```
--------------------------------------------------------------------------------
--
--  Purpose:
--
--    Carry Chain Adder
--
--  Discussion:
--
--
--  Licensing:
--
--    This code is distributed under the GNU LGPL license.
--
--  Modified:
--
--    2012.08.21
--
--  Author:
--
--    Young W. Lim
--
--  Parameters:
--
--    Input:
--
--    Output:
--------------------------------------------------------------------------------

library STD;
use STD.textio.all;

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;


entity adder is
  generic (
    WD     : in natural := 32;
    BD     : in natural := 4 );

  port (
    an     : in   std_logic_vector (WD-1 downto 0) := (others=>'0');
    bn     : in   std_logic_vector (WD-1 downto 0) := (others=>'0');
    ci     : in   std_logic := '0';
    cn     : out  std_logic_vector (WD-1 downto 0) := (others=>'0');
    co     : out  std_logic := '0');

end adder;

architecture cca of adder is

  component Badder is
    generic (
      WD     : in natural := 4);

    port (
      an    : in   std_logic_vector (WD-1 downto 0);
      bn    : in   std_logic_vector (WD-1 downto 0);
      ci    : in   std_logic := '0';
      cn    : out  std_logic_vector (WD-1 downto 0);
      co    : out  std_logic := '0');
  end component;

  type array2d is array (WD/BD-1 downto 0, BD-1 downto 0) of std_logic;
  signal an2d, bn2d, cn2d: array2d  := ((others=> (others=> '0')));
  signal bn: array2d  := ((others=> (others=> '0')));

  type array1d is array (WD/BD-1 downto 0) of std_logic;
  signal s1d := (others=> '0');
```

```
begin


  ILOOP: for i in WD/BD-1 downto 0 generate
        U0: Badder generic map (WD => BD)
                   port map (an => an2d(i, BD-1 downto 0),
                             bn => bn2d(i, BD-1 downto 0),
                             ci => ci1d(i),
                             cn => cn2d(i, BD-1 downto 0),
                             co => co1d(i) );
   end generate ILOOP;




  process (an, bn, ci)
    variable sn : std_logic_vector (WD-1 downto 0) := (others=>'0');
    variable c  : std_logic := '0';
  begin  -- process
    c := ci;
    for i in 0 to WD-1 loop
      sn(i) := an(i) xor bn(i) xor c;
      c := (an(i) and bn(i)) or (an(i) and c) or (bn(i) and c);
    end loop;  -- i

    cn <= sn;
    co <= c;
  end process;

end cca;
```