EE 671 VLSI Design

Assignment 3

32 Bit Brent Kung Adder VHDL synthesis

GHDL

I had never used GHDL before this assignment as we always used Quartus. So, for this assignment I decided to try GHDL instead. Installing and making a workflow for the same was the first issue I had to deal with in this assignment. For viewing waves I decided to install gtkwave software which works well with GHDL.

Making the Brent Kung adder

Understanding how the adder works wasn't really an issue. I had to brush up on my VHDL skills before starting making one. I initially created a behavioral description of the adder after which to follow the template provided in the assignment I decided to convert it to structural description.

The snippets of the code have been added below. For making the testbench I decided to use some help from online sources since the usual quartus testbench was not there on my laptop. In the testbench I use four computations, two with carry and rest without. From the images below we can see that the adder works as intended with enough delay because of and, xor, etc gates having the delay according to the assignment.

Code Snippets:

-Below is the template provided in the assignment:

```vhdl
--  32b Brent Kung adder program
--  Done By: Tejaswee Sulekh, 20D070082


-- simple gates with trivial architectures
library IEEE;
use IEEE.std_logic_1164.all;
entity andgate is
port (A, B: in std_ulogic;
prod: out std_ulogic);
end entity andgate;
architecture trivial of andgate is
begin
prod <= A AND B AFTER 381 ps;
end architecture trivial;

library IEEE;
use IEEE.std_logic_1164.all;
entity xorgate is
port (A, B: in std_ulogic;
uneq: out std_ulogic);
end entity xorgate;
architecture trivial of xorgate is
begin
uneq <= A XOR B AFTER 764 ps;
end architecture trivial;

library IEEE;
use IEEE.std_logic_1164.all;
entity abcgate is
port (A, B, C: in std_ulogic;
abc: out std_ulogic);
end entity abcgate;
architecture trivial of abcgate is
begin
abc <= A OR (B AND C) AFTER 482 ps;
end architecture trivial;
-- A + C.(A+B) with a trivial architecture

library IEEE;
use IEEE.std_logic_1164.all;
entity Cin_map_G is
port(A, B, Cin: in std_ulogic;
Bit0_G: out std_ulogic);
end entity Cin_map_G;
architecture trivial of Cin_map_G is
begin
Bit0_G <= (A AND B) OR (Cin AND (A OR B)) AFTER 764 ps;
end architecture trivial;
```

-This is where I used the template gates to create entity for P and G calculations.

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

-- Calculation of Gn and Pn from Gn-1 and Pn-1
entity pg_calculation is
  port (
    g_in, p_in, g_in_1, p_in_1 : in std_logic; --Defining inputs
    g_out, p_out : out std_logic --Defining outputs
    );
end entity pg_calculation;

architecture behavioral of pg_calculation is
    signal A, B, prod : std_logic;

    component andgate is
            port(
                    A, B : in std_logic;
                    prod : out std_logic
                );
        end component andgate;

    component abcgate is
            port(
                    A, B, C : in std_logic;
                    abc : out std_logic
                );
        end component abcgate;

    begin
    A <= p_in_1;
    B <= p_in;

    pout: andgate port map(A => p_in_1, B => p_in, prod => p_out);
    gout: abcgate port map(A => g_in, B => g_in_1, C => p_in, abc => g_out);
--        g_out <= g_in or (g_in_1 and p_in) after 381 ps;
--        p_out <= p_in and p_in_1 after 150 ps;
end architecture behavioral;

library ieee ;
use ieee.std_logic_1164.all ;
-- Calculation of C_out from C_in, g_in and p_in
entity carry_calculation is
    port (
      c_i, p_in, g_in : in std_logic;
      c_o : out std_logic
      );
  end entity carry_calculation;

  architecture behavioral of carry_calculation is
```

-Halfway in the middle here we start defining the actual adder.

```
    component abcgate is
      port(
            A, B, C : in std_logic;
            abc : out std_logic
          );
        end component abcgate;
      begin
    co: abcgate port map(A => g_in, B => c_i, C => p_in, abc => c_o);

--          c_o <= g_in or (c_i and p_in) after 381 ps;
  end architecture behavioral;

  library ieee ;
  use ieee.std_logic_1164.all ;
  -- Calculation of sum from a, b and c
  entity sum_calculation is
      port (
        a_in, b_in, c_in : in STD_LOGIC_VECTOR(31 downto 0);
        s_out : out STD_LOGIC_VECTOR(31 downto 0)
        );
    end entity sum_calculation;

    architecture behavioral of sum_calculation is
        begin
            s_out <= a_in xor b_in xor c_in after 764 ps;
    end architecture behavioral;

-- adder definition
ibrary ieee ;
se ieee.std_logic_1164.all ;

ntity adder_32b is
    port(
        a, b : in STD_LOGIC_VECTOR(31 downto 0);
        c_in : in std_logic;
        s : out STD_LOGIC_VECTOR(31 downto 0);
        c_out : out std_logic
    );
nd entity adder_32b;

rchitecture structural of adder_32b is
        signal g0, p0, c : STD_LOGIC_VECTOR(31 downto 0);
        signal g1, p1 : STD_LOGIC_VECTOR(15 downto 0);
        signal g2, p2 : STD_LOGIC_VECTOR(7 downto 0);
        signal g3, p3 : STD_LOGIC_VECTOR(3 downto 0);
        signal g4, p4 : STD_LOGIC_VECTOR(1 downto 0);
        signal g5, p5 : std_logic;

        component pg_calculation is
```

```
component pg_calculation is
    port(
            g_in, p_in, g_in_1, p_in_1 : in std_logic;
            g_out, p_out : out std_logic
        );
end component pg_calculation;

component carry_calculation is
    port(
            c_i, p_in, g_in : in std_logic;
            c_o : out std_logic
        );
end component carry_calculation;

component sum_calculation is
    port(
            a_in, b_in, c_in : in STD_LOGIC_VECTOR(31 downto 0);
            s_out : out STD_LOGIC_VECTOR(31 downto 0)
        );
end component sum_calculation;
```

-After all the components have been decided we define the P and G values in the form of a tree as shown in the slides for adder.

```
begin
    c(0) <= c_in;

    -- G_0 calculation
    g0 <= a and b after 381 ps;
    p0 <= a xor b after 764 ps;

    -- G_1 calculation
    G1_0: pg_calculation port map(g_in => g0(1), p_in => p0(1), g_in_1 => g0(0), p_in_1 => p0(0), g_out => g1(0), p_out => p1(0));
    G1_1: pg_calculation port map(g_in => g0(3), p_in => p0(3), g_in_1 => g0(2), p_in_1 => p0(2), g_out => g1(1), p_out => p1(1));
    G1_2: pg_calculation port map(g_in => g0(5), p_in => p0(5), g_in_1 => g0(4), p_in_1 => p0(4), g_out => g1(2), p_out => p1(2));
    G1_3: pg_calculation port map(g_in => g0(7), p_in => p0(7), g_in_1 => g0(6), p_in_1 => p0(6), g_out => g1(3), p_out => p1(3));
    G1_4: pg_calculation port map(g_in => g0(9), p_in => p0(9), g_in_1 => g0(8), p_in_1 => p0(8), g_out => g1(4), p_out => p1(4));
    G1_5: pg_calculation port map(g_in => g0(11), p_in => p0(11), g_in_1 => g0(10), p_in_1 => p0(10), g_out => g1(5), p_out => p1(5));
    G1_6: pg_calculation port map(g_in => g0(13), p_in => p0(13), g_in_1 => g0(12), p_in_1 => p0(12), g_out => g1(6), p_out => p1(6));
    G1_7: pg_calculation port map(g_in => g0(15), p_in => p0(15), g_in_1 => g0(14), p_in_1 => p0(14), g_out => g1(7), p_out => p1(7));
    G1_8: pg_calculation port map(g_in => g0(17), p_in => p0(17), g_in_1 => g0(16), p_in_1 => p0(16), g_out => g1(8), p_out => p1(8));
    G1_9: pg_calculation port map(g_in => g0(19), p_in => p0(19), g_in_1 => g0(18), p_in_1 => p0(18), g_out => g1(9), p_out => p1(9));
    G1_10: pg_calculation port map(g_in => g0(21), p_in => p0(21), g_in_1 => g0(20), p_in_1 => p0(20), g_out => g1(10), p_out => p1(10));
    G1_11: pg_calculation port map(g_in => g0(23), p_in => p0(23), g_in_1 => g0(22), p_in_1 => p0(22), g_out => g1(11), p_out => p1(11));
    G1_12: pg_calculation port map(g_in => g0(25), p_in => p0(25), g_in_1 => g0(24), p_in_1 => p0(24), g_out => g1(12), p_out => p1(12));
    G1_13: pg_calculation port map(g_in => g0(27), p_in => p0(27), g_in_1 => g0(26), p_in_1 => p0(26), g_out => g1(13), p_out => p1(13));
    G1_14: pg_calculation port map(g_in => g0(29), p_in => p0(29), g_in_1 => g0(28), p_in_1 => p0(28), g_out => g1(14), p_out => p1(14));
    G1_15: pg_calculation port map(g_in => g0(31), p_in => p0(31), g_in_1 => g0(30), p_in_1 => p0(30), g_out => g1(15), p_out => p1(15));
```

```
-- G_2 : gnpn_calculation port map(gn_1 => g1, pn_1 => p1, gn => g2, pn => p2);
G2_0: pg_calculation port map(g_in => g1(1), p_in => p1(1), g_in_1 => g1(0), p_in_1 => p1(0), g_out => g2(0), p_out => p2(0))
G2_1: pg_calculation port map(g_in => g1(3), p_in => p1(3), g_in_1 => g1(2), p_in_1 => p1(2), g_out => g2(1), p_out => p2(1))
G2_2: pg_calculation port map(g_in => g1(5), p_in => p1(5), g_in_1 => g1(4), p_in_1 => p1(4), g_out => g2(2), p_out => p2(2))
G2_3: pg_calculation port map(g_in => g1(7), p_in => p1(7), g_in_1 => g1(6), p_in_1 => p1(6), g_out => g2(3), p_out => p2(3))
G2_4: pg_calculation port map(g_in => g1(9), p_in => p1(9), g_in_1 => g1(8), p_in_1 => p1(8), g_out => g2(4), p_out => p2(4))
G2_5: pg_calculation port map(g_in => g1(11), p_in => p1(11), g_in_1 => g1(10), p_in_1 => p1(10), g_out => g2(5), p_out => p2
G2_6: pg_calculation port map(g_in => g1(13), p_in => p1(13), g_in_1 => g1(12), p_in_1 => p1(12), g_out => g2(6), p_out => p2
G2_7: pg_calculation port map(g_in => g1(15), p_in => p1(15), g_in_1 => g1(14), p_in_1 => p1(14), g_out => g2(7), p_out => p2
```

```
-- G_3 : gnpn_calculation port map(gn_1 => g2, pn_1 => p2, gn => g3, pn => p3);
G3_0: pg_calculation port map(g_in => g2(1), p_in => p2(1), g_in_1 => g2(0), p_in_1 => p2(0), g_out => g3(0), p_out => p3(0));
G3_1: pg_calculation port map(g_in => g2(3), p_in => p2(3), g_in_1 => g2(2), p_in_1 => p2(2), g_out => g3(1), p_out => p3(1));
G3_2: pg_calculation port map(g_in => g2(5), p_in => p2(5), g_in_1 => g2(4), p_in_1 => p2(4), g_out => g3(2), p_out => p3(2));
G3_3: pg_calculation port map(g_in => g2(7), p_in => p2(7), g_in_1 => g2(6), p_in_1 => p2(6), g_out => g3(3), p_out => p3(3));

-- G_4 : gnpn_calculation port map(gn_1 => g3, pn_1 => p3, gn => g4, pn => p4);
G4_0: pg_calculation port map(g_in => g3(1), p_in => p3(1), g_in_1 => g3(0), p_in_1 => p3(0), g_out => g4(0), p_out => p4(0));
G4_1: pg_calculation port map(g_in => g3(3), p_in => p3(3), g_in_1 => g3(2), p_in_1 => p3(2), g_out => g4(1), p_out => p4(1));

-- G_5
G5_0: pg_calculation port map(g_in => g4(1), p_in => p4(1), g_in_1 => g4(0), p_in_1 => p4(0), g_out => g5, p_out => p5);
```

-Now once we have the P and G values we can find the final carry and for calculation of the sum we will have to calculate the carry value at each node of the tree. As shown below:

```
        -- Compute Carry out and C16
        C_32: carry_calculation port map(c_i => c_in, g_in => g5, p_in => p5, c_o => c_out);
        C_16: carry_calculation port map(c_i => c(0), g_in => g4(0), p_in => p4(0), c_o => c(16));

        -- Compute other carry
        C_8: carry_calculation port map(c_i => c(0), g_in => g3(0), p_in => p3(0), c_o => c(8));
        C_24: carry_calculation port map(c_i => c(16), g_in => g3(2), p_in => p3(2), c_o => c(24));

        C_4: carry_calculation port map(c_i => c(0), g_in => g2(0), p_in => p2(0), c_o => c(4));
        C_12: carry_calculation port map(c_i => c(8), g_in => g2(2), p_in => p2(2), c_o => c(12));
        C_20: carry_calculation port map(c_i => c(16), g_in => g2(4), p_in => p2(4), c_o => c(20));
        C_28: carry_calculation port map(c_i => c(24), g_in => g2(6), p_in => p2(6), c_o => c(28));

        C_2: carry_calculation port map(c_i => c(0), g_in => g1(0), p_in => p1(0), c_o => c(2));
        C_6: carry_calculation port map(c_i => c(4), g_in => g1(2), p_in => p1(2), c_o => c(6));
        C_10: carry_calculation port map(c_i => c(8), g_in => g1(4), p_in => p1(4), c_o => c(10));
        C_14: carry_calculation port map(c_i => c(12), g_in => g1(6), p_in => p1(6), c_o => c(14));
        C_18: carry_calculation port map(c_i => c(16), g_in => g1(8), p_in => p1(8), c_o => c(18));
        C_22: carry_calculation port map(c_i => c(20), g_in => g1(10), p_in => p1(10), c_o => c(22));
        C_26: carry_calculation port map(c_i => c(24), g_in => g1(12), p_in => p1(12), c_o => c(26));
        C_30: carry_calculation port map(c_i => c(28), g_in => g1(14), p_in => p1(14), c_o => c(30));

        C_1: carry_calculation port map(c_i => c(0), g_in => g0(0), p_in => p0(0), c_o => c(1));
        C_3: carry_calculation port map(c_i => c(2), g_in => g0(2), p_in => p0(2), c_o => c(3));
        C_5: carry_calculation port map(c_i => c(4), g_in => g0(4), p_in => p0(4), c_o => c(5));
        C_7: carry_calculation port map(c_i => c(6), g_in => g0(6), p_in => p0(6), c_o => c(7));
        C_9: carry_calculation port map(c_i => c(8), g_in => g0(8), p_in => p0(8), c_o => c(9));
        C_11: carry_calculation port map(c_i => c(10), g_in => g0(10), p_in => p0(10), c_o => c(11));
        C_13: carry_calculation port map(c_i => c(12), g_in => g0(12), p_in => p0(12), c_o => c(13));
        C_15: carry_calculation port map(c_i => c(14), g_in => g0(14), p_in => p0(14), c_o => c(15));
        C_17: carry_calculation port map(c_i => c(16), g_in => g0(16), p_in => p0(16), c_o => c(17));
        C_19: carry_calculation port map(c_i => c(18), g_in => g0(18), p_in => p0(18), c_o => c(19));
        C_21: carry_calculation port map(c_i => c(20), g_in => g0(20), p_in => p0(20), c_o => c(21));
        C_23: carry_calculation port map(c_i => c(22), g_in => g0(22), p_in => p0(22), c_o => c(23));
        C_25: carry_calculation port map(c_i => c(24), g_in => g0(24), p_in => p0(24), c_o => c(25));
        C_27: carry_calculation port map(c_i => c(26), g_in => g0(26), p_in => p0(26), c_o => c(27));
        C_29: carry_calculation port map(c_i => c(28), g_in => g0(28), p_in => p0(28), c_o => c(29));
        C_31: carry_calculation port map(c_i => c(30), g_in => g0(30), p_in => p0(30), c_o => c(31));

        -- Compute sum
        SUM: sum_calculation port map(a_in => a, b_in => b, c_in => c, s_out => s);
end architecture structural;
```

-This was the adder.vhdl file. The code for testbench has been provided below:

```
entity adder_32b_tb is
end entity adder_32b_tb;

architecture test of adder_32b_tb is

    -- Component under test
    component adder_32b is
        port (
            a, b : in STD_LOGIC_VECTOR(31 downto 0);
            c_in : in std_logic;
            s : out STD_LOGIC_VECTOR(31 downto 0);
            c_out : out std_logic);
      end component adder_32b;

      --  Specifies which entity is bound with the component.
      for adder_32b_0: adder_32b use entity work.adder_32b(structural);
      signal a, b, s : std_logic_vector(31 downto 0);
      signal c_in, c_out : std_logic;
    begin
      --  Component instantiation.
      adder_32b_0: adder_32b
      port map (a => a, b => b, c_in => c_in, s => s, c_out => c_out);

      --  This process does the real job.
      process
        type pattern_type is record
          --  The inputs of the adder.
          a, b : std_logic_vector(31 downto 0);
          c_in : std_logic;
        end record;
        --  The patterns to apply.
        type pattern_array is array (natural range <>) of pattern_type;
        constant patterns : pattern_array :=
          (("01010101010101010101010101010101", "00011010101010101010101010101010", '1'), -- B + A
           ("00000000000000000000000000000000", "00000000000000000000000000001001", '1'),
           ("00000000000000000000000000000001", "00000000000000000000000000001111", '0'),
           ("00110000111100001111000011110000", "00111100001111000011110000111100", '0'));
      begin
        --  Check each pattern.
        for i in patterns'range loop
          --  Set the inputs.
          a <= patterns(i).a;
          b <= patterns(i).b;
          c_in <= patterns(i).c_in;
          --  Wait for the results.
          wait for 15 ns;
        end loop;
        wait;
      end process;
end architecture test;
```

As mentioned above we will be using only four numbers for testing the vhdl file that we created with two of them containing a carry.

Wave Results

When we use gtkwave for obtaining the wave signals for this adder testbench file we get the following result:

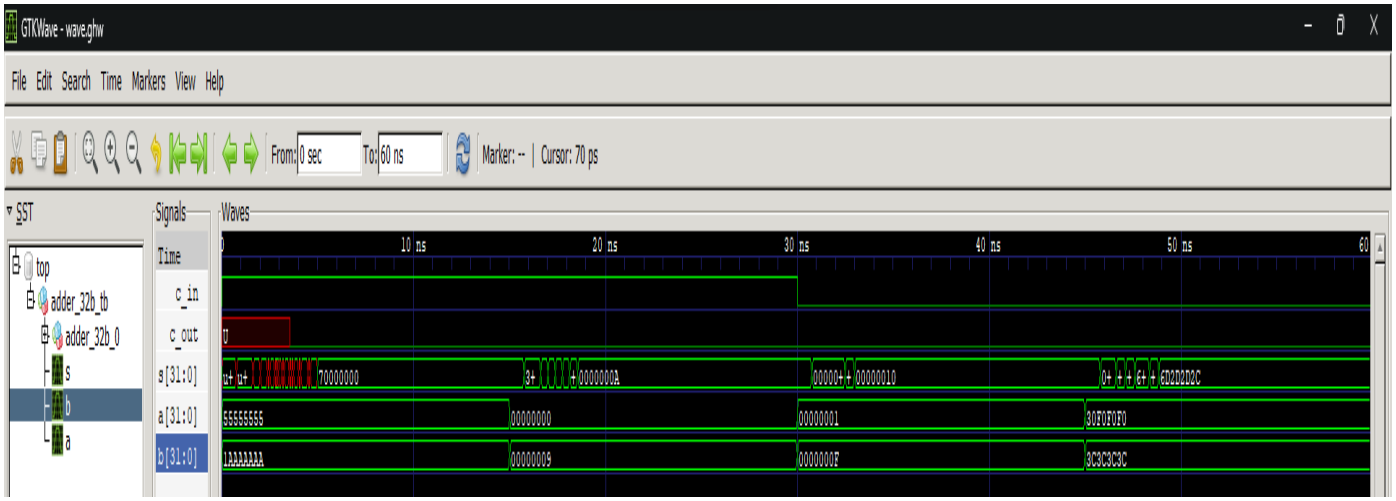
Terminal View:

```
PS C:\Users\tsule> cd C:\Users\tsule\Desktop\Desktop\Course_stuff\Sem7\EE671_VLSI\Assignment3\vhdlfile
s\BrentKungAdder_32bit
PS C:\Users\tsule\Desktop\Desktop\Course_stuff\Sem7\EE671_VLSI\Assignment3\vhdlfiles\BrentKungAdder_32
bit> ghdl -a adder32.vhdl
PS C:\Users\tsule\Desktop\Desktop\Course_stuff\Sem7\EE671_VLSI\Assignment3\vhdlfiles\BrentKungAdder_32
bit> ghdl -a adder_tb.vhdl
PS C:\Users\tsule\Desktop\Desktop\Course_stuff\Sem7\EE671_VLSI\Assignment3\vhdlfiles\BrentKungAdder_32
bit> ghdl -r adder_32b_tb --wave=wave.ghw
PS C:\Users\tsule\Desktop\Desktop\Course_stuff\Sem7\EE671_VLSI\Assignment3\vhdlfiles\BrentKungAdder_32
bit> gtkwave.exe wave.ghw

GTKWave Analyzer v3.3.100 (w)1999-2019 BSI

[0] start time.
[60000000] end time.
```

Wave Signal:



I have added screenshots in the final submission where the signals are more visible.