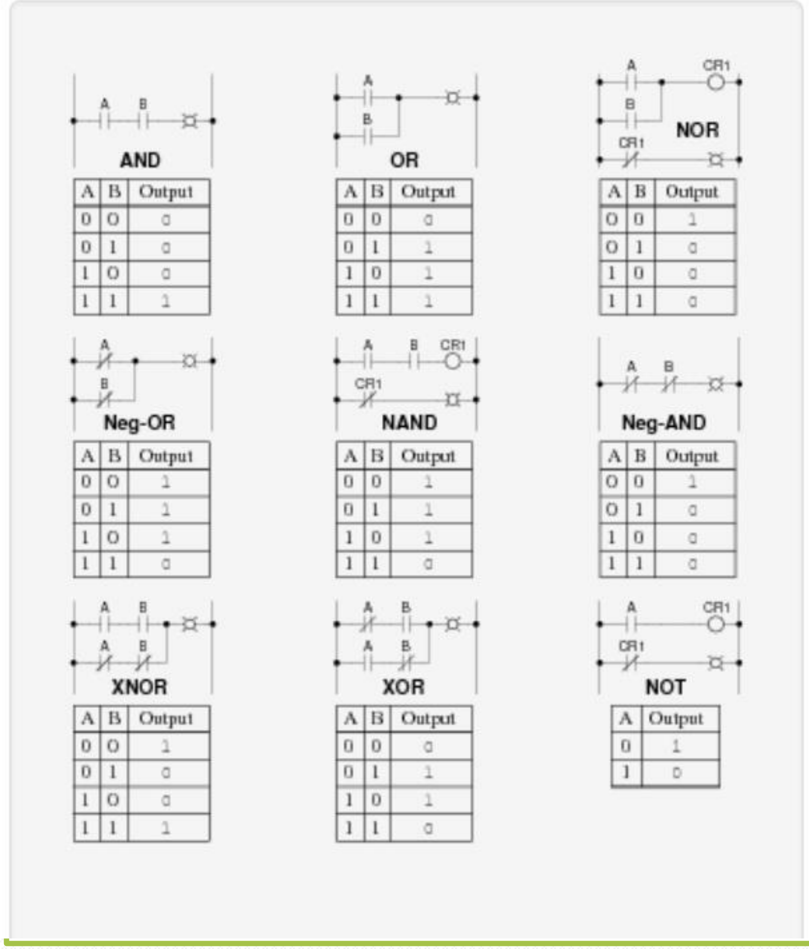
# VHDL: Very High Speed Integrated Circuits Hardware Description Language

Eng: Mohamed Ismail

# Discussion

- What is the Digital Design ?
- What is  truth table ?
- Examples :  AND , OR , NOR ,XOR
- Applications Done By VHDL ?
  ( Report)



AND

| A | B | Output |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

OR

| A | B | Output |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

NOR

| A | B | Output |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

Neg-OR

| A | B | Output |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

NAND

| A | B | Output |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Neg-AND

| A | B | Output |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

XNOR

| A | B | Output |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

XOR

| A | B | Output |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

NOT

| A | Output |
|---|---|
| 0 | 1 |
| 1 | 0 |

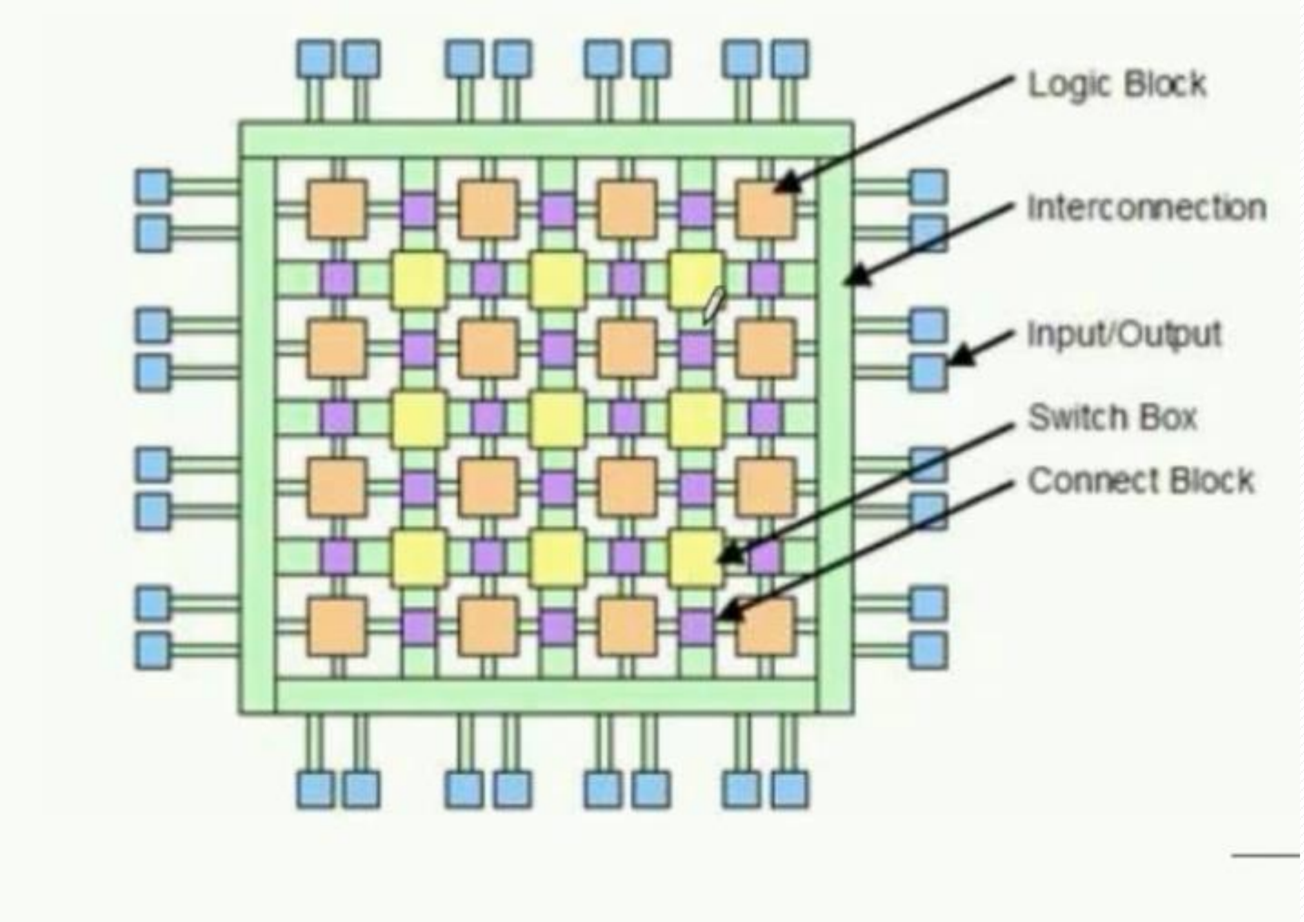
# VHDL Vs prog. Languages

- VHDL is a programming language that allows one to model and develop complex digital systems in a dynamic environment.
- But VHDL is NOT C ... There are some similarities, as with any programming language, but syntax and logic are quite different.
- VHDL is Hardware description language.

Ex:
VHDL uses statements that defines the actual flow of data.....
such as,
 x <= y
 -- this is NOT less than equal to –
this assigns the boolean signal x to the value of boolean signal y...
i.e. x = y this will occur whenever y changes....

# FPGA

# Inside The FPGA



# Downloading a code

# Construction  of HDL code

- Basic constructs of HDL Code :

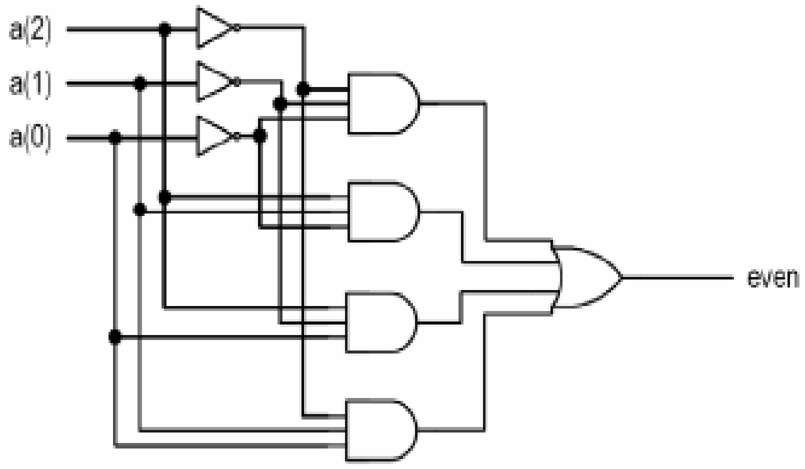
1. Entity
2. Connectivity
3. Concurrency
4. Timing

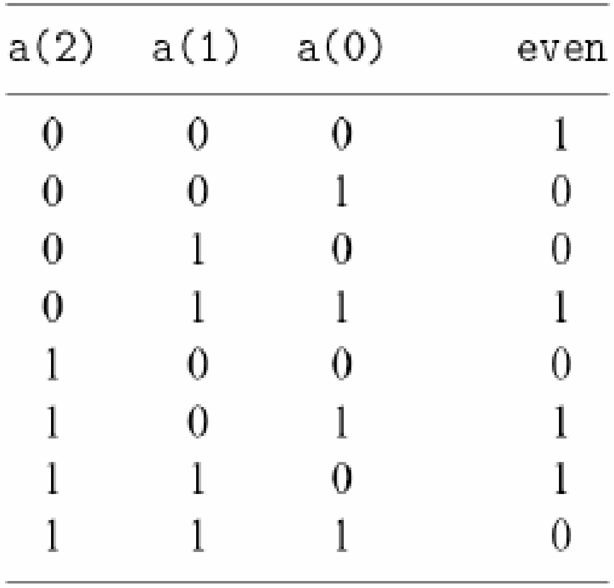
- support Structural implementation .
- Two HDLs used today :

VHDL and Verilog

# Even parity detection circuit

- Input: a(2), a(1), a(0)
- output: even

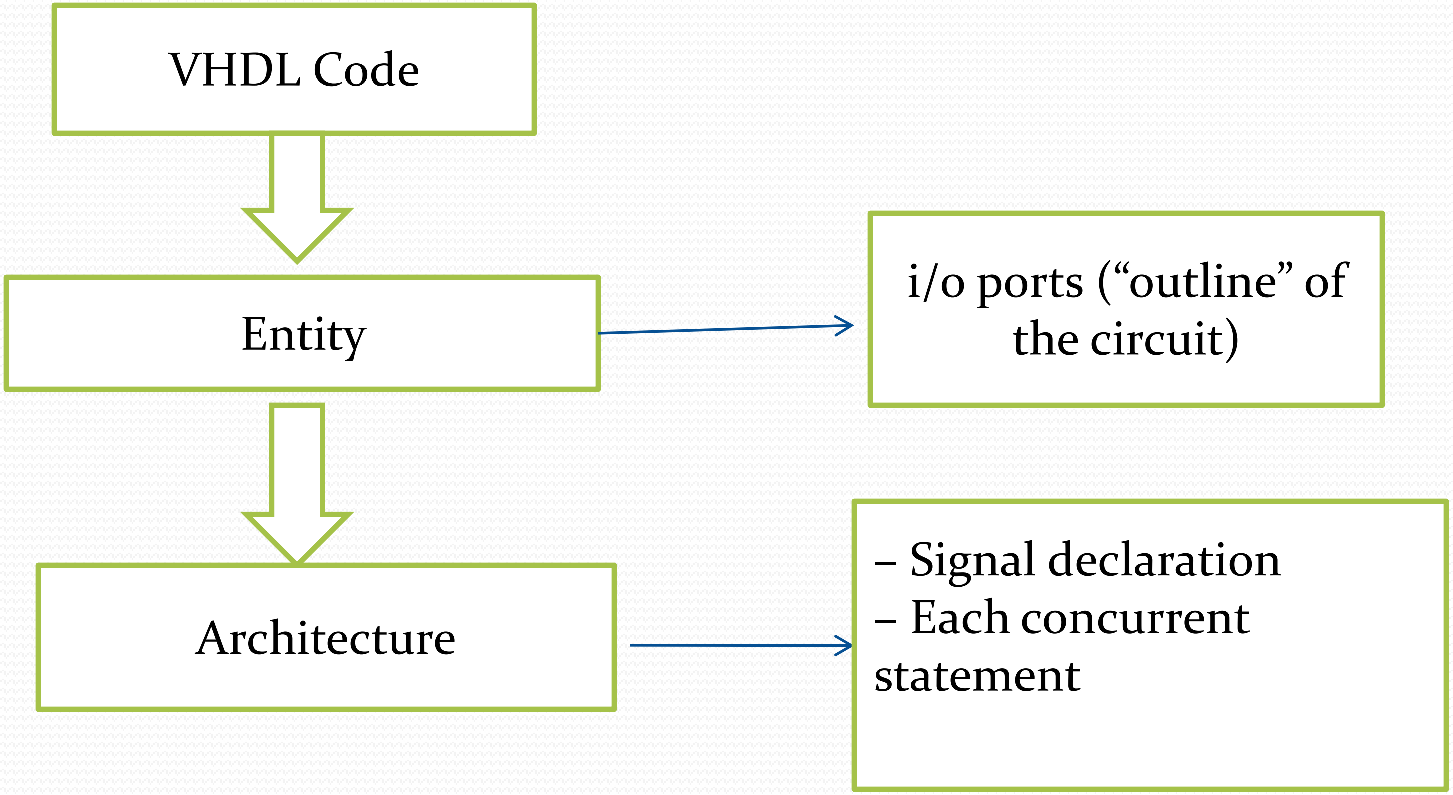




| a(2) | a(1) | a(0) | even |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

$$even = a(2)' \cdot a(1)' \cdot a(0)' + a(2)' \cdot a(1) \cdot a(0) + a(2) \cdot a(1)' \cdot a(0) + a(2) \cdot a(1) \cdot a(0)'$$

# what inside VHDL code ?



VHDL Code

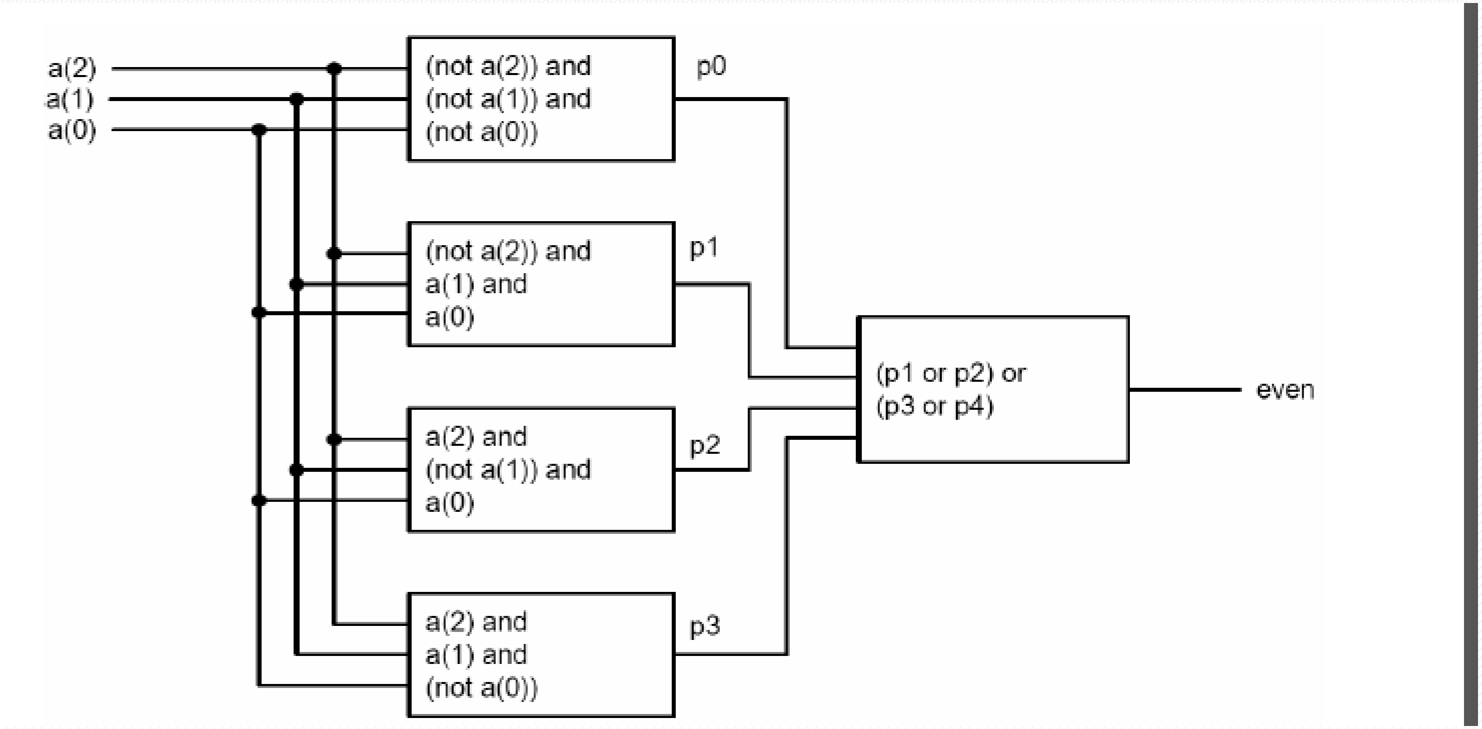
Entity

i/o ports ("outline" of the circuit)

Architecture

– Signal declaration
– Each concurrent statement

# Conceptual interpretation



# VHDL CODE

```vhdl
entity EvenDetector is
   Port ( a : in  STD_LOGIC_VECTOR (2 downto 0);
          even : out  STD_LOGIC);
end EvenDetector;
architecture Behavioral of EvenDetector is
signal p:std_logic_vector (3 downto 0);
begin
p(3)<=(not a(0) ) and a(1) and a(2) ;
p(2)<=(not a(1) ) and a(0) and a(2) ;
p(1)<=(not a(2) ) and a(0) and a(1) ;
p(0)<=(not a(0) ) and (not a(1) ) and (not a(2) ) ;
even <= p(0) or p(1) or p(2) or p(3);
end Behavioral;
```

The entity section contains the outline of the design

Signals can only be defined in this place before the **begin keyword**

# Structural description

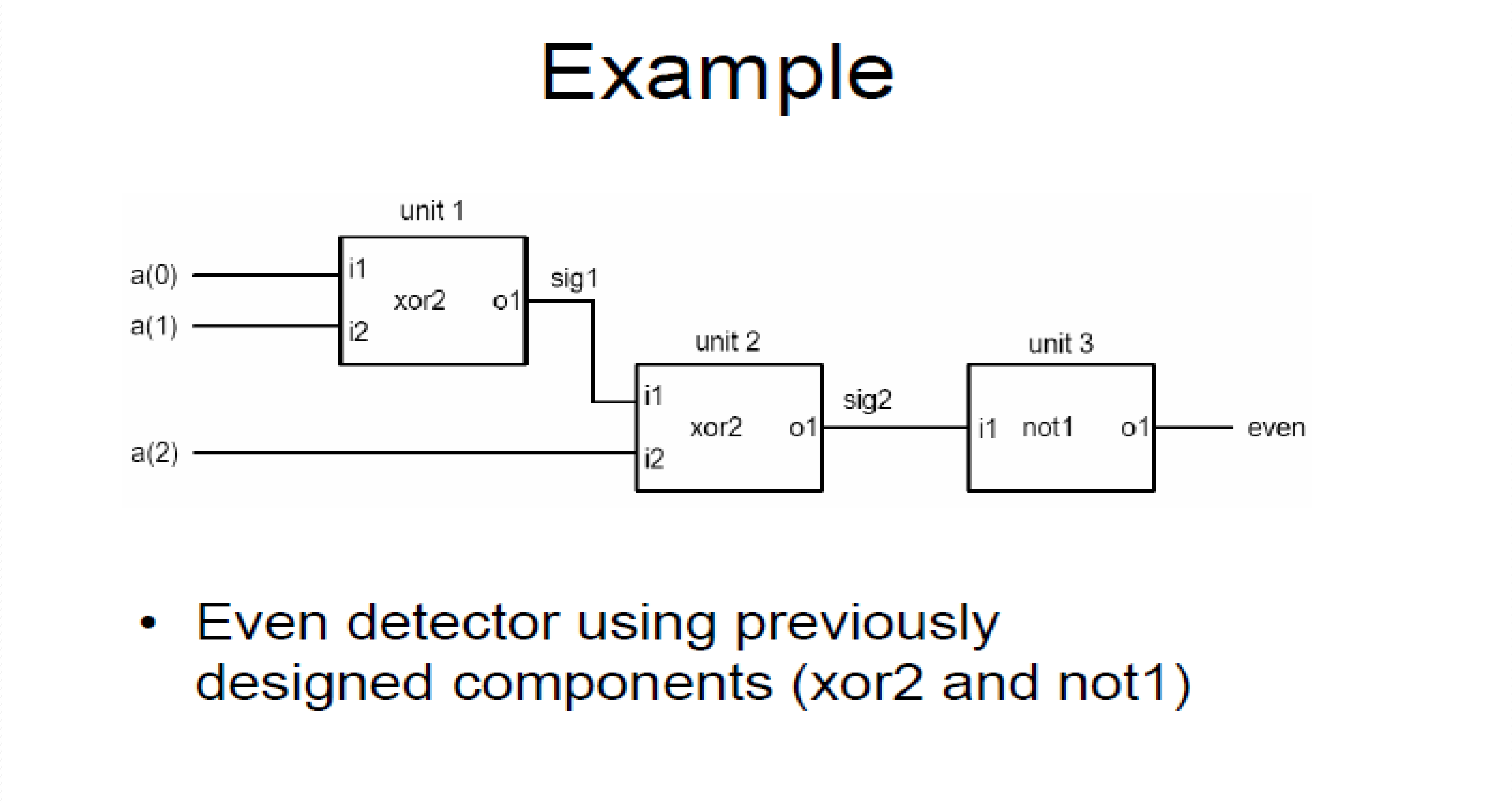
- In structural view, a circuit is constructed by smaller parts.
- Structural description specifies the types of parts and connections.
- Essentially a textual description of a schematic
- Done by using “component” in VHDL
  - – First *declared* (make known)
  - – Then *instantiated* (used)

# XOR

| a | b | output |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Operation : Add the inputs

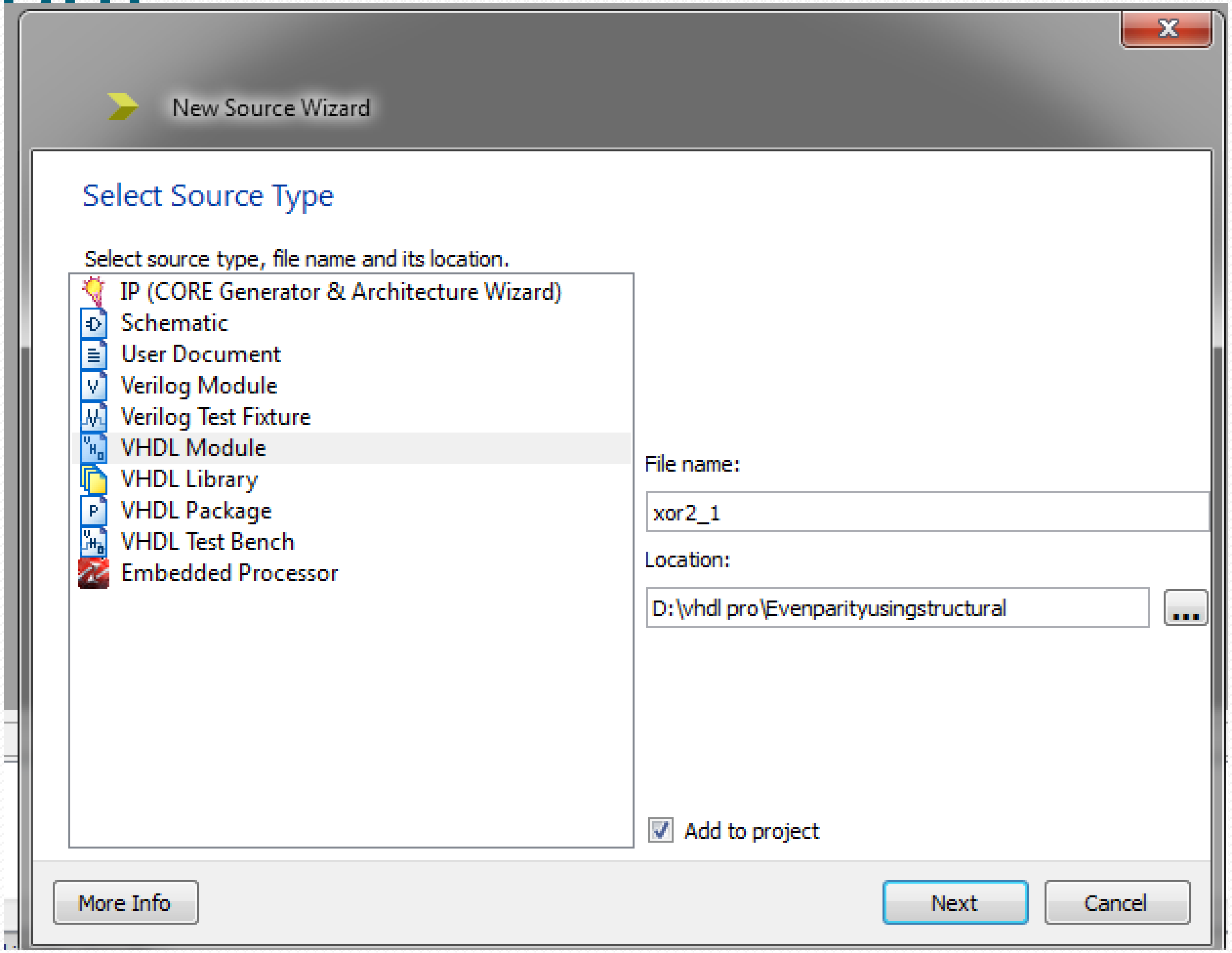
# The even parity using 2 XOR and not



## Example

- Even detector using previously designed components (xor2 and not1)

# Create Xor with 2 inputs and one output



New Source Wizard

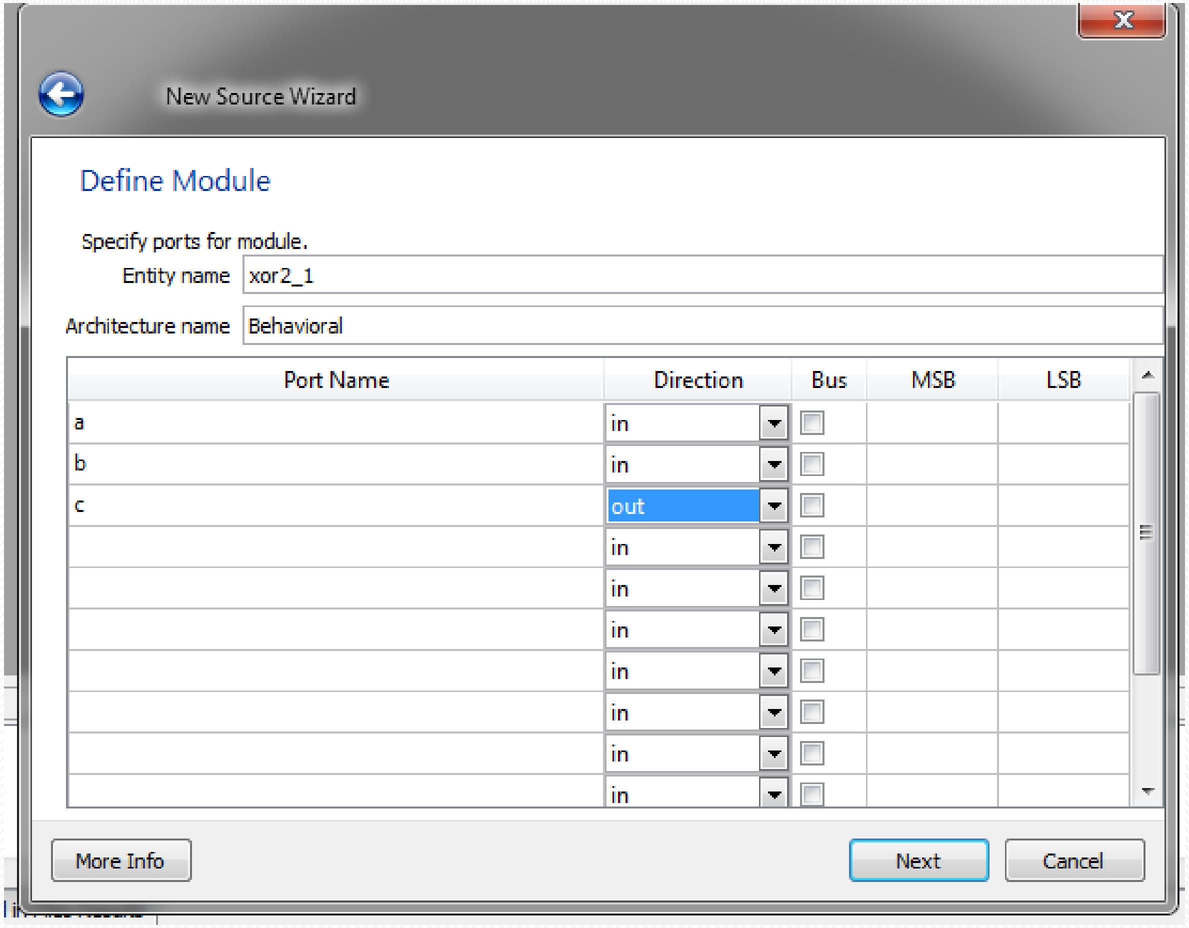
Select Source Type

Select source type, file name and its location.

File name: xor2_1

Location: D:\vhdl pro\Evenparityusingstructural

☑ Add to project



## Define Module

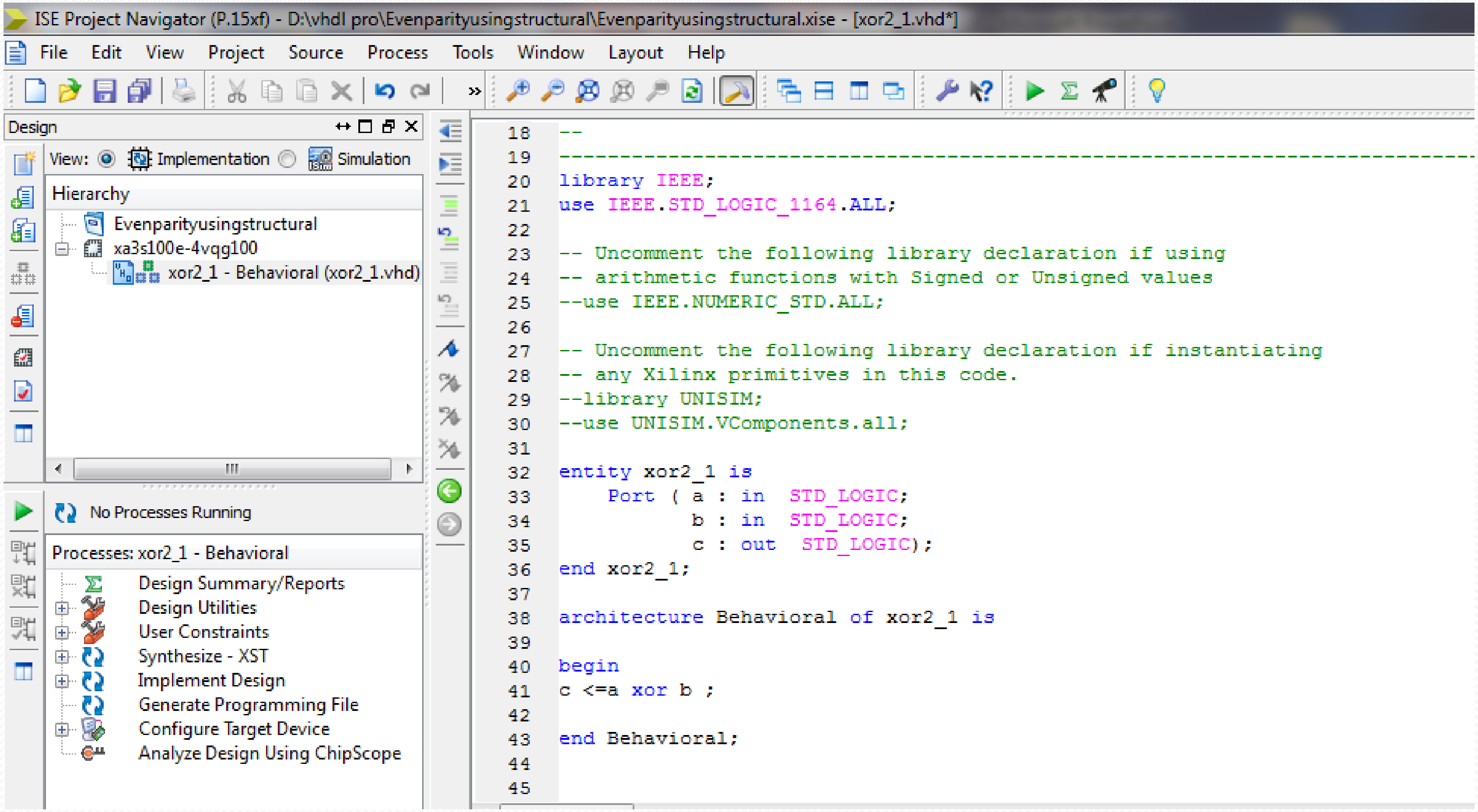
Specify ports for module.

Entity name: xor2_1

Architecture name: Behavioral

| Port Name | Direction | Bus | MSB | LSB |
|---|---|---|---|---|
| a | in | □ | | |
| b | in | □ | | |
| c | out | □ | | |
| | in | □ | | |
| | in | □ | | |
| | in | □ | | |
| | in | □ | | |
| | in | □ | | |
| | in | □ | | |
| | in | □ | | |

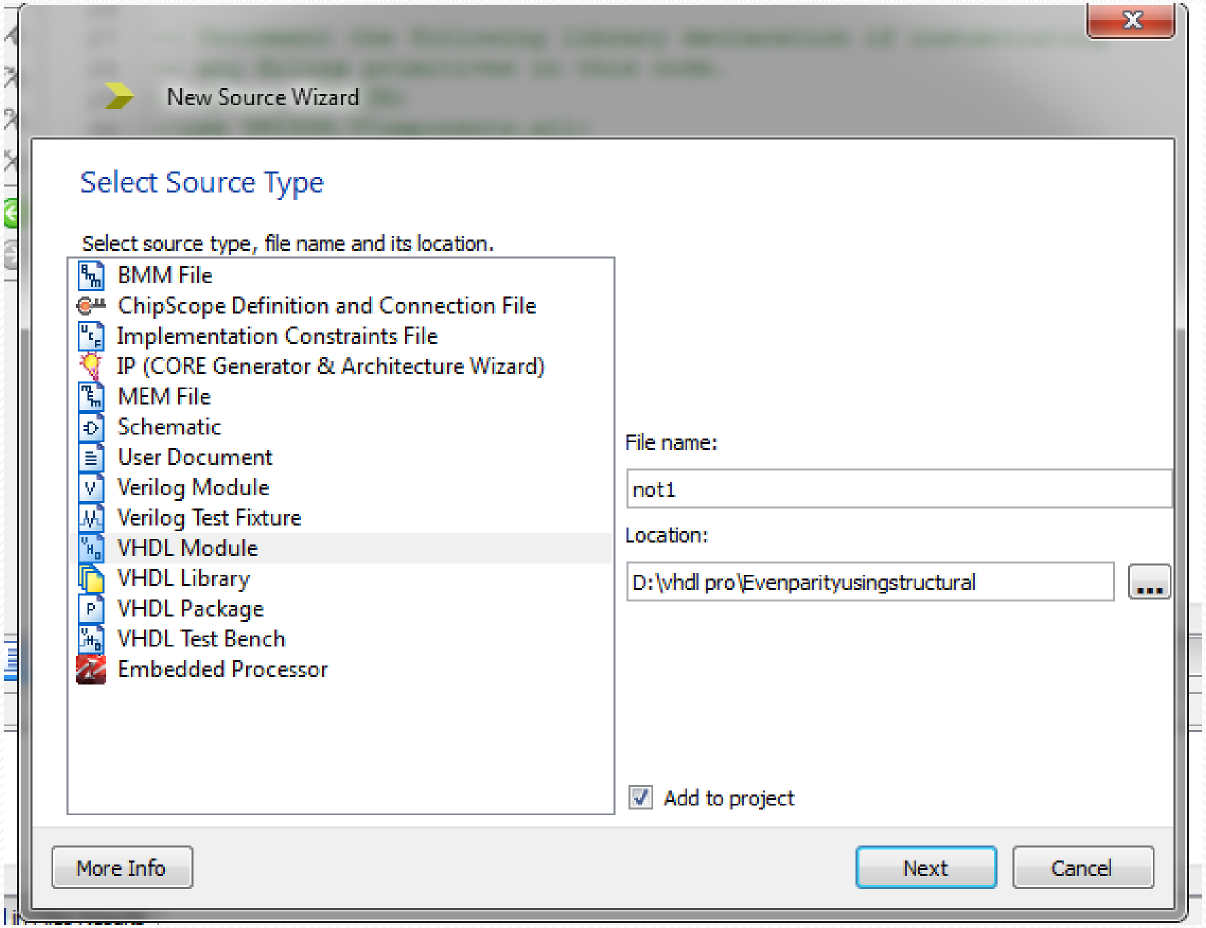
# Define the  Xor  module



ISE Project Navigator (P.15xf) - D:\vhdl pro\Evenparityusingstructural\Evenparityusingstructural.xise - [xor2_1.vhd*]
File
Edit
View
Project
Source
Process
Tools
Window
Layout
Help
Design
View: ● Implementation ○ Simulation
Hierarchy
Evenparityusingstructural
xa3s100e-4vqg100
xor2_1 - Behavioral (xor2_1.vhd)
No Processes Running
Processes: xor2_1 - Behavioral
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
Implement Design
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope

```
--
------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity xor2_1 is
    Port ( a : in  STD_LOGIC;
           b : in  STD_LOGIC;
           c : out  STD_LOGIC);
end xor2_1;

architecture Behavioral of xor2_1 is

begin
c <=a xor b ;

end Behavioral;
```

# Add the not module



New Source Wizard

Select Source Type

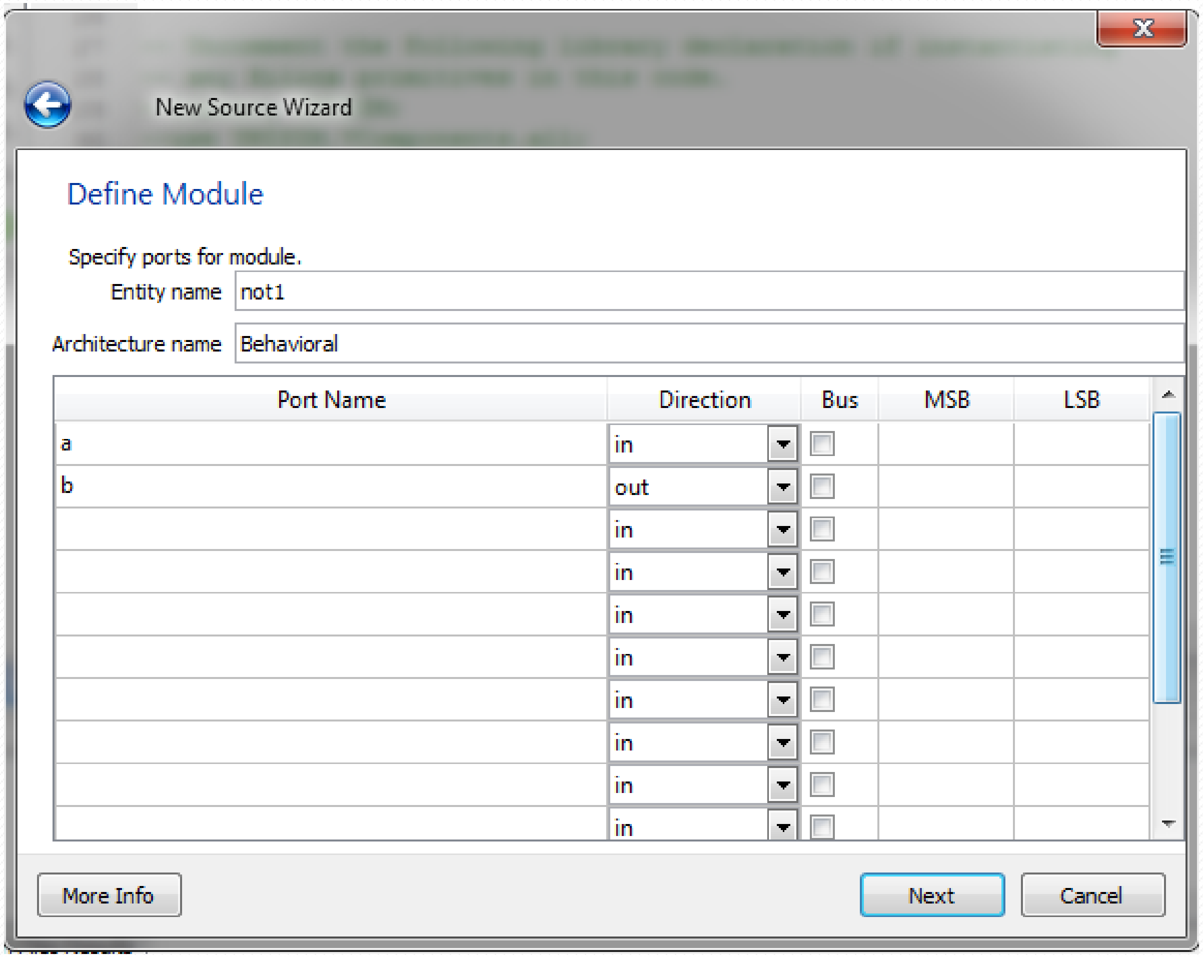
Select source type, file name and its location.

- BMM File
- ChipScope Definition and Connection File
- Implementation Constraints File
- IP (CORE Generator & Architecture Wizard)
- MEM File
- Schematic
- User Document
- Verilog Module
- Verilog Test Fixture
- VHDL Module
- VHDL Library
- VHDL Package
- VHDL Test Bench
- Embedded Processor

File name: not1

Location: D:\vhdl pro\Evenparityusingstructural

☑ Add to project



## Define Module

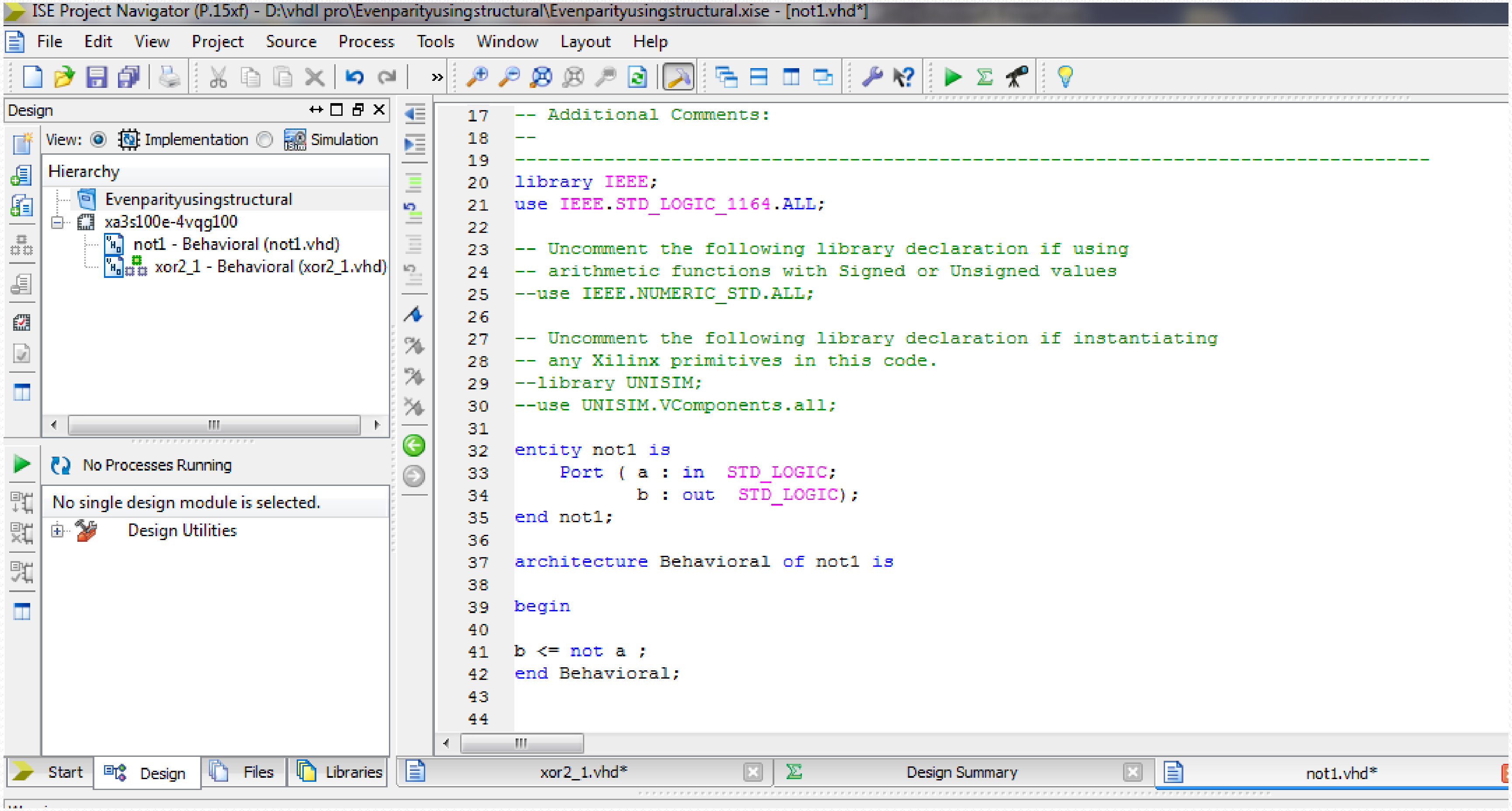
Specify ports for module.

Entity name: not1

Architecture name: Behavioral

| Port Name | Direction | Bus | MSB | LSB |
|---|---|---|---|---|
| a | in | □ | | |
| b | out | □ | | |
| | in | □ | | |
| | in | □ | | |
| | in | □ | | |
| | in | □ | | |
| | in | □ | | |
| | in | □ | | |
| | in | □ | | |
| | in | □ | | |

# Define the not gate



ISE Project Navigator (P.15xf) - D:\vhdl pro\Evenparityusingstructural\Evenparityusingstructural.xise - [not1.vhd*]

View: ● Implementation ○ Simulation

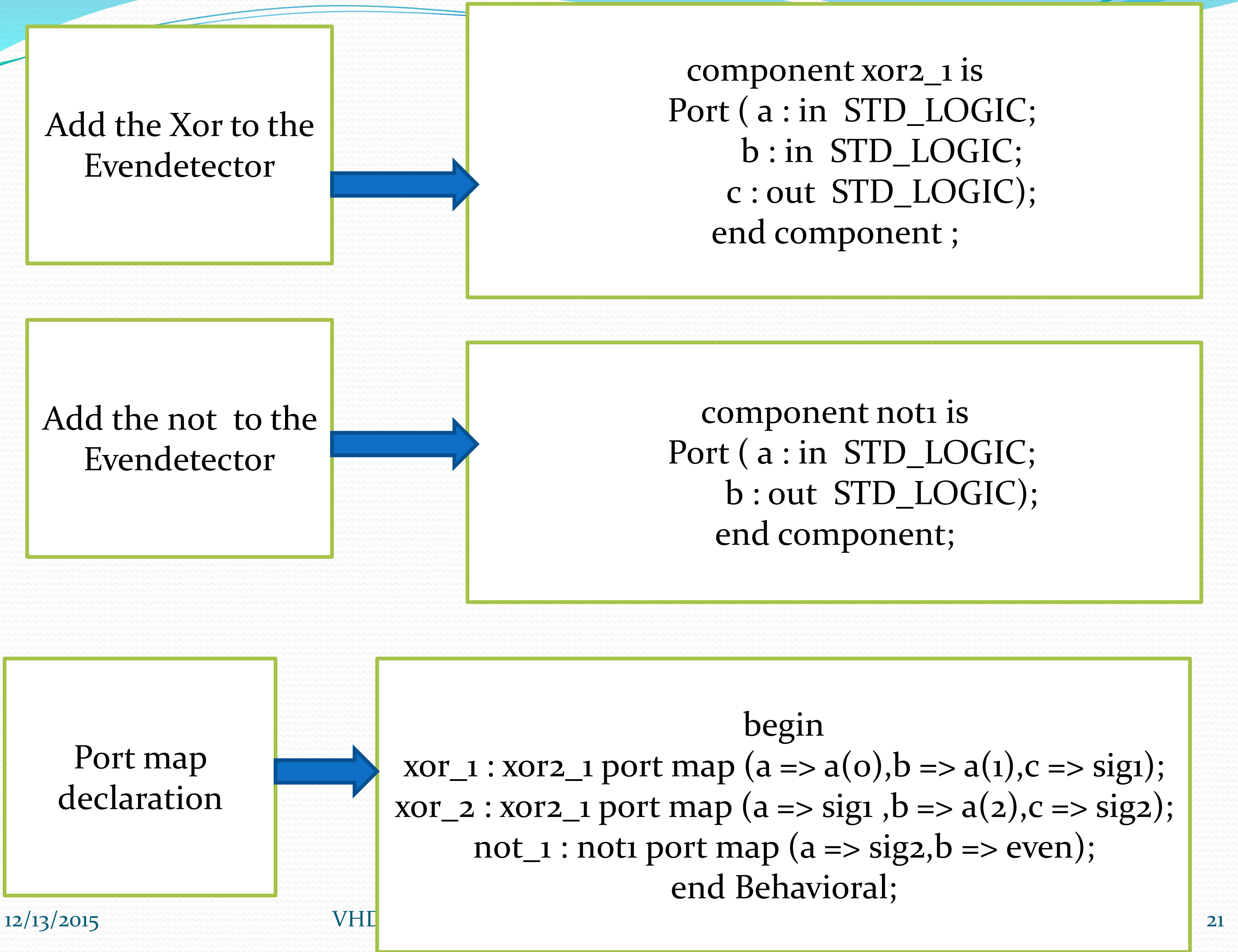
Evenparityusingstructural
xa3s100e-4vqg100
not1 - Behavioral (not1.vhd)
xor2_1 - Behavioral (xor2_1.vhd)

No Processes Running

No single design module is selected.

Design Utilities

```
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity not1 is
    Port ( a : in  STD_LOGIC;
           b : out  STD_LOGIC);
end not1;

architecture Behavioral of not1 is

begin

b <= not a ;
end Behavioral;


```



Add the Xor to the Evendetector

```
component xor2_1 is
Port ( a : in  STD_LOGIC;
       b : in  STD_LOGIC;
       c : out  STD_LOGIC);
end component ;
```

Add the not  to the Evendetector

```
component not1 is
Port ( a : in  STD_LOGIC;
       b : out  STD_LOGIC);
end component;
```

Port map declaration

```
begin
xor_1 : xor2_1 port map (a => a(0),b => a(1),c => sig1);
xor_2 : xor2_1 port map (a => sig1 ,b => a(2),c => sig2);
not_1 : not1 port map (a => sig2,b => even);
end Behavioral;
```

# The VHDL code

```
entity evenparity is
    Port ( a : in  STD_LOGIC_VECTOR (2 downto 0);
           even  : out  STD_LOGIC);
end evenparity;

architecture Behavioral of evenparity is
component xor2_1 is
    Port ( a : in  STD_LOGIC;
           b : in  STD_LOGIC;
           c : out  STD_LOGIC);
end component ;

 component not1 is
    Port ( a : in  STD_LOGIC;
           b : out  STD_LOGIC);
end component;
signal sig1 ,sig2:std_logic;
begin
xor_1 : xor2_1 port map (a => a(0),b => a(1),c => sig1);
xor_2 : xor2_1 port map (a => sig1 ,b => a(2),c => sig2);
not_1 : not1 port map (a => sig2,b => even);


end Behavioral;
```

# Assignment : Next section

- Redo the even parity detector using :
  1. only one xor.
  2. a not circuit .

- Design the Half Adder :
  1. which has 2 inputs :a,b
  2. two outputs :sum , carry