N-bit adder solutions

Solution 1.

The main source code for the n-bit adder. Note the declaration of the full-adder that is defined in a separate file within the project

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

entity Adder is
port (a, b : in bit_vector (3 downto 0);
      cin : in bit;
      s : out bit_vector (3 downto 0);
      cout : out bit );
end;

architecture structural of Adder is

component fa is
port (a, b, cin : in bit;
      cout, s : out bit ) ;
end component;

signal c : bit_vector (3 downto 0);

begin

fa1 : fa port map (a(0), b(0), cin, c(1), s(0));
fa2 : fa port map (a(1), b(1), c(1), c(2), s(1));
fa3 : fa port map (a(2), b(2), c(2), c(3), s(2));
fa4 : fa port map (a(3), b(3), c(3), cout, s(3));

end structural;
```

The full-adder component is defined here. It resides within the project folder in a separate file.

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

entity fa is
port (a, b, cin : in bit;
      cout, s : out bit );
end;
```

```
architecture behavioral of fa is
begin

cout <= (a and b) or (cin and a) or (cin and b);
s <= a xor b xor cin;

end behavioral;
```

This shows the resulting equations – this is part of the report file.


```
ispLEVER Classic 1.2.00.11.33.08  -  Device Utilization Chart         Fri Mar
20 11:48:02 2009

P16V8AS Programmed Logic:
--------------------------------------------------------------------------------

s_3_     = !(  a_3_ & b_3_ & N_3
           #   !a_3_ & !b_3_ & N_3
           #   !a_3_ & b_3_ & !N_3
           #   a_3_ & !b_3_ & !N_3 );

cout     = !(  !a_3_ & !b_3_
           #   !a_3_ & N_3
           #   !b_3_ & N_3 );

s_2_     = !(  !a_2_ & b_2_ & c_2__n
           #   a_2_ & !b_2_ & c_2__n
           #   a_2_ & b_2_ & !c_2__n
           #   !a_2_ & !b_2_ & !c_2__n );

s_1_     = !(  !a_1_ & b_1_ & c_1__n
           #   a_1_ & !b_1_ & c_1__n
           #   a_1_ & b_1_ & !c_1__n
           #   !a_1_ & !b_1_ & !c_1__n );

s_0_     = !(  !cin & a_0_ & b_0_
           #   cin & !a_0_ & b_0_
           #   cin & a_0_ & !b_0_
           #   !cin & !a_0_ & !b_0_ );

c_1__n     = !(  !cin & !a_0_  #   !cin & !b_0_  #   !a_0_ & !b_0_ );

c_2__n     = !(  !a_1_ & !b_1_   #   !a_1_ & !c_1__n   #   !b_1_ & !c_1__n );

N_3     = !(  a_2_ & b_2_   #   a_2_ & c_2__n   #   b_2_ & c_2__n );
```

Solution 2.

Here is the source code for an implementation using arithmetic operator +.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
--use ieee.std_logic_unsigned.all;
--use ieee.std_logic_signed.all;

entity n_adder is

port (a, b : in signed (4 downto 0);
      s : out signed (4 downto 0));
end;

architecture behavioral of n_adder is
begin

s <= a + b;

end behavioral;
```

Here are the resulting equations

```
--------------------------------------------------------------------------------
                                                                          Page 2
ispLEVER Classic 1.2.00.11.33.08  -  Device Utilization Chart         Tue Apr
14 10:39:15 2009

P22V10G Programmed Logic:
--------------------------------------------------------------------------------
---



s_4_     = (  a_4_ & b_4_ & N_28
           #   !a_4_ & !b_4_ & N_28
           #   a_4_ & b_4_ & N_29
           #   !a_4_ & !b_4_ & N_29
           #   a_4_ & b_4_ & N_30
           #   !a_4_ & !b_4_ & N_30
           #   !a_4_ & b_4_ & !N_28 & !N_29 & !N_30
           #   a_4_ & !b_4_ & !N_28 & !N_29 & !N_30 );

s_3_     = (  !a_3_ & b_3_ & N_36
           #   a_3_ & !b_3_ & N_36
           #   a_3_ & b_3_ & !N_36
           #   !a_3_ & !b_3_ & !N_36 );

s_2_     = (  a_2_ & a_1_ & b_2_ & b_1_
           #   !a_2_ & a_1_ & !b_2_ & b_1_
```

```
             #   a_2_ & b_2_ & N_14
             #   !a_2_ & !b_2_ & N_14
             #   !a_2_ & !a_1_ & b_2_ & !N_14
             #   a_2_ & !a_1_ & !b_2_ & !N_14
             #   !a_2_ & b_2_ & !b_1_ & !N_14
             #   a_2_ & !b_2_ & !b_1_ & !N_14 );

s_1_     = (  !a_1_ & !a_0_ & b_1_
             #   a_1_ & !a_0_ & !b_1_
             #   a_1_ & a_0_ & b_1_ & b_0_
             #   !a_1_ & a_0_ & !b_1_ & b_0_
             #   !a_1_ & b_1_ & !b_0_
             #   a_1_ & !b_1_ & !b_0_ );

s_0_     = (  !a_0_ & b_0_
             #   a_0_ & !b_0_ );

N_14     = (  a_1_ & a_0_ & b_0_
             #   a_0_ & b_1_ & b_0_ );

N_28     = (  a_3_ & b_3_ );

N_29     = (  b_3_ & !N_36 );

N_30     = (  a_3_ & !N_36 );

N_36     = (  !a_2_ & !b_2_
             #   !a_2_ & !a_1_ & !N_14
             #   !a_1_ & !b_2_ & !N_14
             #   !a_2_ & !b_1_ & !N_14
             #   !b_2_ & !b_1_ & !N_14 );
```

Solution 3.

This solution uses a for loop.

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

entity add is
port ( a, b : in std_logic_vector(4 downto 0);
      cin : in std_logic;
      s : out std_logic_vector(4 downto 0);
      cout : out std_logic );
end;

architecture behavioral of add is
signal c : std_logic_vector(5 downto 0);
begin
c(0) <= cin;
cout <= c(5);
add: for i in 0 to 4 generate
      s(i) <= a(i) xor b(i) xor c(i);
      c(i+1) <= (a(i) and b(i)) or (a(i) and c(i)) or (b(i) and c(i));
end generate;

end behavioral;
```

Here are the resulting equations – again, this is not a carry-ripple adder implementation.

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

                                                                          Page 2
ispLEVER Classic 1.2.00.11.33.08  -  Device Utilization Chart         Tue Apr
14 10:47:11 2009

P22V10G Programmed Logic:
--------------------------------------------------------------------------------

s_4_     = (  a_4_ & b_4_ & c_4__n
           #   !a_4_ & !b_4_ & c_4__n
           #   !a_4_ & b_4_ & !c_4__n
           #   a_4_ & !b_4_ & !c_4__n );

cout     = (  a_4_ & b_4_
           #   a_4_ & c_4__n
           #   b_4_ & c_4__n );

s_3_     = (  a_3_ & b_3_ & c_3__n
           #   !a_3_ & !b_3_ & c_3__n
           #   !a_3_ & b_3_ & !c_3__n
           #   a_3_ & !b_3_ & !c_3__n );

s_2_     = (  a_2_ & b_2_ & c_2__n
           #   !a_2_ & !b_2_ & c_2__n
```

```
            #   !a_2_ & b_2_ & !c_2__n
            #   a_2_ & !b_2_ & !c_2__n );

s_1_     = (  !a_1_ & b_1_ & N_3
            #   a_1_ & !b_1_ & N_3
            #   a_1_ & b_1_ & !N_3
            #   !a_1_ & !b_1_ & !N_3 );

s_0_     = (  cin & a_0_ & b_0_
            #   !cin & !a_0_ & b_0_
            #   !cin & a_0_ & !b_0_
            #   cin & !a_0_ & !b_0_ );

c_2__n     = (  a_1_ & b_1_
              #   a_1_ & !N_3
              #   b_1_ & !N_3 );

c_3__n     = (  a_2_ & b_2_
              #   a_2_ & c_2__n
              #   b_2_ & c_2__n );

c_4__n     = (  a_3_ & b_3_
              #   a_3_ & c_3__n
              #   b_3_ & c_3__n );

N_3     = (  !cin & !a_0_
           #   !cin & !b_0_
           #   !a_0_ & !b_0_ );
```