VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY FACULTY OF COMPUTER SCIENCE AND ENGINEERING

LOGIC DESIGN WITH HDL

REPORT LAB 1

Students:	Phạm Anh Tài	2350023
	Phạm Ngọc Huy	2352404
	Lục Tấn Phúc	2352935
	Trần Đình Duy Khương	2352638

HO CHI MINH CITY, 20 March 2024

I .Exercise

1) Exercise 1:

a)

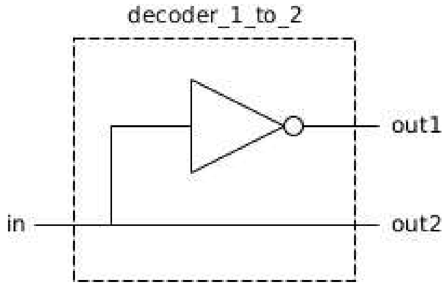
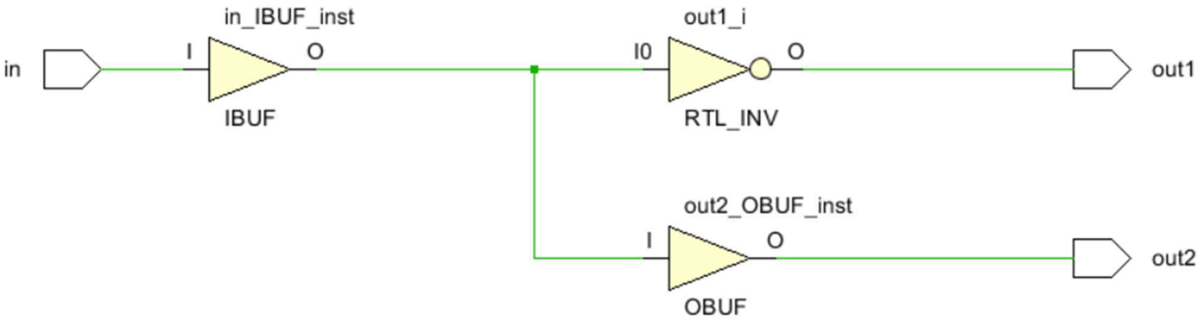
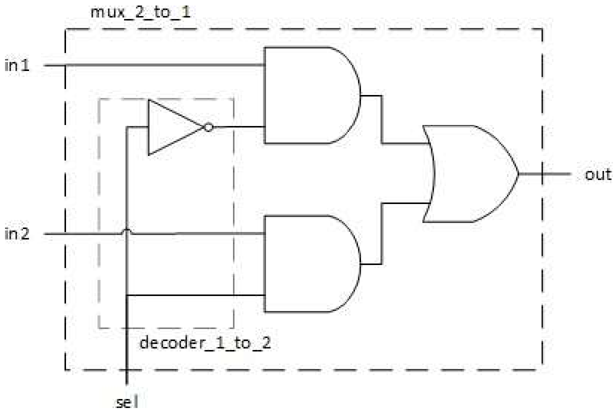
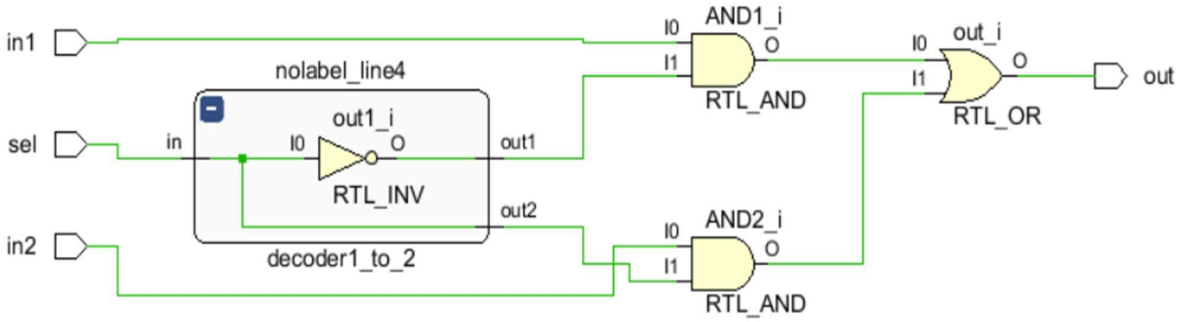


Figure 1: 1-to-2 Decoder



b) Design a 2-to-1 Multiplexer





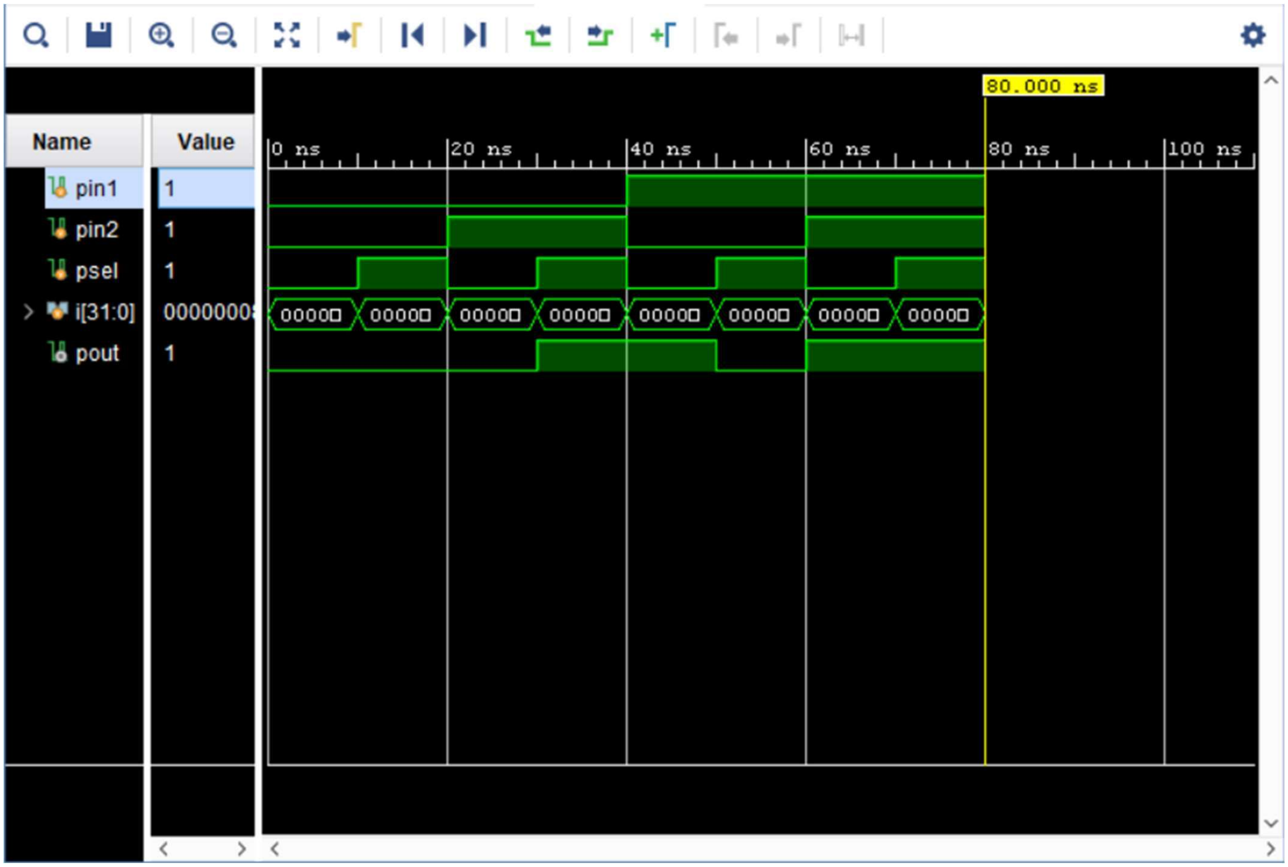
2) Exercise 2

a) Test bench for the 2-to-1 Multiplexer

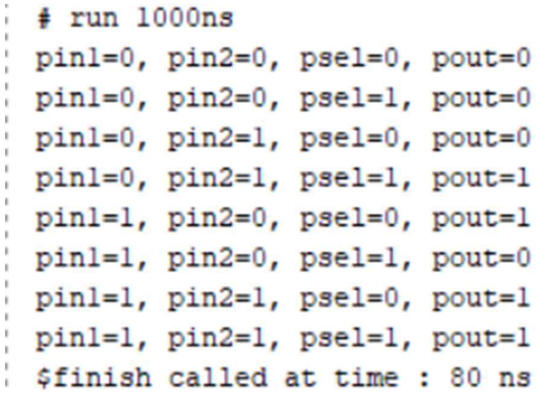
- RTL code: This code represents the actual hardware design or behavior of the digital system. It describes how various registers and combinational logic elements are interconnected to perform a specific function.

- Testbench code: The testbench is used to verify the functionality of the RTL code. It generates input stimuli, applies them to the design under test, and checks the output responses against expected results.

Waveform:

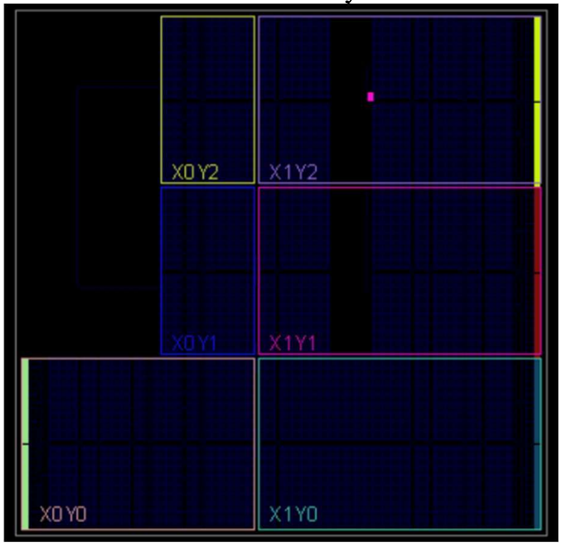


Name	Value
pin1	1
pin2	1
psel	1
i[31:0]	0000000
pout	1



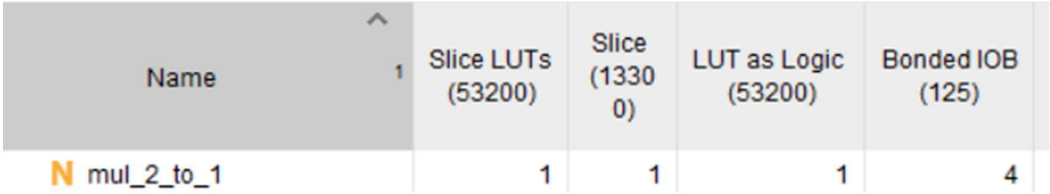
```
# run 1000ns
pin1=0, pin2=0, psel=0, pout=0
pin1=0, pin2=0, psel=1, pout=0
pin1=0, pin2=1, psel=0, pout=0
pin1=0, pin2=1, psel=1, pout=1
pin1=1, pin2=0, psel=0, pout=1
pin1=1, pin2=0, psel=1, pout=0
pin1=1, pin2=1, psel=0, pout=1
pin1=1, pin2=1, psel=1, pout=1
$finish called at time : 80 ns
```

b) Compare the Synthesis's Schematic and the RTL Analysis's schematic



- Synthesis's Schematic represents the synthesized netlist derived from the RTL code.
- RTL Analysis's schematic represents the code by logic gates illustrating the data flow, and structure of the RTL code.

c) After that, run the Implementation, check the Utilization report in the Summary report in Project Summary for used resources.



Name	Slice LUTs (53200)	Slice (13300)	LUT as Logic (53200)	Bonded IOB (125)
N mul_2_to_1	1	1	1	4

3) Exercise 3:

a) Design a half-adder circuit using structural model.

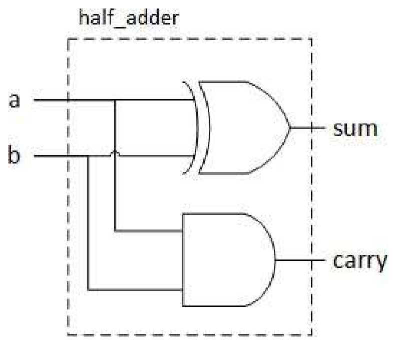
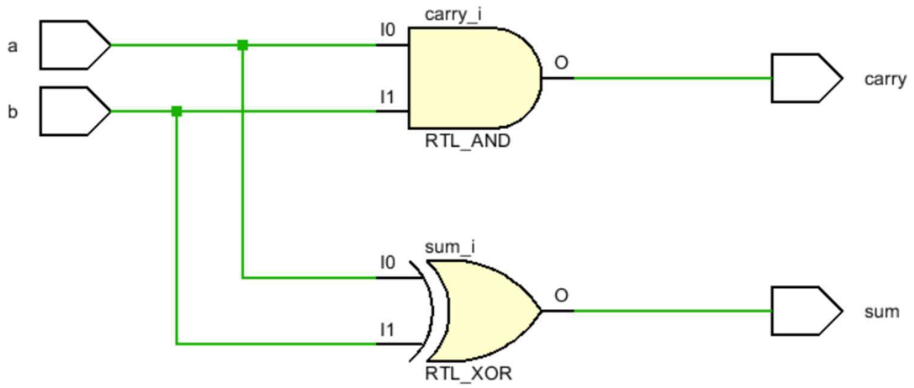


Figure 3: Half-adder



c) Design a full-adder circuit using structural model. Reuse the half-adder module.

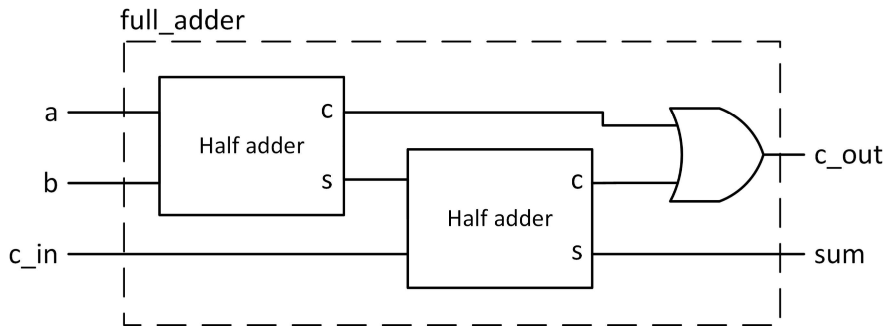
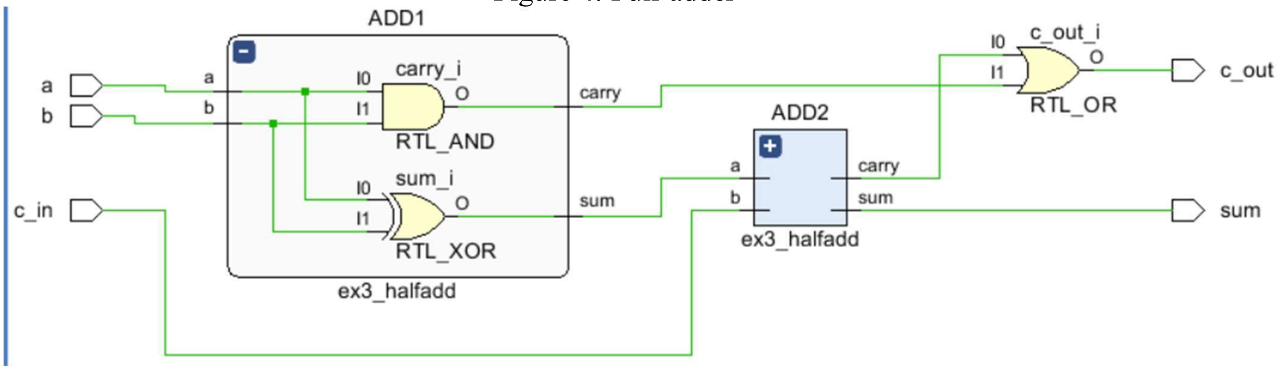
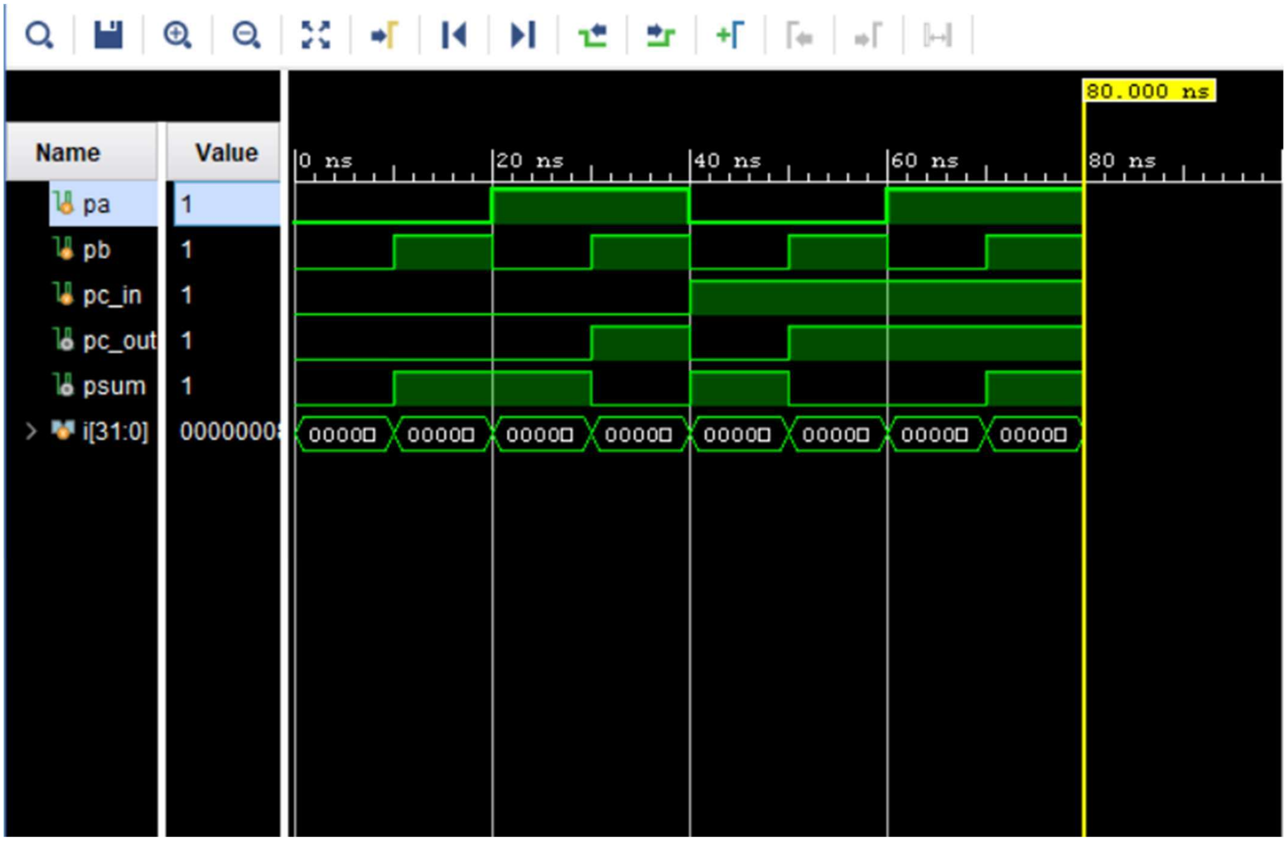


Figure 4: Full-adder



Testbench to evaluate operations of the implemented full-adder circuit



Name	Value
pa	1
pb	1
pc_in	1
pc_out	1
psum	1
i[31:0]	0000000

d) Design a 4-bit ripple carry adder using structural model. Reuse the implemented full-adder.

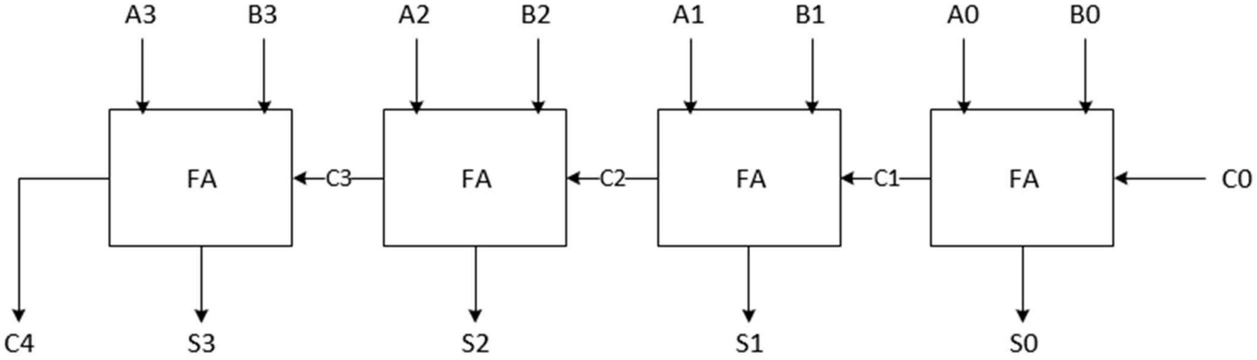
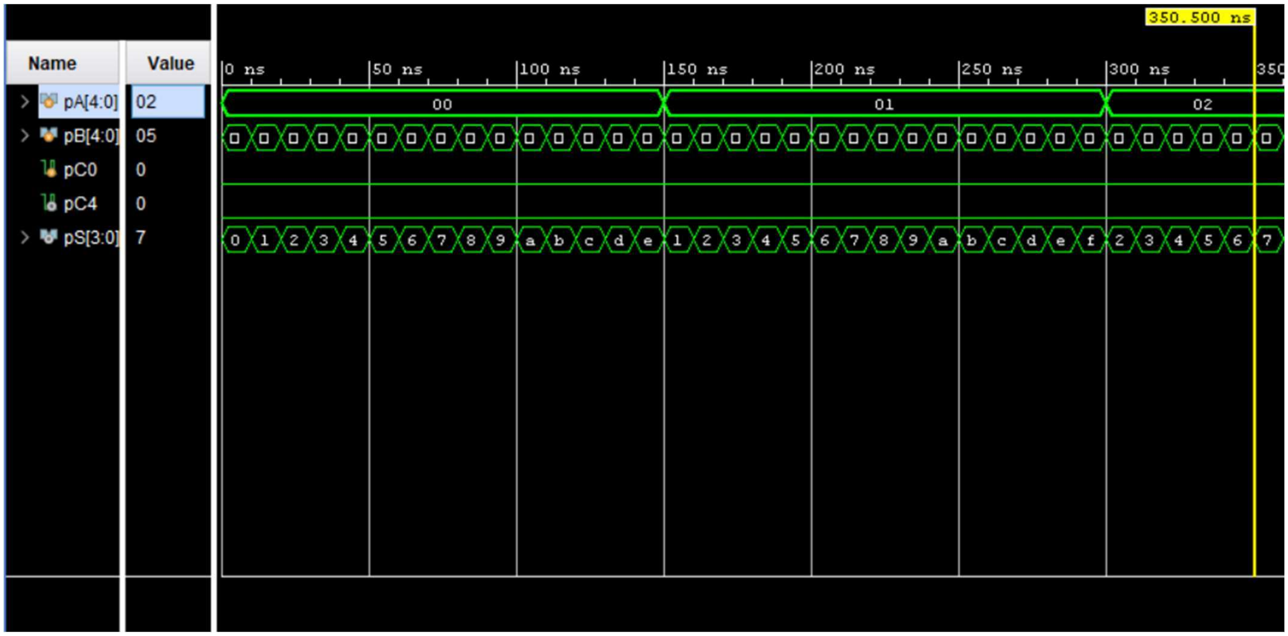


Figure 5: 4-bit ripple carry adder

Testbench to simulate the implemented circuit:



Name	Value
pA[4:0]	02
pB[4:0]	05
pC0	0
pC4	0
pS[3:0]	7

350.500 ns
0 ns
50 ns
100 ns
150 ns
200 ns
250 ns
300 ns
350
00
01
02
0 1 2 3 4 5 6 7 8 9 a b c d e 1 2 3 4 5 6 7 8 9 a b c d e f 2 3 4 5 6 7

4) Exercise 4

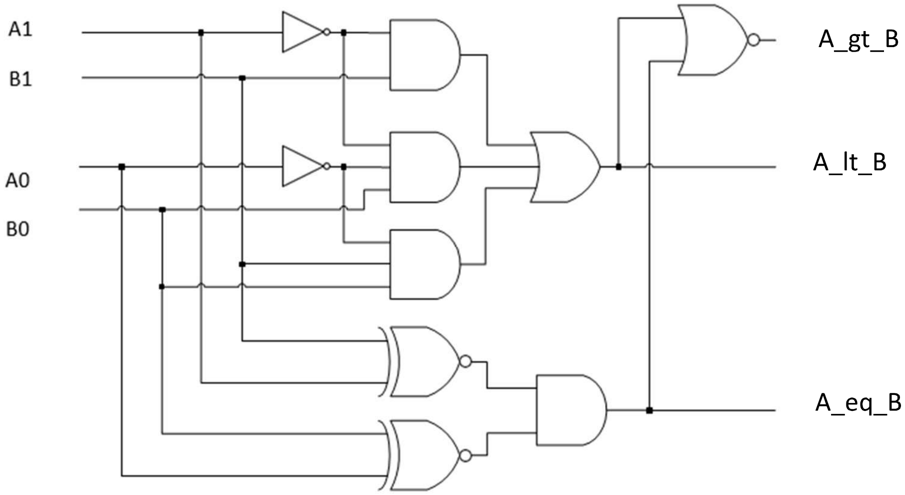
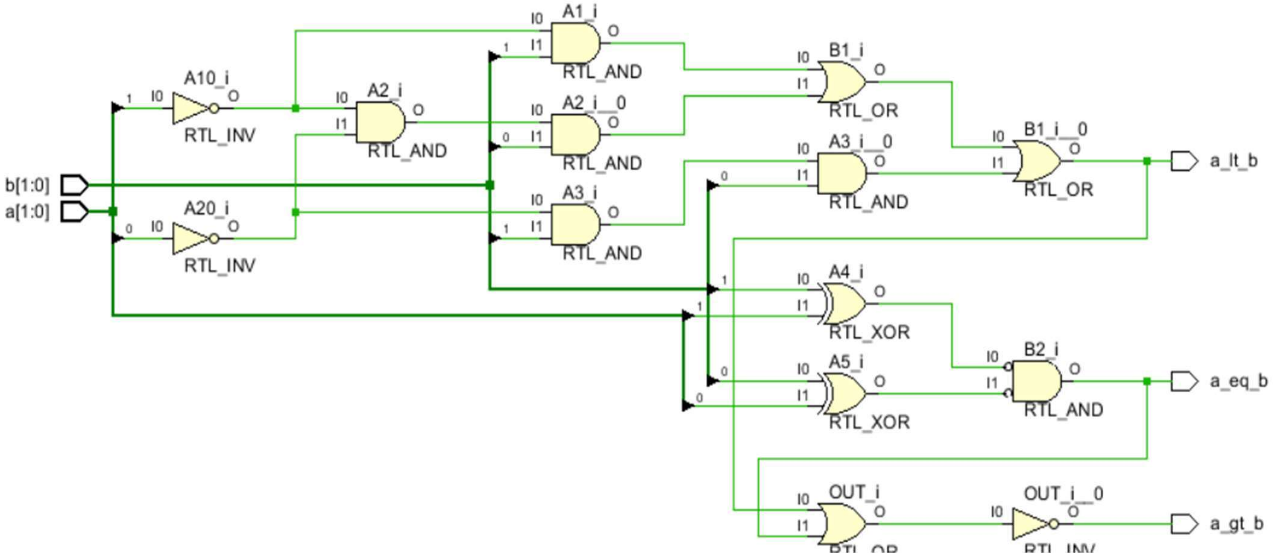
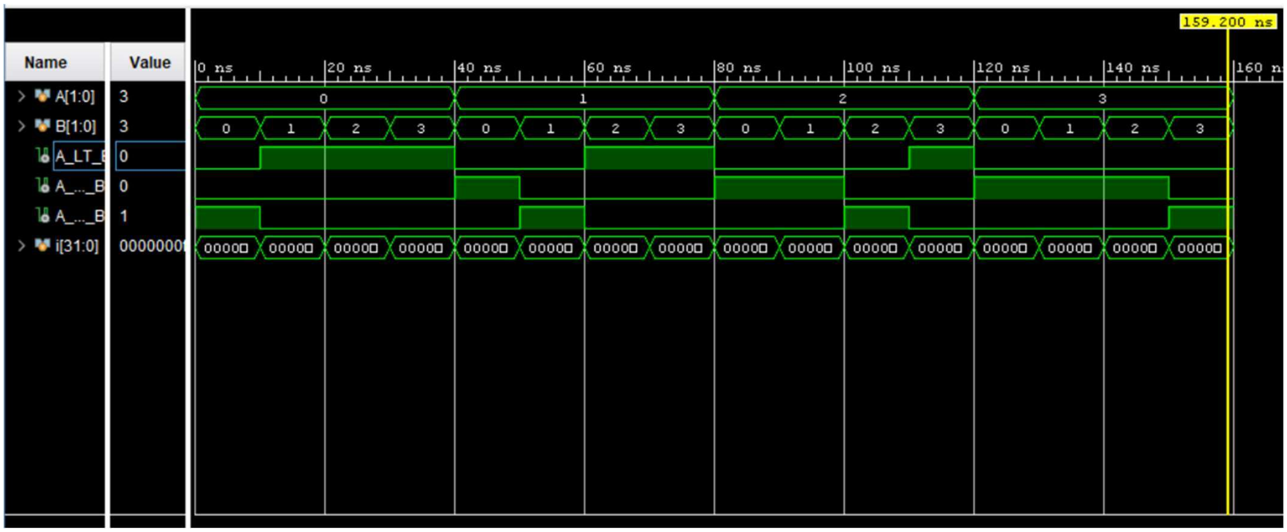


Figure 6: 2-bit comparator





Name	Value
A[1:0]	3
B[1:0]	3
A_LT_B	0
A_..._B	0
A_..._B	1
i[31:0]	0000000

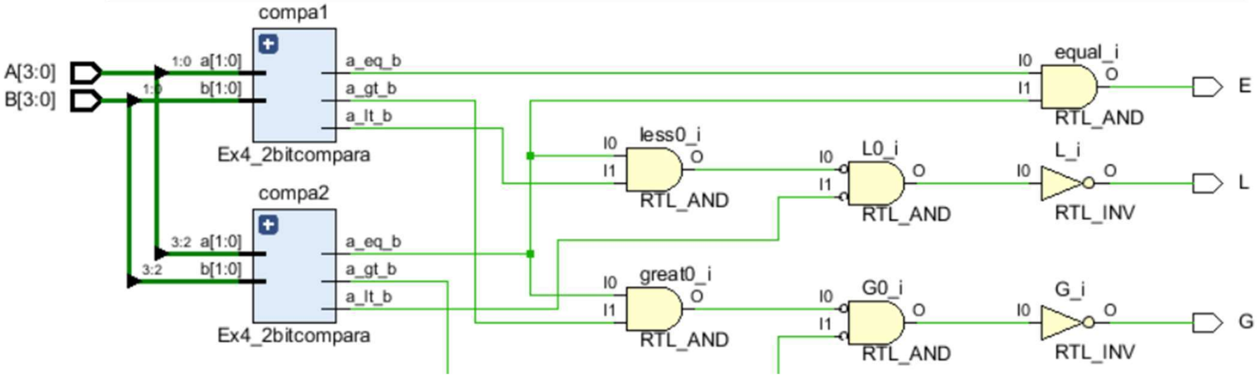
- The functions of each output of the 2-bit comparator:

$$a\_gt\_b = (\overline{a}_1 b_1 + \overline{a}_1 \overline{a}_0 b_0 + \overline{a}_0 b_1 b_0)'((b_1 a_1)(b_0 a_0))' + (\overline{a}_1 b_1 + \overline{a}_1 \overline{a}_0 b_0 + \overline{a}_0 b_1 b_0)(b_1 a_1)(b_0 a_0)$$

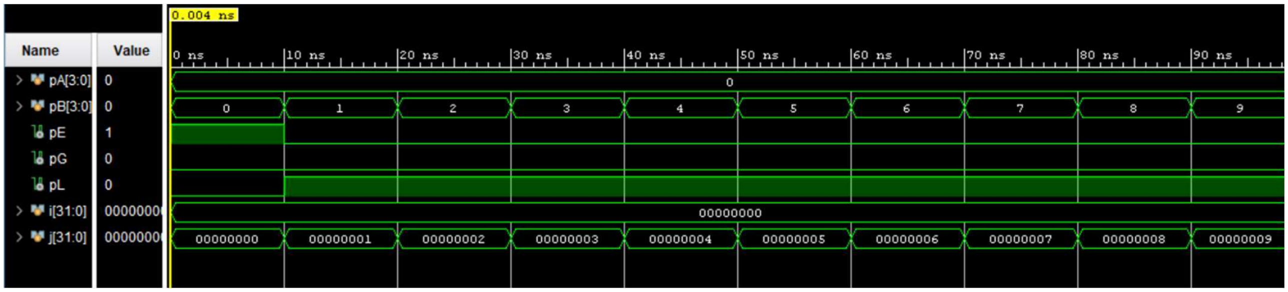
$$a\_lt\_b = \overline{a}_1 b_1 + \overline{a}_1 \overline{a}_0 b_0 + \overline{a}_0 b_1 b_0$$

$$a\_eq\_b = (b_1 a_1)(b_0 a_0)$$

- Schematic of 4-bit comparator:



-Testbench waveform:


0.004 ns
Name
Value
pA[3:0] 0
pB[3:0] 0
pE 1
pG 0
pL 0
i[31:0] 0000000
j[31:0] 0000000
0 ns
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
70 ns
80 ns
90 ns
0
0
1
2
3
4
5
6
7
8
9
00000000
00000000
00000001
00000002
00000003
00000004
00000005
00000006
00000007
00000008
00000009