Experiment 1 Part-A: XOR Using NAND Gates

Name: Arya Agarwal Roll Number:210070012

August 7, 2022

1 Overview of the Experiment

- The purpose of the experiment is to design a XOR gate.
- I used NAND gates to create the design.
- There would be two input bits and only one output bit.

2 Approach to the Experiment

I used two input bits and five NAND gates to complete the experiment. Two NAND gates were used to get the negation of the two input bits . My pen paper design is shown below in Figure 1.

3 Include your design documentation and code if relevant

The two bits A and B are first fed in N1 and N2 NAND gates seperately as shown in Figure 1 so that we get negation of A and negation of B respectively. Then A-bar and B are fed together in N3 gate and B-bar and A in N4.Finally the output bits from both are fed in the N5 NAND gate. Here is my VHDL code. You can refer to Figure 1 for signal names and instance names.

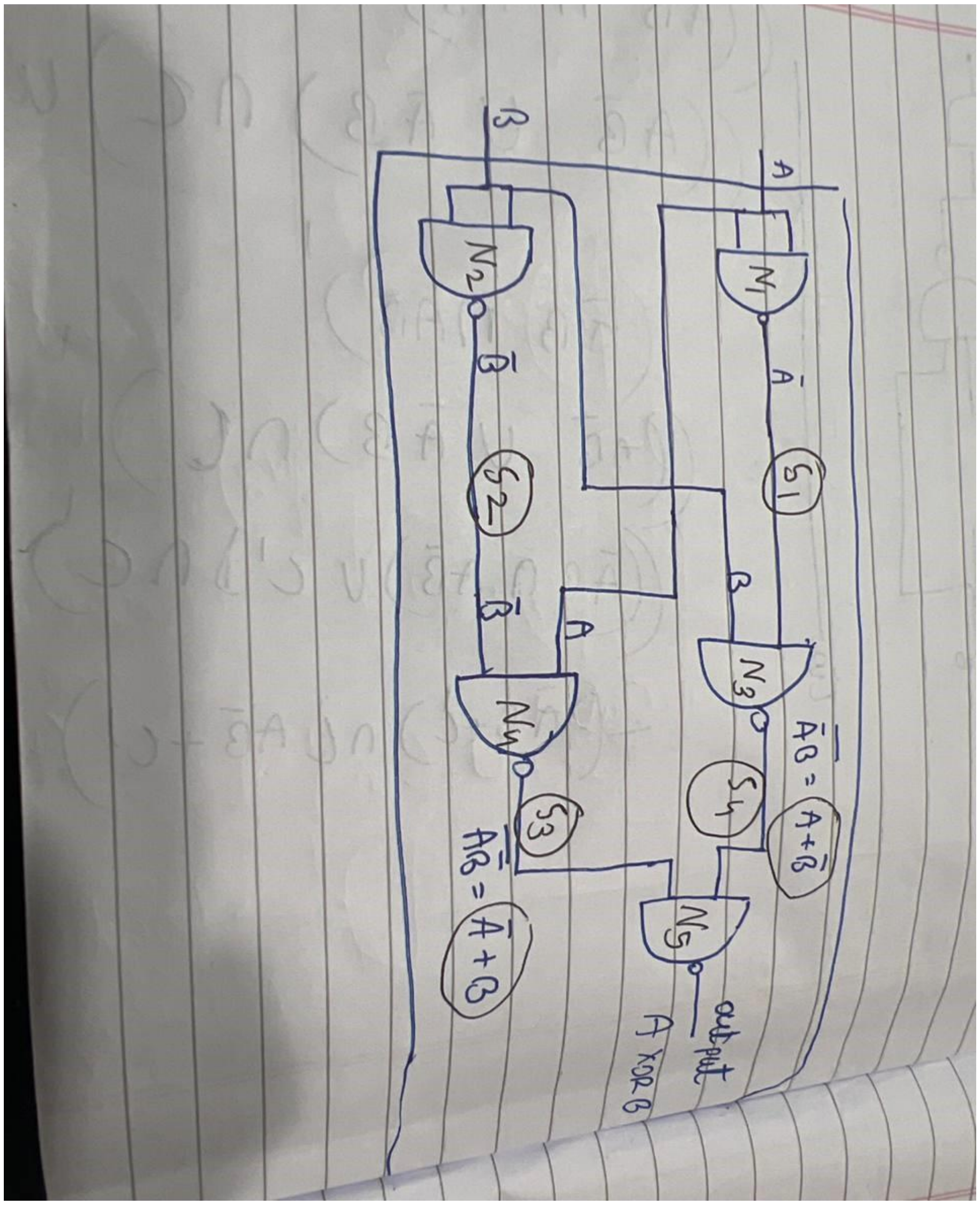
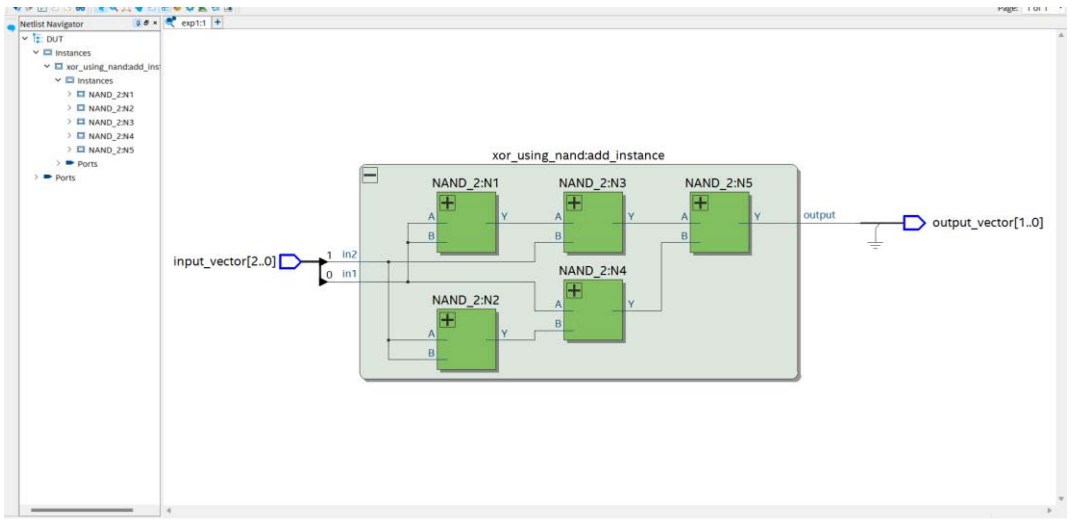

A
B
N1
N2
N3
N4
N5
Ā
B̄
S1
S2
S3
S4
$\overline{\bar{A}B} = \bar{\bar{A}} + \bar{B}$
$\overline{A\bar{B}} = \bar{A} + \bar{\bar{B}}$
output
A XOR B

Figure 1: Pen Paper Design.

VHDL CODE FOR XOR USING NAND GATES

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

library work; use
work.Gates.all;

entity xor_using_nand is
    port( in1, in2:in std_logic;
output:out std_logic;
); end entity xor_using_nand

architecture trivial of xor_using_nand is
    for all: NAND_2
        use entity work.NAND_2(Equations)
    signal s1, s2, s3, s4: std_logic;
  begin

N1:component NAND_2 port
map(in1,in1,s1)
N2:component NAND_2 port
map(in2,in2,s2)
N3:component NAND_2 port
map(s1,in2,s4)
N4:component NAND_2 port
map(in1,s2,s3)
N5:component NAND_2 port
map(s4,s3,output) end
trivial;
```

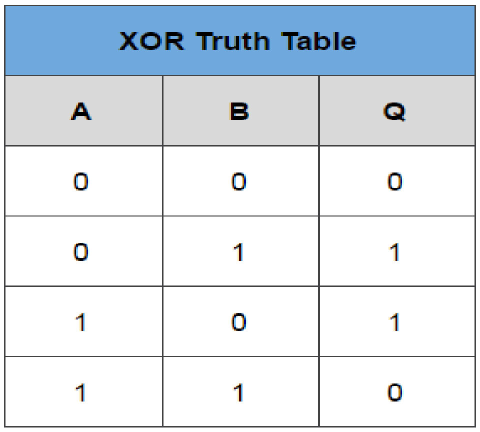
RTL VIEW



4 OBSERVATIONS

Testbench was then finally compiled using the tracefile provided in which the first column contains 2 bits in which the first one is in1 and the second one is in2 and the second column corresponds to the final output. Finally we observe that the digital logic is correct and all the test cases have been successfully passed. You can refer to the truth table and RTL simulation below.

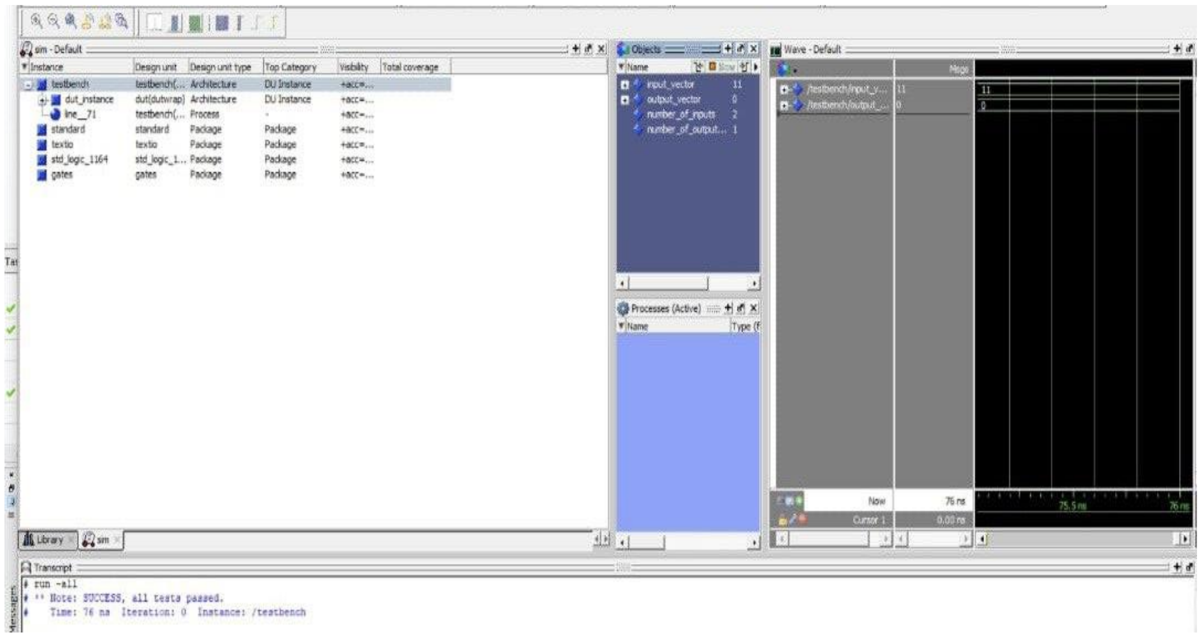
TRUTH TABLE



XOR Truth Table		
A	**B**	**Q**
0	0	0
0	1	1
1	0	1
1	1	0

Here A is in1 and B is in2 and Q is the output.

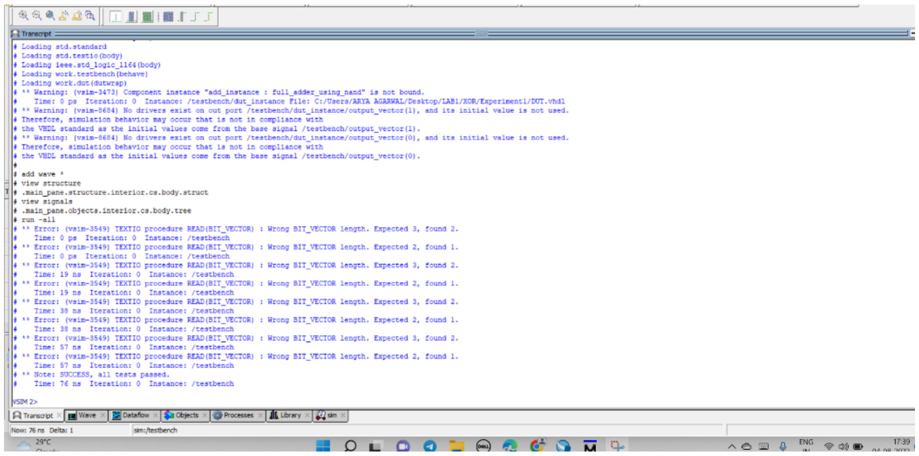
RTL SIMULATION



Instance	Design unit	Design unit type	Top Category	Visibility	Total coverage
testbench	testbench(...	Architecture	DU Instance	+acc=...	
dut_instance	dut(dutwrap)	Architecture	DU Instance	+acc=...	
line__71	testbench(...	Process	-	+acc=...	
standard	standard	Package	Package	+acc=...	
textio	textio	Package	Package	+acc=...	
std_logic_1164	std_logic_1...	Package	Package	+acc=...	
gates	gates	Package	Package	+acc=...	

```
# run -all
# ** Note: SUCCESS, all tests passed.
#    Time: 76 ns  Iteration: 0  Instance: /testbench
```

TRANSCRIPT



```
# Loading std.standard
# Loading std.textio(body)
# Loading ieee.std_logic_1164(body)
# Loading work.testbench(behave)
# Loading work.dut(dutwrap)
# ** Warning: (vsim-3473) Component instance "add_instance : full_adder_using_nand" is not bound.
#    Time: 0 ps  Iteration: 0  Instance: /testbench/dut_instance File: C:/Users/ARYA AGARWAL/Desktop/LAB1/XOR/Experiment1/DUT.vhdl
# ** Warning: (vsim-8684) No drivers exist on out port /testbench/dut_instance/output_vector(1), and its initial value is not used.
# Therefore, simulation behavior may occur that is not in compliance with
# the VHDL standard as the initial values come from the base signal /testbench/output_vector(1).
# ** Warning: (vsim-8684) No drivers exist on out port /testbench/dut_instance/output_vector(0), and its initial value is not used.
# Therefore, simulation behavior may occur that is not in compliance with
# the VHDL standard as the initial values come from the base signal /testbench/output_vector(0).
#
# add wave *
# view structure
# .main_pane.structure.interior.cs.body.struct
# view signals
# .main_pane.objects.interior.cs.body.tree
# run -all
# ** Error: (vsim-3549) TEXTIO procedure READ(BIT_VECTOR) : Wrong BIT_VECTOR length. Expected 3, found 2.
#    Time: 0 ps  Iteration: 0  Instance: /testbench
# ** Error: (vsim-3549) TEXTIO procedure READ(BIT_VECTOR) : Wrong BIT_VECTOR length. Expected 2, found 1.
#    Time: 0 ps  Iteration: 0  Instance: /testbench
# ** Error: (vsim-3549) TEXTIO procedure READ(BIT_VECTOR) : Wrong BIT_VECTOR length. Expected 3, found 2.
#    Time: 19 ns  Iteration: 0  Instance: /testbench
# ** Error: (vsim-3549) TEXTIO procedure READ(BIT_VECTOR) : Wrong BIT_VECTOR length. Expected 2, found 1.
#    Time: 19 ns  Iteration: 0  Instance: /testbench
# ** Error: (vsim-3549) TEXTIO procedure READ(BIT_VECTOR) : Wrong BIT_VECTOR length. Expected 3, found 2.
#    Time: 38 ns  Iteration: 0  Instance: /testbench
# ** Error: (vsim-3549) TEXTIO procedure READ(BIT_VECTOR) : Wrong BIT_VECTOR length. Expected 2, found 1.
#    Time: 38 ns  Iteration: 0  Instance: /testbench
# ** Error: (vsim-3549) TEXTIO procedure READ(BIT_VECTOR) : Wrong BIT_VECTOR length. Expected 3, found 2.
#    Time: 57 ns  Iteration: 0  Instance: /testbench
# ** Error: (vsim-3549) TEXTIO procedure READ(BIT_VECTOR) : Wrong BIT_VECTOR length. Expected 2, found 1.
#    Time: 57 ns  Iteration: 0  Instance: /testbench
# ** Note: SUCCESS, all tests passed.
#    Time: 76 ns  Iteration: 0  Instance: /testbench

VSIM 2>
```

Now: 76 ns Delta: 1   sim:/testbench

Experiment 1 Part-B: FULL ADDER USING NAND GATES

Name:Arya Agarwal Roll Number:210070012

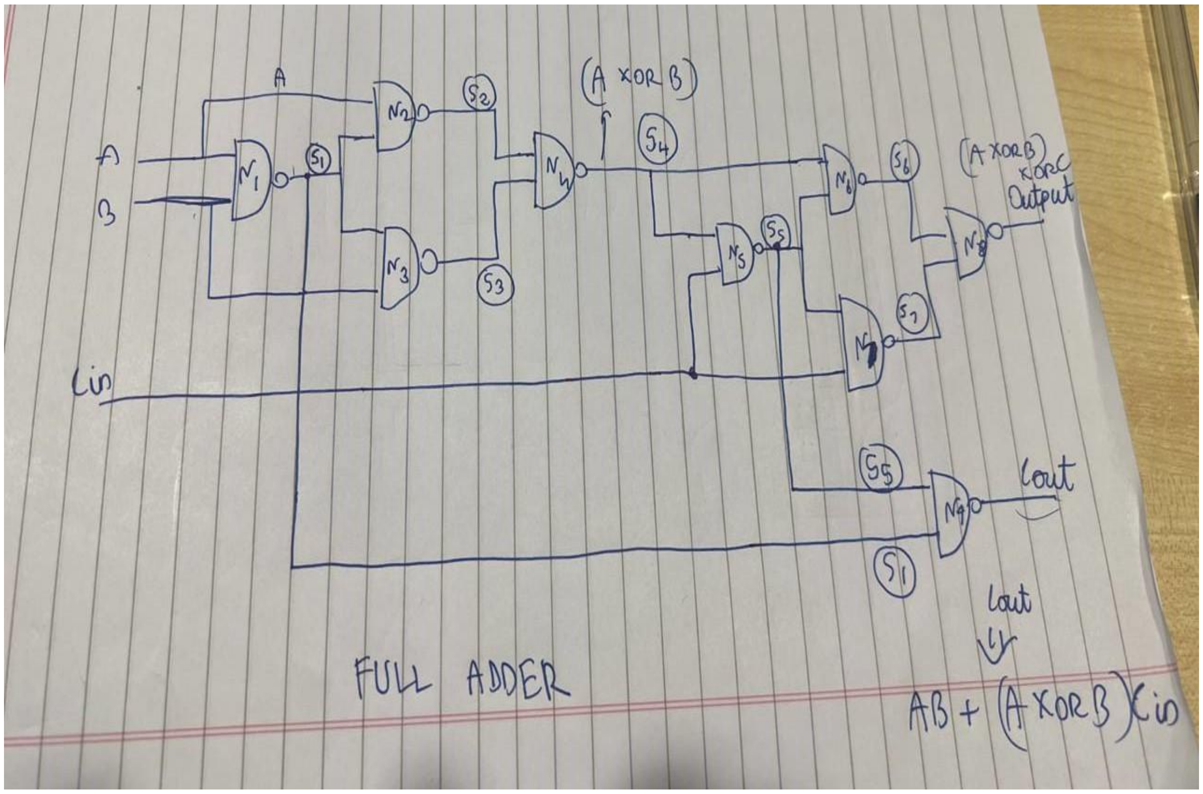
August 7, 2022

1 Overview of the Experiment

- The purpose of the experiment is to design a Full adder.
- We have used only NAND gates to create the design.
- There would be three input bits and two output bits.

2 Approach to the Experiment

I used three input bits out of which one is carry and nine NAND gates to complete the experiment . Four NAND gates were used to get the XOR of A and B. We finally get two outputs one is the sum and other is the carry (Cout). My pen-paper design is shown below.



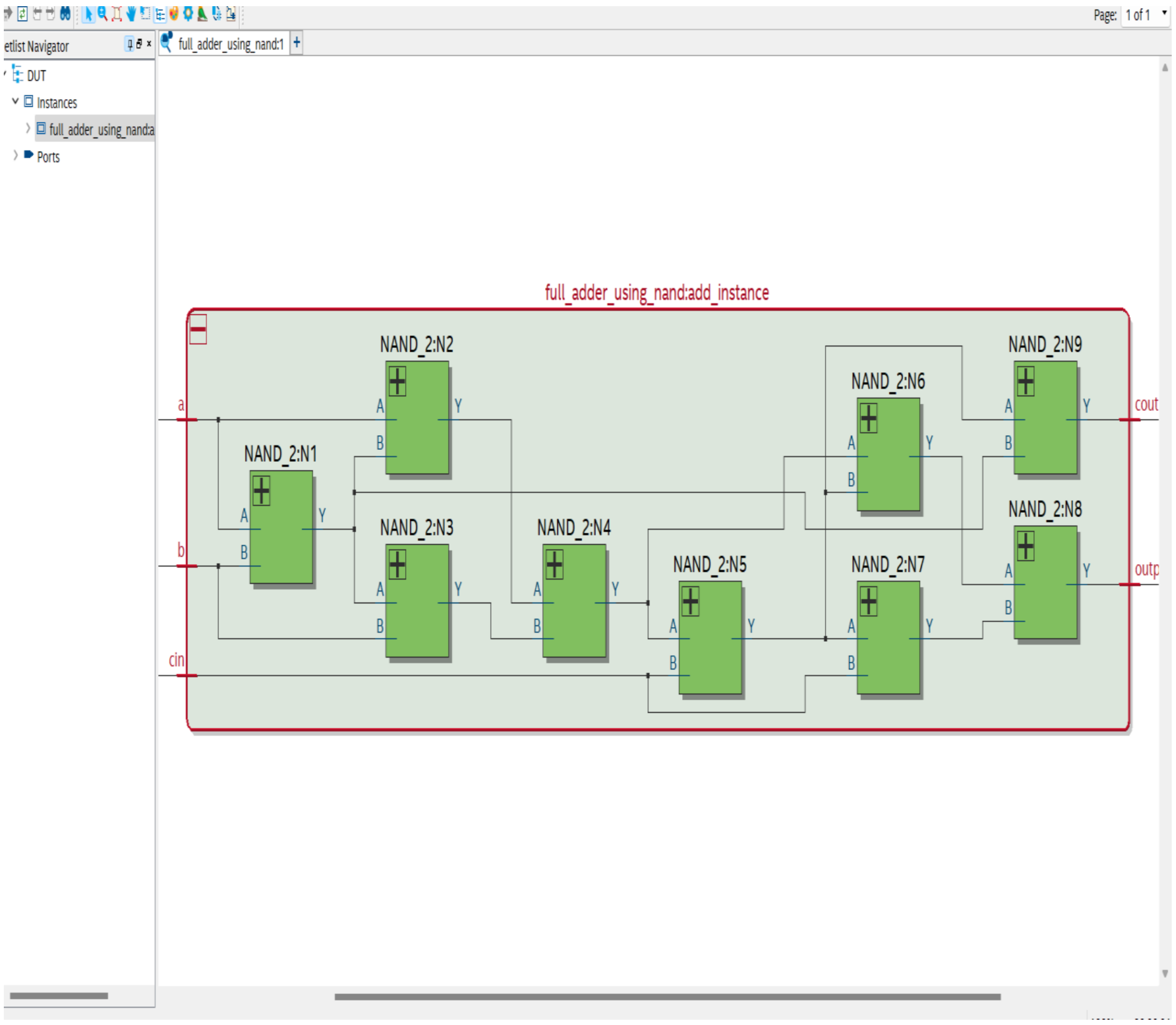
3 Include your design documentation and code if relevant

The NAND gates N1,N2,N3,N4 are designed to get the XOR relation between the bits A and B. Then Cin and (A XOR B) are fed in N5.The gates N6,N7,N8 are used to get the final output which is sum -((A XOR B) XOR Cin). Finally we have to use one NAND gate N9 to get the carry output Cout which is AB+(A XOR B)C . You can refer to the pen-paper diagram for component names and other thongs. Below is my VHDL code.

VHDL CODE FOR FULL ADDER USING NAND GATES

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

library work;
use work.Gates.all;

entity xor_using_nand is
  port(
      in1, in2:in std_logic;
            output:out std_logic;
            );
end entity xor_using_nand

architecture trivial of xor_using_nand is
  for all: NAND_2
    use entity work.NAND_2(Equations)
  signal s1, s2, s3, s4: std_logic;
 begin

N1:component NAND_2
      port map(in1,in1,s1)
N2:component NAND_2
      port map(in2,in2,s2)
N3:component NAND_2
      port map(s1,in2,s4)
N4:component NAND_2
      port map(in1,s2,s3)
N5:component NAND_2
      port map(s4,s3,output)
end trivial;
```

RTL VIEW

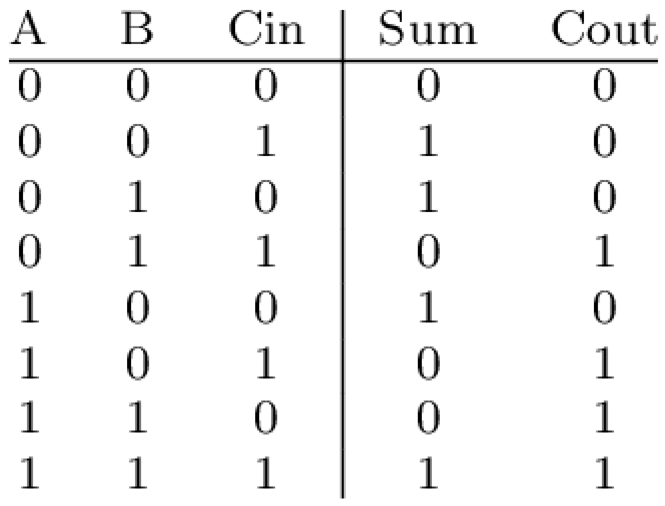


Page: 1 of 1
etlist Navigator
DUT
Instances
full_adder_using_nand:a
Ports
full_adder_using_nand:1
full_adder_using_nand:add_instance
NAND_2:N1
NAND_2:N2
NAND_2:N3
NAND_2:N4
NAND_2:N5
NAND_2:N6
NAND_2:N7
NAND_2:N8
NAND_2:N9
a
b
cin
cout
outp

4 OBSERVATIONS

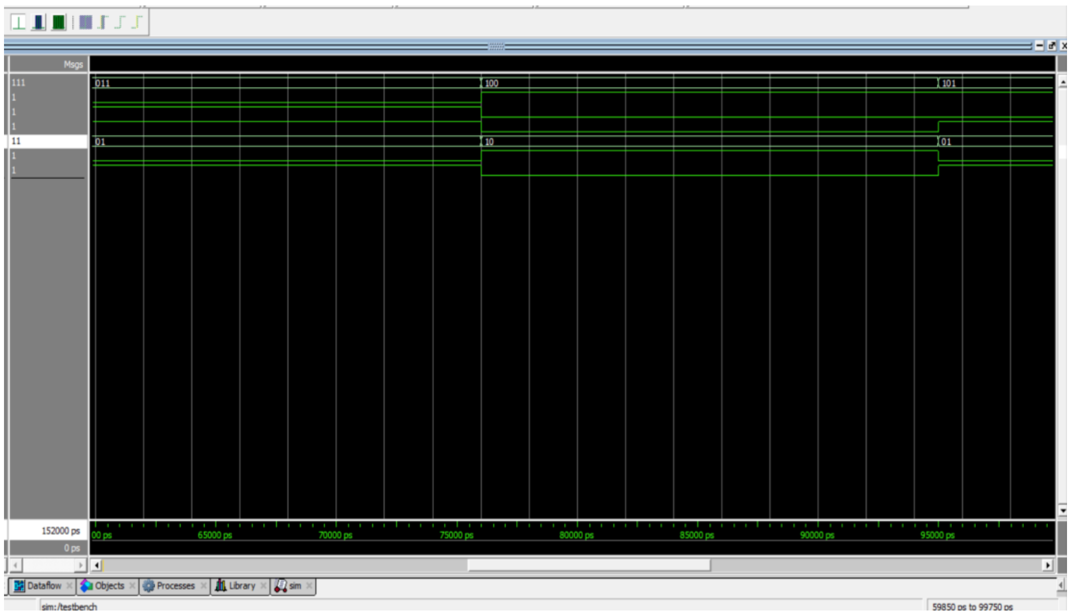
Testbench was then finally compiled using the tracefile provided in which the first column contains 3 bits in which the first one is A, the second one is B and the third input bit is Cin. Second column contains two output bits out of which first one is the sum bit and the second one is the carry bit. Please refer to the truth table given below.

TRUTH TABLE

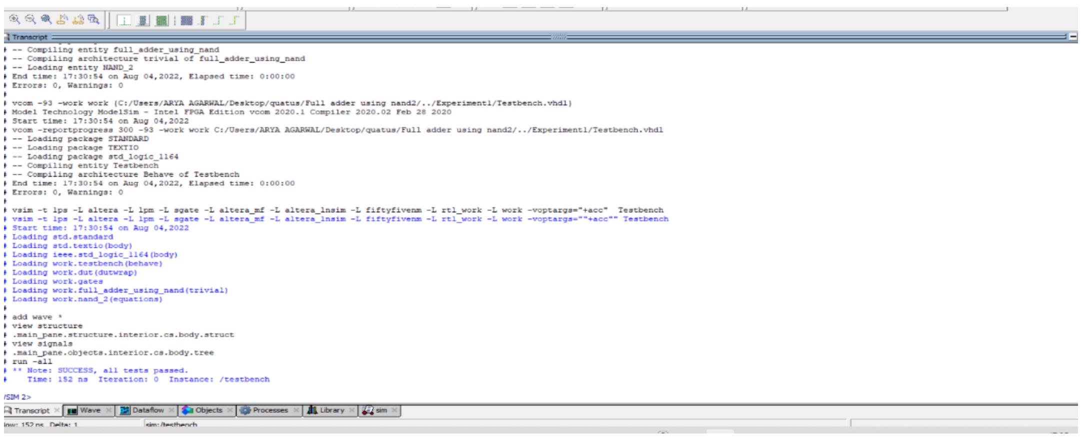


A	B	Cin	Sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

RTL SIMULATION



TRANSCRIPT(All test cases have been passed)



```
-- Compiling entity full_adder_using_nand
-- Compiling architecture trivial of full_adder_using_nand
-- Loading entity NAND_2
End time: 17:30:54 on Aug 04,2022, Elapsed time: 0:00:00
Errors: 0, Warnings: 0

vcom -93 -work work {C:/Users/ARYA AGARWAL/Desktop/quatus/Full adder using nand2/../Experiment1/Testbench.vhdl}
Model Technology ModelSim - Intel FPGA Edition vcom 2020.1 Compiler 2020.02 Feb 28 2020
Start time: 17:30:54 on Aug 04,2022
vcom -reportprogress 300 -93 -work work C:/Users/ARYA AGARWAL/Desktop/quatus/Full adder using nand2/../Experiment1/Testbench.vhdl
-- Loading package STANDARD
-- Loading package TEXTIO
-- Loading package std_logic_1164
-- Compiling entity Testbench
-- Compiling architecture Behave of Testbench
End time: 17:30:54 on Aug 04,2022, 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: 17:30:54 on Aug 04,2022
Loading std.standard
Loading std.textio(body)
Loading ieee.std_logic_1164(body)
Loading work.testbench(behave)
Loading work.dut(dutwrap)
Loading work.gates
Loading work.full_adder_using_nand(trivial)
Loading work.nand_2(equations)

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 tests passed.
   Time: 152 ns  Iteration: 0  Instance: /testbench

VSIM 2>
```