CSCI 150
Introduction to Digital and Computer System Design
Lecture 3: Combinational Logic Design VI

Jetic Gū

Overview

- Focus: Arithmetic Functional Blocks
- Architecture: Combinatory Logical Circuits
- Textbook v4: Ch4 4.3, 4.7; v5: Ch2 2.9, Ch3 3.10
- Core Ideas:

1. Subtraction I
2. VHDL

Review

Unsigned Binary Adder

1-bit Half Adder

- Half adder
  input X, Y
  output S, C

Carries	C
Augend	X
Addend	$+Y$
Sum	S

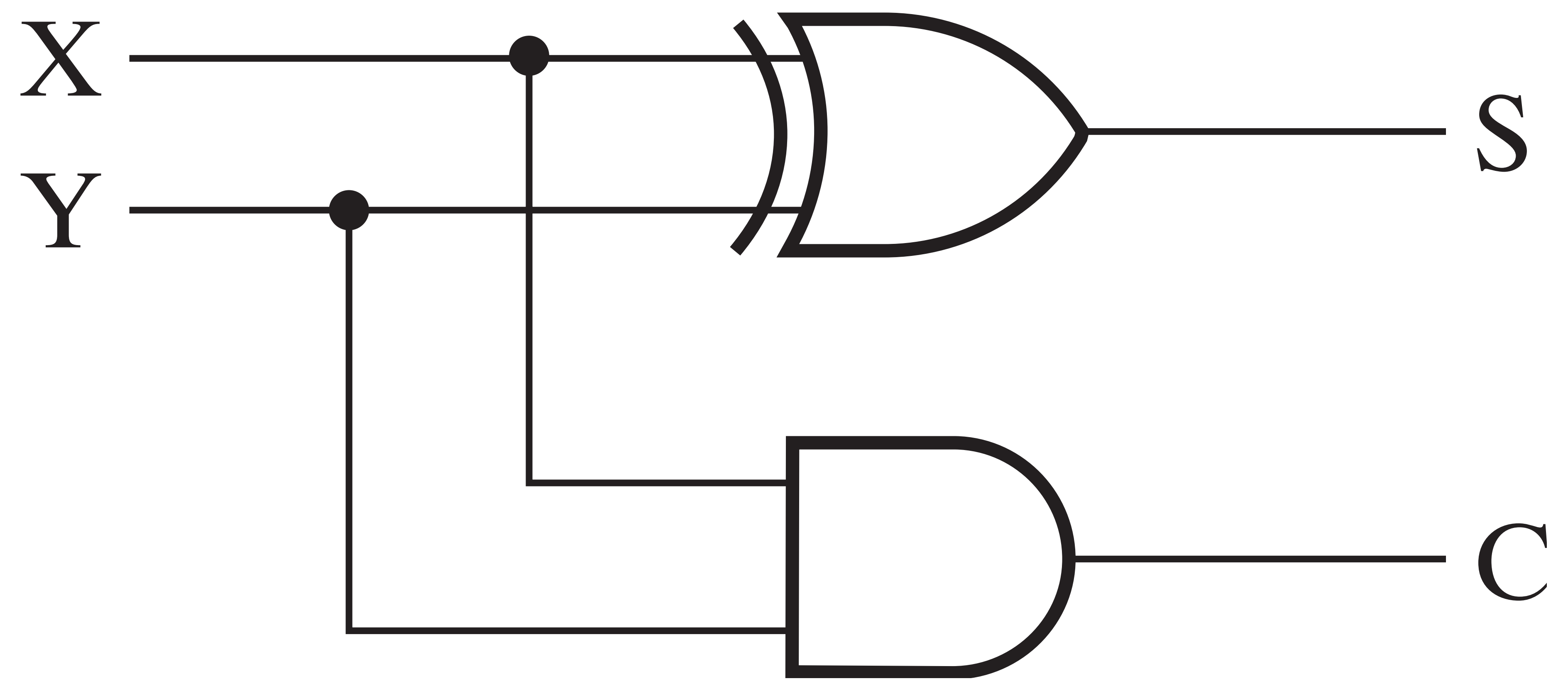
Input
Output

Review

1-bit Half Adder

Carries	C
Augend	X
Addend	$+Y$
Sum	S

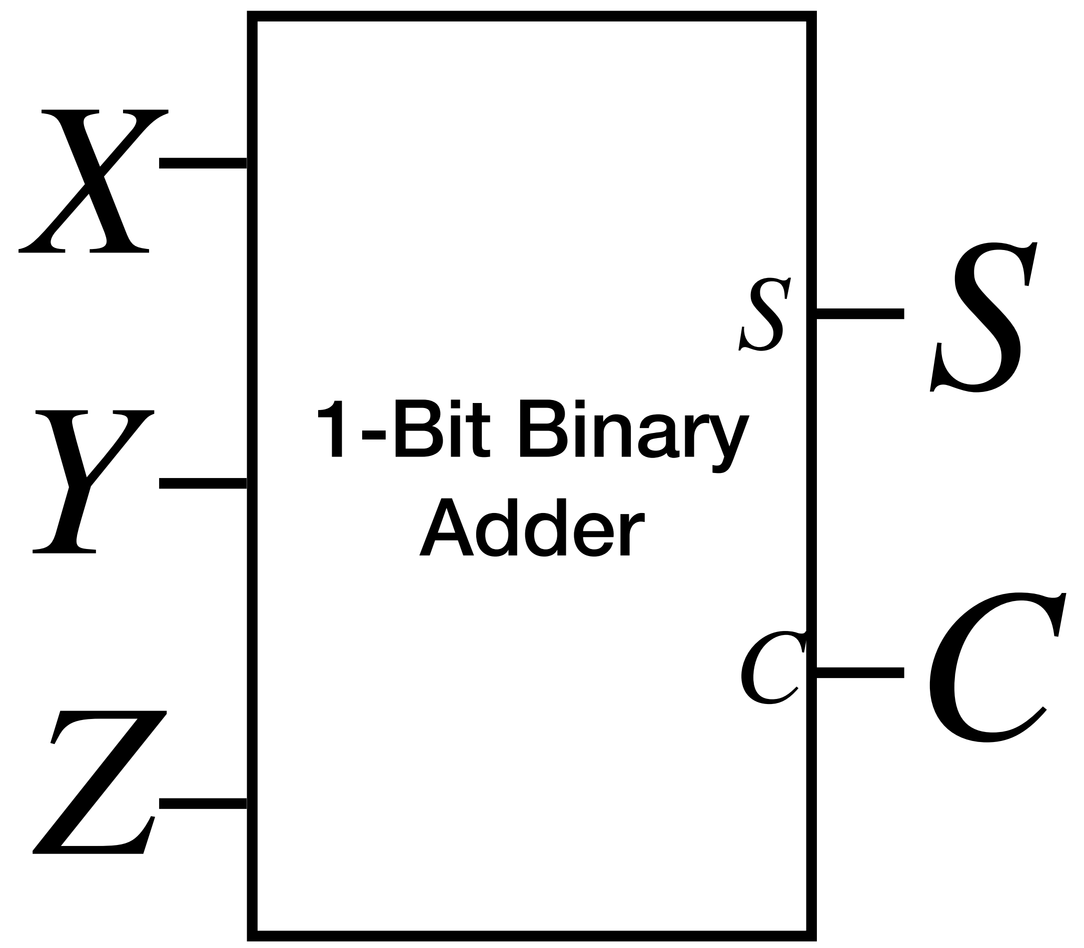
Input
Output

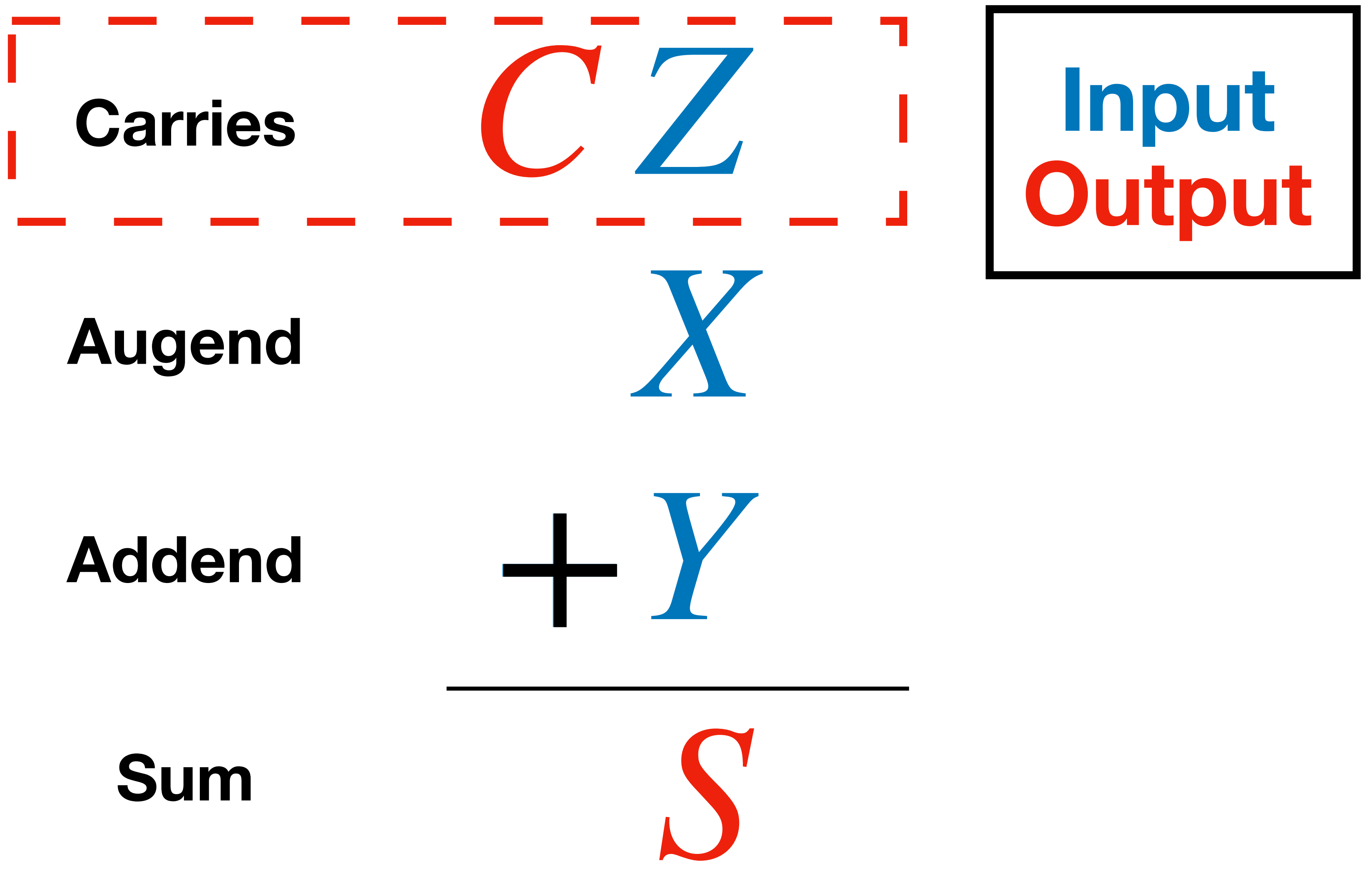


Review

1-bit Full Adder

- Full adder input X, Y, Z; output S, C





Review

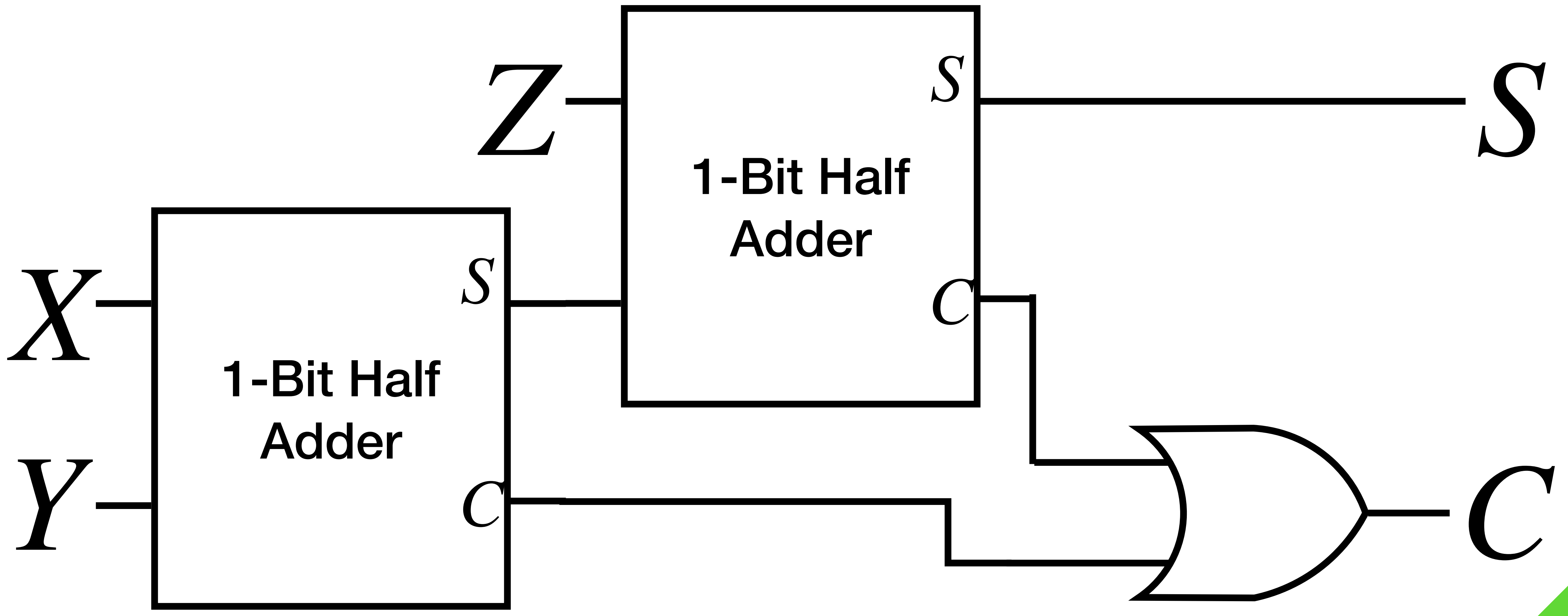
1-bit Full Adder

- Full adder
  input X, Y, Z;
  output S, C

- Half adder1
  input X, Y
  output S', C'

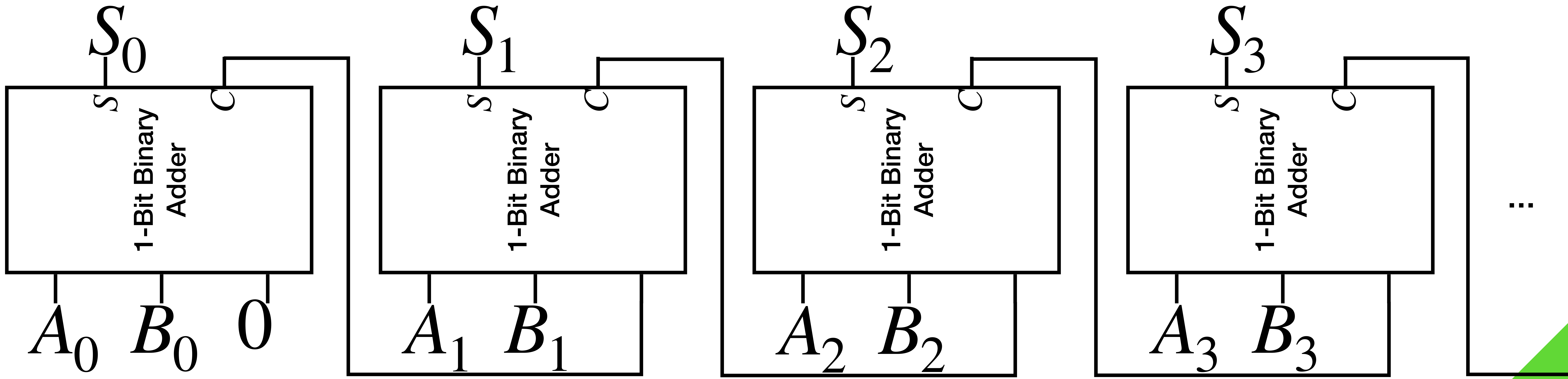
- Half adder2
  input S', Z
  output S, C''

$$C = C' + C''$$



n-bit Full Adder

- Ripple Carry Adder



Unsigned Binary Subtraction I

P1
Subtraction

Unsigned Binary Subtraction

- Input: Minuend and Subtrahend
  Previous borrow

- Output: Last borrow, difference

Borrows	0
Minuend	10110
Subtrahend	-10011
Difference	

Review

P1
Subtraction

Unsigned Binary Subtraction

- Input: Minuend and Subtrahend
  Previous borrow

- Output: Last borrow, difference

Borrows	000110
Minuend	10110
Subtrahend	$-$10011
Difference	00011

Input
Output

This method works when the Minuend is greater than the Subtrahend!

Review

Unsigned Binary Subtraction

$$X > Y, F = X - Y$$

- We learned to perform subtraction, by subtracting the smaller number from the greater number

P1
Subtraction

Unsigned Binary Subtraction

- Input: Minuend and Subtrahend
  Previous borrow

- Output: Last borrow, difference

Borrows	000110
Minuend	10110
Subtrahend	$-$10011
Difference	00011

Input
Output

Concept

P1
Subtraction

Unsigned 1-bit Binary Subtraction

- Input: Minuend X and Subtrahend Y
  Previous borrow Z

- Output: Last borrow B, difference D

	$B\ Z$
Borrows	$1\ 0$
Minuend X	0
Subtrahend Y	$-\ 1$
Difference D	1

Input
Output

Concept

P1
Subtraction

Unsigned 1-bit Binary Subtraction

- Input: Minuend X and Subtrahend Y
  Previous borrow Z

- Output: Last borrow B, difference D

	$B\ Z$
Borrows	$1\ 0$
Minuend X	0
Subtrahend Y	-1
Difference D	1

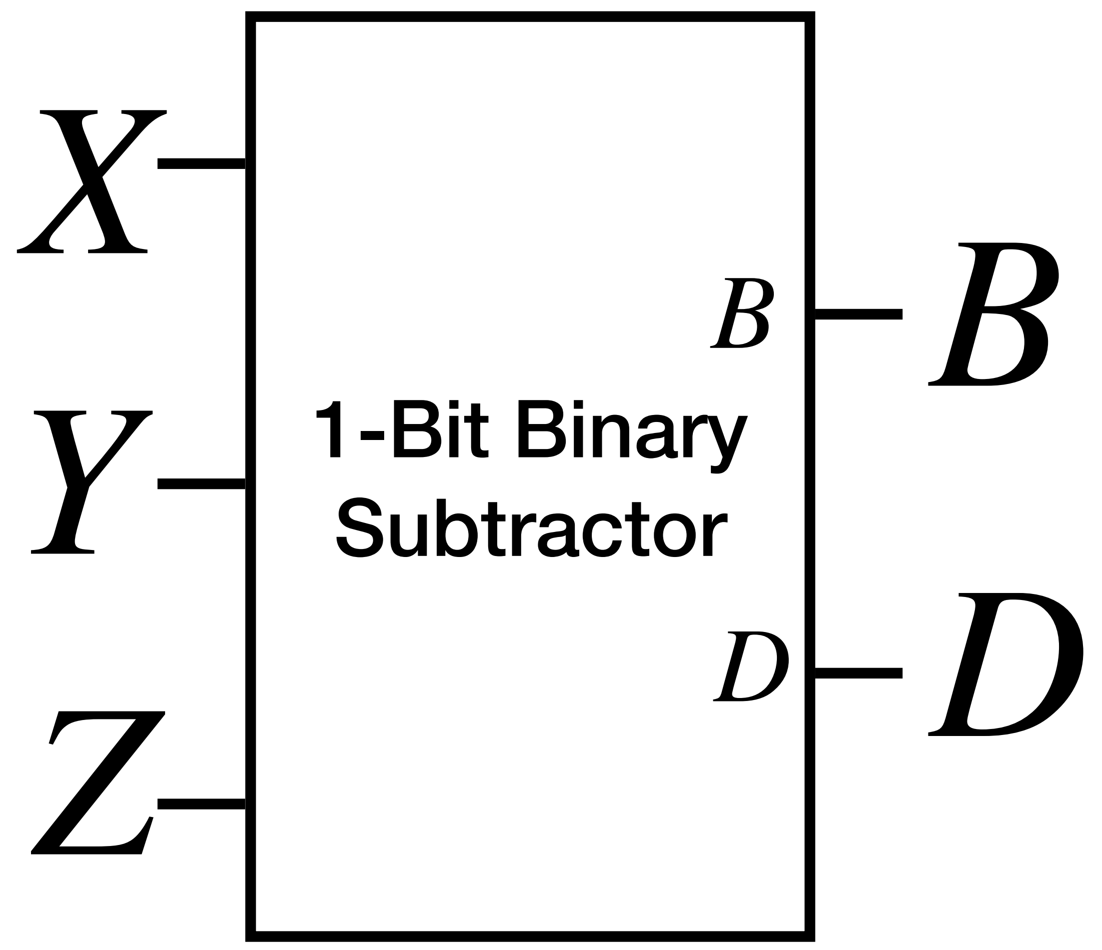
Input
Output

X	Y	Z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Concept

Unsigned 1-bit Binary Subtraction

- Implementation using 3-to-8 Decoder
  - $B(X, Y, Z) = \Sigma m(1,2,3,7)$
  - $D(X, Y, Z) = \Sigma m(1,2,4,7)$



Concept

P1
Subtraction

Unsigned Binary Subtraction

Technology
- 1 bit Unsigned Subtractor

- Input: Minuend and Subtrahend
  Previous borrow

- Output: Last borrow, difference

$$\begin{array}{lr}
 & B \qquad\qquad Z \\
\textbf{Borrows} & 000110 \\
\textbf{Minuend } X & 10110 \\
\textbf{Subtrahend } Y & -\,10011 \\
\hline
\textbf{Difference } D & 00011
\end{array}$$

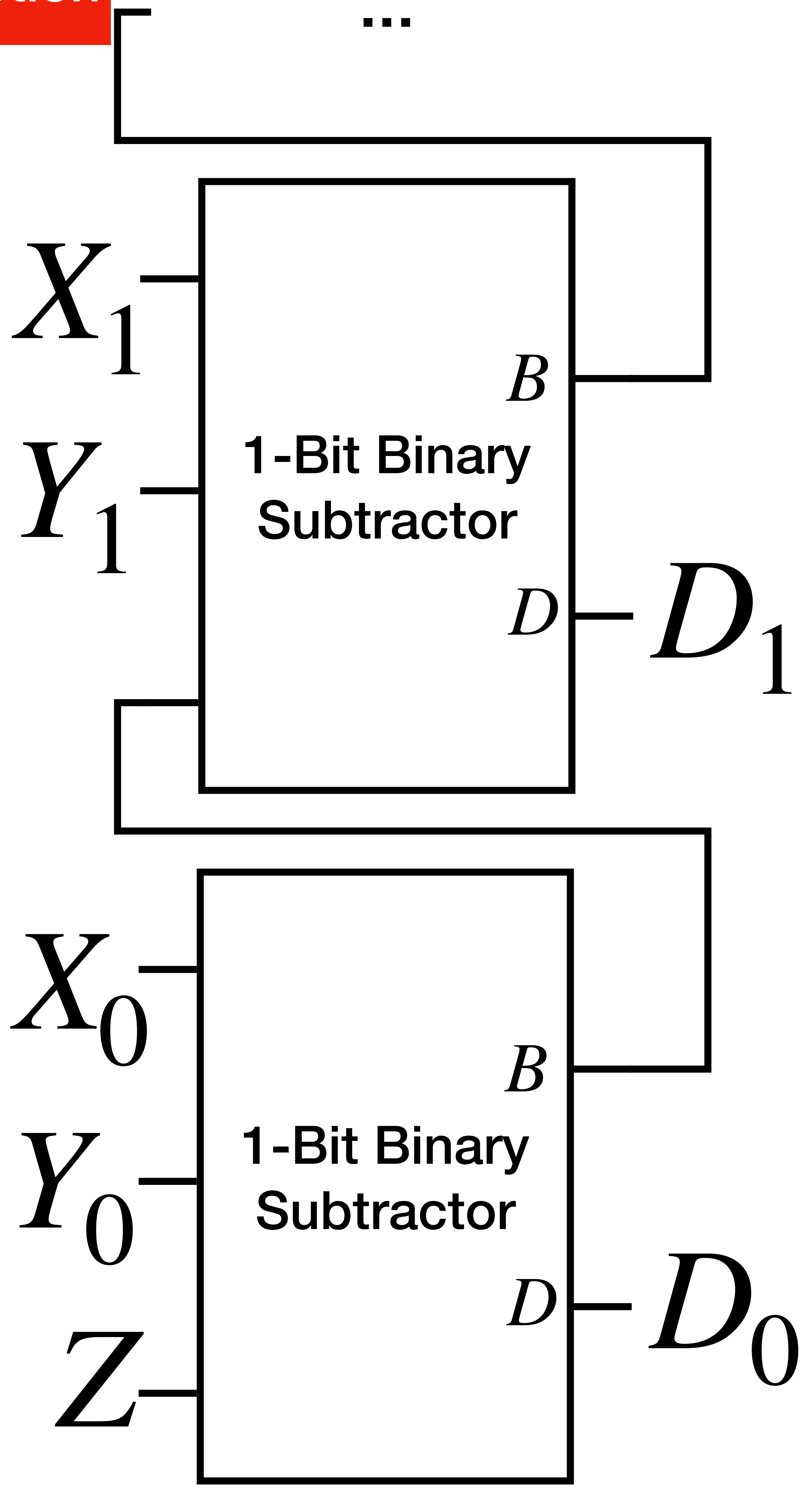
Input
Output

Concept

P1
Subtraction

Unsigned Binary Subtraction

Technology
- 1 bit Unsigned Subtractor



	B Z
Borrows	000110
Minuend $X_{0:n-1}$	10110
Subtrahend $Y_{0:n-1}$	$-$ 10011
Difference $D_{0:n-1}$	00011

Input
Output

Concept

Hardware Description Language

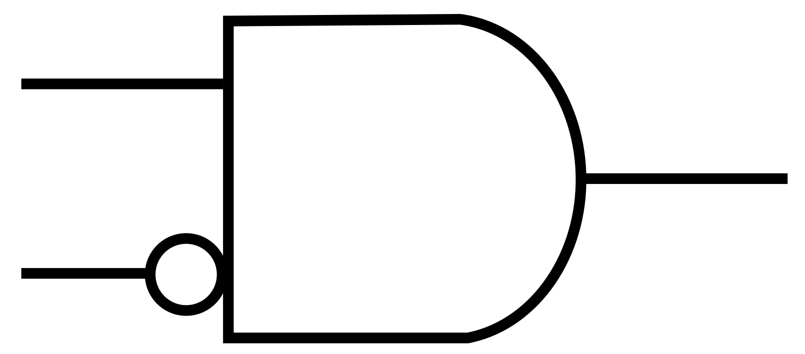
VHDL (VHSIC-HDL): Very High Speed Integrated Circuit Hardware Description Language

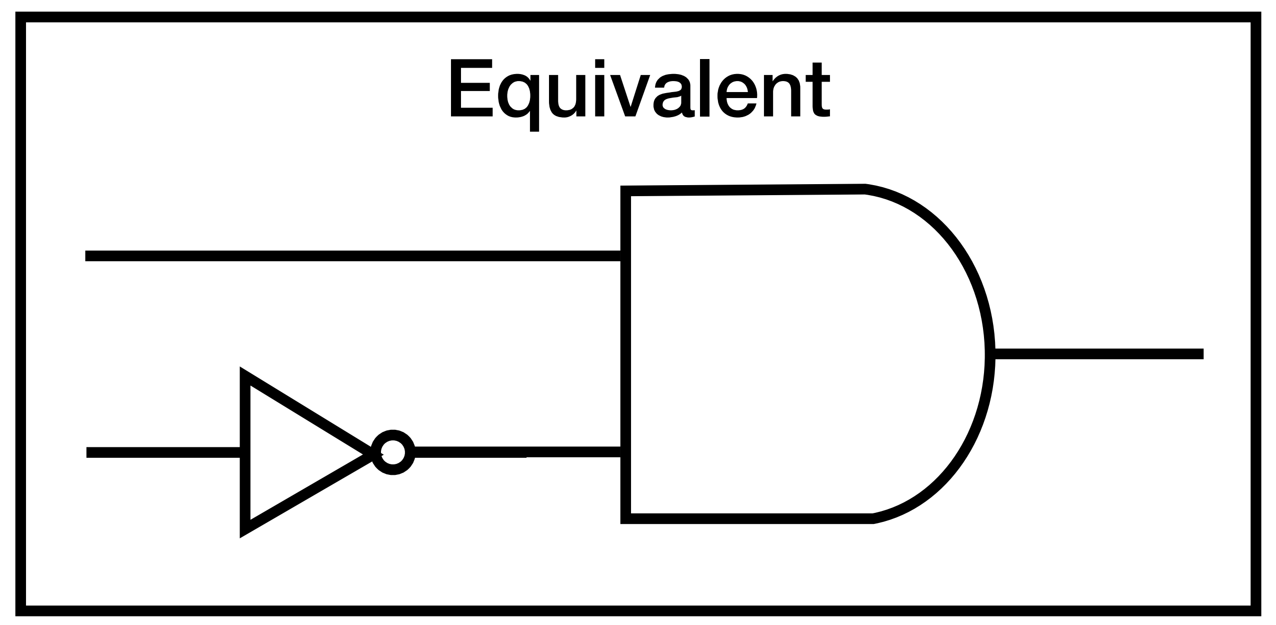
What is HDL

- Designing complex circuits using logic circuit diagrams is inefficient
- Hardware Description Language
  - Like programming language, describes hardware structures and behaviours
  - More efficient
  - Common languages
    - Verilog
    - VHDL

P2
VHDL

Creating a AND1INV model

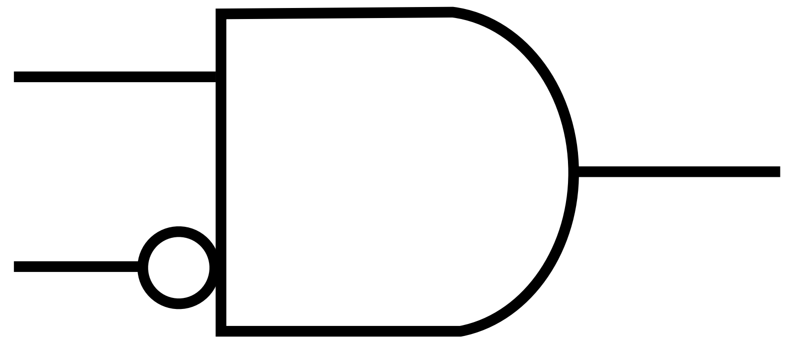


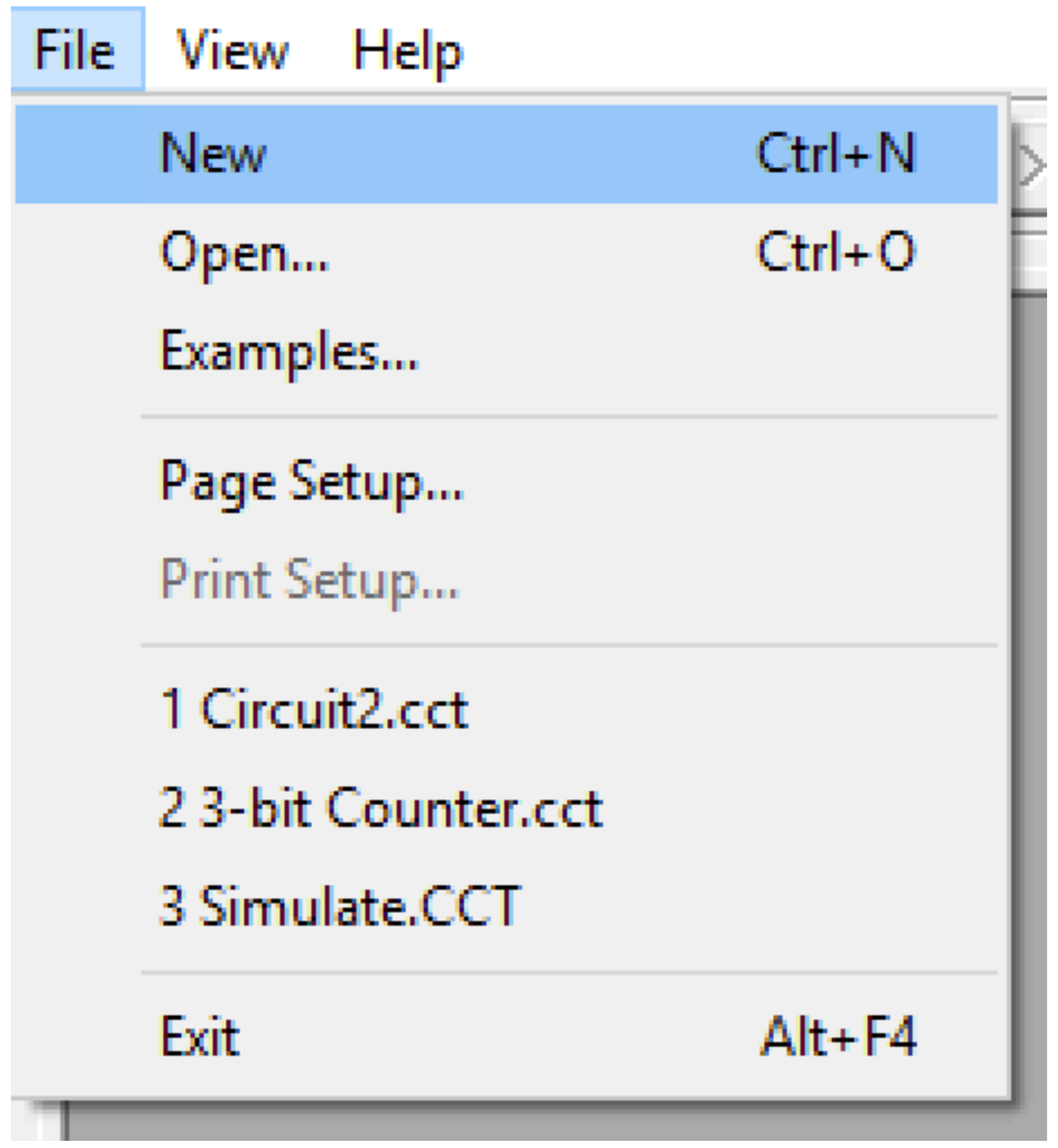


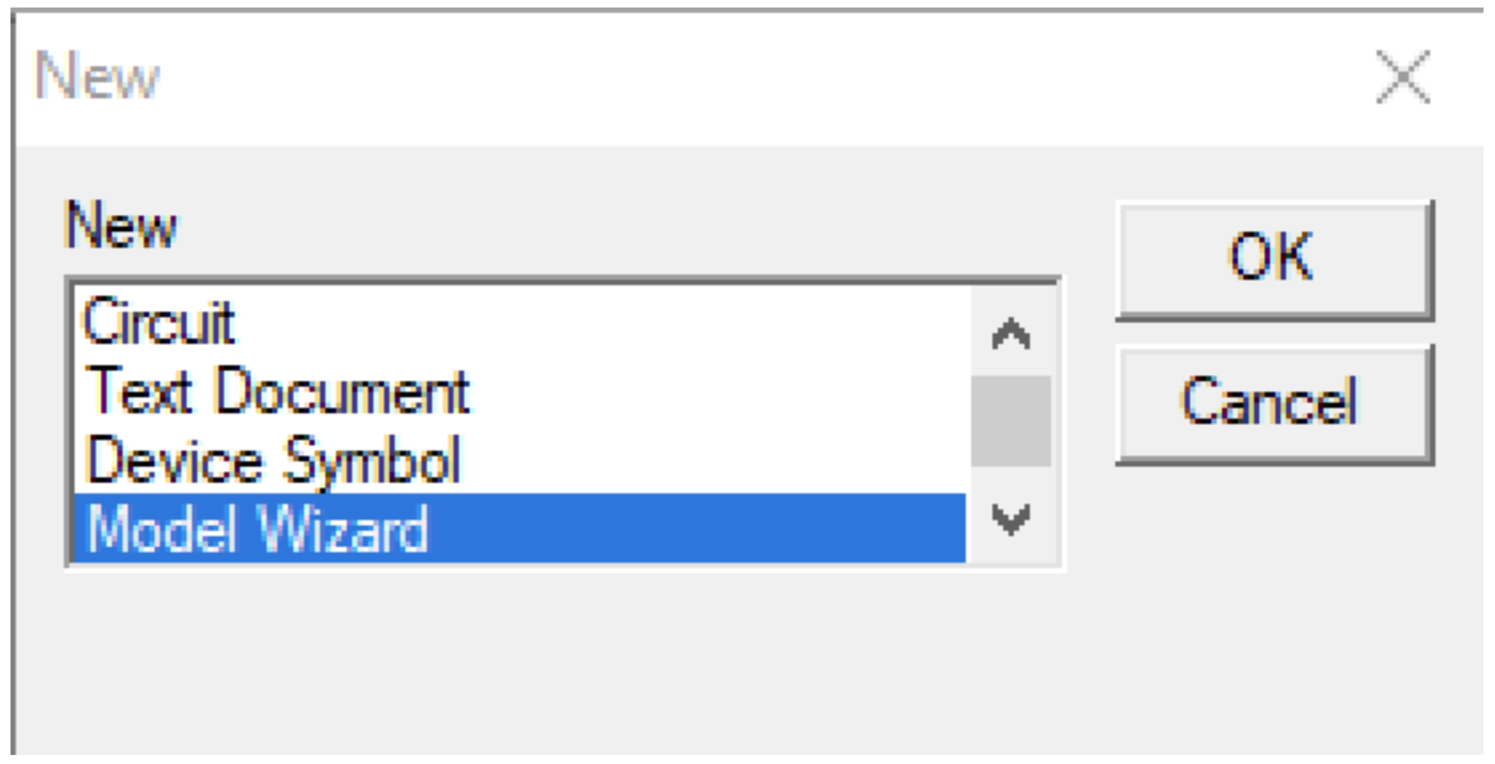
Tutorial

P2
VHDL

Creating a AND1INV model





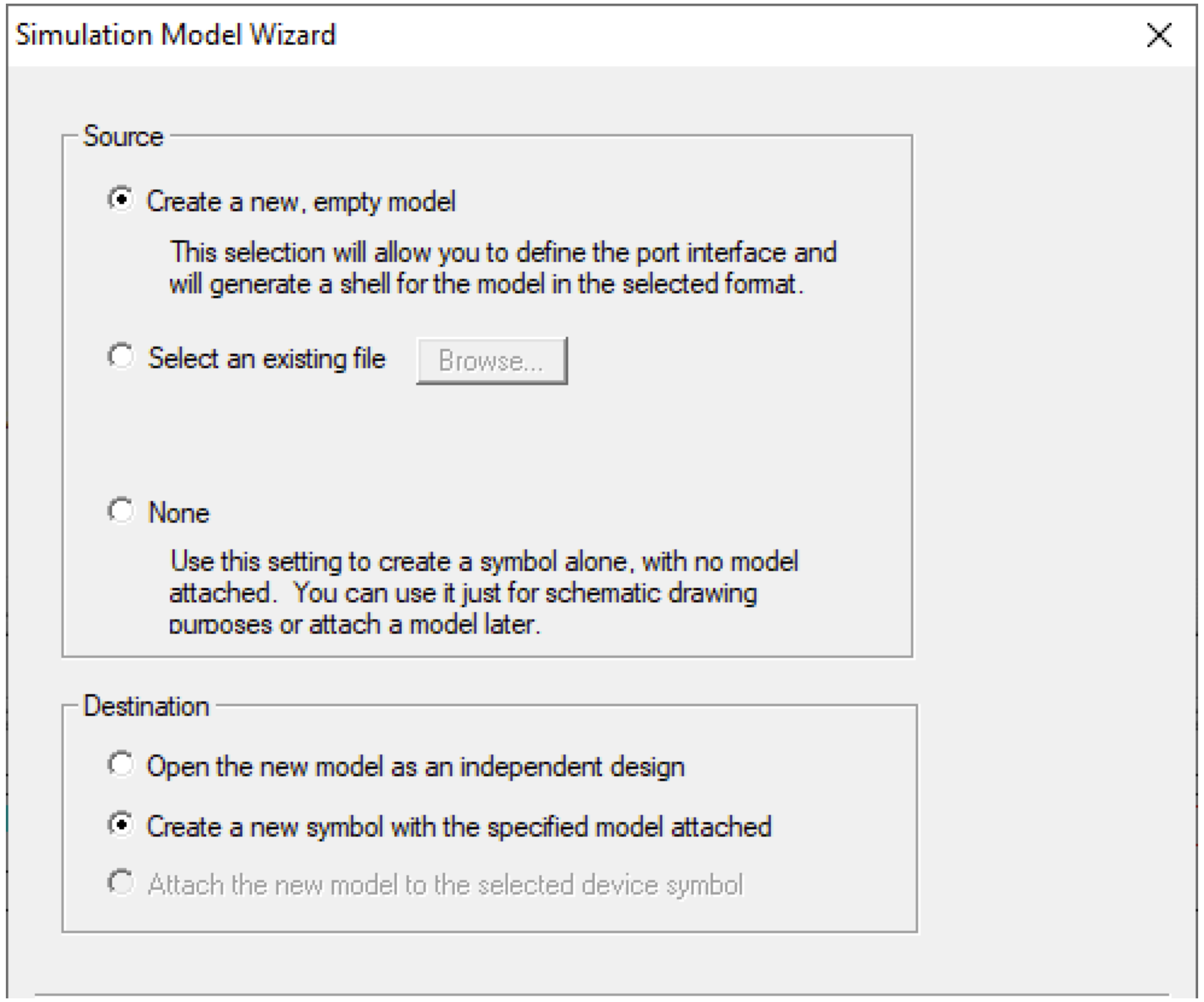


Tutorial

1. Go to the `File` menu, select `New` command. Select `Model Wizard` and click `OK`

P2
VHDL

Creating a AND1INV model



Simulation Model Wizard

Source

(●) Create a new, empty model

This selection will allow you to define the port interface and will generate a shell for the model in the selected format.

( ) Select an existing file Browse...

( ) None

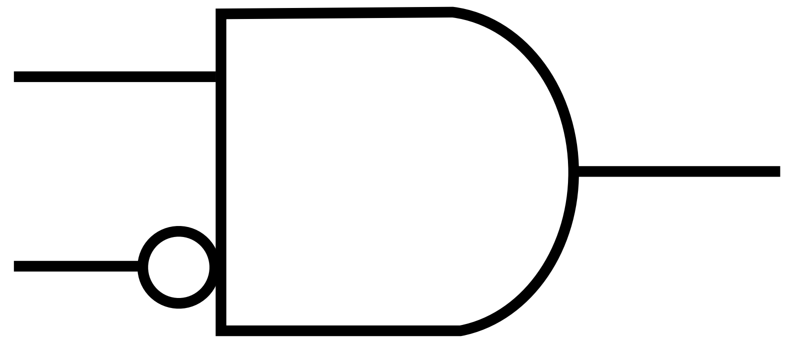
Use this setting to create a symbol alone, with no model attached. You can use it just for schematic drawing purposes or attach a model later.

Destination

( ) Open the new model as an independent design

(●) Create a new symbol with the specified model attached

( ) Attach the new model to the selected device symbol

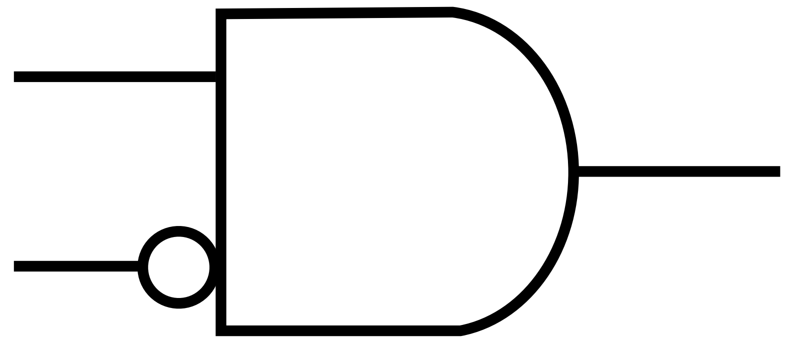


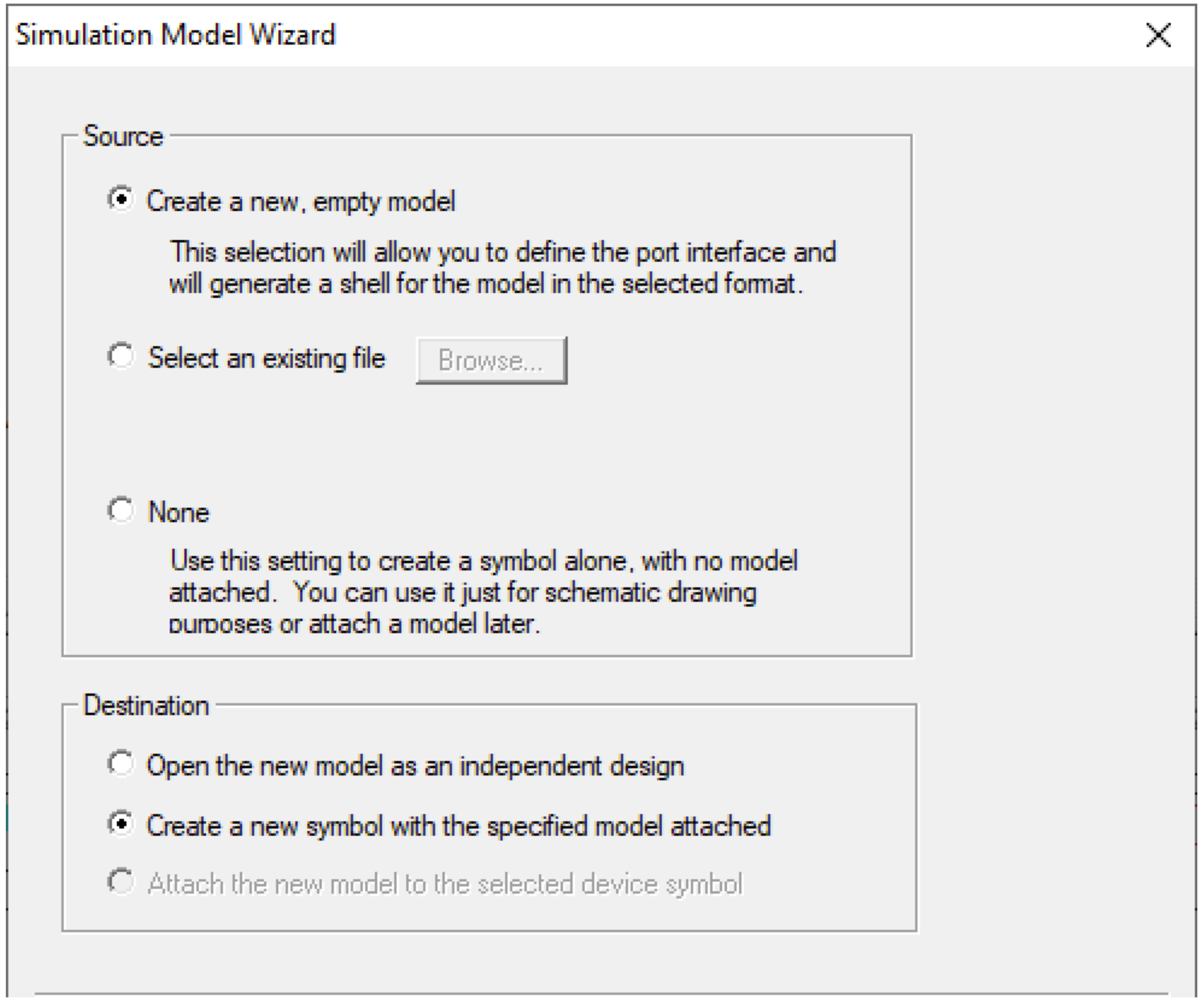
2. **Source:** `Create a new, empty model`, **Destination:** `Create a new symbol with the specified model attached`. Click `Next`

Tutorial

P2
VHDL

Creating a AND1INV model





Simulation Model Wizard

Source

[✓ Create a new, empty model]

This selection will allow you to define the port interface and will generate a shell for the model in the selected format.

Select an existing file

None

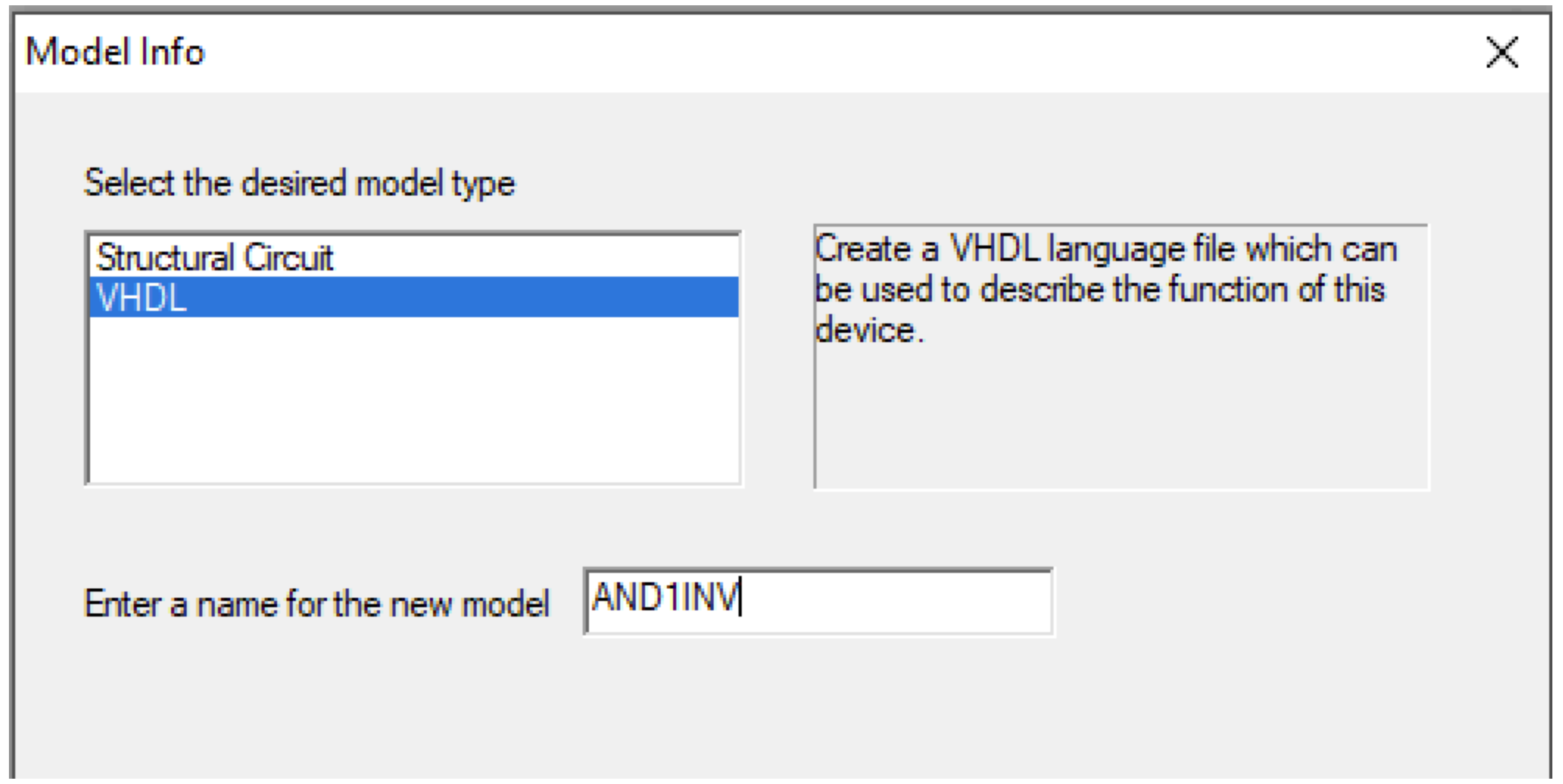
Use this setting to create a symbol alone, with no model attached. You can use it just for schematic drawing purposes or attach a model later.

Destination

Open the new model as an independent design

[✓ Create a new symbol with the specified model attached]

Attach the new model to the selected device symbol



Model Info

Select the desired model type

Structural Circuit
[✓ VHDL]

Create a VHDL language file which can be used to describe the function of this device.

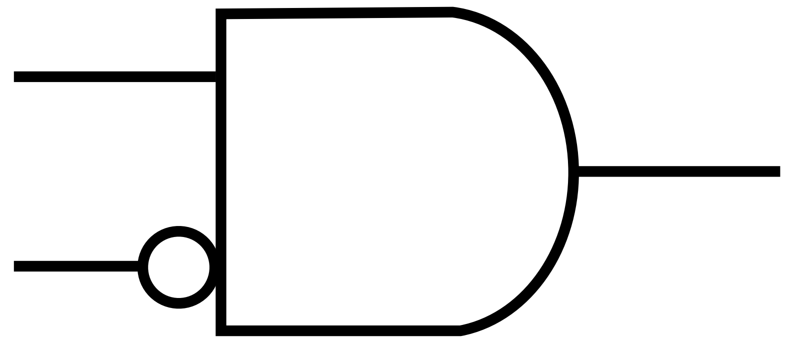
Enter a name for the new model: AND1INV

2. **Source:** `Create…`, **Destination:** `Open the…` . Click `Next` . Select `VHDL`, Enter name `AND1INV`.

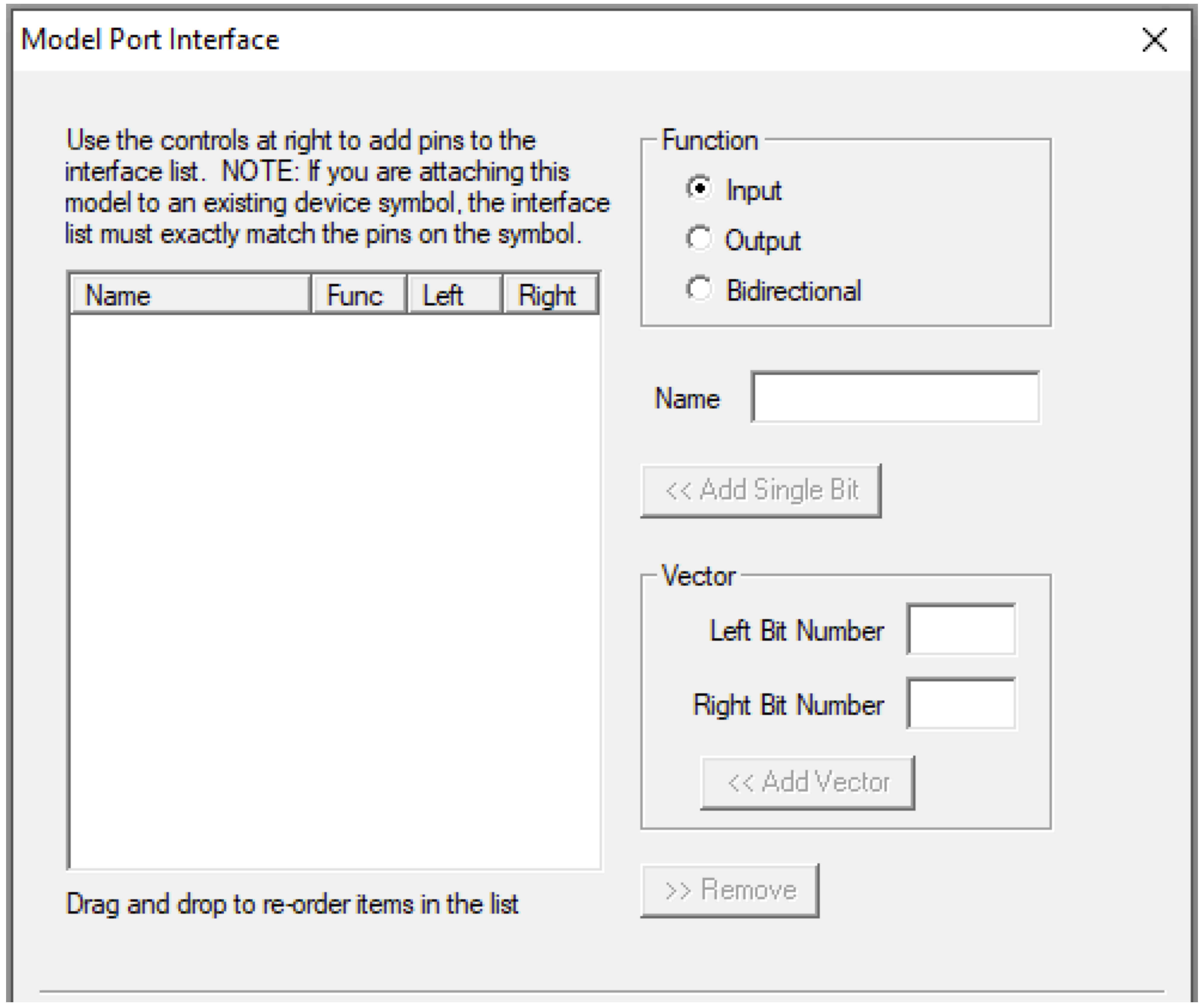
Tutorial

P2
VHDL

Creating a AND1INV model



- This is where you define all inputs and outputs
  - Input: POS
  - Input: NEG
  - Output: Out1



Use the controls at right to add pins to the interface list. NOTE: If you are attaching this model to an existing device symbol, the interface list must exactly match the pins on the symbol.

Function: [✓ Input] Output Bidirectional

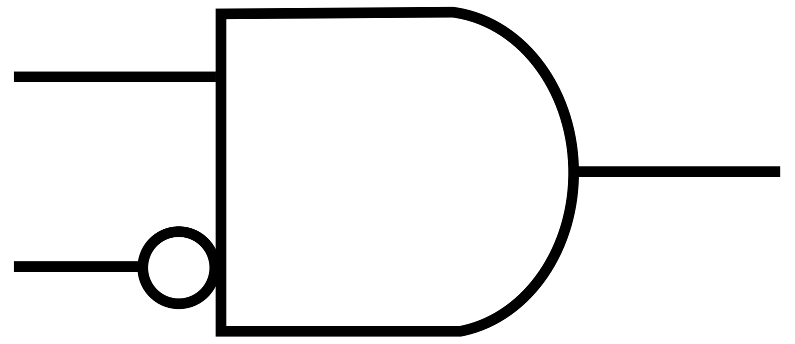
Drag and drop to re-order items in the list

3. **Type the `Name` and select the `Function` accordingly, then press `Add Single Bit`, click `Finish` to create the model file.**

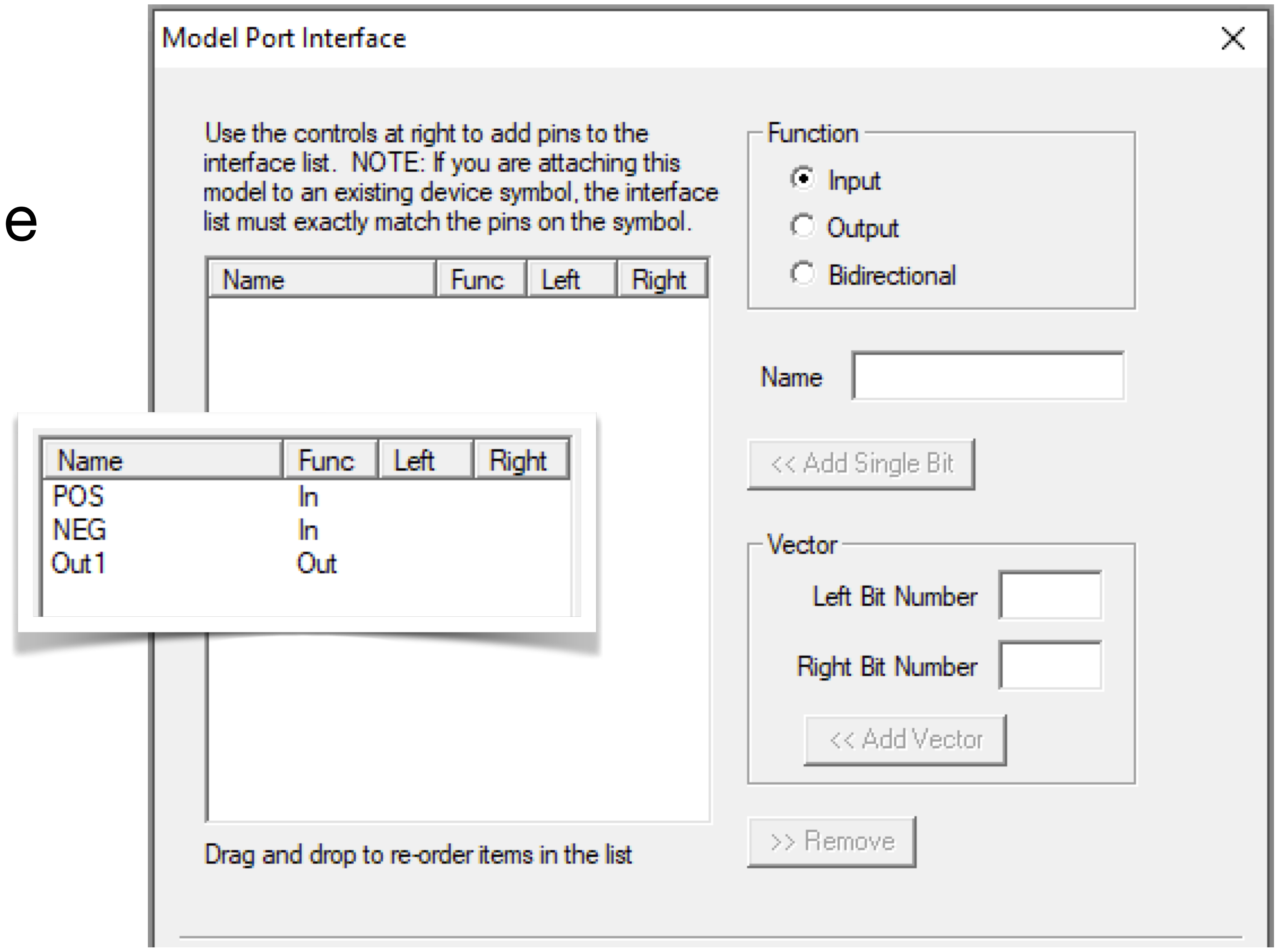
Tutorial

P2
VHDL

Creating a AND1INV model



- This is where you define all inputs and outputs
  - Input: POS
  - Input: NEG
  - Output: Out1



Model Port Interface

Use the controls at right to add pins to the interface list. NOTE: If you are attaching this model to an existing device symbol, the interface list must exactly match the pins on the symbol.

Name	Func	Left	Right
POS	In		
NEG	In		
Out1	Out		

Drag and drop to re-order items in the list

Function: [✓ Input] Output Bidirectional

Name

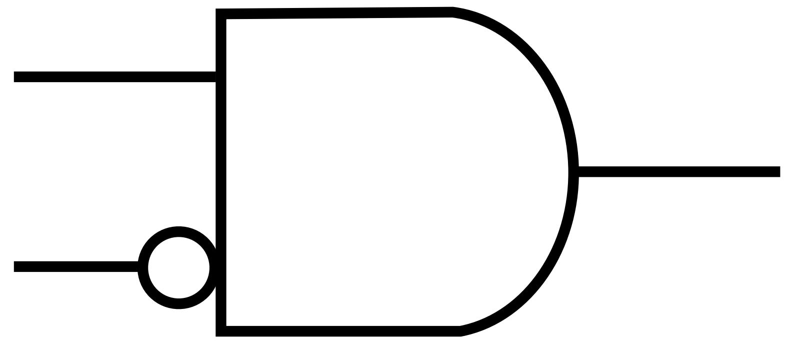
Vector
Left Bit Number
Right Bit Number

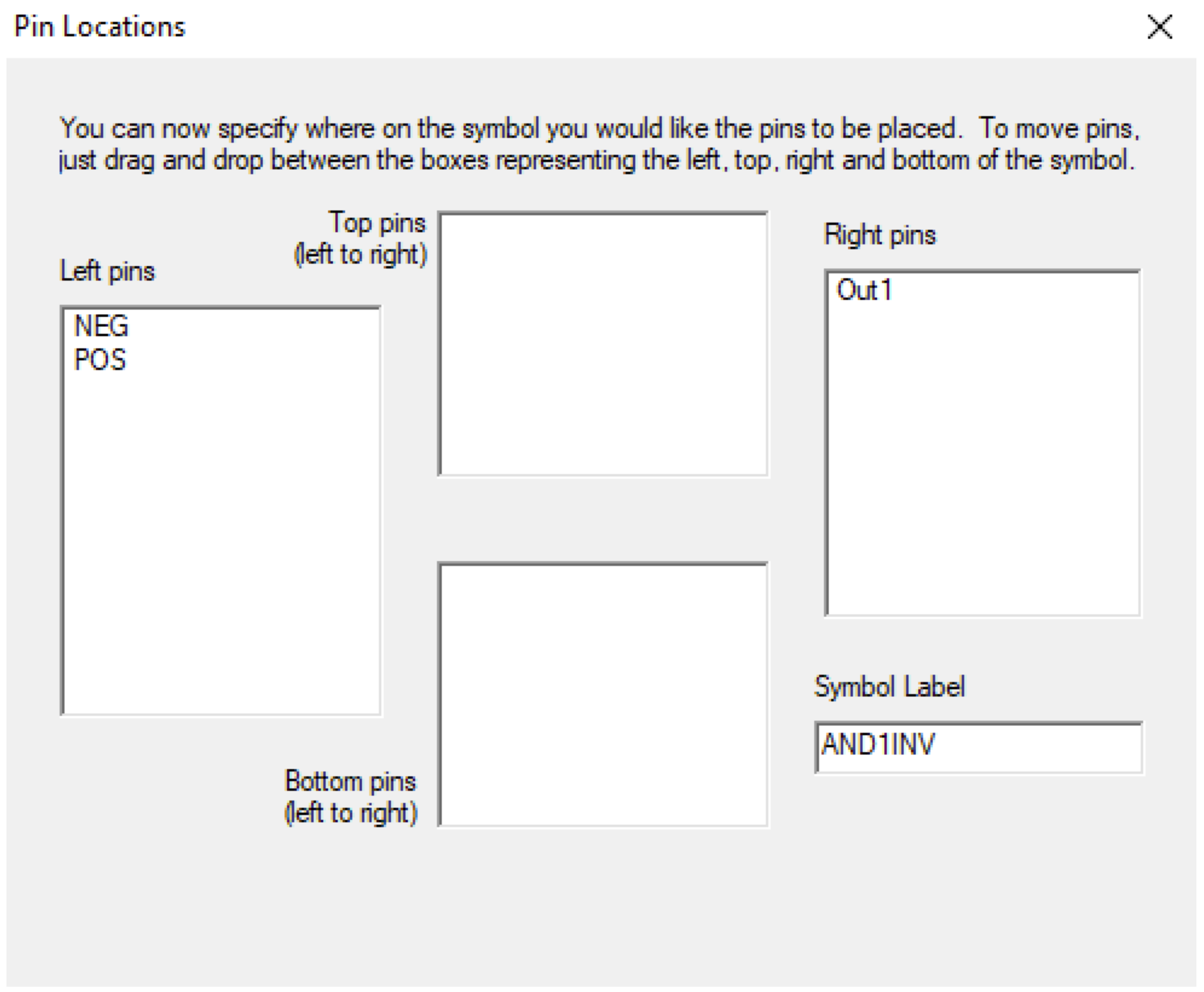
3. Type the `Name` and select the `Function` accordingly, then press `Add Single Bit`, click `Finish` to create the model file.

Tutorial

P2
VHDL

Creating a AND1INV model





Pin Locations

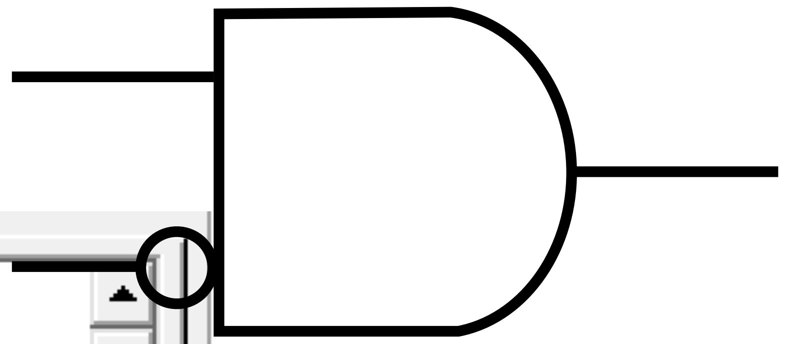
You can now specify where on the symbol you would like the pins to be placed. To move pins, just drag and drop between the boxes representing the left, top, right and bottom of the symbol.

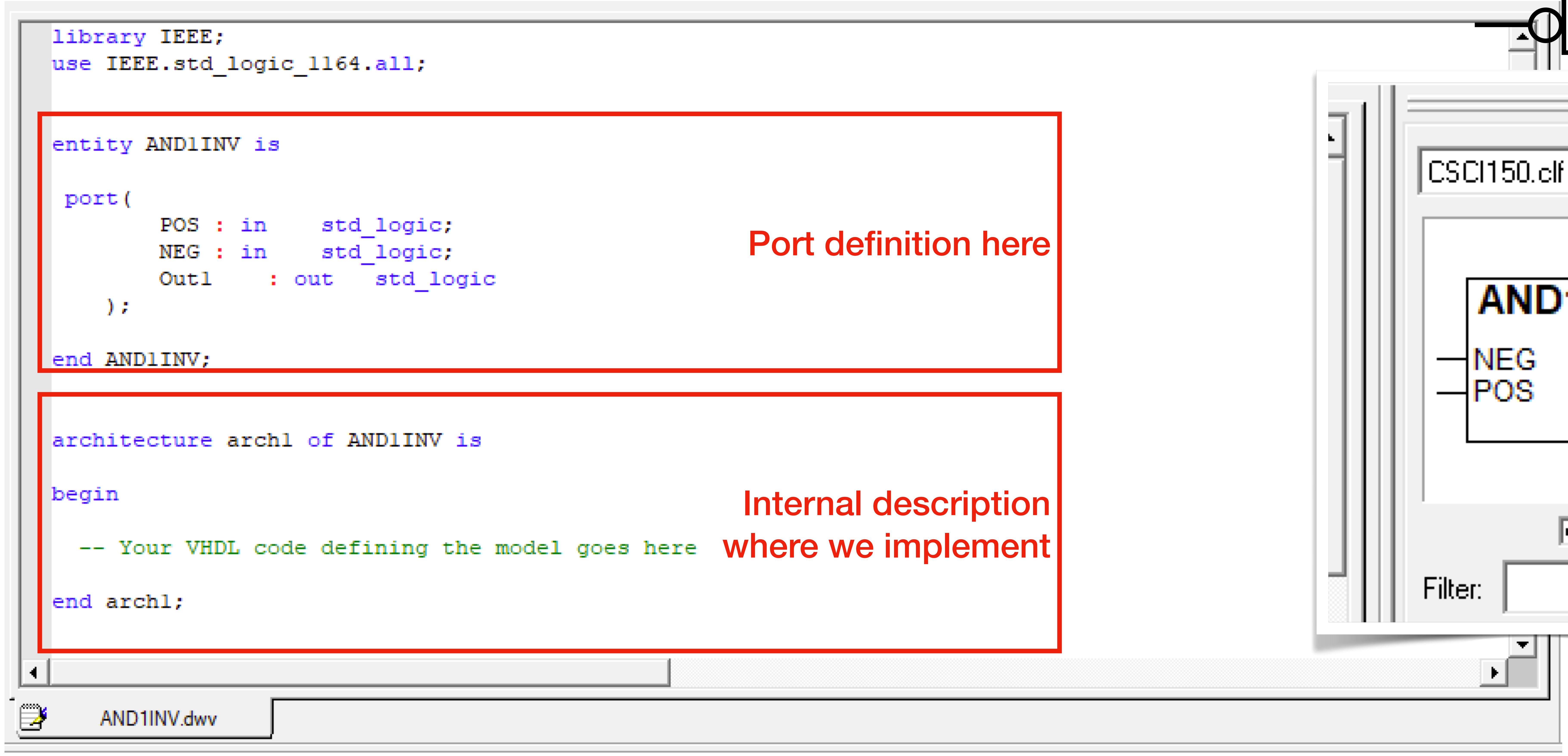
Tutorial

4. The programme will ask you for `Pin Location` assignment. Just click `Next`.

P2
VHDL

Creating a AND1INV model

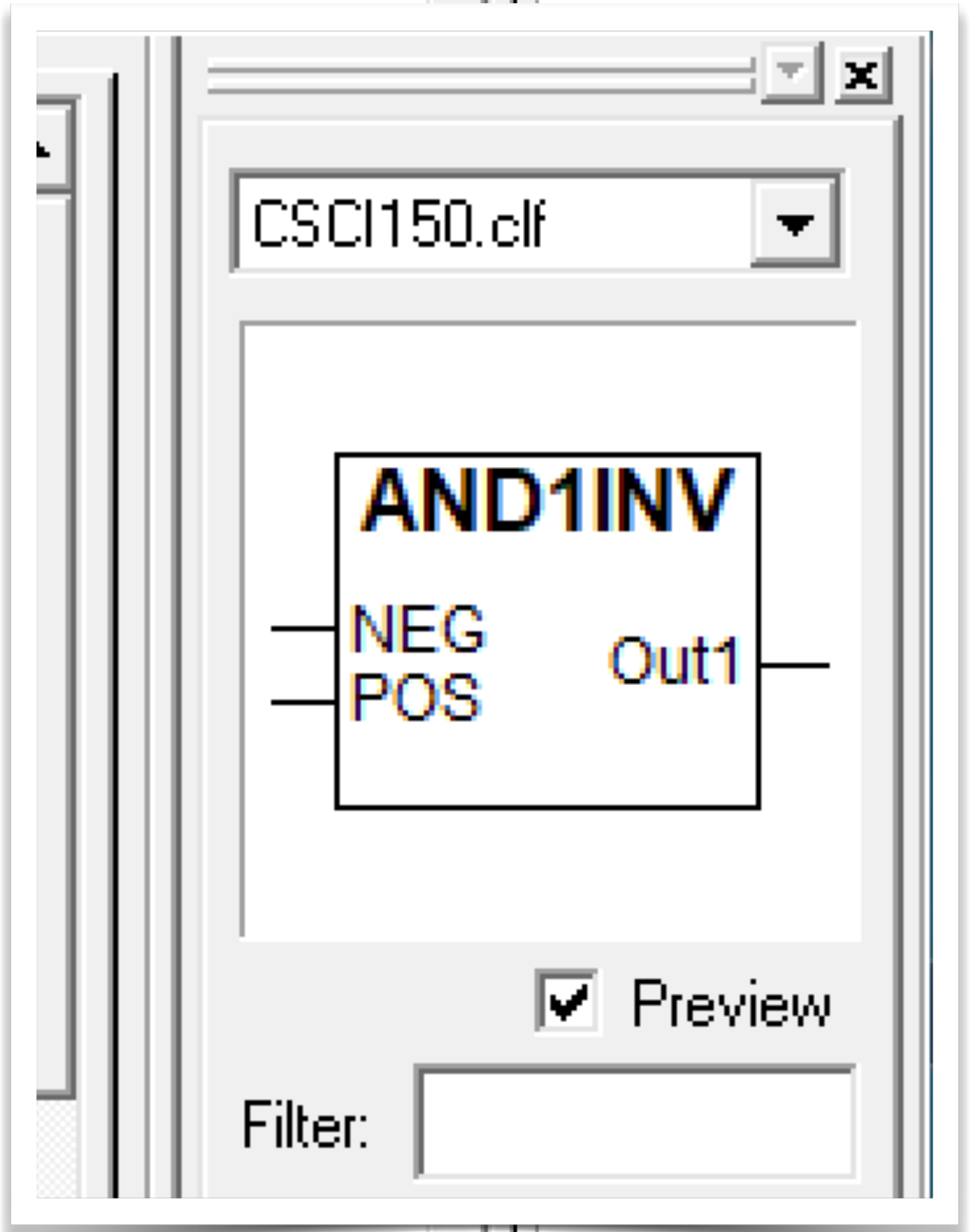




```
library IEEE;
use IEEE.std_logic_1164.all;

entity AND1INV is

 port(
        POS : in    std_logic;
        NEG : in    std_logic;
        Out1     : out    std_logic
    );

end AND1INV;

architecture arch1 of AND1INV is

begin

  -- Your VHDL code defining the model goes here

end arch1;
```

Port definition here

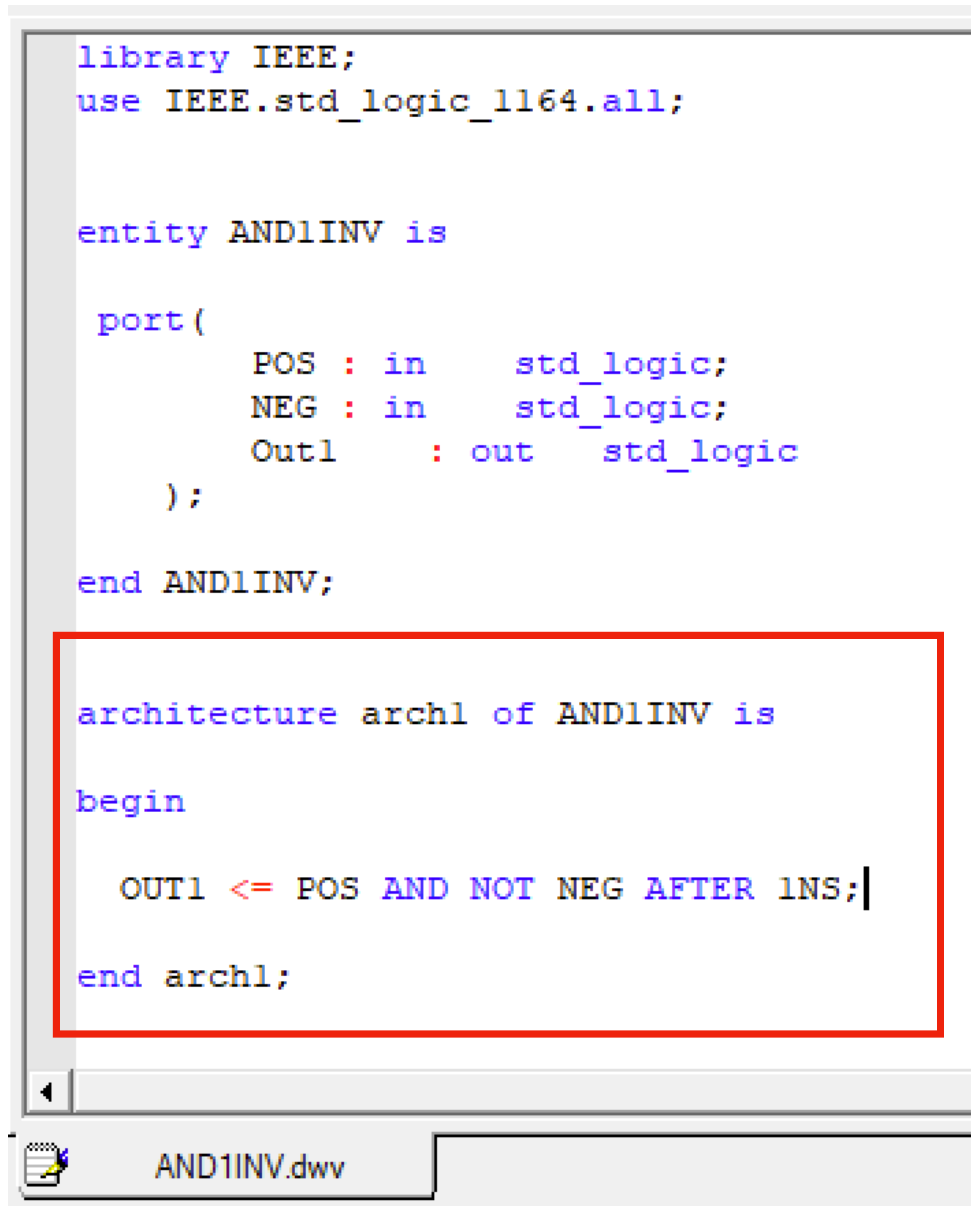
Internal description where we implement



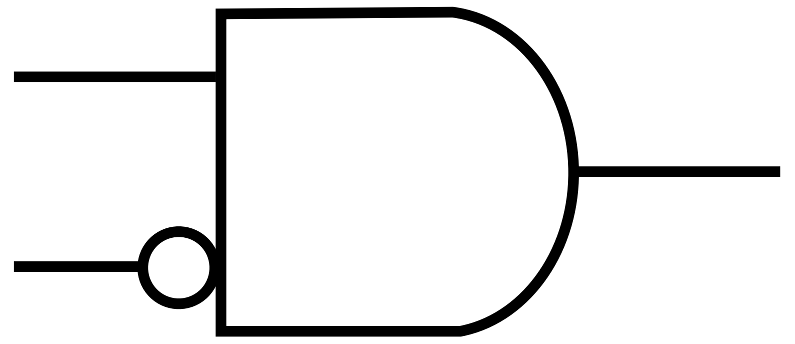
Tutorial

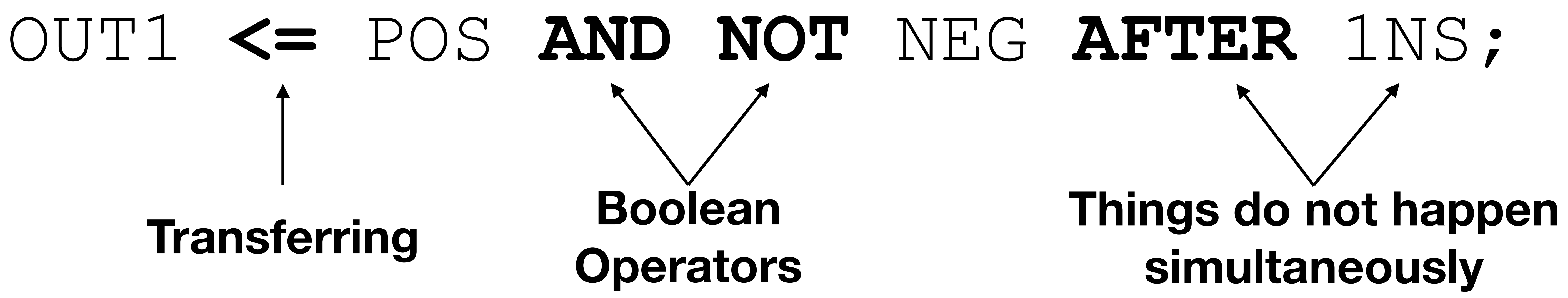
P2
VHDL

Creating a AND1INV model



```
library IEEE;
use IEEE.std_logic_1164.all;


entity AND1INV is

 port(
        POS : in    std_logic;
        NEG : in    std_logic;
        Out1     : out   std_logic
    );

end AND1INV;

architecture arch1 of AND1INV is

begin

  OUT1 <= POS AND NOT NEG AFTER 1NS;

end arch1;
```

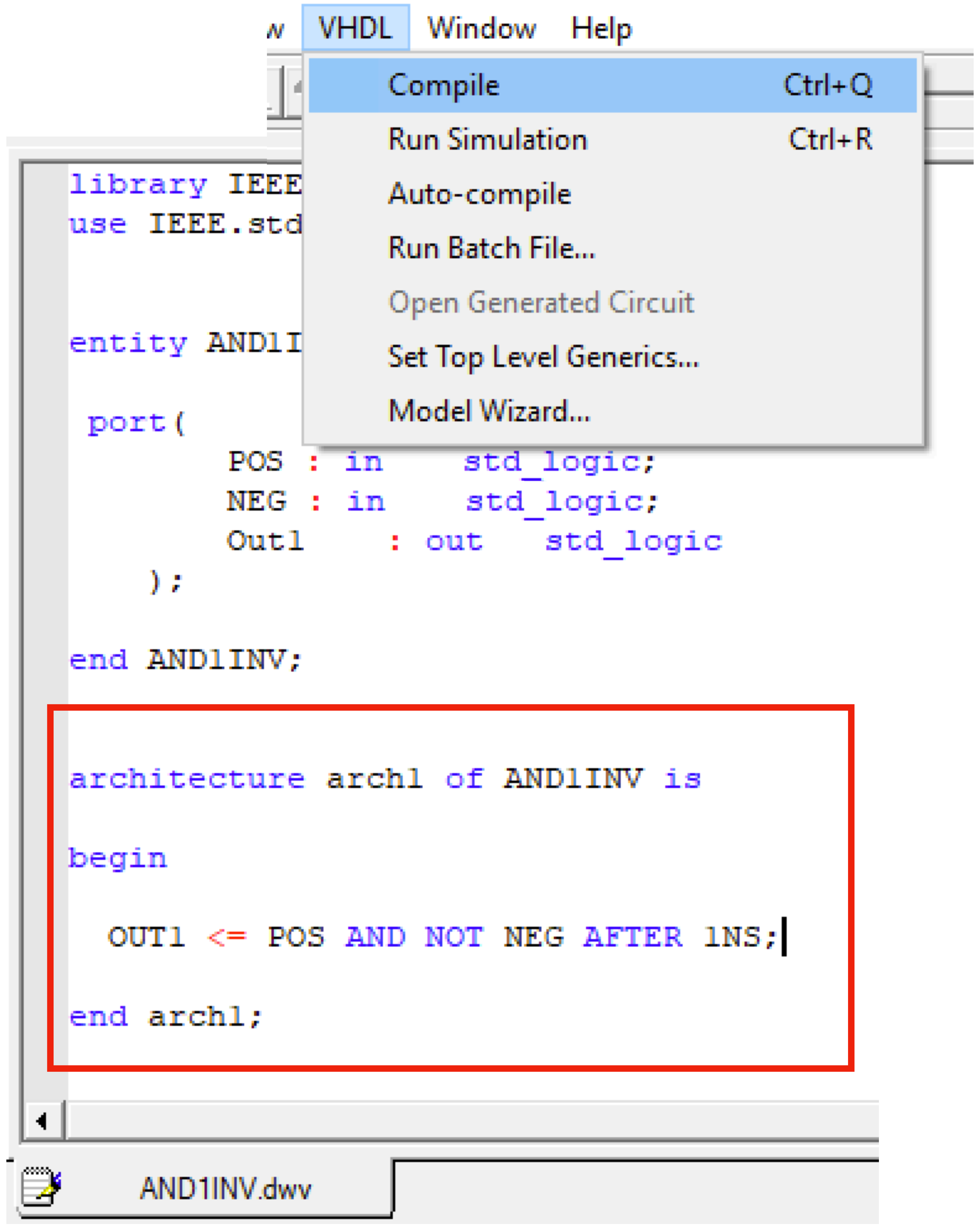



OUT1 <= POS AND NOT NEG AFTER 1NS;
Transferring
Boolean Operators
Things do not happen simultaneously

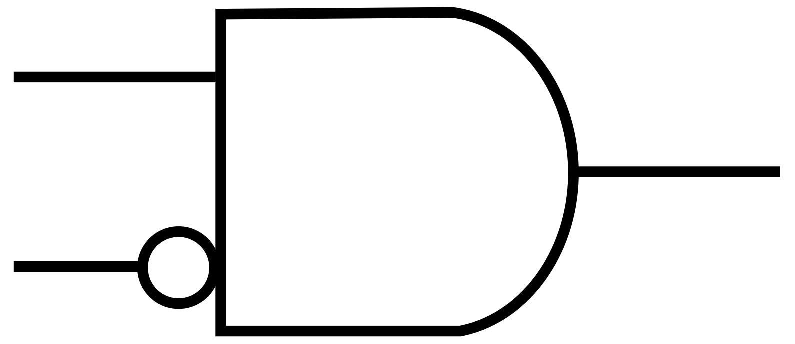
Tutorial

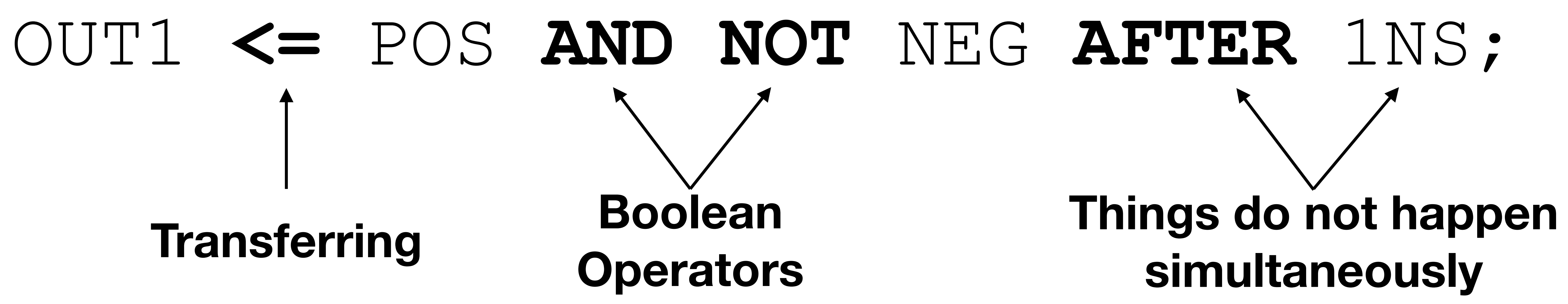
5. Type: `OUT1 <= POS AND NOT NEG AFTER 1NS;` This is the Boolean description of the `OUT1` port

Creating a AND1INV model



```
library IEEE
use IEEE.std

entity AND1I

 port(
        POS : in     std_logic;
        NEG : in     std_logic;
        Out1     : out    std_logic
    );

end AND1INV;

architecture arch1 of AND1INV is

begin

  OUT1 <= POS AND NOT NEG AFTER 1NS;

end arch1;
```

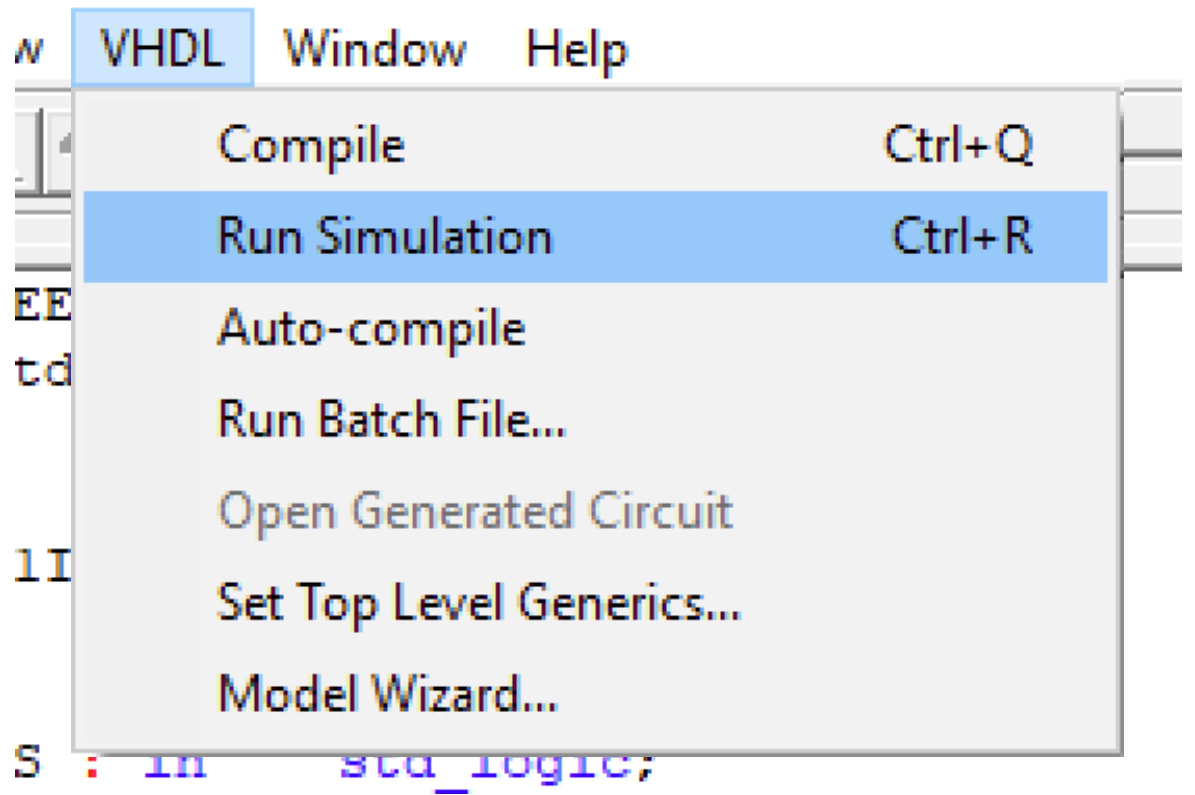


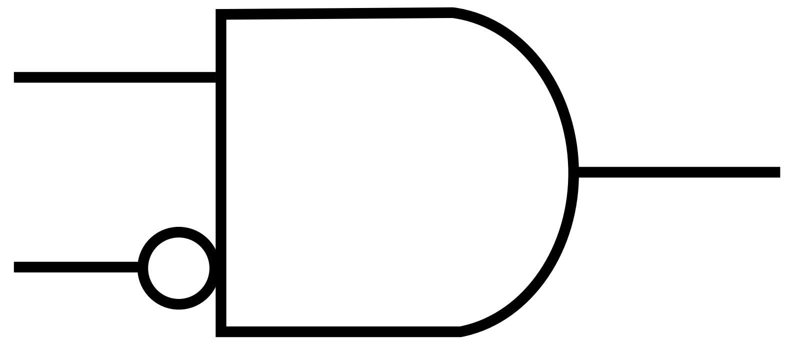

OUT1 <= POS AND NOT NEG AFTER 1NS;
Transferring
Boolean Operators
Things do not happen simultaneously

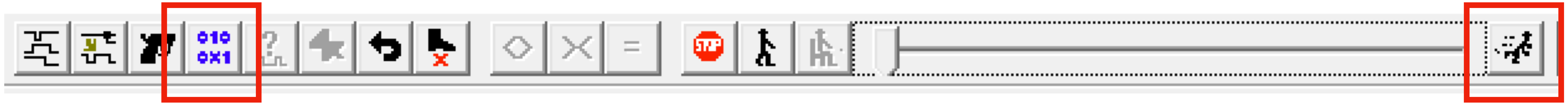
6. `VHDL` -> `Compile`. A message should say `Compile Completed - 0 errors`

P2
VHDL

Run Simulation





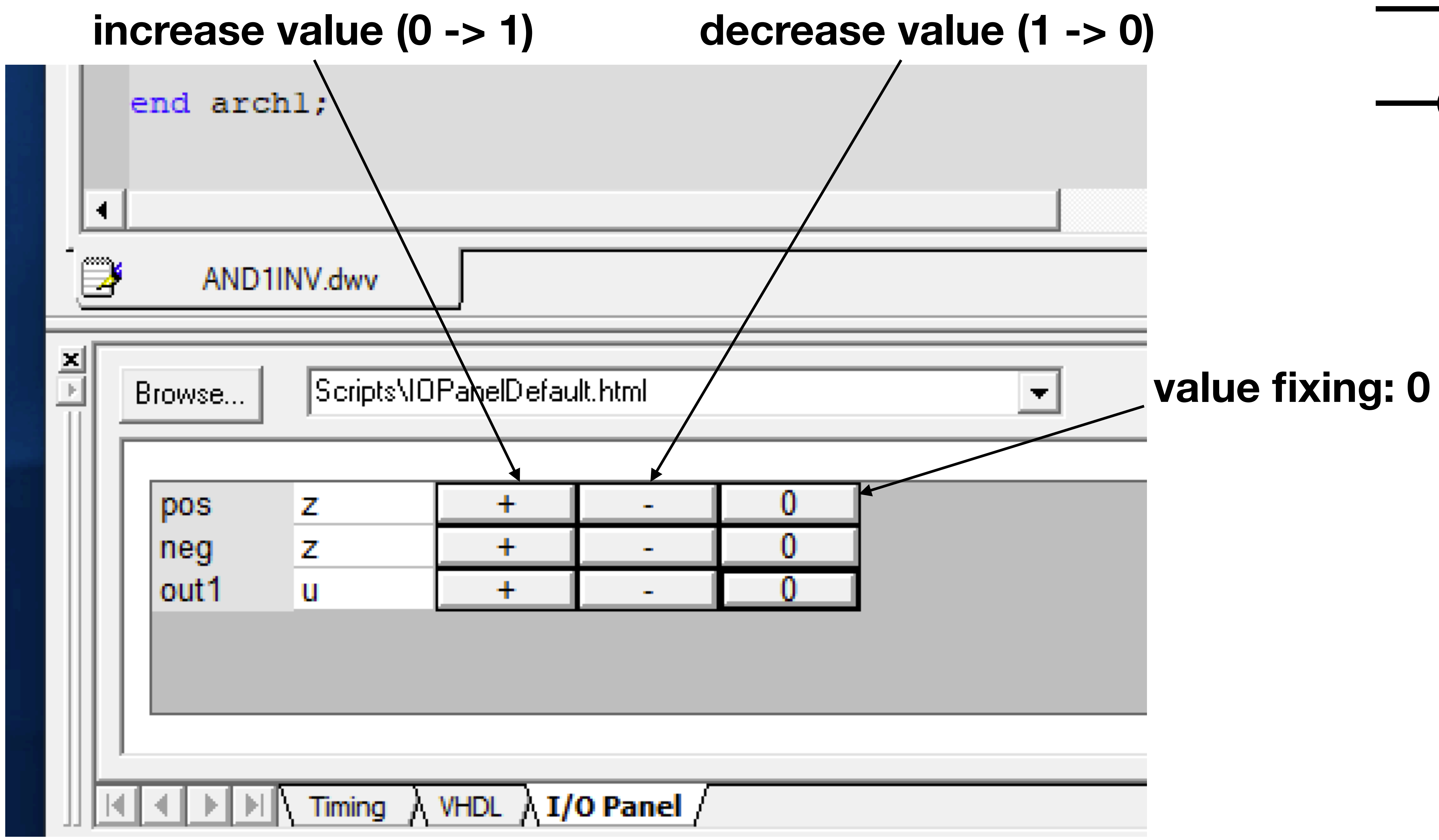


Tutorial

6. `VHDL` -> `Run Simulation`. The text should turn grey (not editable). Click `Run` Button to start simulator, click `I/O panel Page` button

P2
VHDL

Run Simulation

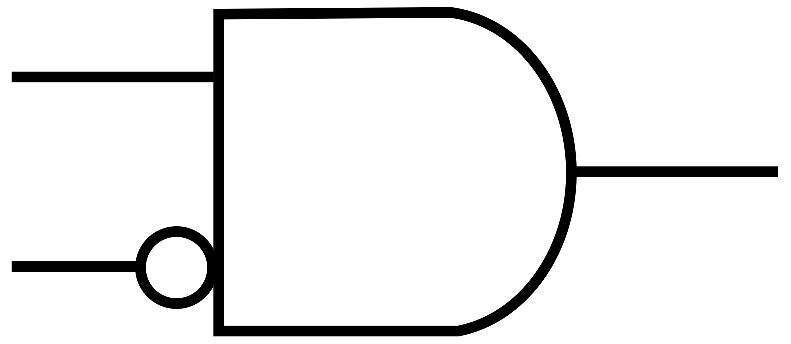


```
end arch1;
```

AND1INV.dwv

Scripts\IOPanelDefault.html

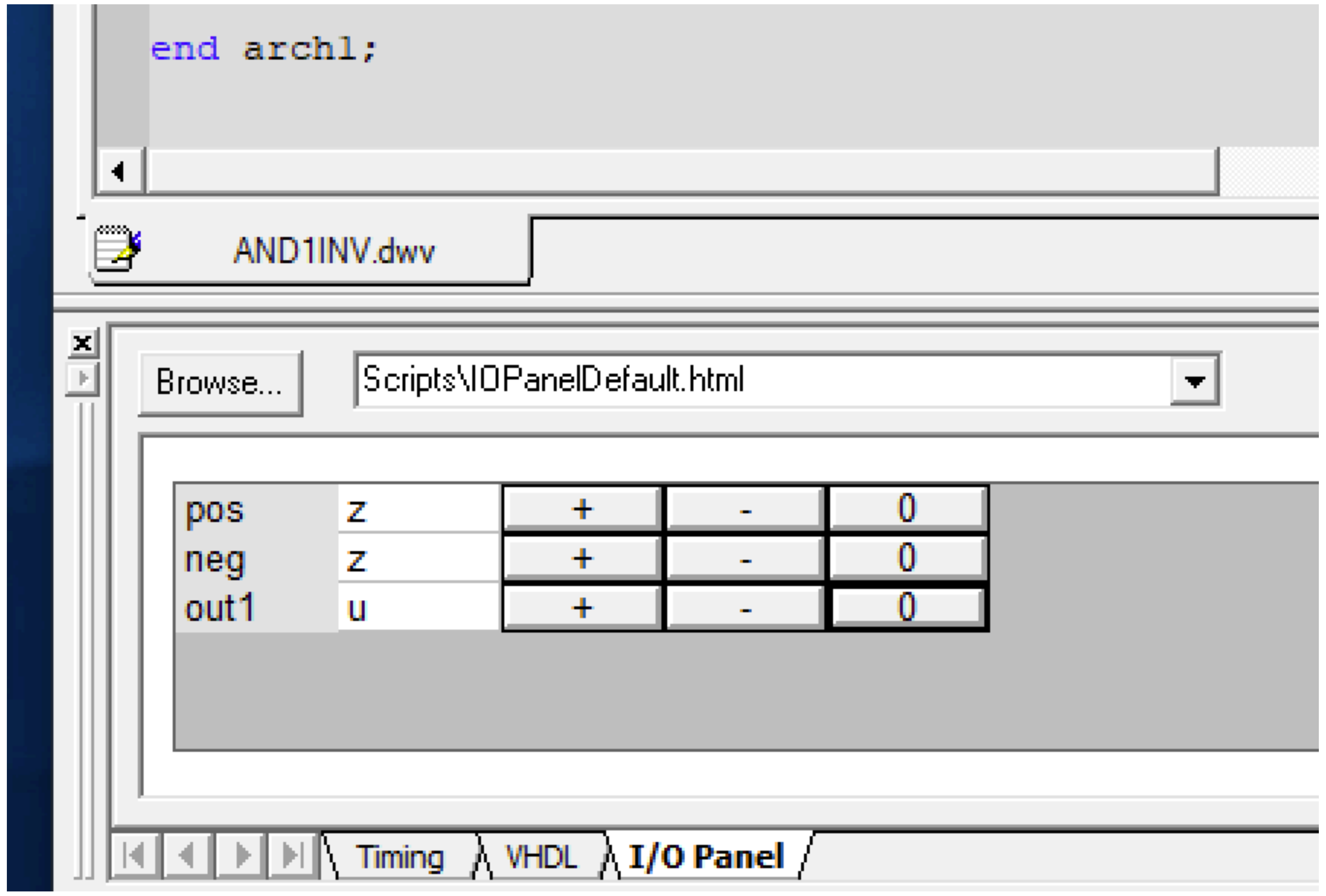
pos	z	+	-	0
neg	z	+	-	0
out1	u	+	-	0



Tutorial

7. Play with the buttons in the I/O panel

Run Simulation

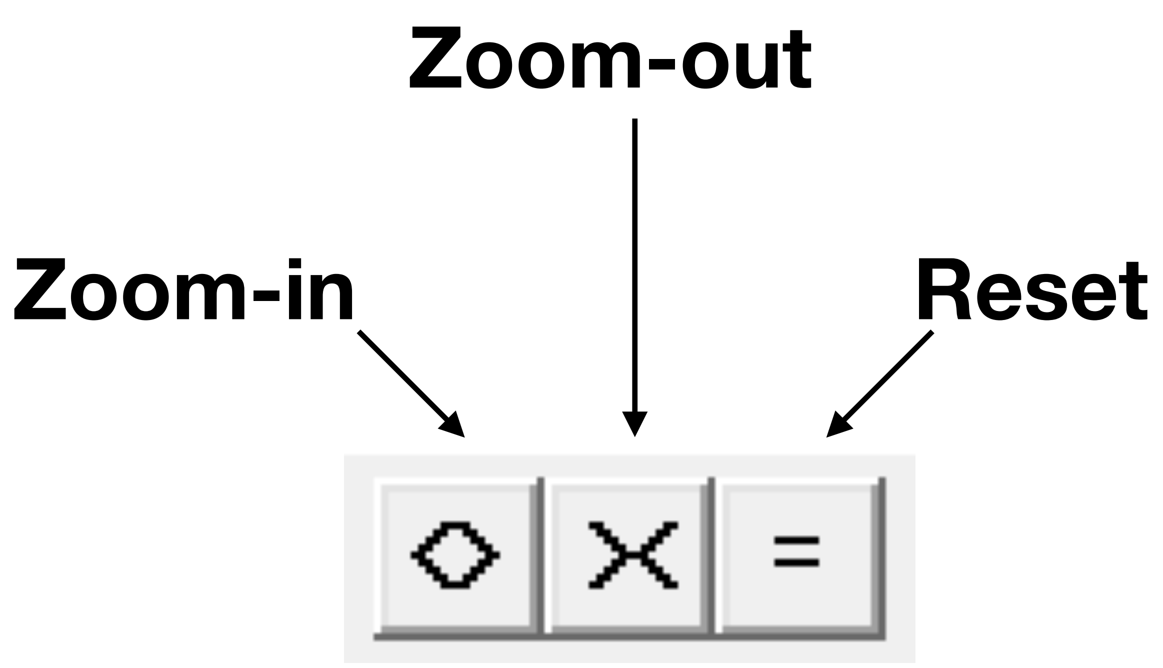


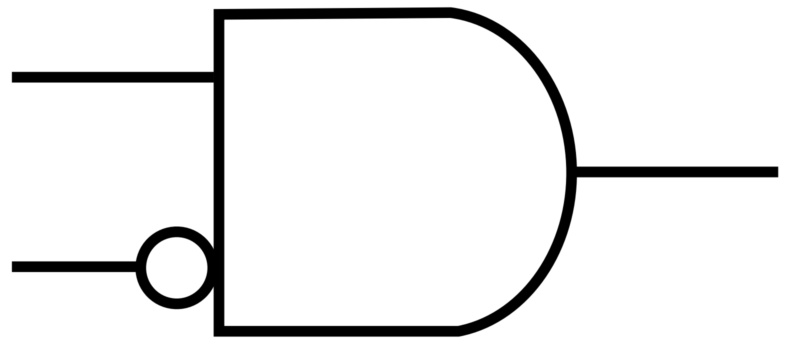
```
end archl;
```

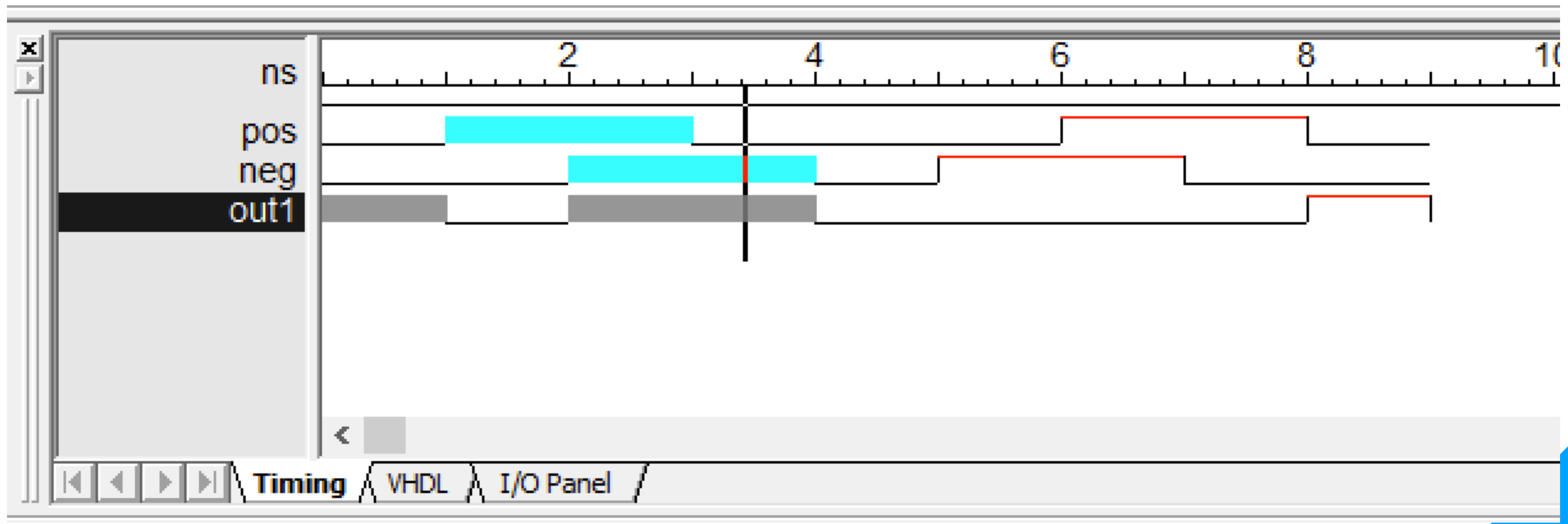
AND1INV.dwv

Browse... Scripts\IOPanelDefault.html

pos	z	+	-	0
neg	z	+	-	0
out1	u	+	-	0



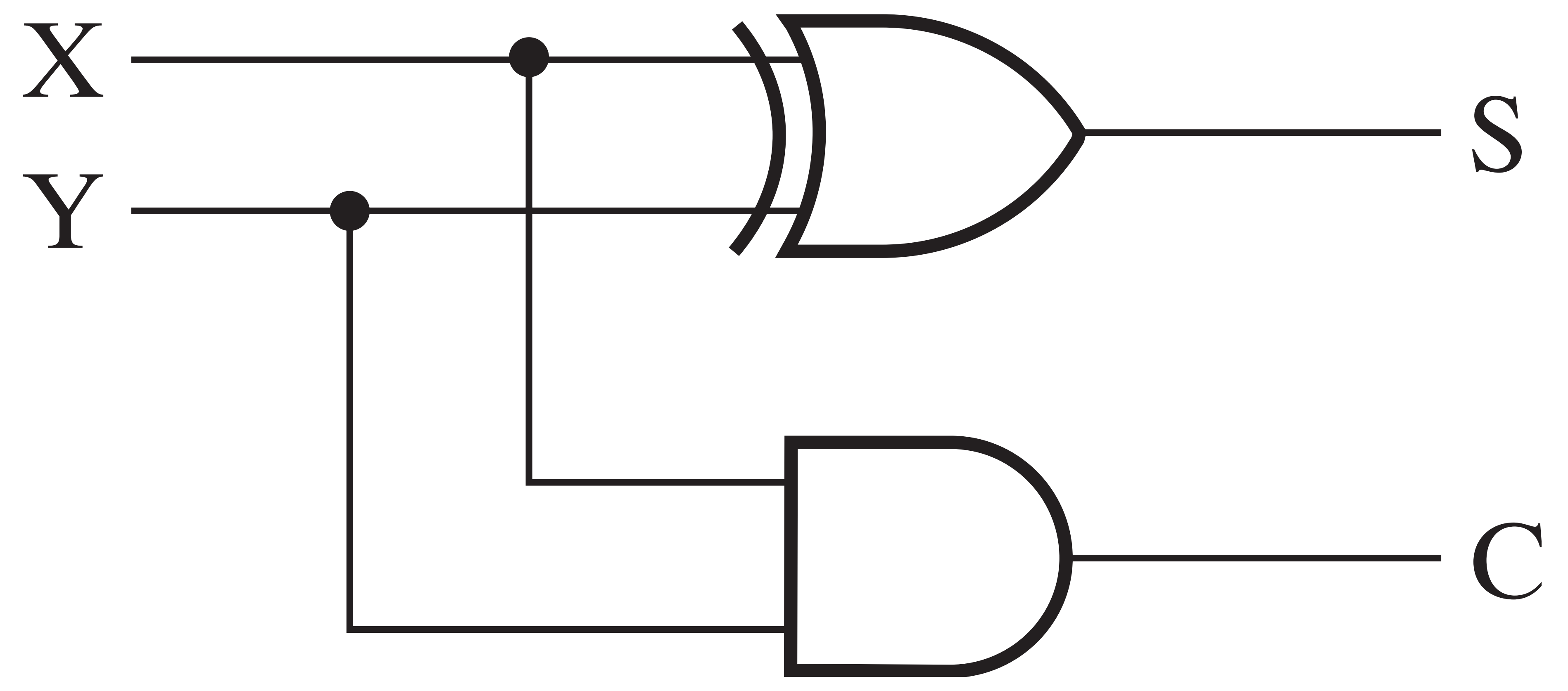




8. Changes are reflected in the Timing Diagram. Use Zoom panel to Zoom In and Out

Exe1: 1-bit Half Adder

- Create a new component in VHDL called `HalfAdder1`
  - Input: X, Y
  - Output: S, C
  - Don't use `AFTER`

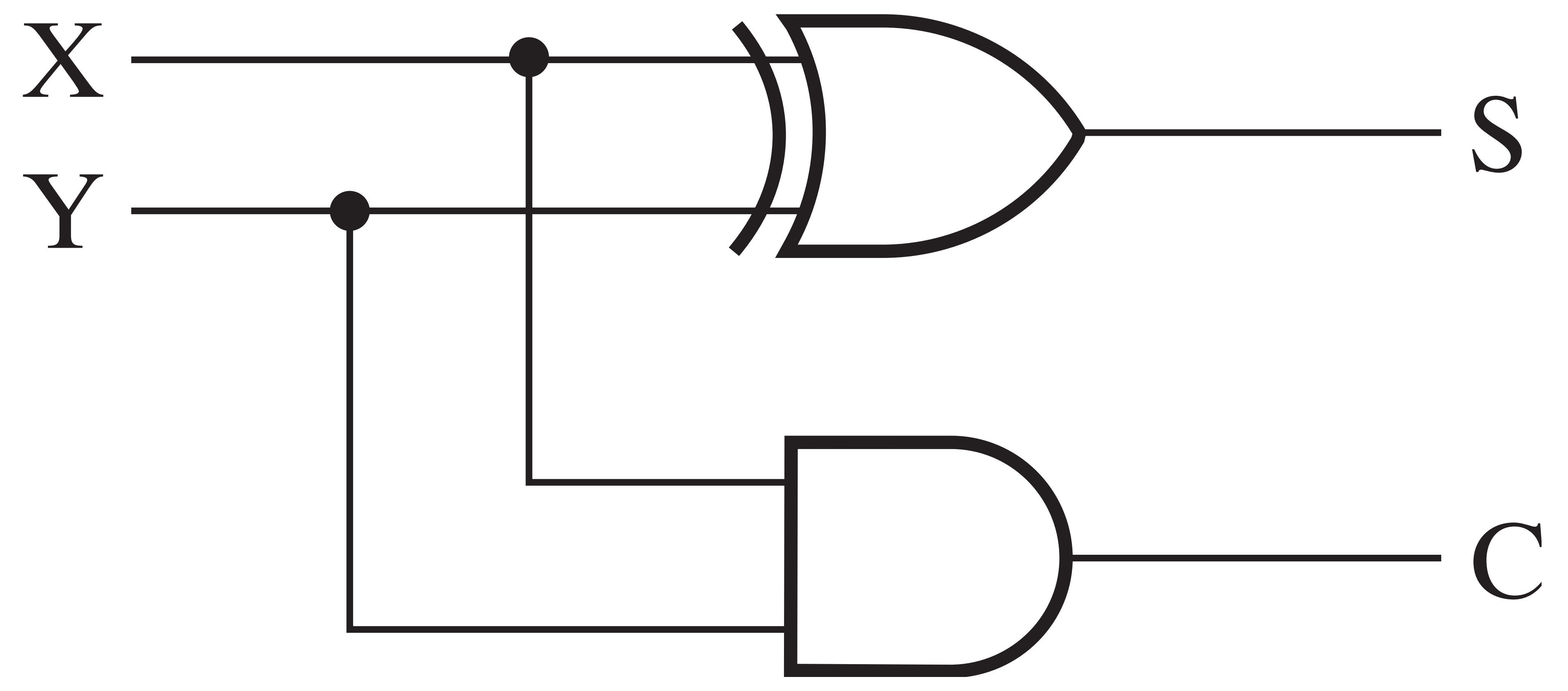


Practice

P2
VHDL

Exe1: 1-bit Half Adder

```
architecture arch1 of HalfAdder is

begin

    S <= X XOR Y;

    C <= X AND Y;

end arch1;
```



Practice