Assignment - 3

32-bit Brent Kung Adder

EE 671 - VLSI Design

Name:	**Satvikkumar Patel**
Roll number:	23M1117
Program:	M.tech (EE5 - Electronic Systems)
Department:	Electrical Engineering
Date of Sub:	7th Oct, 2023

Tuesday	**EE 671: VLSI Design**	Due on Saturday
Sep. 26, 2023	**Assignment 3**	Oct. 07, 2023

Q–1 Describe a 32 bit Brent Kung adder in VHDL and simulate it using a test bench. Your description should use `std_logic` types for various signals. You can use the public domain tool "ghdl" for VHDL simulation.

a) Logic functions AND, XOR, A + B.C and A.B + C.(A+B) are required for computing different orders of G, P and final sum and carry outputs. A template file with entity/architecture pairs for these functions is appended at the end. You should modify that code to implement delays for logic functions as given below:

Function	Delay in ps
AND	300 + last two digit of your roll number
A + B.C	400 + last two digits of your roll number
XOR	600 + 2*last two digits of your roll number
A.B + C.(A+B)	600 + 2*last two digits of your roll number

b) Using the above entities as components, write structural descriptions of each level of the tree for generating various orders of G and P values.
The rightmost blocks of all levels should use the already available value of C0 to compute the output carry directly using the logic block for A.B + C.(A+B) and use these instead of the G values for computation of P and G for the next level.

c) Using the outputs of the tree above, write structural VHDL code for generating the bit wise sum and carry values. Test the final adder with a test bench which reads pairs of 16 bit words and a single bit input carry value from a file, applies these to the adder and compares the result with the expected 16 bit sum and 1 bit carry values stored in the same file.
This test should be carried out for 10 randomly chosen input combinations.
It should use assert statements to flag errors if there is a mismatch between the computed sum/carry and the stored sum/carry.

```
-- 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 300 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 600 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 400 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 600 ps;
end architecture trivial;
```

HDL CODE

The delays for trivial architectures are taken as per the roll number 23M1117.

```vhdl
-- 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 317 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 634 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 417 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 634 ps;
end architecture trivial;


-- Brent Kung Adder


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

entity brent_kung is
```

```
port(A, B : in std_ulogic_vector (31 downto 0);
        Cin : in std_ulogic;
    Sum : out std_ulogic_vector (31 downto 0);
    Cout : out std_ulogic);
end entity;

architecture rtl of brent_kung is

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

component xorgate is
    port (A, B: in std_ulogic;
    uneq: out std_ulogic);
end component;

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

component Cin_map_G is
    port(A, B, Cin: in std_ulogic;
    Bit0_G: out std_ulogic);
end component;

signal g1, p1 : std_ulogic_vector (31 downto 0);
signal g2, p2 : std_ulogic_vector (15 downto 0);
signal g3, p3 : std_ulogic_vector (7 downto 0);
signal g4, p4 : std_ulogic_vector (3 downto 0);
signal g5, p5 : std_ulogic_vector (1 downto 0);
signal g6, p6 : std_ulogic;

signal temp_C : std_ulogic_vector (32 downto 0);
signal temp_s : std_ulogic_vector (31 downto 0);

begin

-- Finding different order Generate and Propogate values.

-- 1st Order:

order_1 : for i in 0 to 31 generate
  if_gen :if i=0 generate
  G_1_0 : Cin_map_G port map (a(i),b(i), cin, g1(i));
end generate if_gen;
  if_gen2 : if i>0 generate
    G_1 : andgate port map (a(i), b(i), g1(i));

end generate if_gen2;
P_1 : xorgate port map (a(i), b(i), p1(i));
end generate order_1;

-- 2nd Order:

order_2 : for i in 0 to 15 generate
    G_2 : abcgate port map (g1((2*i)+1), p1((2*i)+1), g1(2*i), g2(i));
    P_2 : andgate port map (p1((2*i)+1), p1(2*i), p2(i));
end generate order_2;

```

```

-- 3rd Order:
order_3 : for i in 0 to 7 generate
    G_3 : abcgate port map (g2((2*i)+1), p2((2*i)+1), g2(2*i), g3(i));
    P_3 : andgate port map (p2((2*i)+1), p2(2*i), p3(i));
end generate order_3;


-- 4th Order:
order_4 : for i in 0 to 3 generate
    G_4 : abcgate port map (g3((2*i)+1), p3((2*i)+1), g3(2*i), g4(i));
    P_4 : andgate port map (p3((2*i)+1), p3(2*i), p4(i));
end generate order_4;


--5th Order:
order_5 : for i in 0 to 1 generate
    G_5 : abcgate port map (g4((2*i)+1), p4((2*i)+1), g4((2*i)), g5(i));
    P_5 : andgate port map (p4((2*i)+1), p4((2*i)), p5(i));
end generate order_5;



--6th order:

order_6_G_5 : abcgate port map (g5(1), p5(1), g5(0), g6);
order_6_P_5 : andgate port map (p5(1), p5(0), p6);

-- Generate different Carry

temp_c(0) <= cin;
temp_c(1) <= G1(0);
temp_c(2) <= G2(0);
temp_c_3: abcgate port map (G1(2), P1(2), temp_c(2), temp_c(3));
temp_c(4) <= G3(0);
temp_c_5: abcgate port map (G1(4), P1(4), temp_c(4), temp_c(5));
temp_c_6: abcgate port map (G2(2), P2(2), temp_c(4), temp_c(6));
temp_c_7: abcgate port map (G1(6), P1(6), temp_c(6), temp_c(7));
temp_c(8) <= G4(0);
temp_c_9: abcgate port map (G1(8), P1(8), temp_c(8), temp_c(9));
temp_c_10: abcgate port map (G2(4), P2(4), temp_c(8), temp_c(10));
temp_c_11: abcgate port map (G1(10), P1(10), temp_c(10), temp_c(11));
temp_c_12: abcgate port map (G3(2), P3(2), temp_c(8), temp_c(12));
temp_c_13: abcgate port map (G1(12), P1(12), temp_c(12), temp_c(13));
temp_c_14: abcgate port map (G2(6), P2(6), temp_c(12), temp_c(14));
temp_c_15: abcgate port map (G1(14), P1(14), temp_c(14), temp_c(15));
temp_c(16) <= G5(0);
temp_c_17: abcgate port map (G1(16), P1(16), temp_c(16), temp_c(17));
temp_c_18: abcgate port map (G2(8), P2(8), temp_c(16), temp_c(18));
temp_c_19: abcgate port map (G1(18), P1(18), temp_c(18), temp_c(19));
temp_c_20: abcgate port map (G3(4), P3(4), temp_c(16), temp_c(20));
temp_c_21: abcgate port map (G1(20), P1(20), temp_c(20), temp_c(21));
temp_c_22: abcgate port map (G2(10), P2(10), temp_c(20), temp_c(22));
temp_c_23: abcgate port map (G1(22), P1(22), temp_c(22), temp_c(23));
temp_c_24: abcgate port map (G4(2), P4(2), temp_c(16), temp_c(24));
temp_c_25: abcgate port map (G1(24), P1(24), temp_c(24), temp_c(25));
temp_c_26: abcgate port map (G2(12), P2(12), temp_c(24), temp_c(26));
temp_c_27: abcgate port map (G1(26), P1(26), temp_c(26), temp_c(27));
temp_c_28: abcgate port map (G3(6), P3(6), temp_c(24), temp_c(28));
temp_c_29: abcgate port map (G1(28), P1(28), temp_c(28), temp_c(29));
temp_c_30: abcgate port map (G2(14), P2(14), temp_c(28), temp_c(30));
temp_c_31: abcgate port map (G1(30), P1(30), temp_c(30), temp_c(31));
```

```
temp_c(32) <= G6;


sum_out : for i in 0 to 31 generate
  sum_is : xorgate port map (P1(i), temp_c(i), temp_s(i));
end generate sum_out;

sum <= temp_s;
cout <= temp_c(32);

end architecture rtl;
```

TestBench

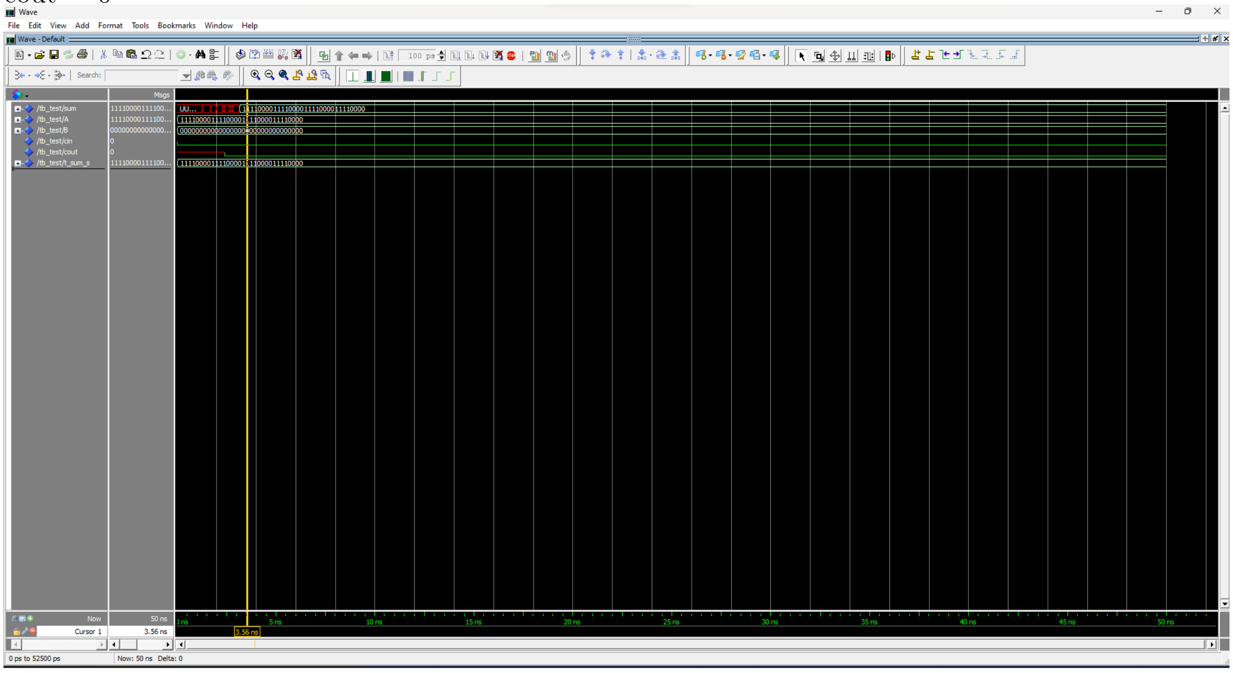
The inputs A, B, Cin, t_sum, and t_cout are taken from a file called **"inputs.txt"**, and used to compare the results and assert statements for success or error.

```
library STD;
library IEEE;
use std.textio.all;
use ieee.std_logic_textio.all;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity tb_test is
end entity;

architecture tb_rtl of tb_test is

component brent_kung is
port(A, B : in std_ulogic_vector (31 downto 0);
        Cin : in std_ulogic;
    Sum : out std_ulogic_vector (31 downto 0);
    Cout : out std_ulogic);
end component;

signal t_sum : std_ulogic_vector(32 downto 0);
signal sum : std_ulogic_vector(31 downto 0);
signal t_cout : std_ulogic;
signal A : std_ulogic_vector(31 downto 0);
signal B : std_ulogic_vector(31 downto 0);
signal cin : std_ulogic;
signal cout : std_ulogic;
signal t_sum_s : std_ulogic_vector(31 downto 0);


begin

process

file infile: text open read_mode is "inputs.txt";

variable t_condition_sum : boolean := FALSE;
variable t_condition_carry : boolean := FALSE;

variable input_line : line;
variable A_in : std_ulogic_vector(31 downto 0);
variable B_in : std_ulogic_vector(31 downto 0);
variable t_sum : std_ulogic_vector(31 downto 0);
variable C_in : std_ulogic;
variable t_cout : std_ulogic;


begin
while not endfile(infile) loop

  readline(infile, input_line);
    read(input_line, A_in);
    read(input_line, B_in);
    read(input_line, C_in);
    read(input_line, t_sum);
    read(input_line, t_cout);

A <= A_in;
```

```
B <= B_in;
cin <= C_in;
t_sum_s <= t_sum;

wait for 50 ns;

if t_sum_s = sum then
    t_condition_sum := TRUE;
  else
    t_condition_sum := FALSE;
  end if;

  if t_cout = cout then
    t_condition_carry := TRUE;
  else
    t_condition_carry := FALSE;
  end if;


assert t_condition_sum report "Success sum!!" severity note;
assert not t_condition_sum report "Error sum" severity note;
assert t_condition_carry report "Success Cout!!" severity note;
assert not t_condition_carry report "error cout!!" severity note;

end loop;


end process;

tb_brent_kung : brent_kung port map (A, B, cin, Sum, Cout);


end architecture tb_rtl;
```

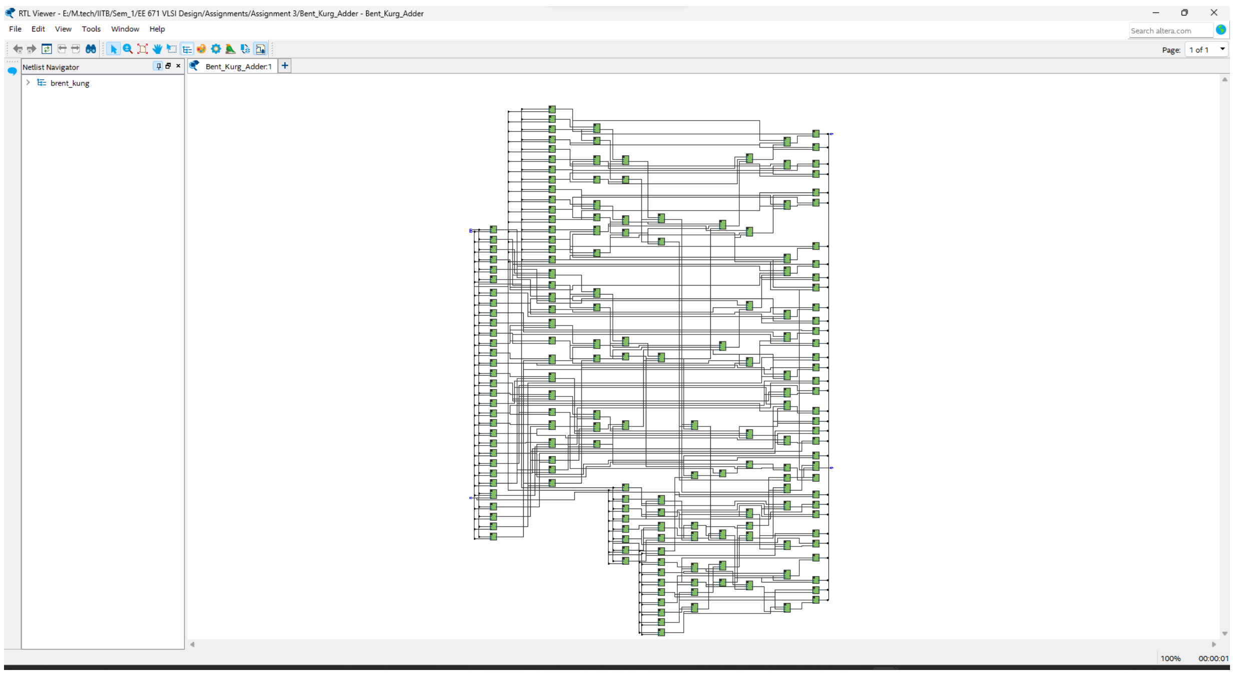
waveform

As seen, the inputs are taken as from the file as
inputs:
A = 11110000111100001111000011110000
B = 00000000000000000000000000000000
Cin = 0
outputs:
sum = 11110000111100001111000011110000
cout = 0



/tb_test/sum 11110000111100... UU... 11110000111100001111000011110000
/tb_test/A 11110000111100... 11110000111100001111000011110000
/tb_test/B 00000000000000... 00000000000000000000000000000000
/tb_test/cin 0
/tb_test/cout 0
/tb_test/t_sum_s 11110000111100... 11110000111100001111000011110000

RTL View


RTL Viewer - E:/M.tech/IITB/Sem_1/EE 671 VLSI Design/Assignments/Assignment 3/Bent_Kurg_Adder - Bent_Kurg_Adder
File Edit View Tools Window Help
Search altera.com
Page: 1 of 1
Netlist Navigator
brent_kung
Bent_Kurg_Adder:1
100%
00:00:01