FULL ADDER

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    09:44:47 08/31/2016
-- Design Name:
-- Module Name:    Fulladder - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
--library IEEE;
--use IEEE.STD_LOGIC_1164.ALL;
--
---- Uncomment the following library declaration if using
```

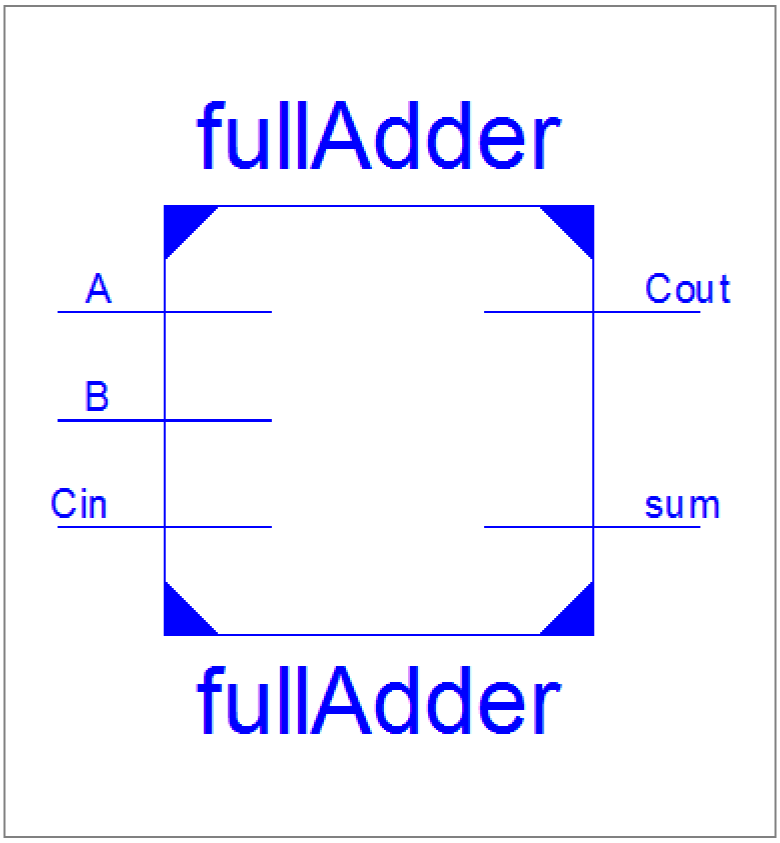
```
---- arithmetic functions with Signed or Unsigned values
----use IEEE.NUMERIC_STD.ALL;
--
---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
----library UNISIM;
----use UNISIM.VComponents.all;
--
--entity Fulladder is
--    Port ( A,B,Cin : in  STD_LOGIC;
--           sum, Cout : out  STD_LOGIC);
--end Fulladder;
--
--architecture Behavioral of Fulladder is
--
--begin
--
--
--end Behavioral;


-- Here we define the AND gate that we need for
-- the Half Adder
library ieee;
use ieee.std_logic_1164.all;
```

```
entity andGate is
   port( A, B : in std_logic;
            F : out std_logic);
end andGate;

architecture func of andGate is
begin
   F <= A and B;
end func;
--*============================

-- Here we define the XOR gate that we need for
-- the Half Adder
library ieee;
use ieee.std_logic_1164.all;

entity xorGate is
   port(        A, B : in std_logic;
            F : out std_logic);
end xorGate;

architecture func of xorGate is
begin
```

```
   F <= A xor B;
end func;
--*===========================

-- At this point we construct the half adder
-- using the AND and XOR gates
library ieee;
use ieee.std_logic_1164.all;

entity halfAdder is
   port( A, B : in std_logic;
    sum, Cout : out std_logic);
end halfAdder;

architecture halfAdder of halfAdder is

   component andGate is -- import AND Gate
      port( A, B : in std_logic;
               F : out std_logic);
   end component;

   component xorGate is -- import XOR Gate
      port( A, B : in std_logic;
               F : out std_logic);
   end component;


begin
   G1 : xorGate port map(A, B, sum);
   G2 : andGate port map(A, B, Cout);
end halfAdder;
--*=====================*=================== END HALF
ADDER


-- Now we define the OR gate that we need for the Full Adder
library ieee;
use ieee.std_logic_1164.all;


entity orGate is
   port(         A, B : in std_logic;
            F : out std_logic);
end orGate;


architecture func of orGate is
begin
   F <= A or B;
end func;
--*==============================*
--*==============================*
```

```
-- We are finally ready to build the Full Adder
library ieee;
use ieee.std_logic_1164.all;

entity fullAdder is
   port( A, B, Cin : in std_logic;
         sum, Cout : out std_logic);
end fullAdder;
--
architecture fullAdder of fullAdder is

   component halfAdder is --import Half Adder entity
      port( A, B  : in std_logic;
        sum, Cout : out std_logic);
   end component;

   component orGate is --import OR Gate entity
      port( A, B : in std_logic;
             F : out std_logic);
   end component;

   signal halfTohalf, halfToOr1, halfToOr2: std_logic;

begin
```

```
  G1: halfAdder port map(A, B, halfTohalf, halfToOr1);

  G2: halfAdder port map(halfTohalf, Cin, sum, halfToOr2);

  G3: orGate port map(halfToOr1, halfToOr2, Cout);

end fullAdder;

---------------------------------------------------------END

---------------------------------------------------------END
```


fullAdder
A
B
Cin
Cout
sum
fullAdder