EE671: Assignment 3

Name: Anoushka Dey

Roll No: 210010010

October 2023

1 32 Bit Brent Kung Adder

For this assignment, I have used the Quartus Prime software for writing the VHDL code as well as for the simulation. I first started off with changing the delay values of the gates as in the template code provided. I created the gates.vhdl file and using the last two digits of my roll number which are '10', I made the requisite changes in the delay values. This was followed by the main structural description of the 32 bit Brent Kung Adder.
I first started with the instantiation of the gates and the declaration of the Brent Kung entity. The inputs to the adder are the 32 bit values of A and B, the numbers to be added and the value of the input carry Cin. The outputs are the 32 bit sum denoted by S and the output carry denoted by Cout. The first level P and G values were generated using the generate function, using the xor gate for P and the and gate for G. These values were stored in 32 bit std_logic_vectors. The only difference in the case of the G vector was the value of G(0). This was directly calculated using the Cin_map_G by plugging in the values of A(0), B(0) and Cin. The carry after every stage is denoted by a bit in the 33 bit 'c' vector, c(0) denoting Cin and c(16) denoting Cout and all intermediate values in the vector denoting the intermediate carry values. The next steps involved the generation of the 2 bit, 4 bit, 8 bit, 16 bit and 32 bit P and G values.
After all P and G values for every stage was generated, I went ahead with the carry generation. Finally the sum vector was also computed using the generate function. The final output consists of Cout and the sum S. I also wrote the testbench for the simulation of the adder. The 10 testcases were randomly generated using python and the testbench was modified to indicate if all testcases pass successfully or if they do not. The following figures show the waveform obtained on simulation on Quartus Prime using the ModelSim-Altera simulator. These is the VHDL code files that I have written:

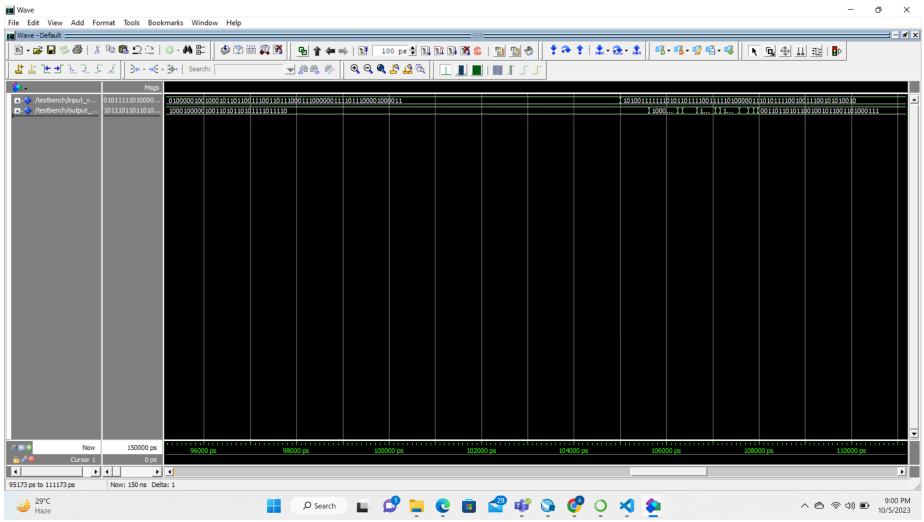
```
--Structural Description of the 32 bit Brent Kung Adder
library IEEE;
use IEEE.std_logic_1164.all;
```

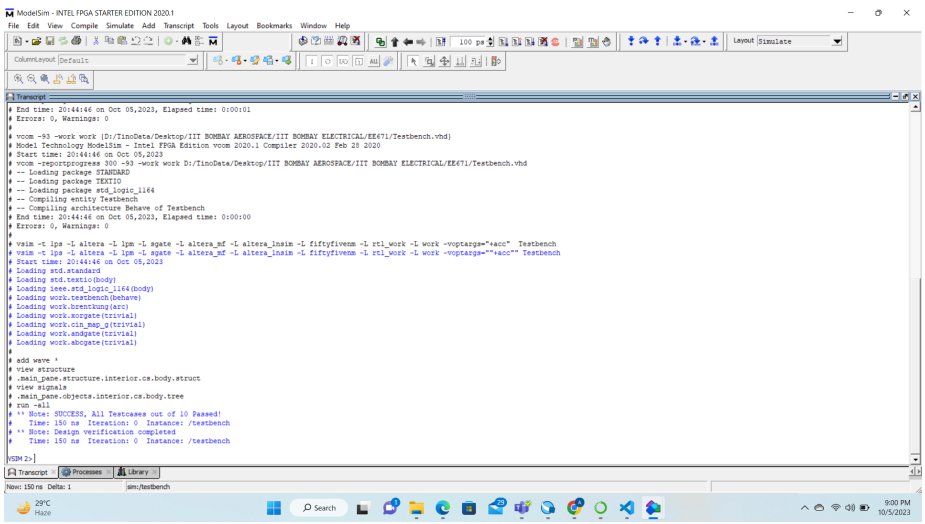
```
entity brentkung is
port(A,B: in std_logic_vector(31 downto 0);
Cin: in std_logic; S: out std_logic_vector(31 downto 0);
Cout: out std_logic);
end entity;

architecture arc of brentkung is


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

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

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

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

--Signal Declarations
signal p0,g0: std_logic_vector(31 downto 0);
signal c: std_logic_vector(32 downto 0);
signal p1_10,g1_10,p1_32,g1_32,p1_54,g1_54,p1_76,g1_76,p1_98,g1_98,
p1_1110,g1_1110,p1_1312,g1_1312,p1_1514,g1_1514,p1_1716,g1_1716,p1_1918,
g1_1918,p1_2120,g1_2120,p1_2322,g1_2322,p1_2524,g1_2524,p1_2726,
g1_2726,p1_2928,g1_2928,p1_3130,g1_3130:std_logic;
signal p2_30,g2_30,p2_74,g2_74,p2_118,g2_118,p2_1512,g2_1512,p2_1916
,g2_1916,p2_2320,g2_2320,p2_2724,g2_2724,p2_3128,g2_3128:std_logic;
signal p3_70,g3_70,p3_158,g3_158,p3_2316,g3_2316,p3_3124,g3_3124:std_logic;
signal p4_150,g4_150,p4_3116,g4_3116:std_logic;
signal p5_310,g5_310:std_logic;

begin
```

```
c(0)<=Cin;
--Single bit Pi values
xor_p0: for i in 0 to 31 generate
x0:xorgate port map(A=>A(i),B=>B(i),uneq=>p0(i));
end generate xor_p0;

--Single bit Gi values
cmap1: Cin_map_G port map(A=>a(0),B=>b(0),Cin=>Cin, Bit0_G=>g0(0));
and_g0: for i in 1 to 31 generate
a0:andgate port map(A=>A(i),B=>B(i),prod=>g0(i));
end generate and_g0;

--Two bit Pi values
and1: andgate port map(A=>p0(0),B=>p0(1),prod=>p1_10);
and2: andgate port map(A=>p0(2),B=>p0(3),prod=>p1_32);
and3: andgate port map(A=>p0(4),B=>p0(5),prod=>p1_54);
and4: andgate port map(A=>p0(6),B=>p0(7),prod=>p1_76);
and5: andgate port map(A=>p0(8),B=>p0(9),prod=>p1_98);
and6: andgate port map(A=>p0(10),B=>p0(11),prod=>p1_1110);
and7: andgate port map(A=>p0(12),B=>p0(13),prod=>p1_1312);
and8: andgate port map(A=>p0(14),B=>p0(15),prod=>p1_1514);
and9: andgate port map(A=>p0(16),B=>p0(17),prod=>p1_1716);
and10: andgate port map(A=>p0(18),B=>p0(19),prod=>p1_1918);
and11: andgate port map(A=>p0(21),B=>p0(20),prod=>p1_2120);
and12: andgate port map(A=>p0(22),B=>p0(23),prod=>p1_2322);
and13: andgate port map(A=>p0(24),B=>p0(25),prod=>p1_2524);
and14: andgate port map(A=>p0(26),B=>p0(27),prod=>p1_2726);
and15: andgate port map(A=>p0(28),B=>p0(29),prod=>p1_2928);
and16: andgate port map(A=>p0(30),B=>p0(31),prod=>p1_3130);


--Two bit Gi values
abc1: abcgate port map(A=>g0(1),B=>p0(1),C=>g0(0),abc=>g1_10);
abc2: abcgate port map(A=>g0(3),B=>p0(3),C=>g0(2),abc=>g1_32);
abc3: abcgate port map(A=>g0(5),B=>p0(5),C=>g0(4),abc=>g1_54);
abc4: abcgate port map(A=>g0(7),B=>p0(7),C=>g0(6),abc=>g1_76);
abc5: abcgate port map(A=>g0(9),B=>p0(9),C=>g0(8),abc=>g1_98);
abc6: abcgate port map(A=>g0(11),B=>p0(11),C=>g0(10),abc=>g1_1110);
abc7: abcgate port map(A=>g0(13),B=>p0(13),C=>g0(12),abc=>g1_1312);
abc8: abcgate port map(A=>g0(15),B=>p0(15),C=>g0(14),abc=>g1_1514);
abc9: abcgate port map(A=>g0(17),B=>p0(17),C=>g0(16),abc=>g1_1716);
abc10: abcgate port map(A=>g0(19),B=>p0(19),C=>g0(18),abc=>g1_1918);
abc11: abcgate port map(A=>g0(21),B=>p0(21),C=>g0(20),abc=>g1_2120);
abc12: abcgate port map(A=>g0(23),B=>p0(23),C=>g0(22),abc=>g1_2322);
abc13: abcgate port map(A=>g0(25),B=>p0(25),C=>g0(24),abc=>g1_2524);
abc14: abcgate port map(A=>g0(27),B=>p0(27),C=>g0(26),abc=>g1_2726);
```

```
abc15: abcgate port map(A=>g0(29),B=>p0(29),C=>g0(28),abc=>g1_2928);
abc16: abcgate port map(A=>g0(31),B=>p0(31),C=>g0(30),abc=>g1_3130);

--Four bit Pi values
and17: andgate port map(A=>p1_10,B=>p1_32,prod=>p2_30);
and18: andgate port map(A=>p1_54,B=>p1_76,prod=>p2_74);
and19: andgate port map(A=>p1_1110,B=>p1_98,prod=>p2_118);
and20: andgate port map(A=>p1_1514,B=>p1_1312,prod=>p2_1512);
and21: andgate port map(A=>p1_1716,B=>p1_1918,prod=>p2_1916);
and22: andgate port map(A=>p1_2120,B=>p1_2322,prod=>p2_2320);
and23: andgate port map(A=>p1_2524,B=>p1_2726,prod=>p2_2724);
and24: andgate port map(A=>p1_2928,B=>p1_3130,prod=>p2_3128);

--Four bit Gi values
abc17: abcgate port map(A=>g1_32,B=>p1_32,C=>g1_10,abc=>g2_30);
abc18: abcgate port map(A=>g1_76,B=>p1_76,C=>g1_54,abc=>g2_74);
abc19: abcgate port map(A=>g1_1110,B=>p1_1110,C=>g1_98,abc=>g2_118);
abc20: abcgate port map(A=>g1_1514,B=>p1_1514,C=>g1_1312,abc=>g2_1512);
abc21: abcgate port map(A=>g1_1918,B=>p1_1918,C=>g1_1716,abc=>g2_1916);
abc22: abcgate port map(A=>g1_2322,B=>p1_2322,C=>g1_2120,abc=>g2_2320);
abc23: abcgate port map(A=>g1_2726,B=>p1_2726,C=>g1_2524,abc=>g2_2724);
abc24: abcgate port map(A=>g1_3130,B=>p1_3130,C=>g1_2928,abc=>g2_3128);


--Eight bit Pi values
and25: andgate port map(A=>p2_74,B=>p2_30,prod=>p3_70);
and26: andgate port map(A=>p2_1512,B=>p2_118,prod=>p3_158);
and27: andgate port map(A=>p2_2320,B=>p2_1916,prod=>p3_2316);
and28: andgate port map(A=>p2_3128,B=>p2_2724,prod=>p3_3124);

--Eight bit Gi values
abc25: abcgate port map(A=>g2_74, B=>p2_74, C=>g2_30,abc=>g3_70);
abc26: abcgate port map(A=>g2_1512, B=>p2_1512, C=>g2_118,abc=>g3_158);
abc27: abcgate port map(A=>g2_2320, B=>p2_2320, C=>g2_1916,abc=>g3_2316);
abc28: abcgate port map(A=>g2_3128, B=>p2_3128, C=>g2_2724,abc=>g3_3124);


--Sixteen bit Pi values
and29: andgate port map(A=>p3_158,B=>p3_70,prod=>p4_150);
and30: andgate port map(A=>p3_3124,B=>p3_2316,prod=>p4_3116);

--Sixteen bit Gi values
abc29: abcgate port map(A=>g3_158,B=>p3_158,C=>g3_70,abc=>g4_150);
abc30: abcgate port map(A=>g3_3124,B=>p3_3124,C=>g3_2316,abc=>g4_3116);

--Thirty two bit Pi value
```

```
and31: andgate port map(A=>p4_150,B=>p4_3116,prod=>p5_310);

--Thirty two bit Gi value
abc31: abcgate port map(A=>g4_3116,B=>p4_3116,C=>g4_150,abc=>g5_310);

--Carry Declarations
c(1)<=g0(0);
carry2:abcgate port map(A=>g1_10,B=>p1_10,C=>Cin,abc=>c(2));
carry3:abcgate port map(A=>g0(2),B=>p0(2),C=>c(2),abc=>c(3));
carry4:abcgate port map(A=>g2_30,B=>p2_30,C=>Cin,abc=>c(4));
carry5:abcgate port map(A=>g0(4),B=>p0(4),C=>c(4),abc=>c(5));
carry6:abcgate port map(A=>g1_54,B=>p1_54,C=>c(4),abc=>c(6));
carry7:abcgate port map(A=>g0(6),B=>p0(6),C=>c(6),abc=>c(7));
carry8:abcgate port map(A=>g3_70,B=>p3_70,C=>Cin,abc=>c(8));
carry9:abcgate port map(A=>g0(8),B=>p0(8),C=>c(8),abc=>c(9));
carry10:abcgate port map(A=>g1_98,B=>p1_98,C=>c(8),abc=>c(10));
carry11:abcgate port map(A=>g0(10),B=>p0(10),C=>c(10),abc=>c(11));
carry12:abcgate port map(A=>g2_118,B=>p2_118,C=>c(8),abc=>c(12));
carry13:abcgate port map(A=>g0(12),B=>p0(12),C=>c(12),abc=>c(13));
carry14:abcgate port map(A=>g1_1312,B=>p1_1312,C=>c(12),abc=>c(14));
carry15:abcgate port map(A=>g0(14),B=>p0(14),C=>c(14),abc=>c(15));
carry16:abcgate port map(A=>g4_150,B=>p4_150,C=>Cin,abc=>c(16));
carry17:abcgate port map(A=>g0(16),B=>p0(16),C=>c(16),abc=>c(17));
carry18:abcgate port map(A=>g1_1716,B=>p1_1716,C=>c(16),abc=>c(18));
carry19:abcgate port map(A=>g0(18),B=>p0(18),C=>c(18),abc=>c(19));
carry20:abcgate port map(A=>g2_1916,B=>p2_1916,C=>c(16),abc=>c(20));
carry21:abcgate port map(A=>g0(20),B=>p0(20),C=>c(20),abc=>c(21));
carry22:abcgate port map(A=>g1_2120,B=>p1_2120,C=>c(20),abc=>c(22));
carry23:abcgate port map(A=>g0(22),B=>p0(22),C=>c(22),abc=>c(23));
carry24:abcgate port map(A=>g3_2316,B=>p3_2316,C=>c(16),abc=>c(24));
carry25:abcgate port map(A=>g0(24),B=>p0(24),C=>c(24),abc=>c(25));
carry26:abcgate port map(A=>g1_2524,B=>p1_2524,C=>c(24),abc=>c(26));
carry27:abcgate port map(A=>g0(26),B=>p0(26),C=>c(26),abc=>c(27));
carry28:abcgate port map(A=>g2_2724,B=>p2_2724,C=>c(24),abc=>c(28));
carry29:abcgate port map(A=>g0(28),B=>p0(28),C=>c(28),abc=>c(29));
carry30:abcgate port map(A=>g1_2928,B=>p1_2928,C=>c(28),abc=>c(30));
carry31:abcgate port map(A=>g0(30),B=>p0(30),C=>c(30),abc=>c(31));
carry32:abcgate port map(A=>g5_310,B=>p5_310,C=>Cin,abc=>c(32));
Cout<=c(32);

--Sum Generation
sums: for i in 0 to 31 generate
x2:xorgate port map(A=>p0(i),B=>c(i),uneq=>S(i));
end generate sums;
end arc;
```


Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
/TestBench/input_v...
/TestBench/output_...
Now
150000 ps
Cursor 1
0 ps
95173 ps to 111173 ps
Now: 150 ns Delta: 1
29°C
Haze
Search
9:00 PM
10/5/2023


ModelSim - INTEL FPGA STARTER EDITION 2020.1
File Edit View Compile Simulate Add Transcript Tools Layout Bookmarks Window Help
Transcript
End time: 20:44:46 on Oct 05,2023, Elapsed time: 0:00:01
Errors: 0, Warnings: 0
#
vcom -93 -work work {D:/TinoData/Desktop/IIT BOMBAY AEROSPACE/IIT BOMBAY ELECTRICAL/EE671/Testbench.vhd}
Model Technology ModelSim - Intel FPGA Edition vcom 2020.1 Compiler 2020.02 Feb 28 2020
Start time: 20:44:46 on Oct 05,2023
vcom -reportprogress 300 -93 -work work D:/TinoData/Desktop/IIT BOMBAY AEROSPACE/IIT BOMBAY ELECTRICAL/EE671/Testbench.vhd
-- Loading package STANDARD
-- Loading package TEXTIO
-- Loading package std_logic_1164
-- Compiling entity Testbench
-- Compiling architecture Behave of Testbench
End time: 20:44:46 on Oct 05,2023, Elapsed time: 0:00:00
Errors: 0, Warnings: 0
#
vsim -t 1ps -L altera -L lpm -L sgate -L altera_mf -L altera_lnsim -L fiftyfivenm -L rtl_work -L work -voptargs="+acc" Testbench
vsim -t 1ps -L altera -L lpm -L sgate -L altera_mf -L altera_lnsim -L fiftyfivenm -L rtl_work -L work -voptargs=""+acc"" Testbench
Start time: 20:44:46 on Oct 05,2023
Loading std.standard
Loading std.textio(body)
Loading ieee.std_logic_1164(body)
Loading work.testbench(behave)
Loading work.brentkung(arc)
Loading work.xorgate(trivial)
Loading work.cin_map_g(trivial)
Loading work.andgate(trivial)
Loading work.abcgate(trivial)
#
add wave *
view structure
.main_pane.structure.interior.cs.body.struct
view signals
.main_pane.objects.interior.cs.body.tree
run -all
** Note: SUCCESS, All Testcases out of 10 Passed!
Time: 150 ns Iteration: 0 Instance: /testbench
** Note: Design verification completed
Time: 150 ns Iteration: 0 Instance: /testbench
VSIM 2>
Transcript
Processes
Library
Now: 150 ns Delta: 1
sim:/testbench
29°C
Haze
Search
9:00 PM
10/5/2023