```
--------------------------------------------------------------------------------
--
--  Purpose:
--
--    Carry Chain Adder
--
--  Discussion:
--
--
--  Licensing:
--
--    This code is distributed under the GNU LGPL license.
--
--  Modified:
--
--    2012.08.22
--
--  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;

--------------------------------------------------------------------------
-- an : 1st operand (WD-bit)
-- bn : 2nd operand (WD-bit)
-- ci : carry in (1-bit)
-- cn : result (WD-bit)
-- co : carry out (1-bit)
--------------------------------------------------------------------------


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;

  constant ND : natural := WD/BD;

  type array2d is array (ND-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 (ND-1 downto 0) of std_logic;
  signal ci1d, co1d : array1d := (others=> '0');
  signal g1d, p1d : array1d := (others=> '0');
  signal qi1d, qo1d : array1d := (others=> '0');

begin

  -- ND Adders of BD-bit
  -- ci1d(i) : carry in of the i-th BD-bit adder
  -- co1d(i) : carry out of the i-th BD-bit adder
  -- cn2d(i, j) : j-th bit of the result of the i-th BD-bit adder
  ILOOP: for i in ND-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;


  -- Carry Chain GP Logic
  -- g1d(i) : carry generation of the i-th BD-bit adder
  --          cn2d(i) > BD-1
  -- p1d(i) : carry propagation of the i-th BD-bit adder
  --          cn2d(i) = BD-1

  g1d(i) <= co1d(i);

  process (cn2d)
    variable tmp2d: array2d  := ((others=> (others=> '0')));
    variable tmp : std_logic := 0;
  begin

    tmp2d := cn2d;

    for i in ND-1 downto 0 loop
        tmp := 0;
        for j in BD-1 downto 0 loop
            tmp := tmp and tmp2d(i, j);
        end loop;
        p1d(i) <= tmp;
    end loop;

  end process;

  -- Carry Chain Cell
  -- qi1d(i) : input of a carry chain cell
  -- qo1d(i) : output of a carry chain cell

  process (ci, qo1d)
    variable tmp1d : array1d := (others=> '0');
  begin

    tmp1d := qo1d;

    for i in ND-1 downto 1 loop
        qi1d(i) := qo1d(i-1);
    end loop;

    qi1d(0) := ci;
```

```
  end process;


  process (p1d, g1d, qi1d)
    variable tmp1d_p, tmp1d_g, tmp1d_qi : array1d := (others=> '0');
  begin

    tmp1d_p := p1d;
    tmp1d_g := g1d;
    tmp1d_qi := qi1d;

    for i in ND-1 downto 0 loop
        if (tmp1d_p(i)) then
           qo1d(i) <= tmp1d_qi(i);
        else
           qo1d(i) <= tmp1d_g(i);
        end if;
    end loop;

  end process;



end cca;
```