```
::::::::::::::
c1.adder.vhdl
::::::::::::::
--------------------------------------------------------------------------------
--
--  Purpose:
--
--    Bianary Adder Entity
--
--  Discussion:
--
--
--  Licensing:
--
--    This code is distributed under the GNU LGPL license.
--
--  Modified:
--
--    2012.04.03
--
--  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;


::::::::::::::
c1.adder.cla.vhdl
::::::::::::::
--------------------------------------------------------------------------------
--
--  Purpose:
--
--    Carry Lookahead Adder
--
--  Discussion:
--
--
--  Licensing:
--
--    This code is distributed under the GNU LGPL license.
--
--  Modified:
--
--    2014.03.11
--
--  Author:
--
--    Young W. Lim
--
--  Parameters:
--
--    Input: an(31:0), bn(31:0), ci
--
--    Output: {co, cn(31:0)} = an(31:0) + bn(31:0) + ci
--------------------------------------------------------------------------------

library STD;
use STD.textio.all;

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




architecture cla of adder is

  component ahead is
    generic (
      BD     : in natural := 4);
```

```
  port (
    g    : in   std_logic_vector (BD-1 downto 0) := (others=>'0');
    p    : in   std_logic_vector (BD-1 downto 0) := (others=>'0');
    ci   : in   std_logic := '0';
    co   : out  std_logic_vector (BD-1 downto 0) := (others=>'0'));
end component;

constant ND : natural := WD/BD; -- (8 = 32/4)



-------------------------------------------------------------------
-- ga       : array(WD) <= an and bn (G array)
-- pa       : array(WD) <= an xor bn (P array)
-- ca       : array(WD) <= co2d from carry ahead logic (Carry array)
-------------------------------------------------------------------
signal ga : std_logic_vector (WD-1 downto 0) := (others=>'0');
signal pa : std_logic_vector (WD-1 downto 0) := (others=>'0');
signal ca : std_logic_vector (WD-1 downto 0) := (others=>'0');

-------------------------------------------------------------------
-- ga2d, pa2d       : array(ND, BD) <= ga, pa
-- co2d             : array(ND, BD) => ca
-- ci1d             : array(ND) carry in of each carry ahead logic
-------------------------------------------------------------------
type array2d is array (ND-1 downto 0) of std_logic_vector (BD-1 downto 0);
signal ga2d : array2d := ((others=> (others=> '0'))); -- hold ga(31:0) data
signal pa2d : array2d := ((others=> (others=> '0'))); -- hold pa(31:0) data
signal co2d : array2d := ((others=> (others=> '0'))); -- hold co(31:0) data

type array1d is array (ND-1 downto 0) of  std_logic;
signal ci1d : array1d := (others=> '0');


procedure ToA2d
  (signal a : in std_logic_vector (WD-1 downto 0);
   signal a2d : out array2d ) is
  variable tmp2d: array2d  := ((others=> (others=> '0')));
  variable tmpv : std_logic_vector (WD-1 downto 0) := (others=>'0');
begin
  tmpv := a;

  for i in ND-1 downto 0 loop
      tmp2d(i) := tmpv((i+1)*BD-1 downto i*BD);
      a2d(i) <= tmp2d(i);
  end loop;

end ToA2d;
```

```
  procedure FromA2d
    (signal a2d : in  array2d;
     signal a : out std_logic_vector (WD-1 downto 0) ) is
    variable tmp2d: array2d  := ((others=> (others=> '0')));
    variable tmpv : std_logic_vector (WD-1 downto 0) := (others=>'0');
  begin
    tmp2d := a2d;

    for i in ND-1 downto 0 loop
        tmpv((i+1)*BD-1 downto i*BD) := tmp2d(i);
    end loop;

    a <= tmpv;
  end FromA2d;


begin

  ------------------------------------------------------------------------
  -- ND Ahead Logics of BD-bit
  ------------------------------------------------------------------------
  -- ci1d(i)    : cin of the i-th BD-bit ahead logic
  -- co2d(i, j) : j-th bit carry of the i-th BD-bit ahead logic
  ------------------------------------------------------------------------
  ILOOP: for i in ND-1 downto 0 generate
        U0:ahead generic map (BD => BD)
                    port map (g => ga2d(i),
                              p => pa2d(i),
                              ci => ci1d(i),
                              co => co2d(i) );
  end generate ILOOP;


  ------------------------------------------------------------------------
  -- cn   <= pa, ca, ci
  -- co   <= ci(31)
  ------------------------------------------------------------------------
  ga <= an and bn;
  pa <= an xor bn;
  cn <= pa xor (ca(WD-2 downto 0) & ci);
  co <= ca(WD-1);


  ToA2d(ga, ga2d);
  ToA2d(pa, pa2d);

  FromA2d(co2d, ca);
```

```
  process (co2d, ci)
    variable c : array1d := (others=> '0');
    variable d : std_logic_vector (BD-1 downto 0) := (others=>'0');
  begin  -- process
    c(0) := ci;
    for i in 1 to ND-1 loop
     d := co2d(i-1);
     c(i) := d(BD-1);
    end loop;  -- i

    ci1d <= c;
  end process;

end cla;



::::::::::::::
c1.adder.cla.ahead.vhdl
::::::::::::::
--------------------------------------------------------------------------------
--
--  Purpose:
--
--    Carry Ahead Logic of CLA
--
--  Discussion:
--
--
--  Licensing:
--
--    This code is distributed under the GNU LGPL license.
--
--  Modified:
--
--    2014.03.11
--
--  Author:
--
--    Young W. Lim
--
--  Parameters:
--
--    Input: g(3:0), p(3:0), ci
--
--    Output: co(3:0)
--------------------------------------------------------------------------------

library StD;
use StD.textio.all;
```

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



entity ahead is
  generic (
    BD     : in natural := 4);

  port (
    g    : in   std_logic_vector (3 downto 0) := (others=>'0');
    p    : in   std_logic_vector (3 downto 0) := (others=>'0');
    ci   : in   std_logic := '0';
    co   : out  std_logic_vector (3 downto 0) := (others=>'0'));
end ahead;


architecture rtl of ahead is
  signal t10, t20, t30 : std_logic := '0';
  signal      t21, t31 : std_logic := '0';
  signal           t32 : std_logic := '0';
begin

    t10 <= p(1) and p(0);
    t20 <= p(2) and p(1) and p(0);
    t30 <= p(3) and p(2) and p(1) and p(0);

    t21 <= p(2) and p(1);
    t31 <= p(3) and p(2) and p(1);

    t32 <= p(3) and p(2);



    co(0) <= g(0) or (p(0) and ci);
    co(1) <= g(1) or (p(1) and g(0)) or (t10 and ci);
    co(2) <= g(2) or (p(2) and g(1)) or (t21 and g(0)) or (t20 and ci);
    co(3) <= g(3) or (p(3) and g(2)) or (t32 and g(1)) or (t31 and g(0)) or (t30 and ci);

end rtl;
```