**Experiment / Assignment / Tutorial No. 7**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **Batch: A1 Roll No.: 1911004 Experiment / assignment / tutorial No.: 7** |

|  |
| --- |
| **Title:** VHDL programming for gates |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Objective:** Implements a simple OR, AND, XOR, NOR, NAND, XNOR gate in VHDL

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected Outcome of Experiment:**

**CO4:** Implement digital networks using VHDL.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

* ModelSim Software Link:

https://www.mentor.com/company/higher\_ed/modelsim-student-edition

* J. Bhasker, “VHDL Primer”, Pearson Education
* Douglas L. Perry, “VHDL Programming by Example”, Tata McGraw Hill
* http://esd.cs.ucr.edu/labs/tutorial/

**Pre Lab/ Prior Concepts:**

VHDL is an acronym for VHSlC Hardware Description Language (VHSIC is an acronym for Very High Speed Integrated Circuits). It is a hardware description language that can be used to model a digital system at many levels of abstraction ranging from the algorithmic level to the gate level. The complexity of the digital system being modeled could vary from that of a simple gate to a complete digital electronic system, or anything in between. The digital system can also be described hierarchically. Timing can also be explicitly modeled in the same description.

**VHDL Programming Structure**

Entity and Architecture are the two main basic programming structures in VHDL.

**Entity:** Entity can be seen as the black box view of the system. We define the inputs and outputs of the system which we need to interface. It is used to declare the I/O ports of the circuit.

Eg:

Entity ANDGATE is

Port (A: in std\_logic;

B: in std\_logic;

Y: out std\_logic);

End entity ANDGATE;

Entity name ANDGATE is given by the programmer, each entity must have a name.

**Architecture:** Architecture defines what is in our black box that we described using ENTITY. The description code resides within architecture portion. Either behavioral or structural models can be used to describe our system in the architecture. In Architecture we will have interconnections, processes, components, etc.

Eg:

Architecture AND1 of ANDGATE is

--declarations

Begin

--statements

Y <= A AND B;

End architecture AND1;

Entity name or architecture name is user defined. Identifiers can have uppercase alphabets, lowercase alphabets, and numbers and underscore (\_). First letter of identifier must be an alphabet and identifier cannot end with an underscore. In VHDL, keywords and user identifiers are case insensitive.

VHDL is strongly typed language i.e. every object must be declared. Standardized design libraries are typically used and are included prior to the entity declaration. This is accomplished by including the code "library ieee;" and "use ieee.std\_logic\_1164.all;"

**Implementation Details:**

**VHDL program code**

**OR**

library ieee;

use ieee.std\_logic\_1164.all;

--------------------------------------

entity OR\_ent is

port( x: in std\_logic;

y: in std\_logic;

F: out std\_logic

);

end OR\_ent;

---------------------------------------

architecture OR\_arch of OR\_ent is

begin

process(x, y)

begin

-- compare to truth table

if ((x='0') and (y='0')) then

F <= '0';

else

F <= '1';

end if;

end process;

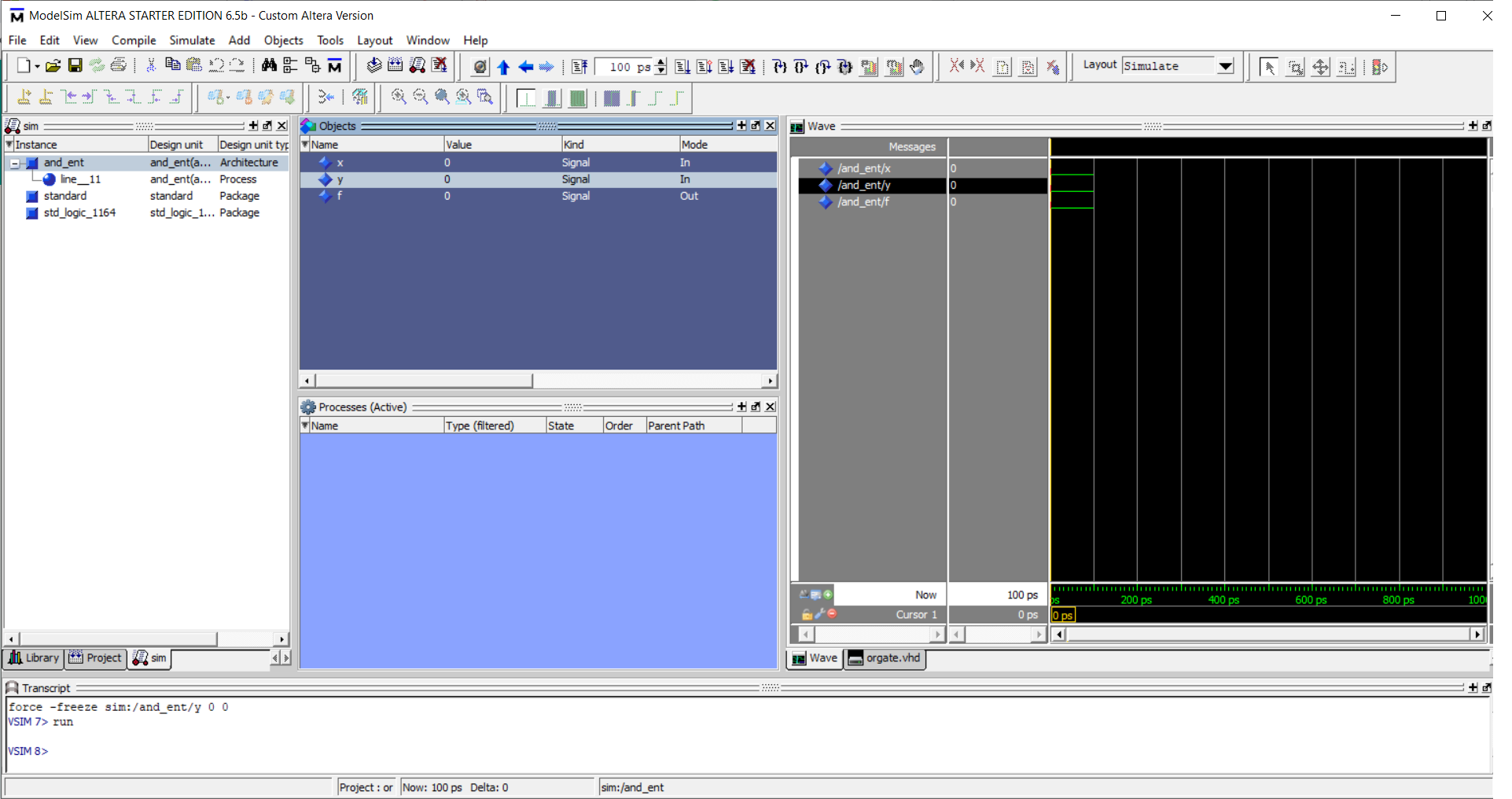
end OR\_arch;

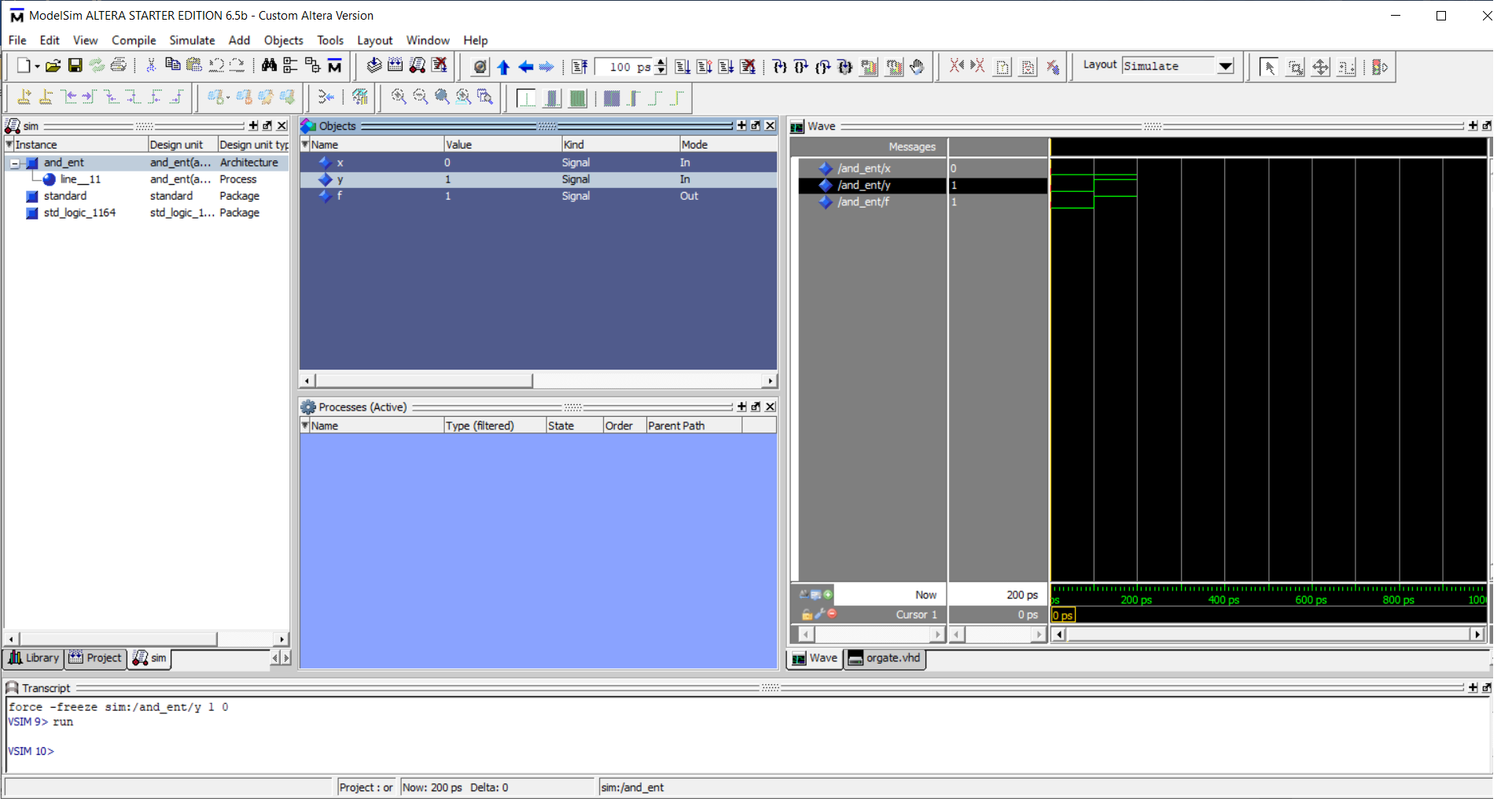
architecture OR\_beh of OR\_ent is

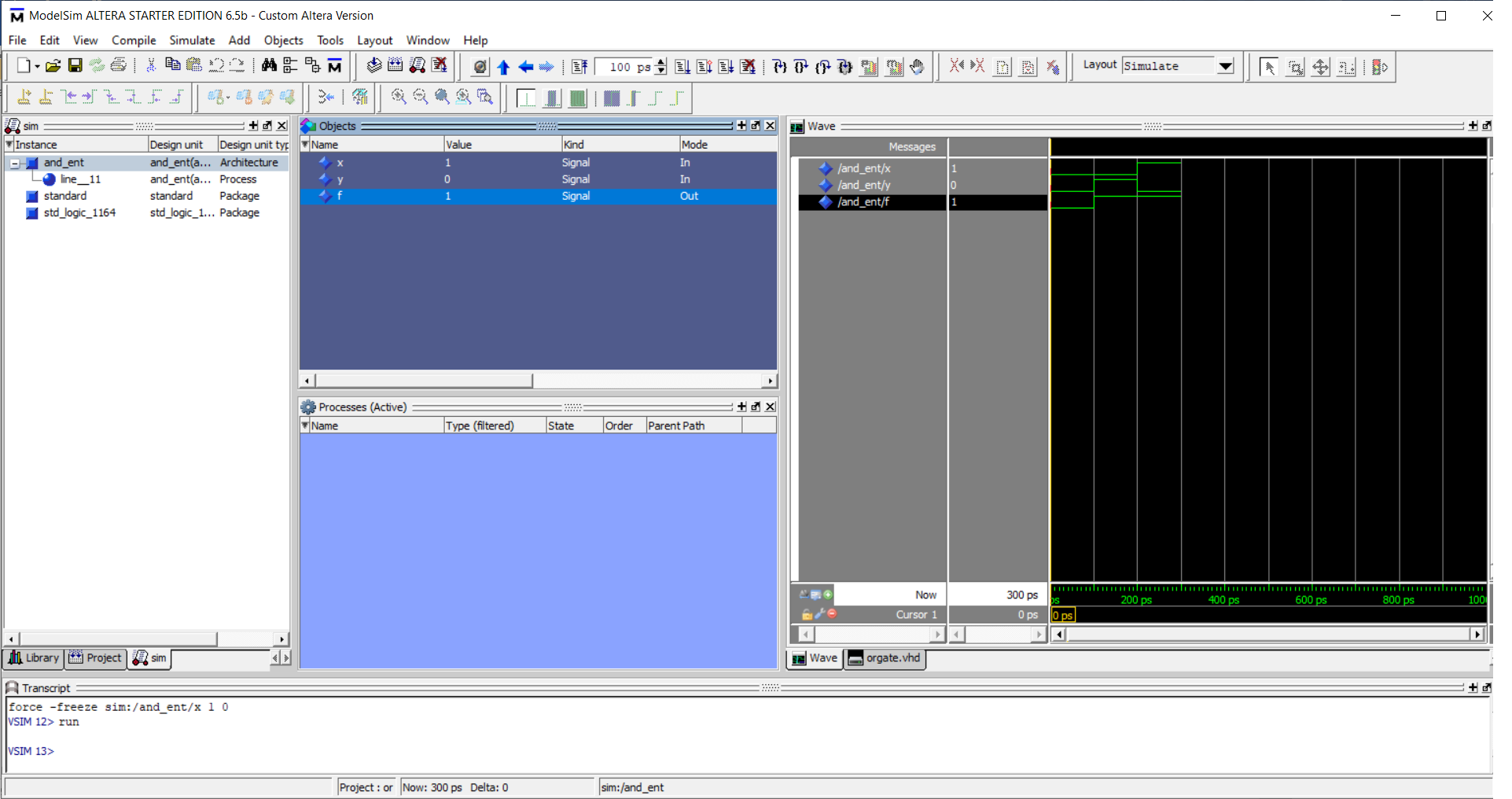
begin

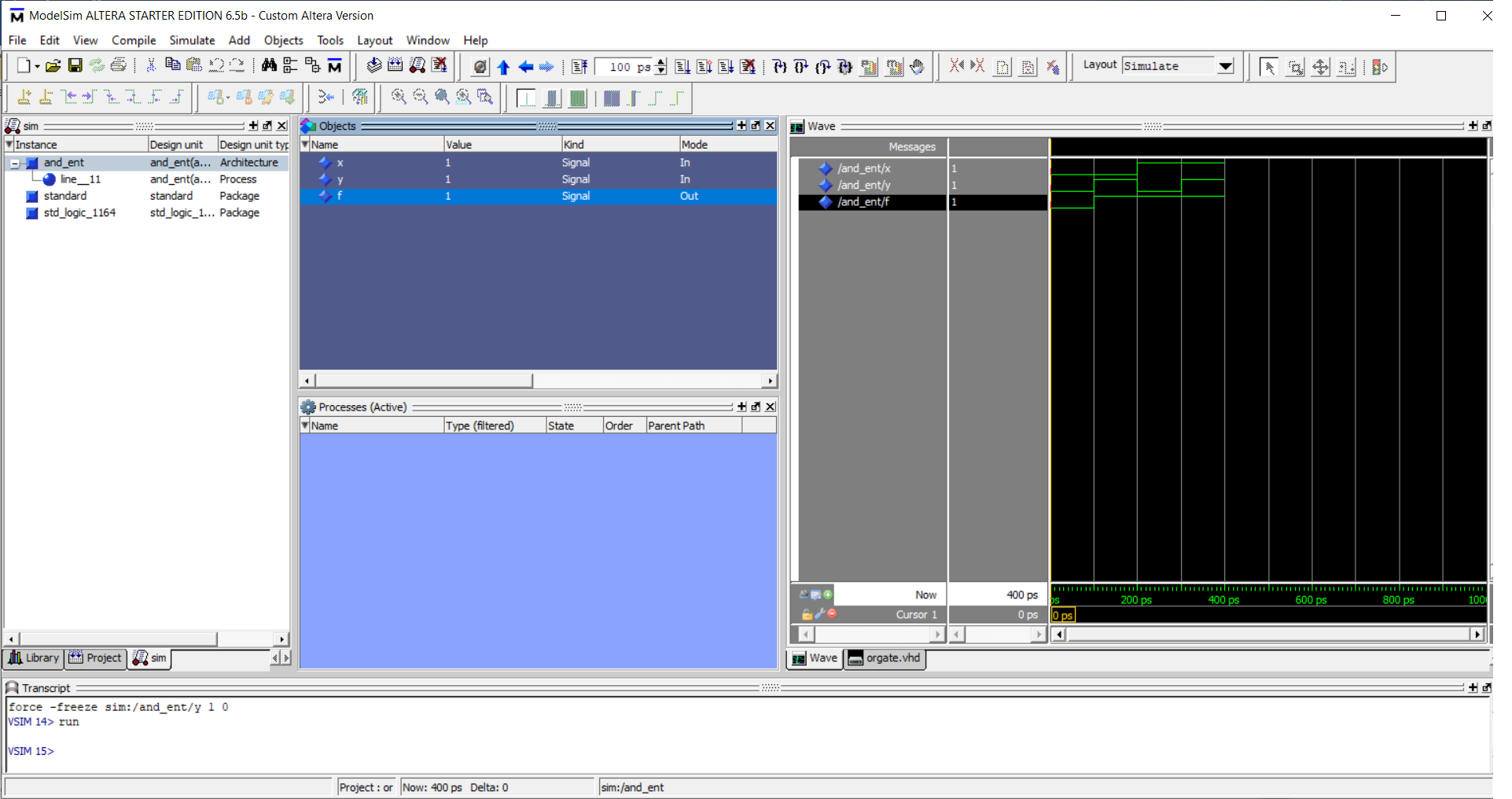
F <= x or y;

end OR\_beh;









**AND**

library ieee;

use ieee.std\_logic\_1164.all;

--------------------------------------

entity AND\_ent is

port( x: in std\_logic;

y: in std\_logic;

F: out std\_logic

);

end AND\_ent;

---------------------------------------

architecture AND\_arch of AND\_ent is

begin

process(x, y)

begin

-- compare to truth table

if ((x='0') and (y='0')) then

F <= '0';

else

F <= '1';

end if;

end process;

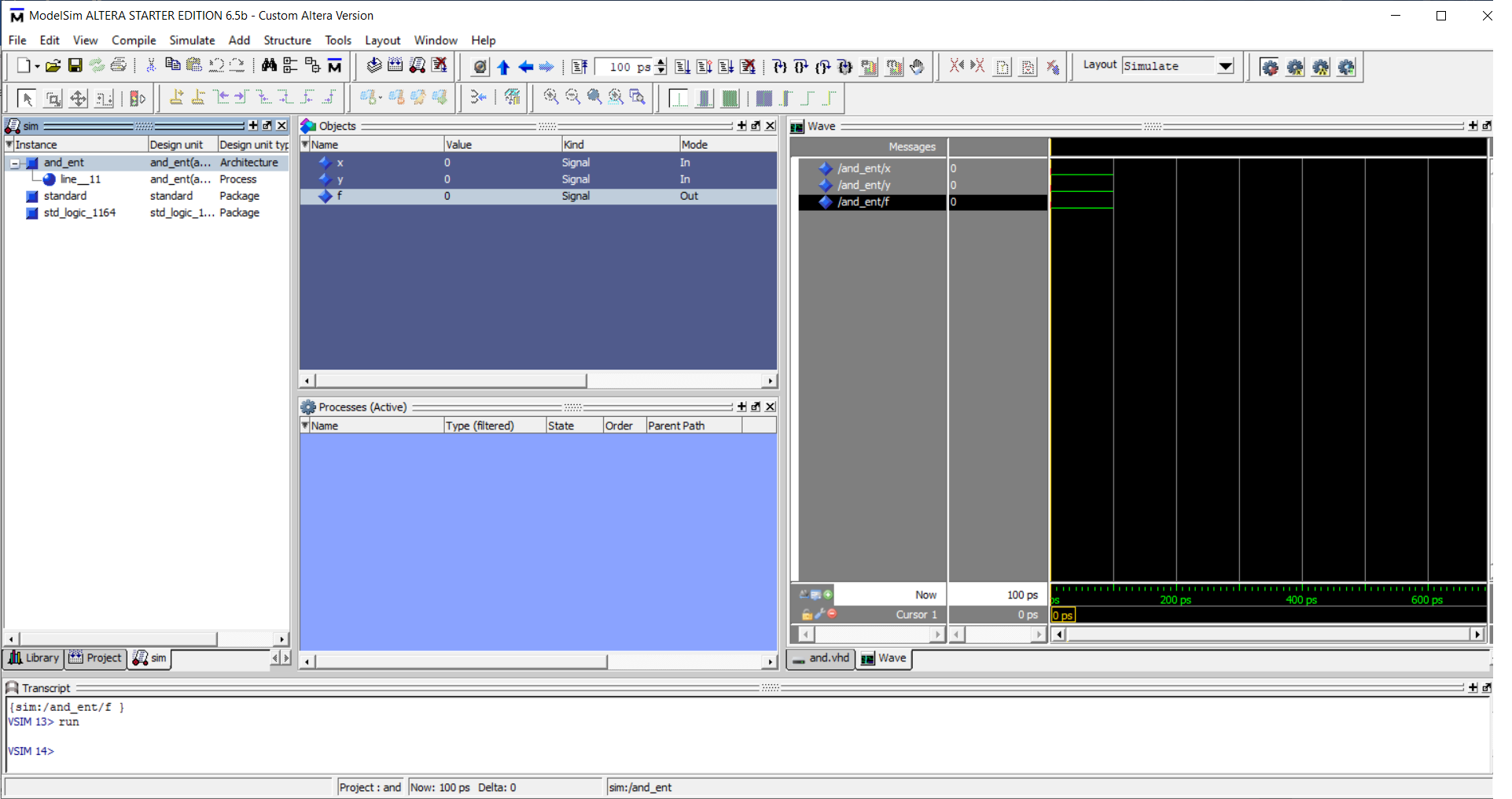
end AND\_arch;

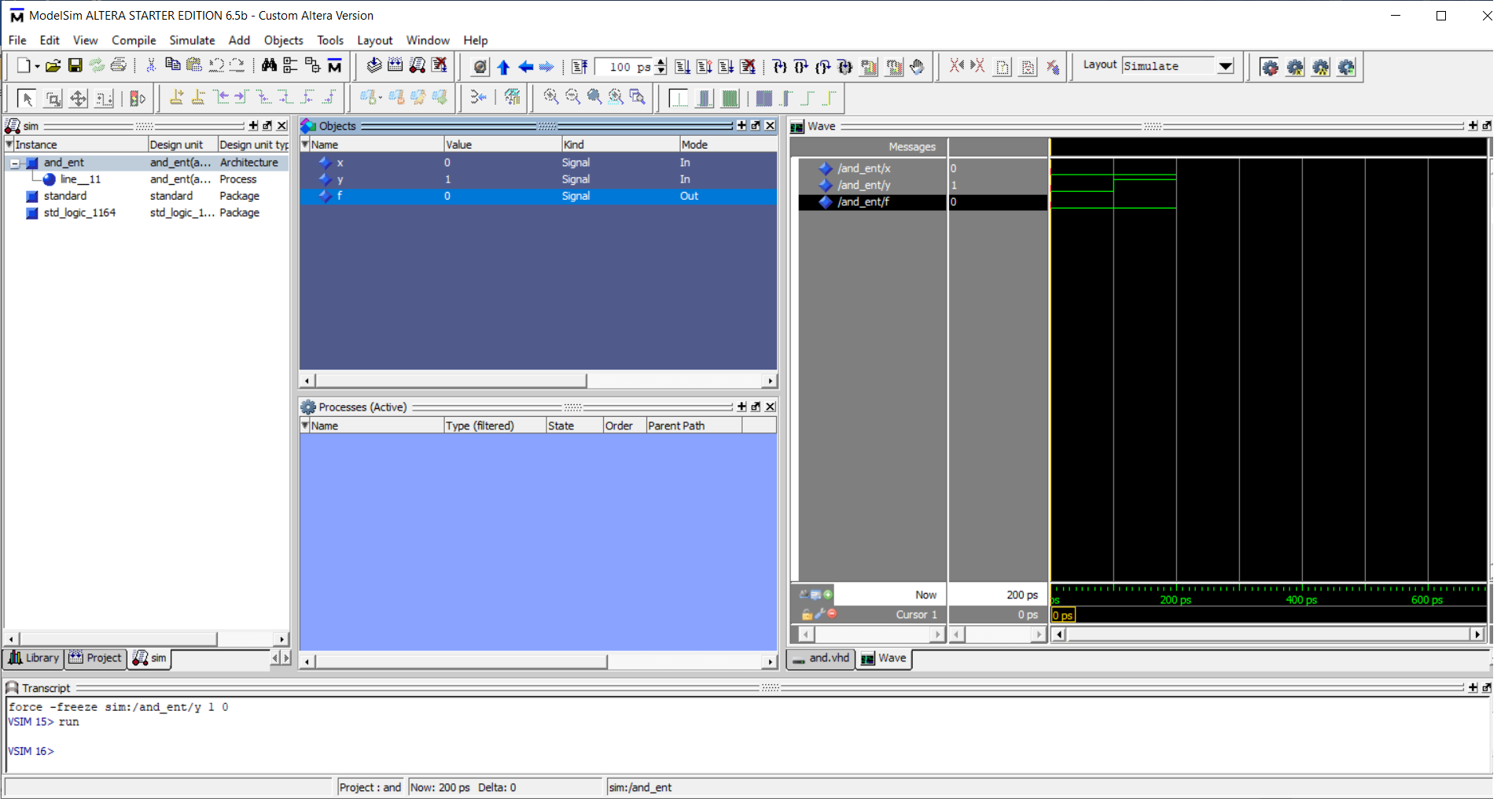
architecture AND\_beh of AND\_ent is

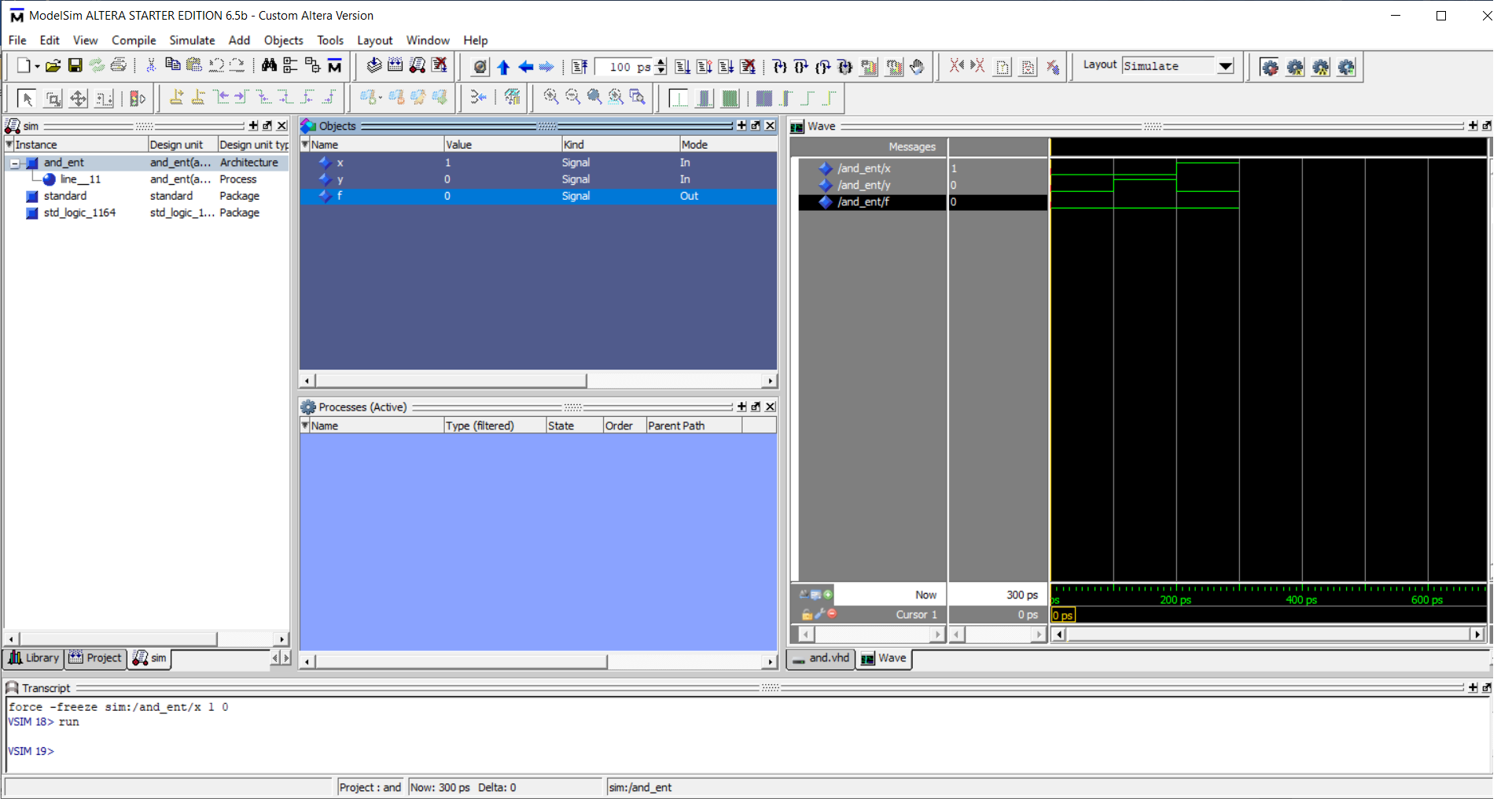
begin

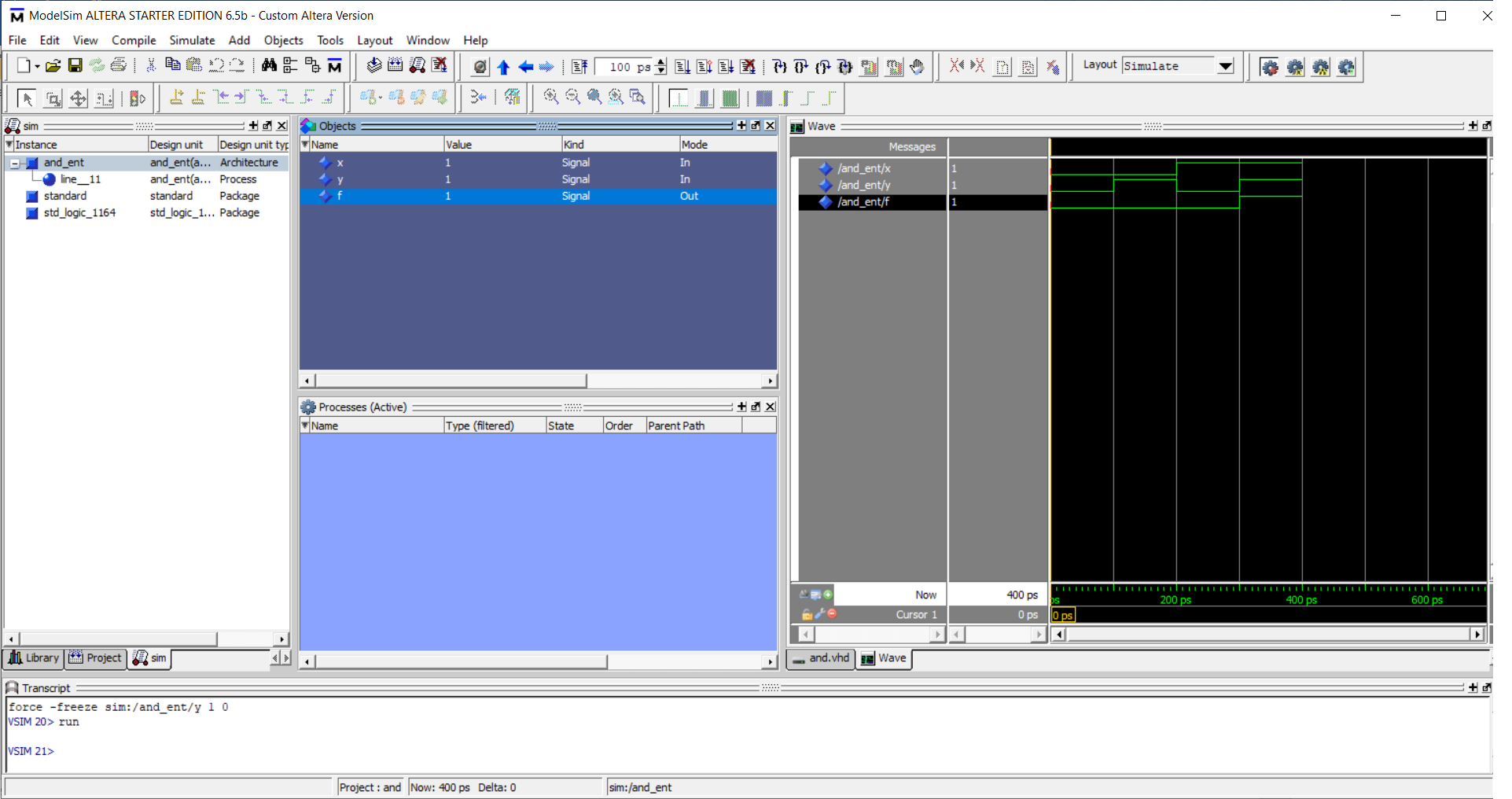
F <= x and y;

end AND\_beh;









**NAND**

library ieee;

use ieee.std\_logic\_1164.all;

--------------------------------------

entity NAND\_ent is

port( x: in std\_logic;

y: in std\_logic;

F: out std\_logic

);

end NAND\_ent;

---------------------------------------

architecture NAND\_arch of NAND\_ent is

begin

process(x, y)

begin

-- compare to truth table

if ((x='0') and (y='0')) then

F <= '1';

else

F <= '0';

end if;

end process;

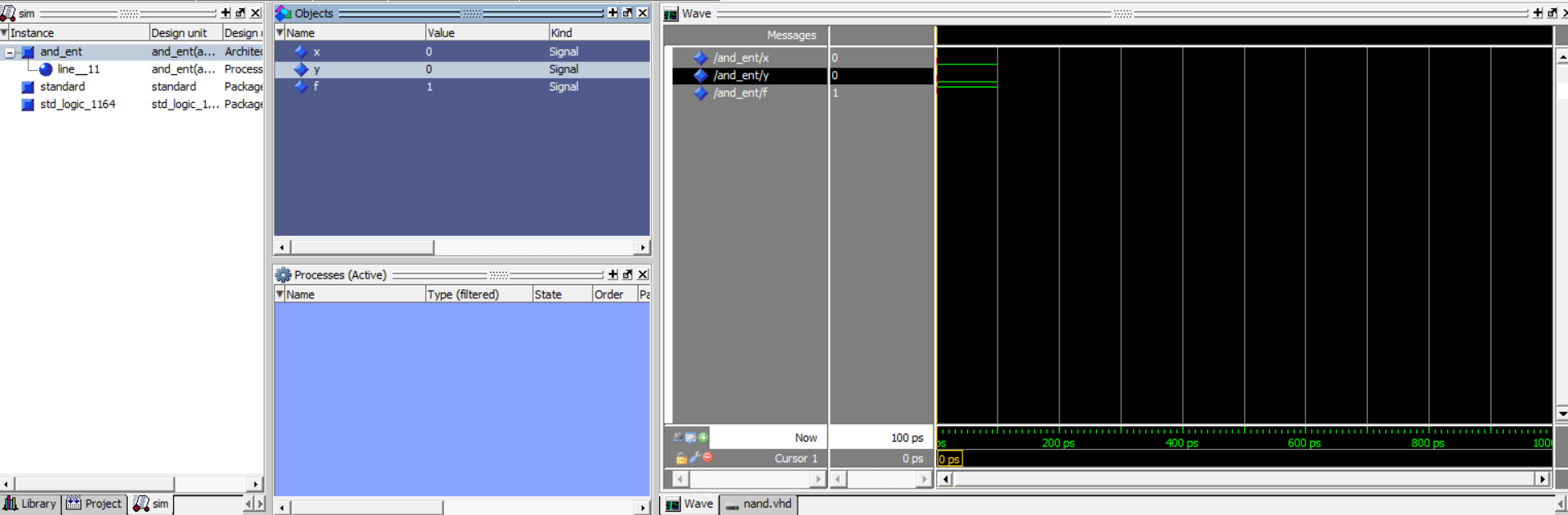
end NAND\_arch;

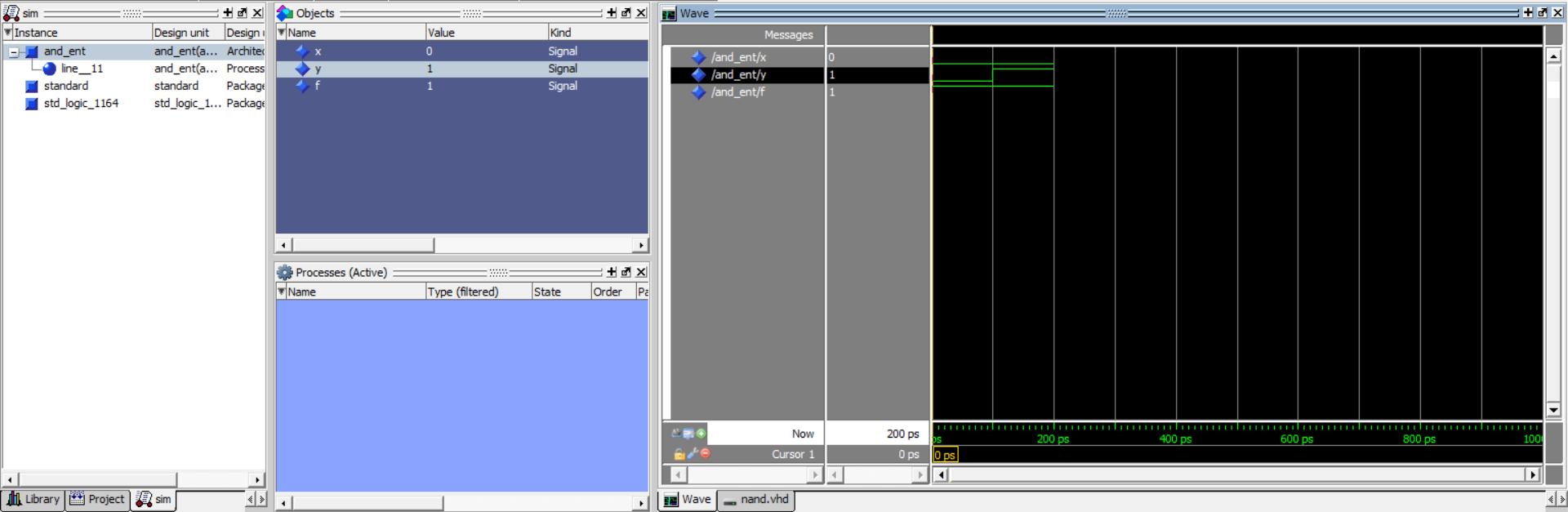
architecture NAND\_beh of NAND\_ent is

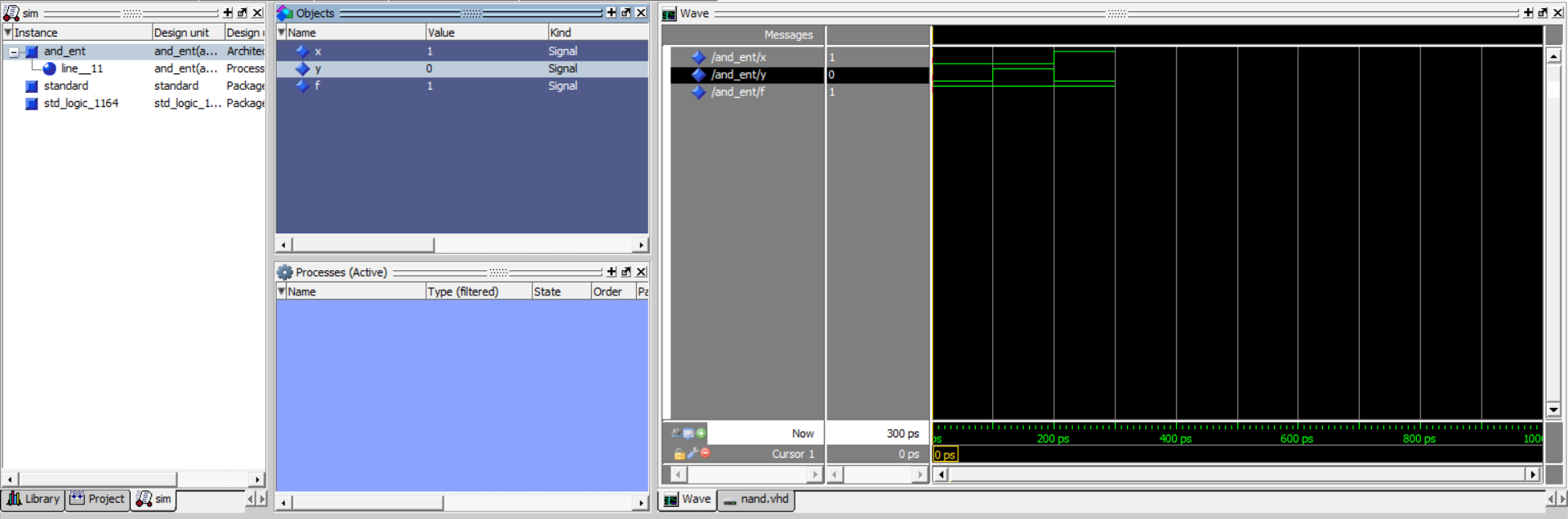
begin

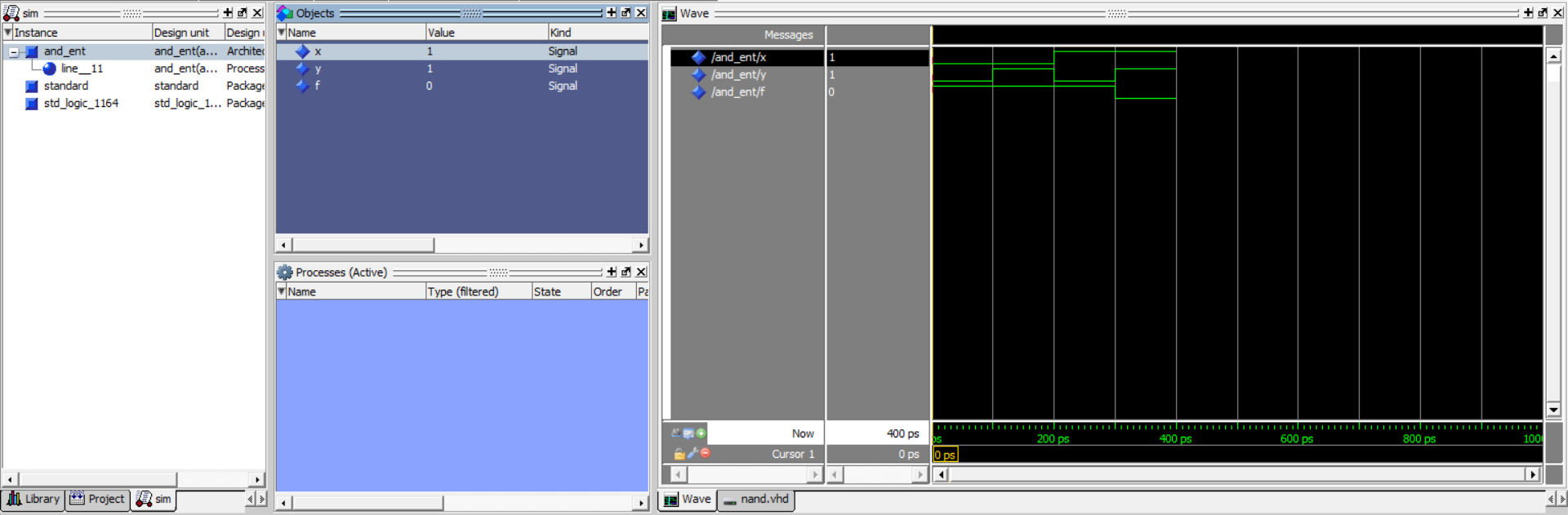
F <= x nand y;

end NAND\_beh;









**NOR**

library ieee;

use ieee.std\_logic\_1164.all;

--------------------------------------

entity NOR\_ent is

port( x: in std\_logic;

y: in std\_logic;

F: out std\_logic

);

end NOR\_ent;

---------------------------------------

architecture NOR\_arch of NOR\_ent is

begin

process(x, y)

begin

-- compare to truth table

if ((x='0') and (y='0')) then

F <= '1';

else

F <= '0';

end if;

end process;

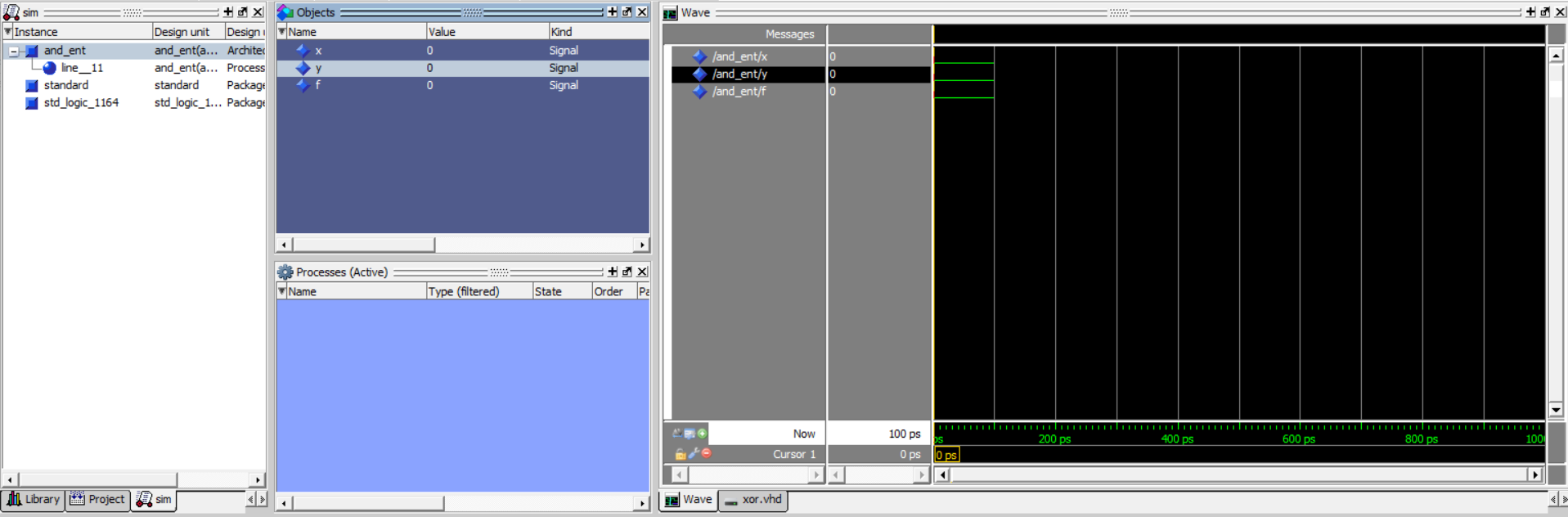
end NOR\_arch;

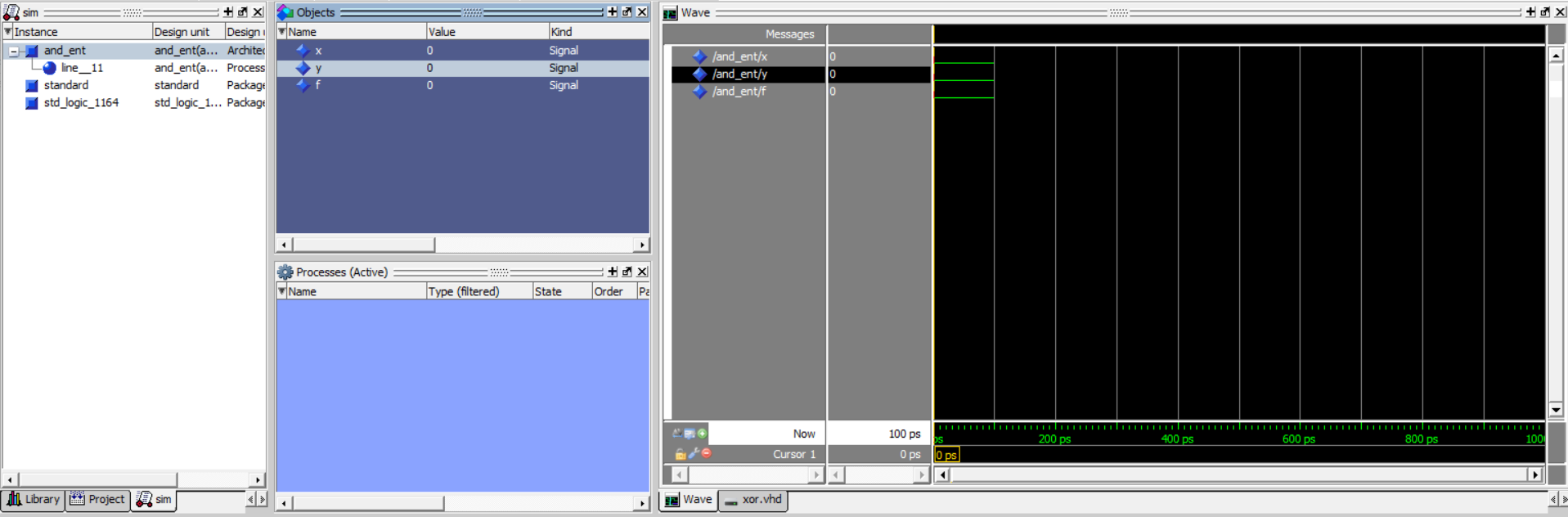
architecture NOR\_beh of NOR\_ent is

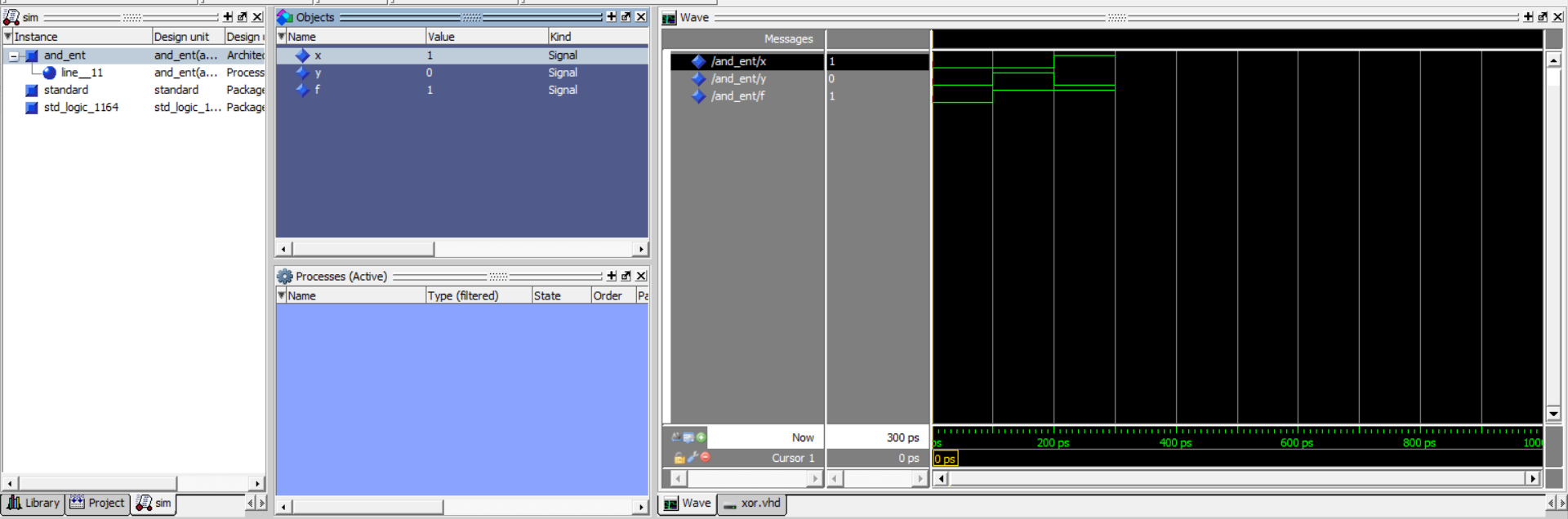
begin

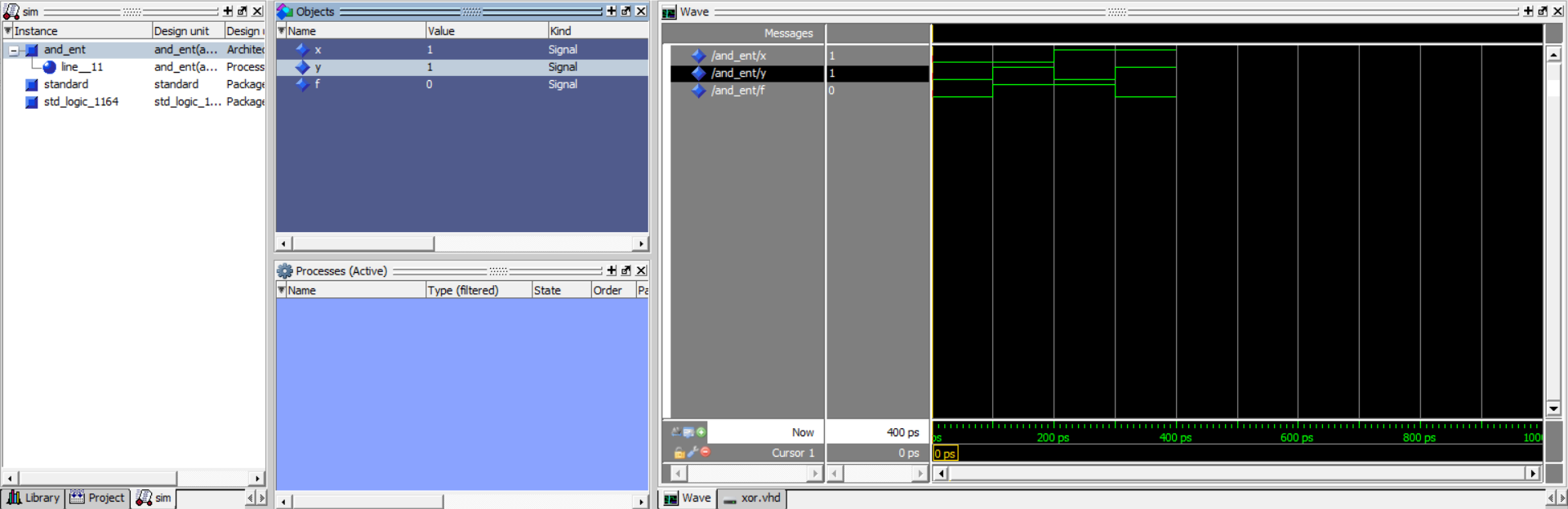
F <= x nor y;

end NOR\_beh;









**XOR**

library ieee;

use ieee.std\_logic\_1164.all;

--------------------------------------

entity XOR\_ent is

port( x: in std\_logic;

y: in std\_logic;

F: out std\_logic

);

end XOR\_ent;

---------------------------------------

architecture XOR\_arch of XOR\_ent is

begin

process(x, y)

begin

-- compare to truth table

if (x=y) then

F <= '0';

else

F <= '1';

end if;

end process;

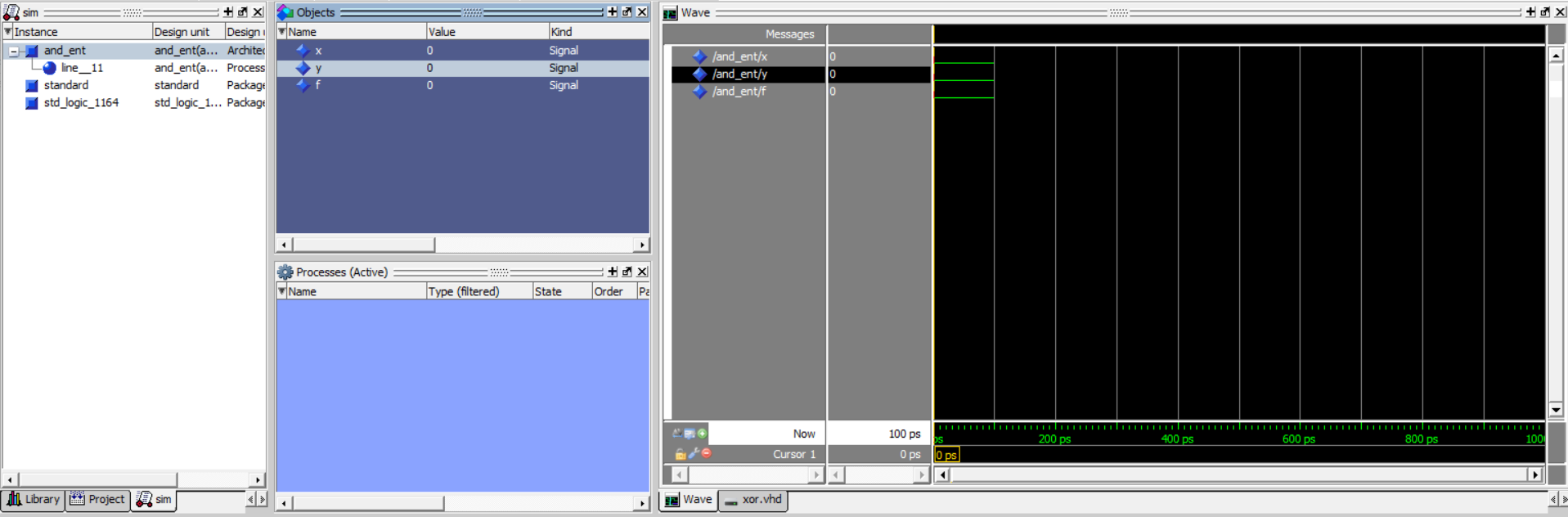
end XOR\_arch;

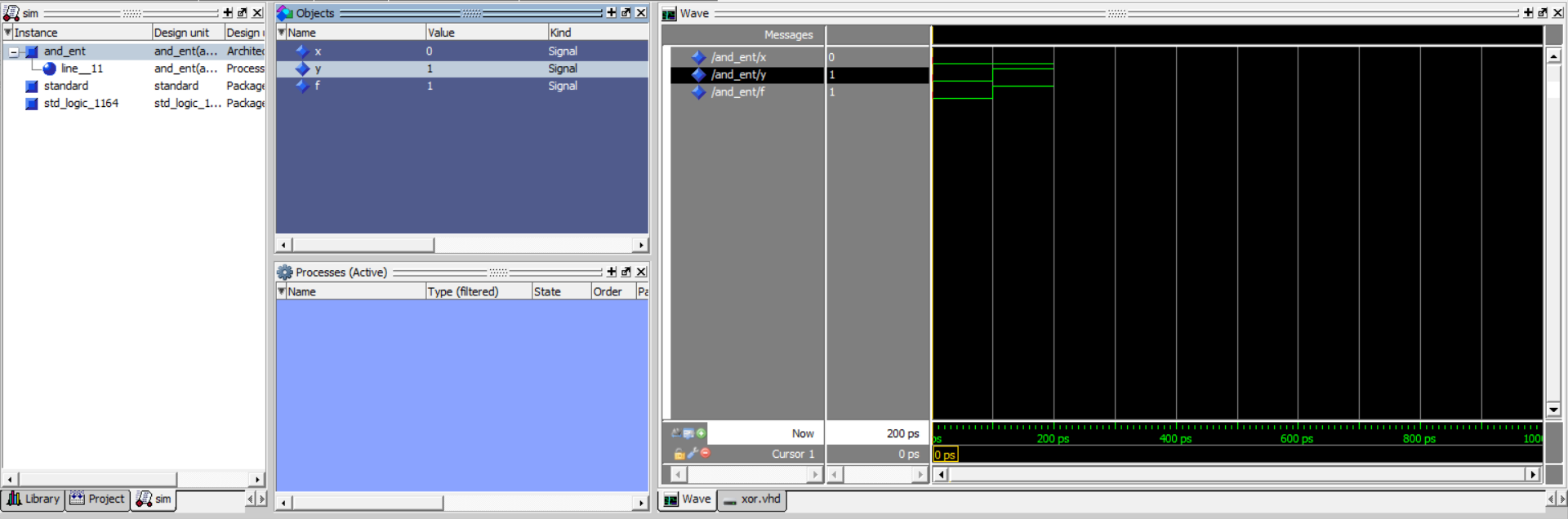
architecture XOR\_beh of XOR\_ent is

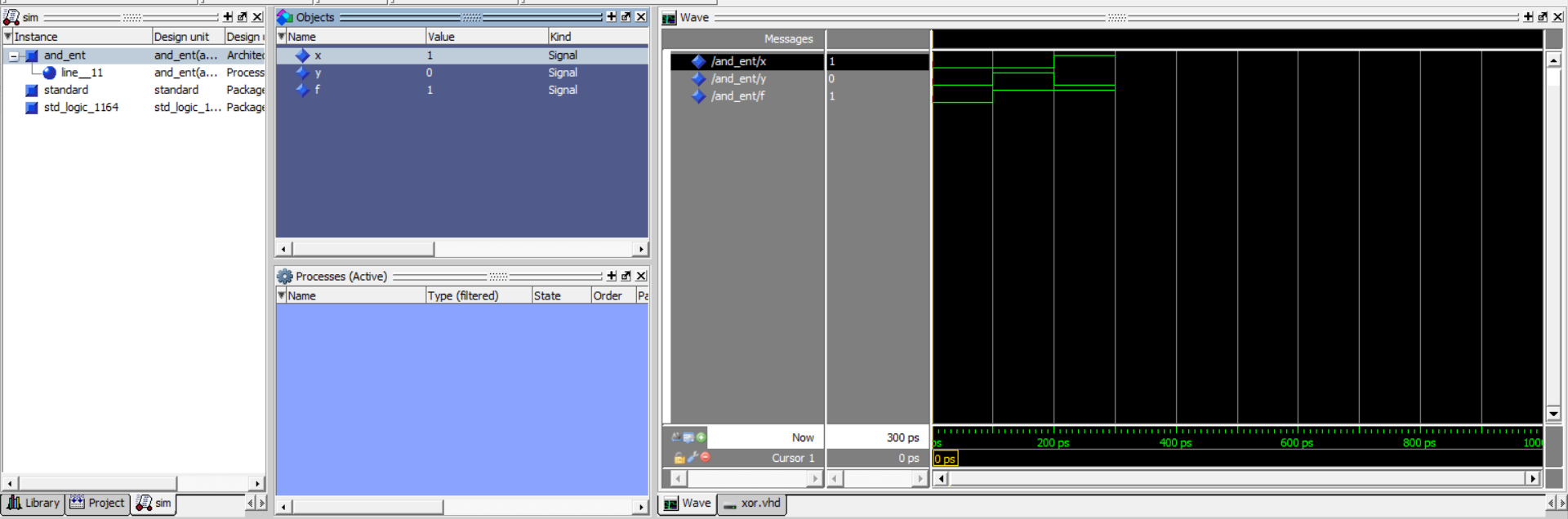
begin

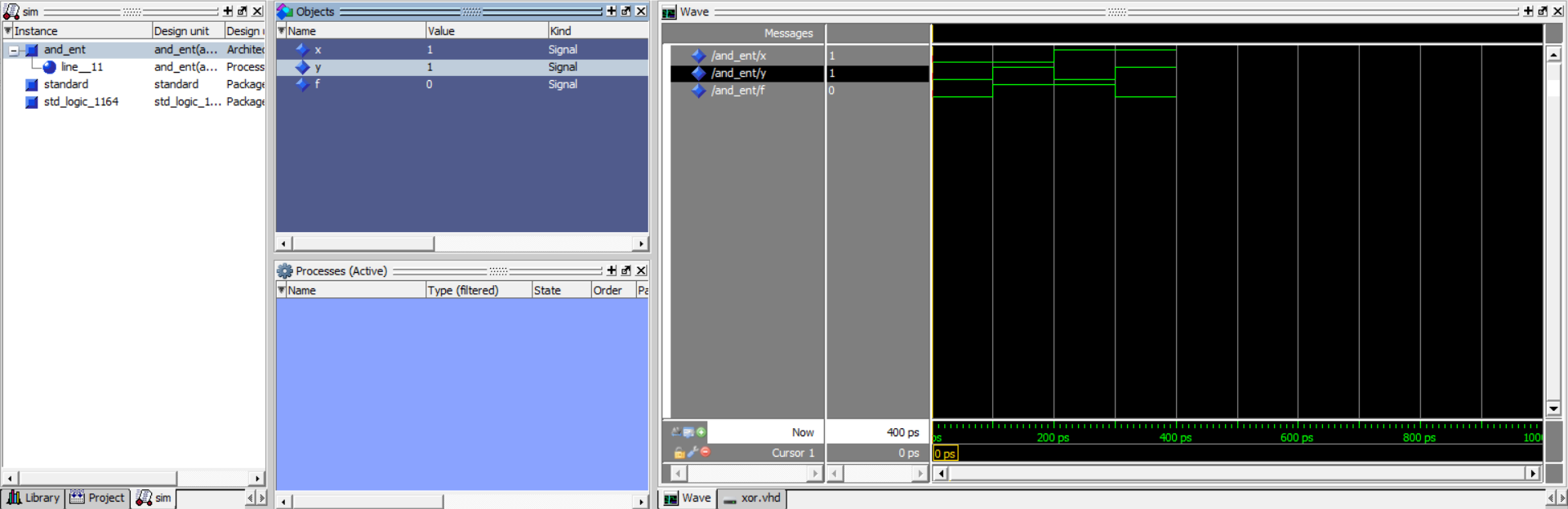
F <= x xor y;

end XOR\_beh;









**XNOR**

library ieee;

use ieee.std\_logic\_1164.all;

--------------------------------------

entity XNOR\_ent is

port( x: in std\_logic;

y: in std\_logic;

F: out std\_logic

);

end XNOR\_ent;

---------------------------------------

architecture XNOR\_arch of XNOR\_ent is

begin

process(x, y)

begin

-- compare to truth table

if (x=y) then

F <= '1';

else

F <= '0';

end if;

end process;

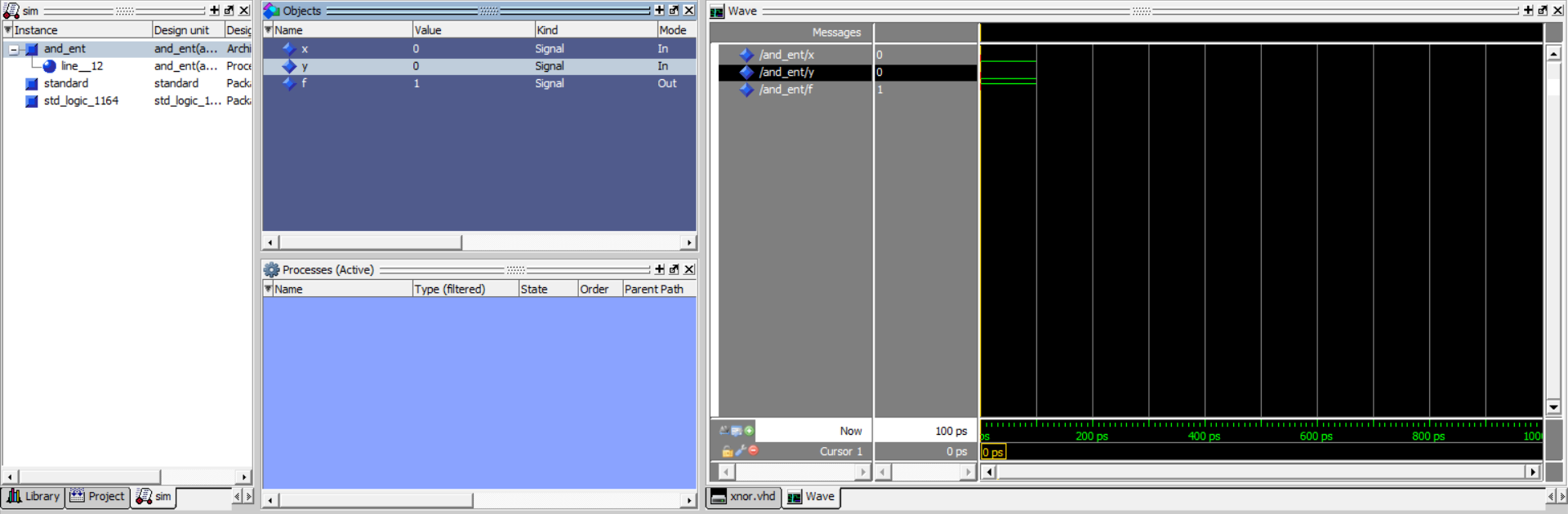
end XNOR\_arch;

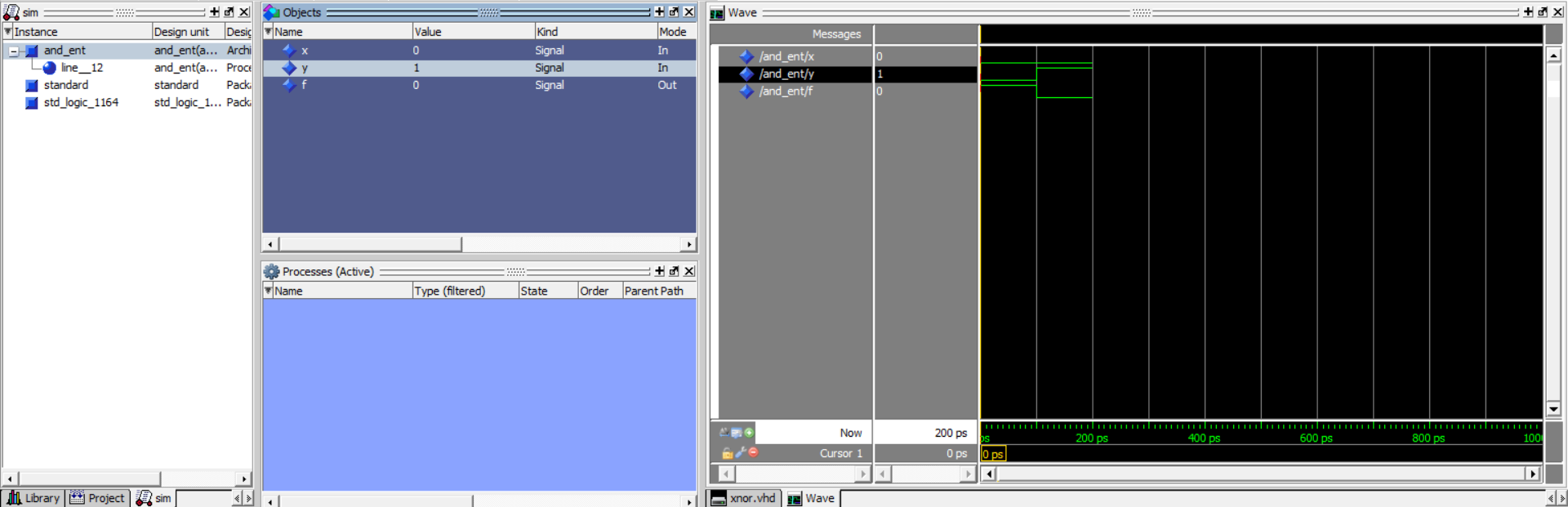
architecture XNOR\_beh of XNOR\_ent is

begin

F <= x xnor y;

end XNOR\_beh;





**NOT**

library ieee;

use ieee.std\_logic\_1164.all;

--------------------------------------

entity NOT\_ent is

port( x: in std\_logic;

y: in std\_logic;

F: out std\_logic

);

end NOT\_ent;

---------------------------------------

architecture NOT\_arch of NOT\_ent is

begin

process(x, y)

begin

-- compare to truth table

if (x='0') then

F <= '0';

else

F <= '1';

end if;

end process;

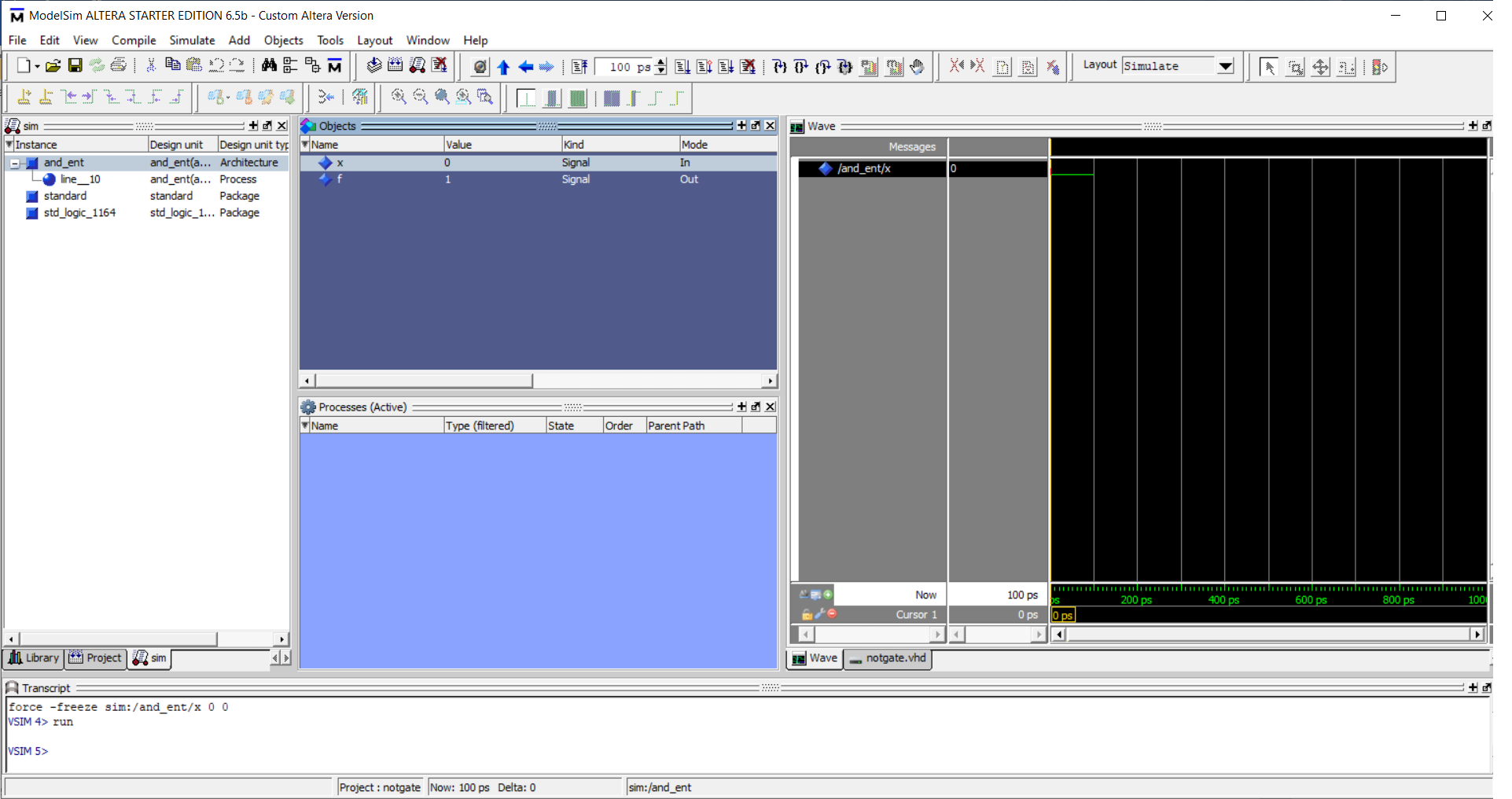
end NOT\_arch;

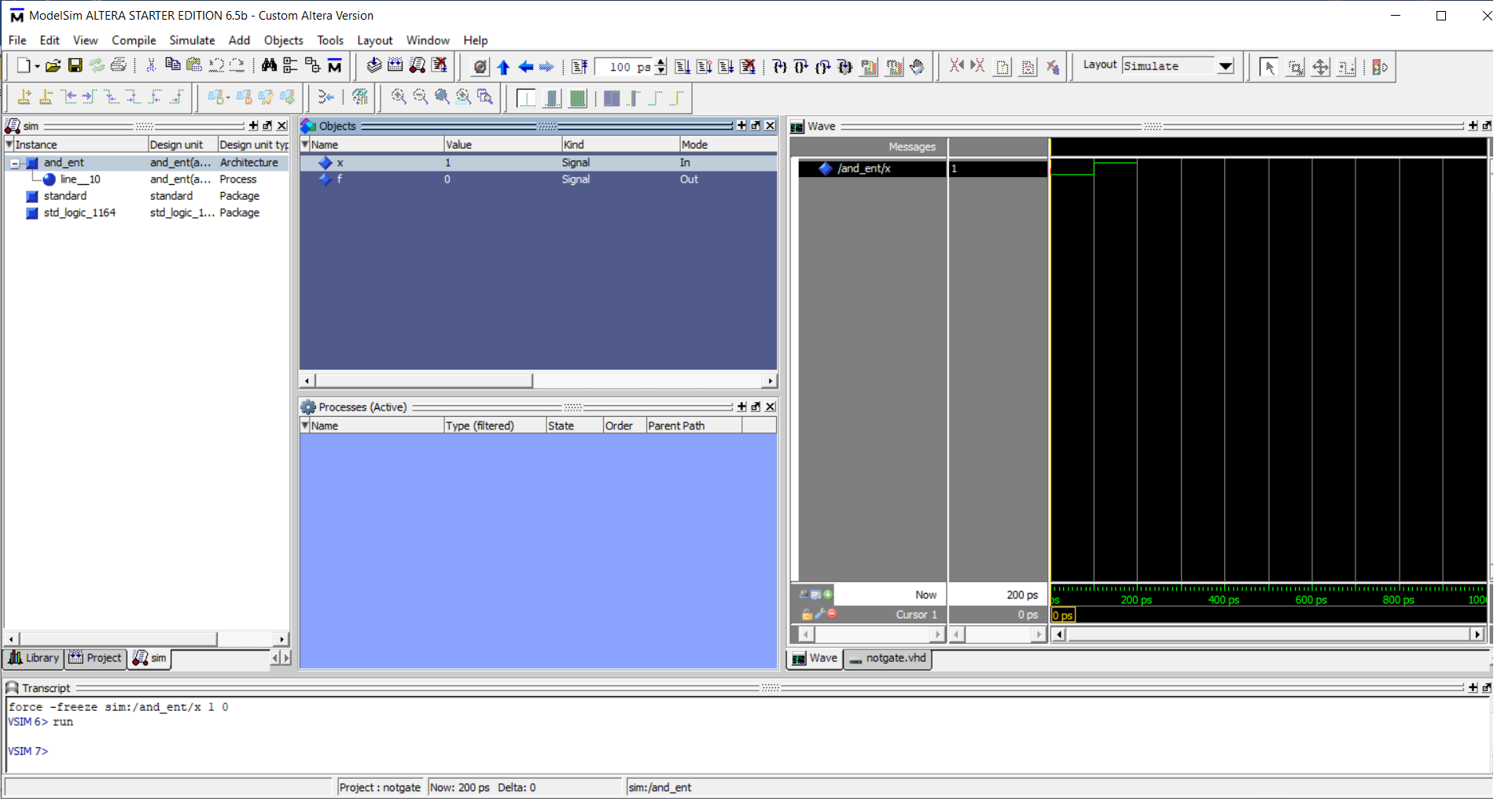
architecture NOT\_beh of NOT\_ent is

begin

F <= x not y;

end NOT\_beh;





**Conclusion:**

We successfully implemented logic gates using VHLD programming; thus, obtained correct output.

**Post Lab Descriptive Questions:**

1. What are two types of HDL?

Hardware description languages need the ability to Describe ; Simulate at behavioural,structural ,mixed level. and to synthesize (structure from behaviour)

1. **VHDL**

it stands for very high-speed integrated circuit hardware description language. It is a programming language used to model a digital system by dataflow, behavioral and structural style of modeling.It is used in electronic design automation to describe digital and mixedsignal systems such as field-programmable gate arrays and integrated circuits

1. **Verilog**

it standardized as IEEE 1364, is a hardware description language (HDL) used to model electronic systems. It is most commonly used in the design and verification of digital circuits at the register-transfer level of abstraction. It is also used in the verification of analog circuits and mixed-signal circuits, as well as in the design of genetic circuits