4 bit Subtractor

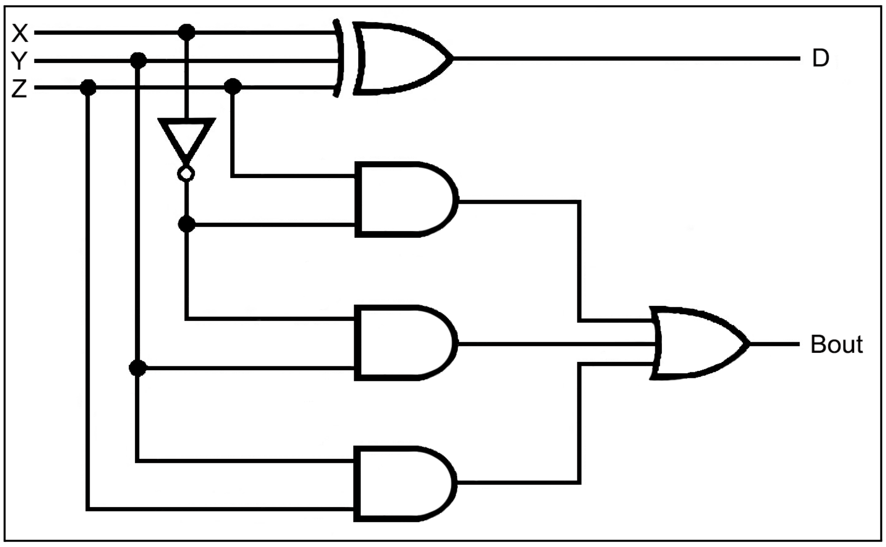
> Full subtractor:

D = X xor Y xor Z
Bout/borrow = X’Z or X’Y or YZ

Truth Table

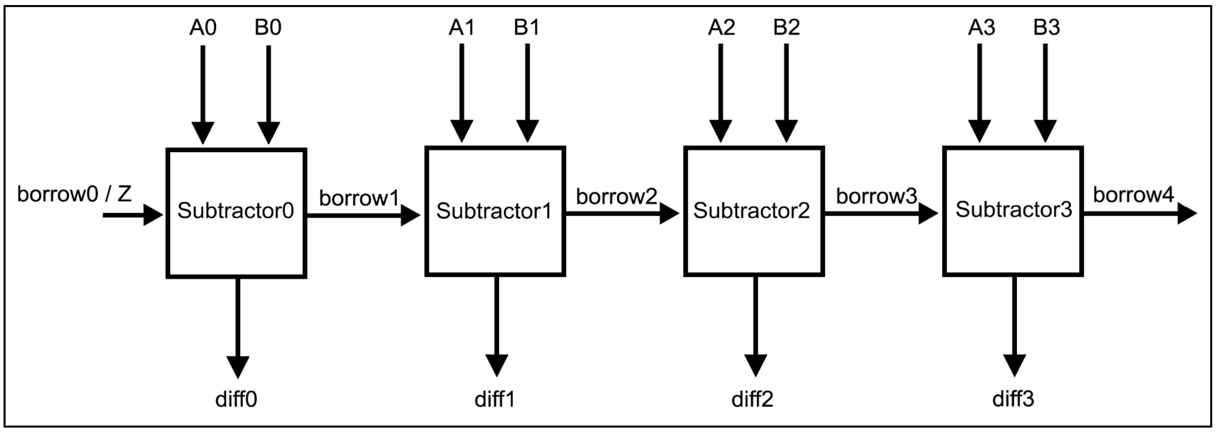
X	Y	Z	D	Bout / borrow
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Full subtractor Circuit:



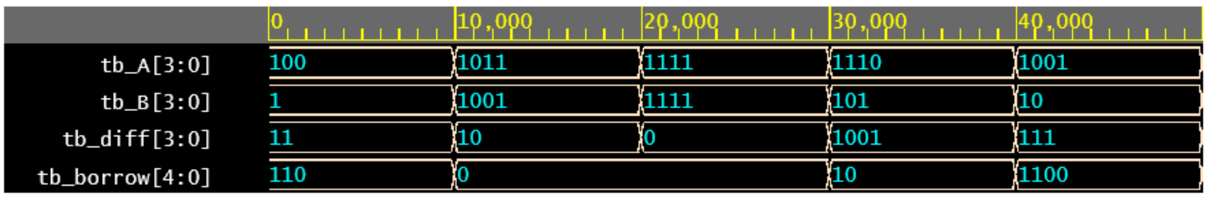
To create a 4-bit subtractor, a connection of 4 complete subtractors is needed. Each subtractor performs a two-bit subtraction taking into account the borrowing of the previous least significant bit. The subtractors take inputs A0-B0, A1-B1, A2-B3, A3-B0 respectively, with subtractor 0 taking Z as its third input, and each subsequent subtractor taking the borrowing of the previous subtractor as its third input and the total difference to be induced with the diff.

> 4 bit Subtractor circuit:



> Demonstration with 5 subtractions:
0ns : 0100 - 0001 with expected difference = 0011
10ns: 1011 - 1001 with expected difference = 0010
20ns: 1111 - 1111 with expected difference = 0000
30ns: 1110 - 0101 with expected difference = 1001
40ns: 1001 - 0010 with expected difference = 0111
In all 3 circuits/models the results of the simulation are as follows:

Run Time: 50ns


0
10,000
20,000
30,000
40,000
tb_A[3:0] 100 1011 1111 1110 1001
tb_B[3:0] 1 1001 1111 101 10
tb_diff[3:0] 11 10 0 1001 111
tb_borrow[4:0] 110 0 10 1100

Circuit Implementations:

*The testbench remains the same for all three models.

> DataFlow Model:

```
library IEEE;
use IEEE.std_logic_1164.all;

-- Create full subtractor
entity Subtractor1_flow is
    port (X,Y,Z: in bit ;
            D,bout: out bit);
end Subtractor1_flow;

architecture dataflow of Subtractor1_flow is
begin
    D <= X xor Y xor Z;
    bout <= (not (X) and Z) or (not (X) and Y) or (Y and Z);
end dataflow;

-- Create 4bit subtractor
entity Subtractor4bit is port (
            A, B: in bit_vector (3 downto 0); -- Δήλωση αριθμών για αφαίρεση
            diff: out bit_vector (3 downto 0);  -- Η διαφορά των αριθμών
            borrow: inout bit_vector(4 downto 0)); -- δάνεισμα κάθε αφαιρέτη
end Subtractor4bit;
-- Using Subtractor1_flow to create the 4 bit subtractor
architecture structure of Subtractor4bit is
component Subtractor1_flow
     port (X,Y,Z: in bit ;
            D,bout: out bit);
end component;

begin
  g0: for index in 0 to 3 generate
            u0: Subtractor1_flow port map (
            X=> A(index), Y=> B(index), D=> diff(index), Z=> borrow(index),
            bout=> borrow (index+1));
  end generate;
end structure;
```

> Behavioural/Algorithmic Model:

```
library IEEE;
use IEEE.std_logic_1164.all;

-- Create full subtractor
entity Subtractor1_Struct is
    port (X,Y,Z: in bit ;
            D,bout: out bit);
end Subtractor1_Struct;
architecture algorithmic of Subtractor1_Struct is
begin
p0: process(X,Y,Z)
variable s: bit_vector (2 downto 0);
begin
s:= X&Y&Z;
 if s = "000" then D <= '0'; bout <= '0';
  elsif s = "001" then D <= '1'; bout <= '1';
  elsif s = "010" then D <= '1'; bout <= '1';
  elsif s = "011" then D <= '0'; bout <= '1';
  elsif s = "100" then D <= '1'; bout <= '0';
  elsif s = "101" then D <= '0'; bout <= '0';
  elsif s = "110" then D <= '0'; bout <= '0';
  elsif s = "111" then D <= '1'; bout <= '1';
  end if;
end process;
end algorithmic;

-- Create 4bit subtractor
entity Subtractor4bit is port (
            A, B: in bit_vector (3 downto 0); -- Declare numbers to subtract
            diff: out bit_vector (3 downto 0);  --  Difference of two numbers
            borrow: inout bit_vector(4 downto 0));  -- Borrow of each subtractor
end Subtractor4bit;

-- Using Subtractor1_flow to create the 4 bit subtractor
architecture structure of Subtractor4bit is
component Subtractor1_Struct
  port (X,Y,Z: in bit ;
            D,bout: out bit);
end component;

begin
  g0: for index in 0 to 3 generate
            u0: Subtractor1_Struct port map (
            X=> A(index), Y=> B(index), D=> diff(index), Z=> borrow(index),
            bout=> borrow (index+1));
   end generate;
end structure;
```

> Structural Model:

```
library IEEE;
use IEEE.std_logic_1164.all;
-- Create full subtractor
entity Subtractor1_Struct is
      port (X,Y,Z: in bit; bout, D: out bit;
      g1And,g2And,g3And: inout bit ); -- ‘g#And’ act as the and gates
end Subtractor1_Struct;

-- Creating entities
entity xor3 is -- XOR 3 εισοδων
  port (X1,X2,X3: in bit; O: out bit);
end xor3;

entity inv is -- inverter
    port (X1: in bit; O: out bit);
end inv;

entity and2 is --AND gate with 2 inputs
    port (X1,X2: in bit; O: inout bit);
end and2;

entity or3 is -- OR gate with 3 inputs
    port (X1,X2,X3: in bit; O: out bit);
end or3;

architecture structure of Subtractor1_Struct is
  component inv
          port (X1: in bit; O: out bit);
  end component;

  component and2
          port(X1, X2 : in bit; O: inout bit);
  end component;

  component xor3
          port(X1, X2, X3 : in bit; O: out bit);
  end component;

  component or3
          port(X1,X2,X3: in bit; O:out bit);
  end component;

signal s0:bit;
begin
          u0: inv port map(X, s0);
          u1: xor3 port map(X,Y,Z,D);
          u2: and2 port map(s0,Z,g1And);
          u3: and2 port map(s0,Y,g2And);
          u4: and2 port map(Y,Z,g3And);
```

```
        u5: or3 port map(g1And,g2And,g3And,bout);
end structure;
entity Subtractor4bit is port (
        A, B: in bit_vector (3 downto 0);
        diff: out bit_vector (3 downto 0);
        borrow: inout bit_vector(4 downto 0));
end Subtractor4bit;

architecture structure of Subtractor4bit is
component Subtractor1_Struct
  port (X,Y,Z: in bit; bout, D: out bit;
  g1And,g2And,g3And: inout bit );
end component;

begin
g0: for index in 0 to 3 generate
u0: Subtractor1_Struct port map (
X=> A(index), Y=> B(index), D=> diff(index), Z=> borrow(index),
bout=> borrow (index+1));

end generate;
end structure;

--  Component functions
architecture behave of xor3 is
begin
  O<= X1 xor X2 xor X3;
end behave;

architecture behave of inv is
begin
O<= not X1;
end behave;

architecture behave of and2 is
begin
O<= X1 and X2;
end behave;

architecture behave of or3 is
begin
O<= X1 or X2 or X3;
end behave;
```

Testbench

```
library IEEE;
use IEEE.std_logic_1164.all;

entity test_Subtractor4bit is
end test_Subtractor4bit;

architecture testbench of test_Subtractor4bit is

component Subtractor4bit port (
A, B: in bit_vector (3 downto 0);
diff: out bit_vector (3 downto 0);
borrow: inout bit_vector(4 downto 0));
end component;

signal tb_A: bit_vector(3 downto 0);
signal tb_B: bit_vector(3 downto 0);
signal tb_diff: bit_vector(3 downto 0);

begin
u0: Subtractor4bit port map(
A=> tb_A,
B=> tb_B,
diff=> tb_diff);

process
begin
-- 0100 – 0001 = 0011
tb_A<="0100";
tb_B<="0001";
wait for 10 ns;
--1100 – 0101 = 0111
tb_A<="1100";
tb_B<="0101";
wait for 10 ns;
-- 1111 – 1111 = 0000
tb_A<="1111";
tb_B<="1111";
wait for 10 ns;
-- 0100 – 0001 = 0011
tb_A<="0110";
tb_B<="0011";
wait for 10 ns;
-- 1000 – 0101 = 0110
tb_A<="1100";
tb_B<="0010";
wait for 10 ns;

end process;
end testbench;
```