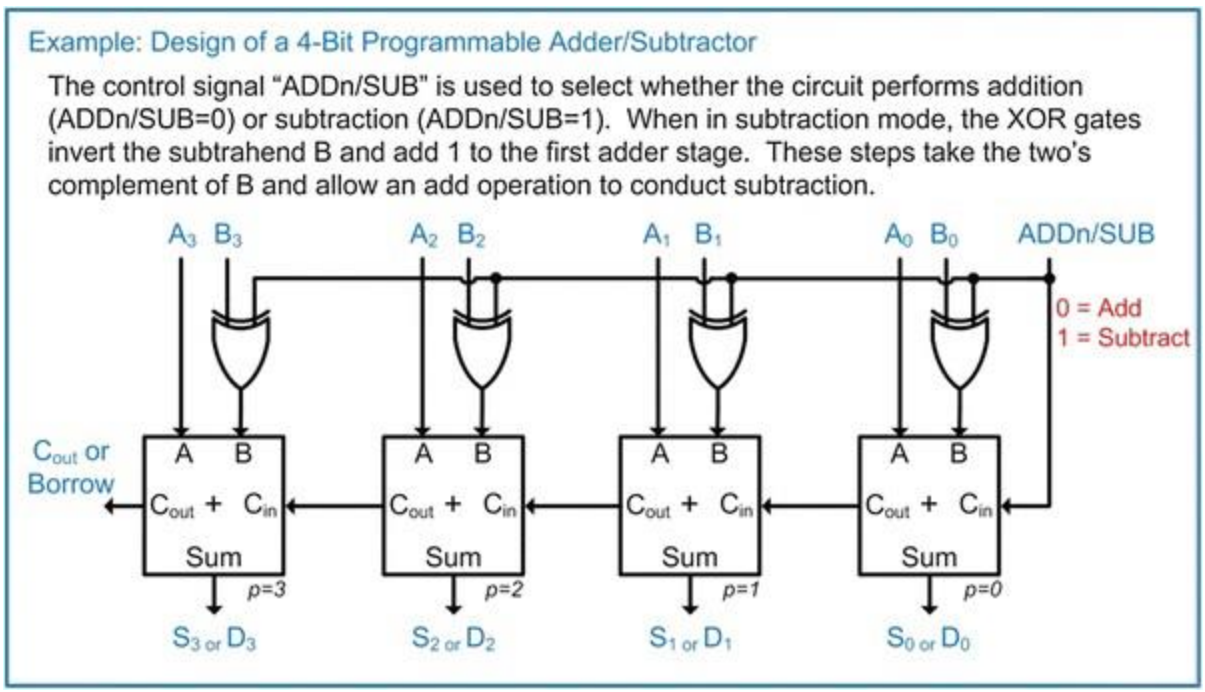
Написати VHDL модел понашања за 8-битни програмабилни сабирач/одузимач. Модел треба да има улаз “ADDn_SUB” који треба да управља да ли се систем понаша као сабирач (0) или као одузимач (1). Модел треба да ради са потпуно комплементираним означеним бројевима. Резултат операција ће бити на порту “Sum_Diff”. Модел треба да поставља излаз “Cout” када сабирање креира пренос или када одузимање креира позајмицу. Коло такође треба да постави излаз Vout када има прекорачења код резултата операција у двоструком комплементу. Дефиниција ентитета за дизајн је приказана на слици 12.8. Направити тест бенч за исцрпну проверу овог модела за све улазе.

```
entity add_n_sub_8bit is
  port (A, B      : in  std_logic_vector(7 downto 0);
        ADDn_SUB  : in  std_logic;
        Sum_Diff  : out std_logic_vector(7 downto 0);
        Cout_Bout : out std_logic;
        Vout      : out std_logic);
end entity;
```



Слика. 12.6 Ентитет програмабилног сабирача/одузимача

Решење задатка 12.1.8




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

entity half_adder is
      port (A, B      : in std_logic;
            Sum, Cout : out std_logic);
end entity;

architecture half_adder_arch of half_adder is
begin
      Sum <= A xor B after 1 ns;
      Cout <= A and B after 1 ns;
```

```
end architecture;


-- full_adder.vhd
library ieee;
use ieee.std_logic_1164.all;

entity full_adder is
      port (A, B, Cin : in std_logic;
            Sum, Cout : out std_logic);
end entity;

architecture full_adder_arch of full_adder is
      component half_adder
            port (A, B : in std_logic;
            Sum, Cout  : out std_logic);
      end component;
      signal HA1_Sum, HA1_Cout, HA2_Cout : std_logic;
begin
      HA1 : half_adder port map (A, B, HA1_Sum, HA1_Cout);
      HA2 : half_adder port map (HA1_Sum, Cin, Sum, HA2_Cout);
      Cout <= HA1_Cout or HA2_Cout after 1 ns;
end architecture;

-- add_sub.vhd
library ieee;
use ieee.std_logic_1164.all;

entity add_n_sub_8bit is
      port (A, B      : in std_logic_vector(7 downto 0);
            ADDn_SUB  : in std_logic;
            Sum_Diff  : out std_logic_vector(7 downto 0);
            Cout_Bout : out std_logic;
            Vout      : out std_logic);
end entity;

architecture add_n_sub_8bit_arch of add_n_sub_8bit is
      component full_adder
            port (A, B, Cin : in std_logic;
                  Sum, Cout : out std_logic);
      end component;

      signal C1, C2, C3, C4, C5, C6, C7 : std_logic;
      signal BC                         : std_logic_vector(7 downto 0);
      signal i                          : integer;
begin
      for i in 0 to 7 loop
            BC(i) <= ADDn_SUB xor B(i);
      end loop;

      A0 : full_adder port map (A(0), BC(0), ADDn_SUB, Sum_Diff(0), C1);
      A1 : full_adder port map (A(1), BC(1), C1, Sum_Diff(1), C2);
      A2 : full_adder port map (A(2), BC(2), C2, Sum_Diff(2), C3);
      A3 : full_adder port map (A(3), BC(3), C3, Sum_Diff(3), C4);
      A4 : full_adder port map (A(4), BC(4), C4, Sum_Diff(4), C5);
      A5 : full_adder port map (A(5), BC(5), C5, Sum_Diff(5), C6);
      A6 : full_adder port map (A(6), BC(6), C6, Sum_Diff(6), C7);
      A7 : full_adder port map (A(7), BC(7), C7, Sum_Diff(7), Cout_Bout);
      Vout = C7 xor Cout_Bout;
end architecture;

-- add_sub_tb.vhd
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
```

```
entity add_n_sub_8bit_TB is
end entity;

architecture add_n_sub_8bit_TB_arch of add_n_sub_8bit_TB is
      component add_n_sub_8bit
            port (A, B      : std_logic_vector(7 downto 0);
                  ADDn_SUB  : std_logic;
                  Sum_Diff  : std_logic_vector(7 downto 0);
                  Cout_Bout : std_logic;
                  Vout      : std_logic);
      end component;
      signal A_TB, B_TB, Sum_Diff_TB : std_logic_vector(7 downto 0);
      signal ADDn_SUB_TB, Cout_Bout_TB, Vout_TB : std_logic;
begin
      DUT : add_n_sub port map (A_TB, B_TB, ADDn_SUB_TB, Sum_Diff_TB, Cout_Bout _TB,
Vout_TB);
      STIM : process
      begin
            for i in 0 to 255 loop
                  for j in 0 to 255 loop
                        A_TB <= std_logic_vector(to_unsigned(i,8));
                        B_TB <= std_logic_vector(to_unsigned(j,8));
                        wait for 30 ns;
                  end loop;
            end loop;
      end process;
end architecture;
```