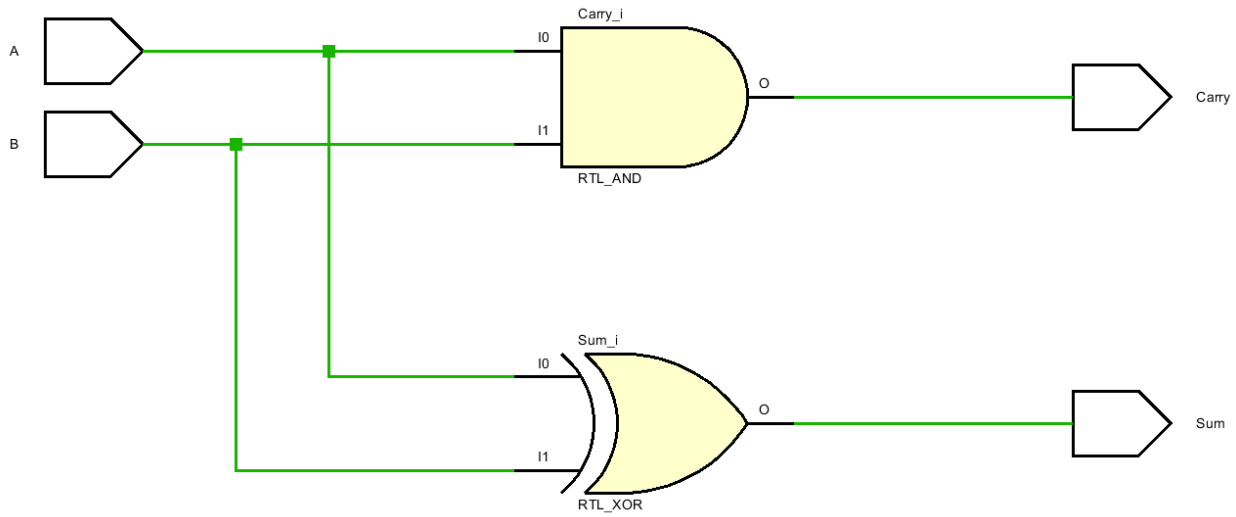
Carry Look Ahead Adder Subtractor

Half Adder – Design Source File

```
----------------------------------------------------------------------------------

-- Group 17
-- Senevirathne S.M.P.U.
--
-- Create Date: 04/16/2024 12:02:05 PM
-- Design Name: Carry_Look_Ahead_Adder_Subtractor
-- Module Name: Half_Adder - Behavioral
-- Project Name: Nanoprocessor
-- Target Devices: Basys3 Board

----------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Half_Adder is
    Port ( A : in STD_LOGIC; --First input bit
           B : in STD_LOGIC; --Second input bit
           Sum : out STD_LOGIC; --Sum output bit
```

```
        Carry : out STD_LOGIC); --Carry output bit
end Half_Adder;

architecture Behavioral of Half_Adder is

begin
   --Defining sum and carry of the half adder
   Sum <= A XOR B;
   Carry <= A AND B;
end Behavioral;
```

Half Adder – Design Schematic



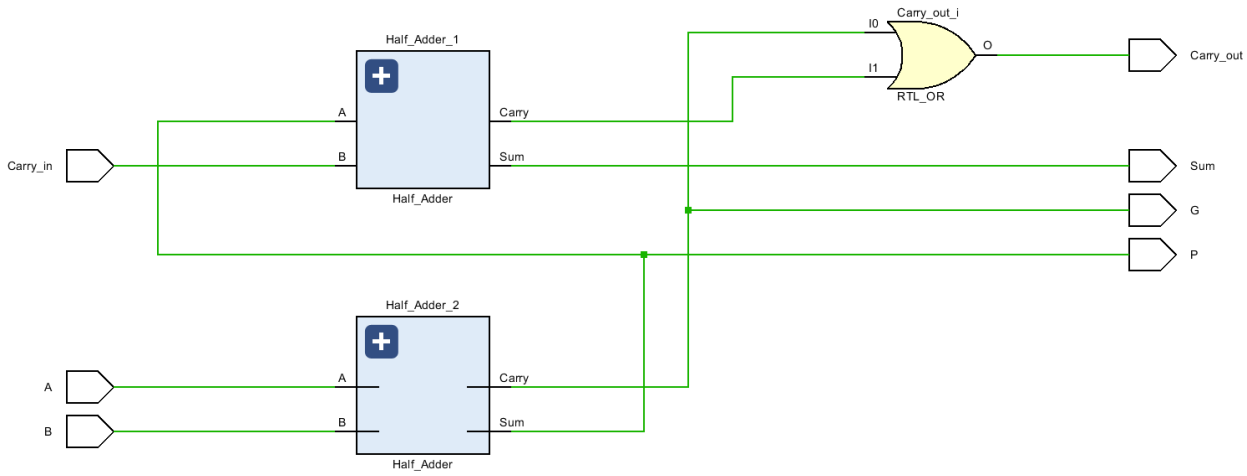
Full Adder – Design Source File

```
-- Group 17
-- Senevirathne S.M.P.U.
--
-- Create Date: 04/16/2024 12:27:08 PM
-- Design Name: Carry_Look_Ahead_Adder_Subtractor
-- Module Name: Full_Adder - Behavioral
-- Project Name: Nanoprocessor
-- Target Devices: Basys3 Board

----------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Full_Adder is
   Port ( A : in STD_LOGIC; --First input bit
          B : in STD_LOGIC; --Second input bit
          Carry_in : in STD_LOGIC; --Carry input bit
          Sum : out STD_LOGIC; --Sum output bit
          Carry_out : out STD_LOGIC; --Carry output bit
          P : out STD_LOGIC; --Propergate output bit
          G : out STD_LOGIC); --Generate output bit
end Full_Adder;

architecture Behavioral of Full_Adder is
component Half_Adder
   Port ( A : in STD_LOGIC; --First input bit
     B : in STD_LOGIC; --Second input bit
     Sum : out STD_LOGIC; --Sum output bit
```

```
    Carry : out STD_LOGIC); --Carry output bit
end component;

SIGNAL A1, A2, B1, B2: std_logic; --signal inputs of the two half adders
SIGNAL Sum1, Sum2, Carry1, Carry2: std_logic; --signal outputs of the two half adders
begin
  Half_Adder_2 : Half_Adder --mapping the bottom half adder
  port map(
    A => A2,
    B => B2,
    Sum => Sum2,
    Carry => Carry2
  );
  Half_Adder_1 : Half_Adder --mapping the top half adder
  port map(
    A => A1,
    B => B1,
    Sum => Sum1,
    Carry => Carry1
  );
--Assigning the inputs to the bottom half adder
A2 <= A;
B2 <= B;

--Defining generation and propergation bits
P <= Sum2;
G <= Carry2;

--Assigning the inputs to the top half adder
```

```
B1 <= Carry_in;
A1 <= Sum2;

--Defining Carry_out and sum
Carry_out <= Carry2 OR Carry1;
Sum <= Sum1;
end Behavioral;
```

Full Adder – Design Schematic

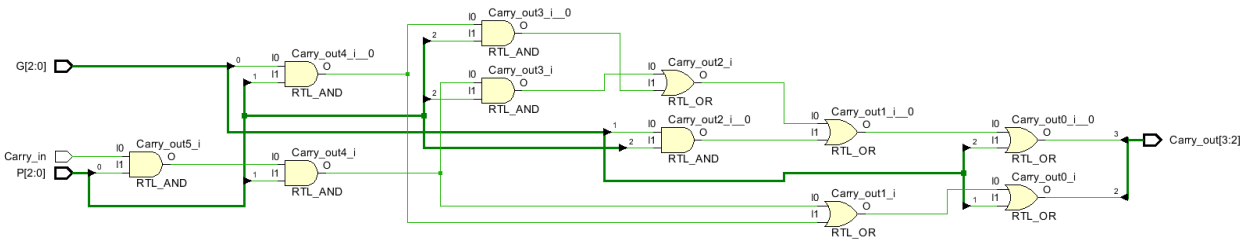


Carry Look Ahead Logic – Design Source File

```
----------------------------------------------------------------------------------

-- Group 17
-- Senevirathne S.M.P.U.
--
-- Create Date: 04/16/2024 01:35:19 PM

Senevirathne S.M.P.U. – 220599M
```

```
-- Design Name: Carry_Look_Ahead_Adder_Subtractor
-- Module Name: Carry_Look_Ahead - Behavioral
-- Project Name: Nanoprocessor
-- Target Devices: Basys3 Board

----------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Carry_Look_Ahead is
   Port ( P : in STD_LOGIC_VECTOR (2 downto 0); --Propergate input bus
        G : in STD_LOGIC_VECTOR (2 downto 0); --Propergate output bus
        Carry_in : in STD_LOGIC; --Carry input bit
        Carry_out : out STD_LOGIC_VECTOR (3 downto 2)); --Carry output bus
end Carry_Look_Ahead;

architecture Behavioral of Carry_Look_Ahead is

begin
--Defining Carry_out(2)
Carry_out(2) <= (Carry_in AND P(0) AND P(1)) OR (G(0) AND P(1)) OR G(1);
--Defining Carry_out(1)
Carry_out(3) <= (Carry_in AND P(0) AND P(1) AND P(2)) OR (G(0) AND P(1) AND P(2)) OR (G(1) AND
P(2)) OR G(2);
end Behavioral;
```

Carry Look Ahead Logic – Design Schematic



Carry Look Ahead Adder Subtractor – Design Source File

-- Group 17

-- Senevirathne S.M.P.U.

--

-- Create Date: 04/16/2024 02:13:42 PM

-- Design Name: Carry_Look_Ahead_Adder_Subtractor

-- Module Name: Carry_Look_Ahead_Adder_Subtractor - Behavioral

-- Project Name: Nanoprocessor

-- Target Devices: Basys3 Board

---

library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

entity Carry_Look_Ahead_Adder_Subtractor is

    Port (A : in STD_LOGIC_VECTOR (3 downto 0); --Bus for the first binary number

```
        B : in STD_LOGIC_VECTOR (3 downto 0); --Bus for the second binary number
        Add_Sub : in STD_LOGIC; --Should we add B to A or should we subtract B from A? (1 for subtraction, 0 for addition)
        EN : in STD_LOGIC; --Enable input
        S : inout STD_LOGIC_VECTOR (3 downto 0); --Bus for the binary sum output
        Sign : inout STD_LOGIC; --Sign Bit: 1 for negative, 0 for positive
        Overflow : inout STD_LOGIC; --Overflow bit: Is there an overflow?
        --When the number is negative, is the number in range [-0,-8]: if yes, there is no overflow => Overflow =0; if no, there is an overflow => Overflow = 1
        --When the number is positive, is the number in range [+0, 7]: if yes, there is no overflow => Overflow =0; if no, there is an overflow => Overflow = 1
        Zero : out STD_LOGIC); --Is the output zero(0000)?
end Carry_Look_Ahead_Adder_Subtractor;

architecture Behavioral of Carry_Look_Ahead_Adder_Subtractor is
component Full_Adder is
    Port ( A : in STD_LOGIC; --First input bit
        B : in STD_LOGIC; --Second input bit
        Carry_in : in STD_LOGIC; --Carry input bit
        Sum : out STD_LOGIC; --Sum output bit
        Carry_out : out STD_LOGIC; --Carry output bit
        P : out STD_LOGIC; --Propergate output bit
        G : out STD_LOGIC); --Generate output bit
end component;

component Carry_Look_Ahead is
    Port ( P : in STD_LOGIC_VECTOR (2 downto 0); --Propergate input bus
        G : in STD_LOGIC_VECTOR (2 downto 0); --Propergate output bus
        Carry_in : in STD_LOGIC; --Carry input bit
        Carry_out : out STD_LOGIC_VECTOR (3 downto 2)); --Carry output bus
```

```
end component;

--Signal the inputs and outputs of internal components
SIGNAL A0, B0, C1, P0, G0: std_logic; --FA0
SIGNAL A1, B1, P1, G1: std_logic; --FA1
SIGNAL A2, B2, C2, P2, G2: std_logic; --FA2
SIGNAL A3, B3, C3: std_logic; --FA3

--An and Bn are n th bits of the binary numbers A and B inserted to the relevant(n th) Full Adder
--Cn is the carry bit generated by adding the n th bits of A and B
--Pn and Gn are propergation and generation bits of the relevant(n th) Full Adder
begin
  Full_Adder_0 : Full_Adder --mapping first full adder
    port map(A => A0,
    B => B0,
    Carry_in => Add_Sub, --C0
    Sum => S(0),
    Carry_out => C1,
    P => P0,
    G => G0
  );
  Full_Adder_1 :Full_Adder --mapping second full adder
    port map(A => A1,
    B => B1,
    Carry_in => C1,
    Sum => S(1),
    P => P1,
    G => G1
  );
```

```
  Full_Adder_2 :Full_Adder --mapping third full adder
    port map(A => A2,
    B => B2,
    Carry_in => C2,
    Sum => S(2),
    P => P2,
    G => G2
  );
  Full_Adder_3 :Full_Adder --mapping last full adder
    port map(A => A3,
    B => B3,
    Carry_in => C3,
    Sum => S(3),
    Carry_out => Sign --carry output of the Carry Look Ahead Adder Subtractor
  );
  Carry_Look_Ahead_0 : Carry_Look_Ahead --mapping carry look ahead logic unit
    port map(P(0) => P0, P(1) => P1, P(2) => P2,
    G(0) => G0, G(1) => G1, G(2) => G2,
    Carry_in => Add_Sub, --C0
    Carry_out(2) => C2, Carry_out(3) => C3
  );

--Defining inputs of FA0 (A0, B0)
A0 <= A(0) AND EN;
B0 <= (EN AND B(0)) XOR Add_Sub;

--Defining inputs of FA1 (A1, B1)
A1 <= A(1) AND EN;
B1 <= (EN AND B(1)) XOR Add_Sub;
```

--Defining inputs of FA2 (A2, B2)

A2 <= A(2) AND EN;

B2 <= (EN AND B(2)) XOR Add_Sub;

--Defining inputs of FA3 (A3, B3)

A3 <= A(3) AND EN;

B3 <= (EN AND B(3)) XOR Add_Sub;

--Defining the overflow bit

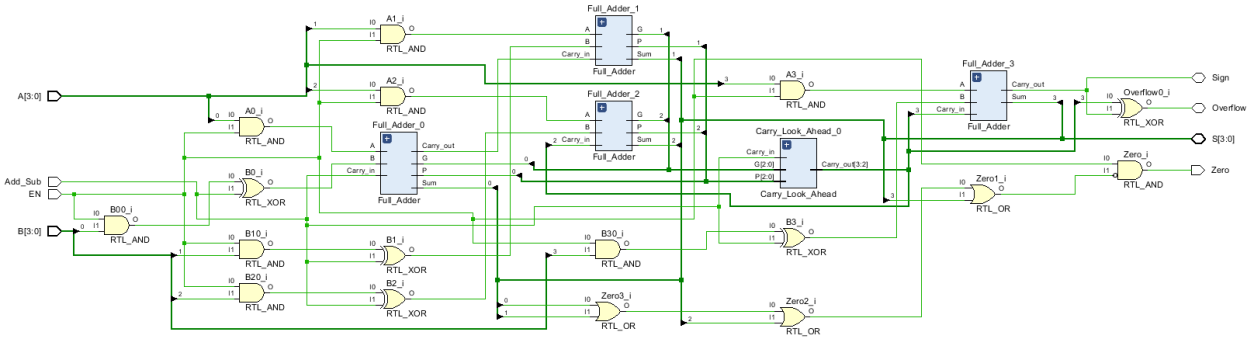
Overflow <= C3 XOR Sign;

--Defining the zero flag

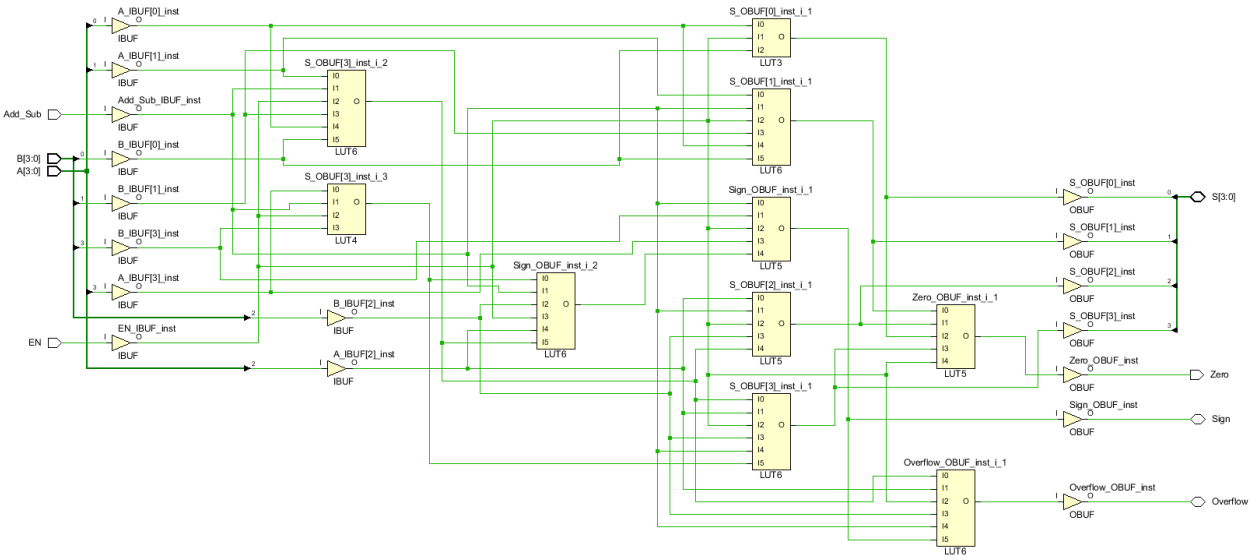
Zero <= EN AND NOT(S(0) OR S(1) OR S(2) OR S(3));

end Behavioral;

Carry Look Ahead Adder Subtractor – Design Schematic



Carry Look Ahead Adder Subtractor – Implemented Design Schematic



Carry Look Ahead Adder Subtractor – Simulation File

-- Group 17

-- Senevirathne S.M.P.U.

--

-- Create Date: 04/16/2024 04:38:31 PM

-- Design Name: Carry_Look_Ahead_Adder_Subtractor

-- Module Name: Carry_Look_Ahead_Adder_Subtractor_TB - Behavioral

-- Project Name: Nanoprocessor

-- Target Devices: Basys3 Board

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

entity Carry_Look_Ahead_Adder_Subtractor_TB is
end Carry_Look_Ahead_Adder_Subtractor_TB;

architecture Behavioral of Carry_Look_Ahead_Adder_Subtractor_TB is
  component Carry_Look_Ahead_Adder_Subtractor is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0); --Bus for the first binary number
        B : in STD_LOGIC_VECTOR (3 downto 0); --Bus for the second binary number
        Add_Sub : in STD_LOGIC; --Should we add B to A or should we subtract B from A? (1 for subtraction, 0 for addition)
        EN : in STD_LOGIC; --Enable input
        S : inout STD_LOGIC_VECTOR (3 downto 0); --Bus for the binary sum output
        Sign : inout STD_LOGIC; --Sign Bit: 1 for negative, 0 for positive
        Overflow : inout STD_LOGIC; --Overflow bit: Is there an overflow?
        --When the number is negative, is the number in range [-0,-8]: if yes, there is no overflow => Overflow =0; if no, there is an overflow => Overflow = 1
        --When the number is positive, is the number in range [+0, 7]: if yes, there is no overflow => Overflow =0; if no, there is an overflow => Overflow = 1
        Zero : out STD_LOGIC); --Is the output zero(0000)?
  end component;

  --Signal the inputs and outputs of Carry Look Ahead Adder Subtractor
  SIGNAL A, B, S : std_logic_vector(3 downto 0);
  SIGNAL Add_Sub, EN, Sign, Overflow, Zero: std_logic;
begin
  Carry_Look_Ahead_Adder_Subtractor_Sim : Carry_Look_Ahead_Adder_Subtractor --mapping Carry Look Ahead Adder Subtractor for simulation
```

```
    port map(
      A => A,
      B => B,
      Add_Sub => Add_Sub,
      EN => EN,
      S => S,
      Sign => Sign,
      Overflow => Overflow,
      Zero => Zero
    );

  process begin
    wait for 100ns;
    EN <= '1';
    Add_Sub <= '0';
    A <= "0000";
    B <= "0000"; -- 0+0 = 0000(0), Carry = 0 (+0), Overflow = 0(C=0, positive; O=0, No overflow => in range [+0,7] )

    wait for 100ns;
    A <= "1001";
    B <= "0010"; -- -7+(2) = -5 = 1011(d), Carry =1, Overflow =0 (C=1, negative; O= 1, No overflow => in range [-0,-8] )

    wait for 100ns;
    A <= "0110";
    B <= "0110"; -- 6+(6) = 12 = 1100(c), Carry =0, Overflow =1 (C=0, positive; O= 0, overflowed positively => Above 7 )

    wait for 100ns;
```

```
A <= "0001";
B <= "0010"; -- 1+2 = 3 = 0011(3), Carry = 0, Overflow = 0(C=0, positive; O=0, No overflow => in range [+0,7] )

wait for 100ns;
A <= "1000";
B <= "0111"; -- -8+7 = -1 = 1111(f), Carry = 1, Overflow = 0(C=1, negative; O=1, No overflow => in range [-0,-8] )

wait for 100ns;
A <= "0111";
B <= "1000"; -- 7+(-8) = -1 = 1111(f), Carry = 1, Overflow = 0(C=1, negative; O=1, No overflow => in range [-0,-8] )

wait for 100ns;
A <= "0111";
B <= "1001"; -- 7+(-7) = 0 = 0000(0), Carry =0 (+0), Overflow =0 (C=0, positive; O= 0, No overflow => in range [+0,7] )

wait for 100ns;
A <= "0100";
B <= "1101"; -- 4+(-3) = 1 = 0001(1), Carry =0, Overflow =0 (C=0, positive; O= 0, No overflow => in range [+0,7] )

wait for 100ns;
A <= "0100";
B <= "1100"; -- 4+(-4) = 0 = 0000(0), Carry =0 (+0), Overflow =0 (C=0, positive; O= 0, No overflow => in range [+0,7] )

wait for 100ns;
A <= "1001";
```

```
    B <= "0001"; -- -7+(1) = -6 = 1010(a), Carry =1, Overflow =0 (C=1, negative; O= 0, No overflow => in range [-0,-8] )

    wait for 100ns;

    A <= "1001";

    B <= "0010"; -- -7+(2) = -5 = 1011(b), Carry =1, Overflow =0 (C=1, negative; O= 0, No overflow => in range [-0,-8] )

    wait for 100ns;

    Add_Sub <= '1';

    A <= "0000";

    B <= "0000"; -- 0-0 = 0000(0), Carry = 1 (-0), Overflow = 0(C=1, negative; O=0, No overflow => in range [-0,-8])

    wait for 100ns;

    A <= "1001";

    B <= "0010"; -- -7-(2) = -9 = 0111(7), Carry =1, Overflow =1 (C=1, negative; O= 1, overflowed negatively => Below -8 )

    wait for 100ns;

    A <= "0110";

    B <= "0110"; -- 6-(6) = 0 = 0000(0), Carry =1 (-0), Overflow =0 (C=1, negative; O= 0, No overflow => in range[-0-8] )

    wait for 100ns;

    A <= "0001";

    B <= "0010"; -- 1-2 = -1 = 1111(f), Carry = 1, Overflow = 0(C=1, negative; O=0, No overflow => in range [-0,-8])

    wait for 100ns;

    A <= "1000";
```

B <= "0111"; -- -8-7 = -15 = 0001(1), Carry = 1, Overflow = 1(C=1, negative; O=1, overflowed negatively => Below -8)

wait for 100ns;

A <= "0111";

B <= "1000"; -- 7-(-8) = 15 = 1111(f), Carry =0, Overflow = 1 (C=0, positive; O=1, overflowed positively => Above 7)

wait for 100ns;

A <= "0111";

B <= "1001"; -- 7-(-7) = 14 = 1110(e), Carry =0, Overflow =1 (C=0, positive; O= 1, overflowed positively => Above 7)

wait for 100ns;

A <= "0100";

B <= "1101"; -- 4-(-3) = 7 = 0111(7), Carry =0, Overflow =0 (C=0, positive; O= 0, No overflow => in range [+0,7])

wait for 100ns;

A <= "0100";

B <= "1100"; -- 4-(-4) = 8 = 1000(8), Carry =0, Overflow =1 (C=0, positive; O= 0, overflowed positively => Above 7)

wait for 100ns;

A <= "1001";

B <= "0001"; -- -7-(1) = -8 = 1000(8), Carry =1, Overflow =0 (C=1, negative; O= 0, No overflow => in range [-0,-8])

wait for 100ns;

A <= "1001";

```
    B <= "0010"; -- -7-(2) = -9 = 0111(7), Carry =1, Overflow =1 (C=1, negative; O= 1, overflowed negatively => Below -8 )

    wait;

  end process;


end Behavioral;
```

Carry Look Ahead Adder Subtractor – Timing Diagram