21.06.20 13:45

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

Jetic Gū

2020 Summer Semester (S2)

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

P0
Binary Adder

1-bit Half Adder

- Half adder
  input X, Y
  output S, C

Carries	C
Augend	X
Addend	$+Y$
Sum	S

Input
Output

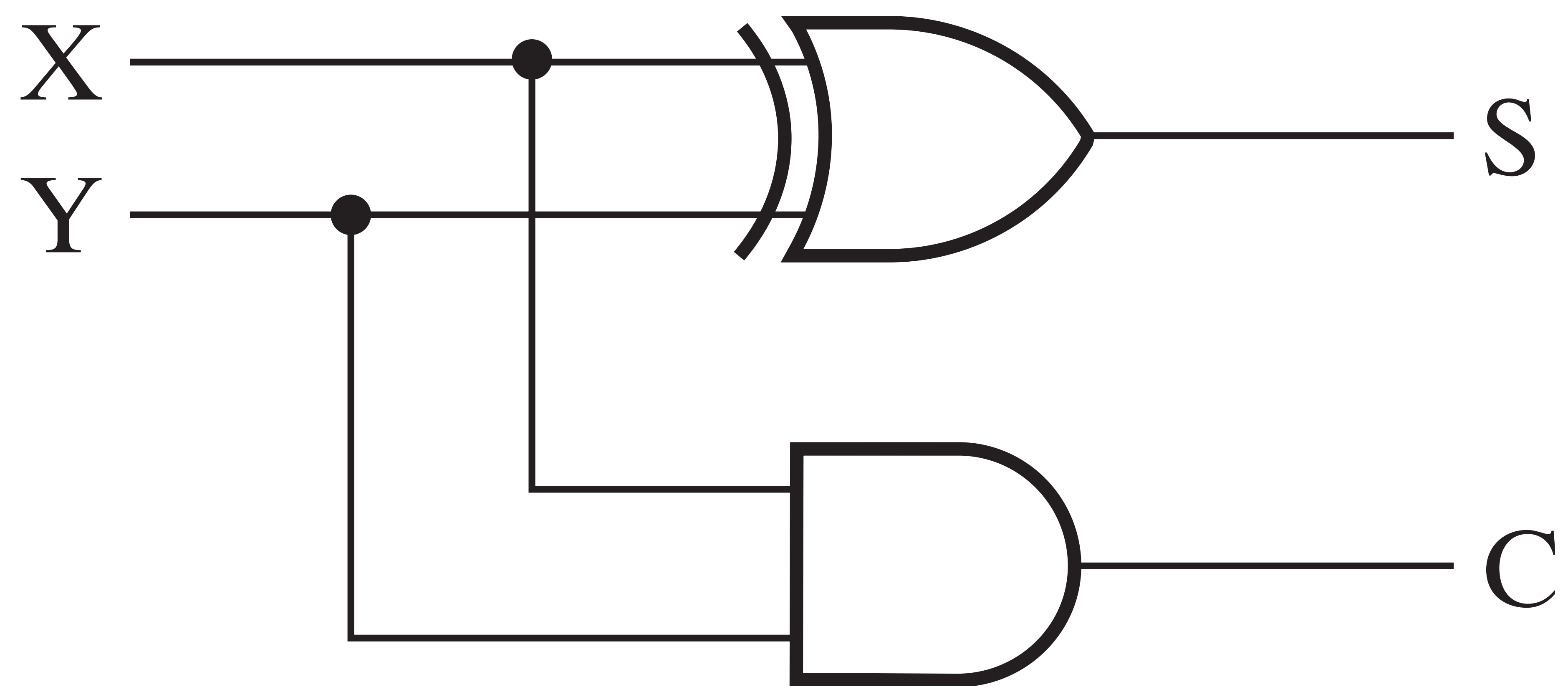
Review

P0
Binary Adder

1-bit Half Adder

Carries	C
Augend	X
Addend	$+Y$
Sum	S

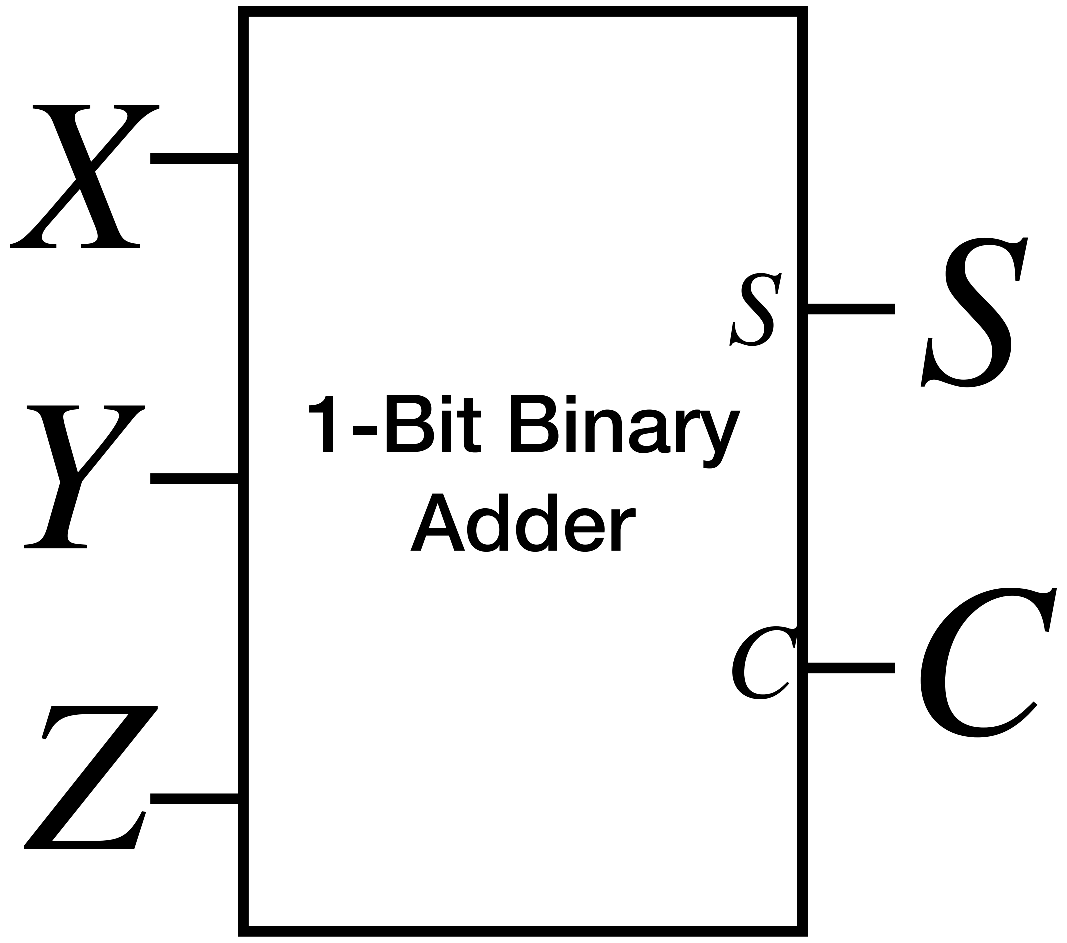
Input
Output

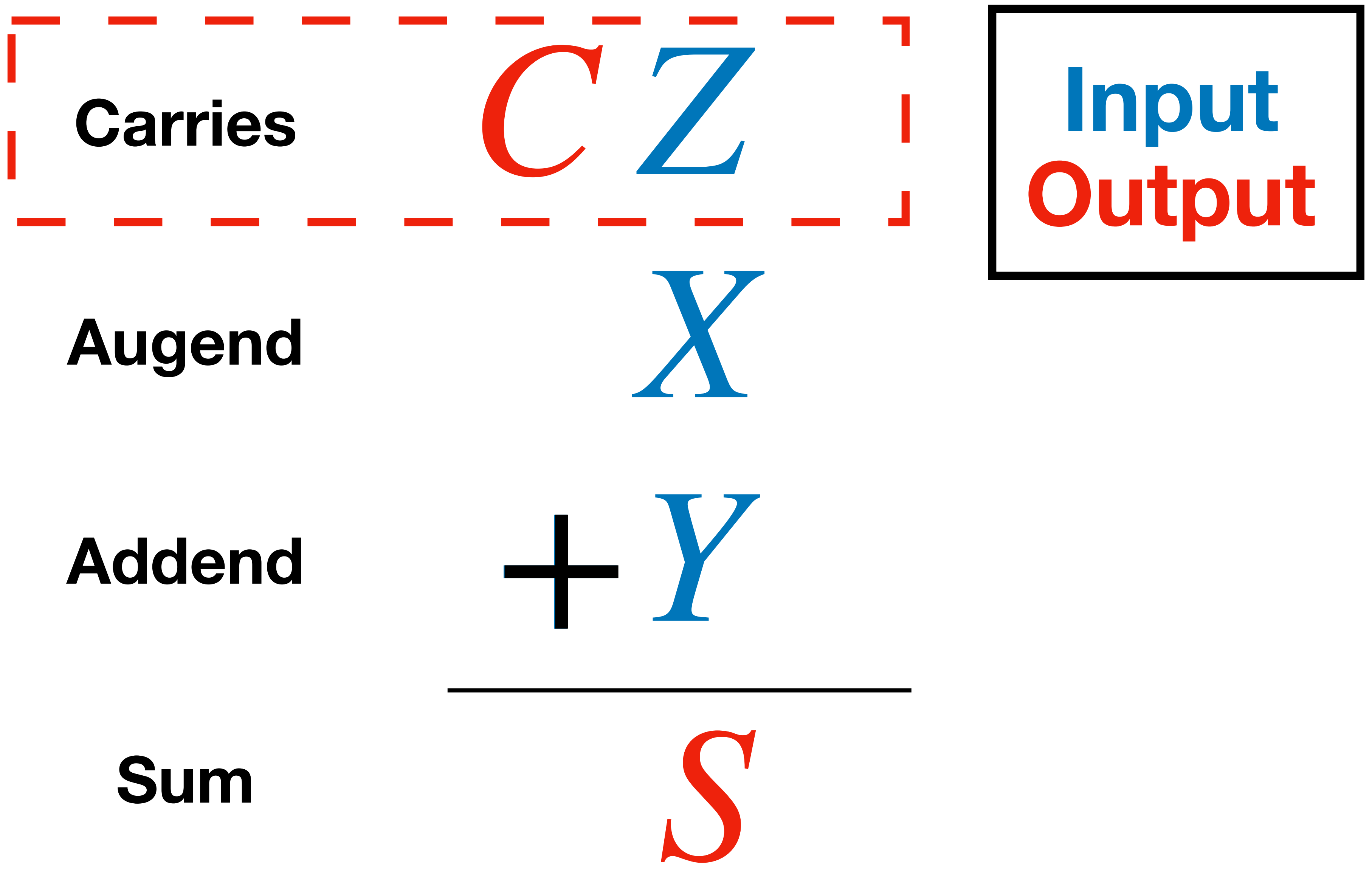


Review

1-bit Full Adder

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





Review

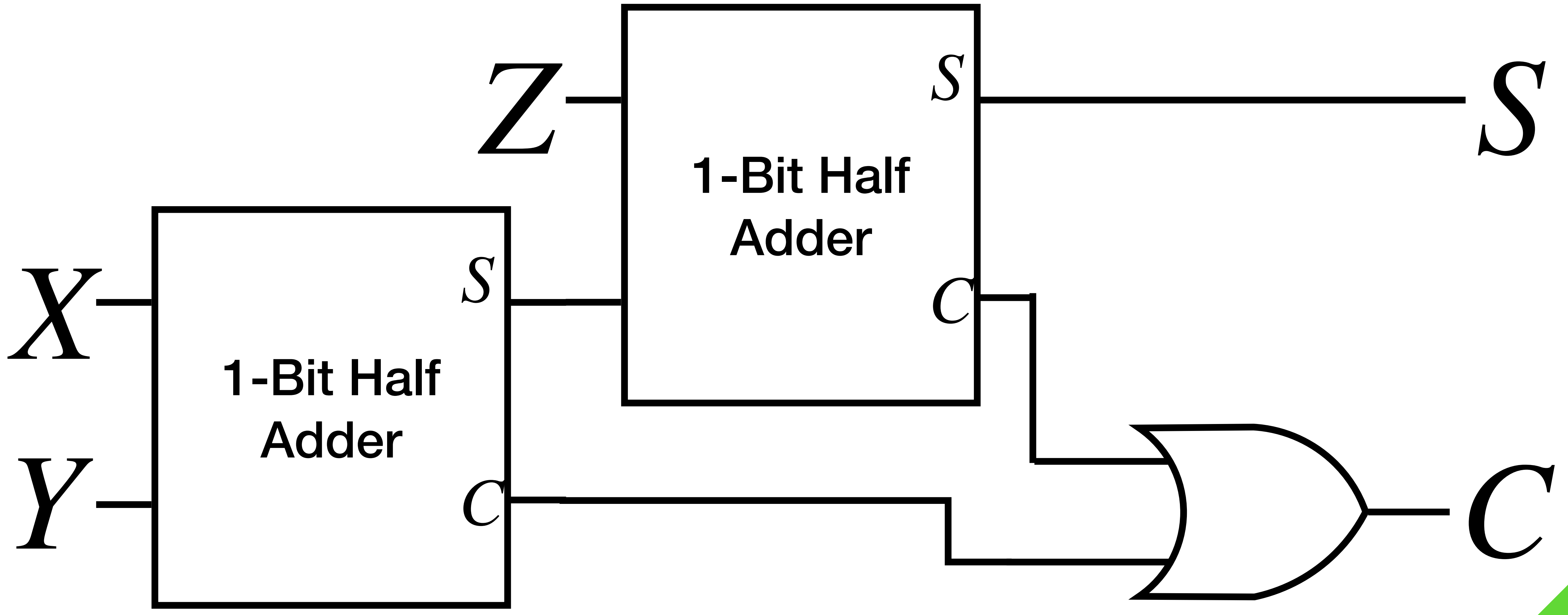
1-bit Full Adder

- 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''

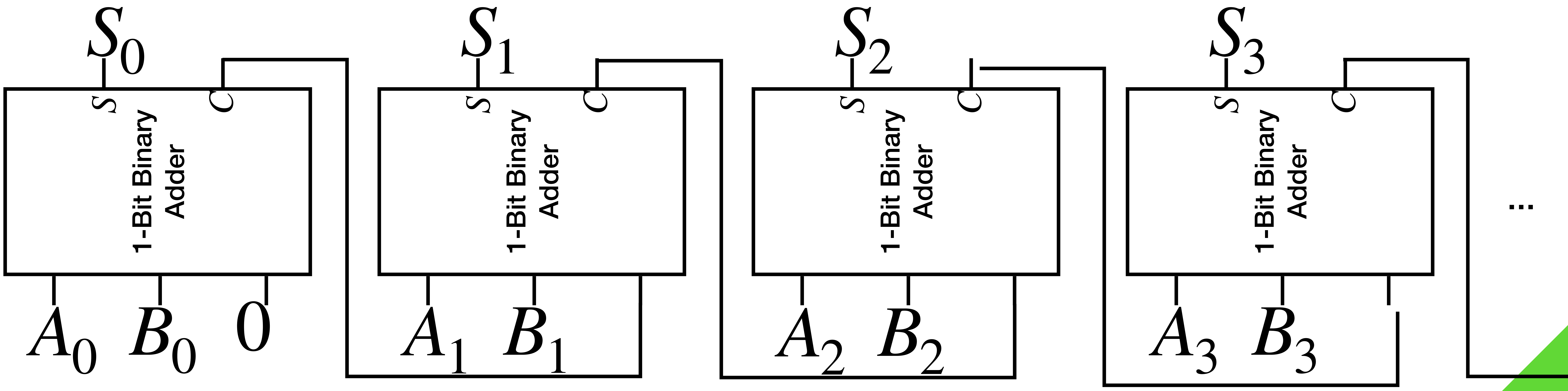
$$C = C' + C''$$



Review

n-bit Full Adder

- Ripple Carry Adder



Review

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

P1 Subtraction

Unsigned Binary Subtraction

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

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

Review

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

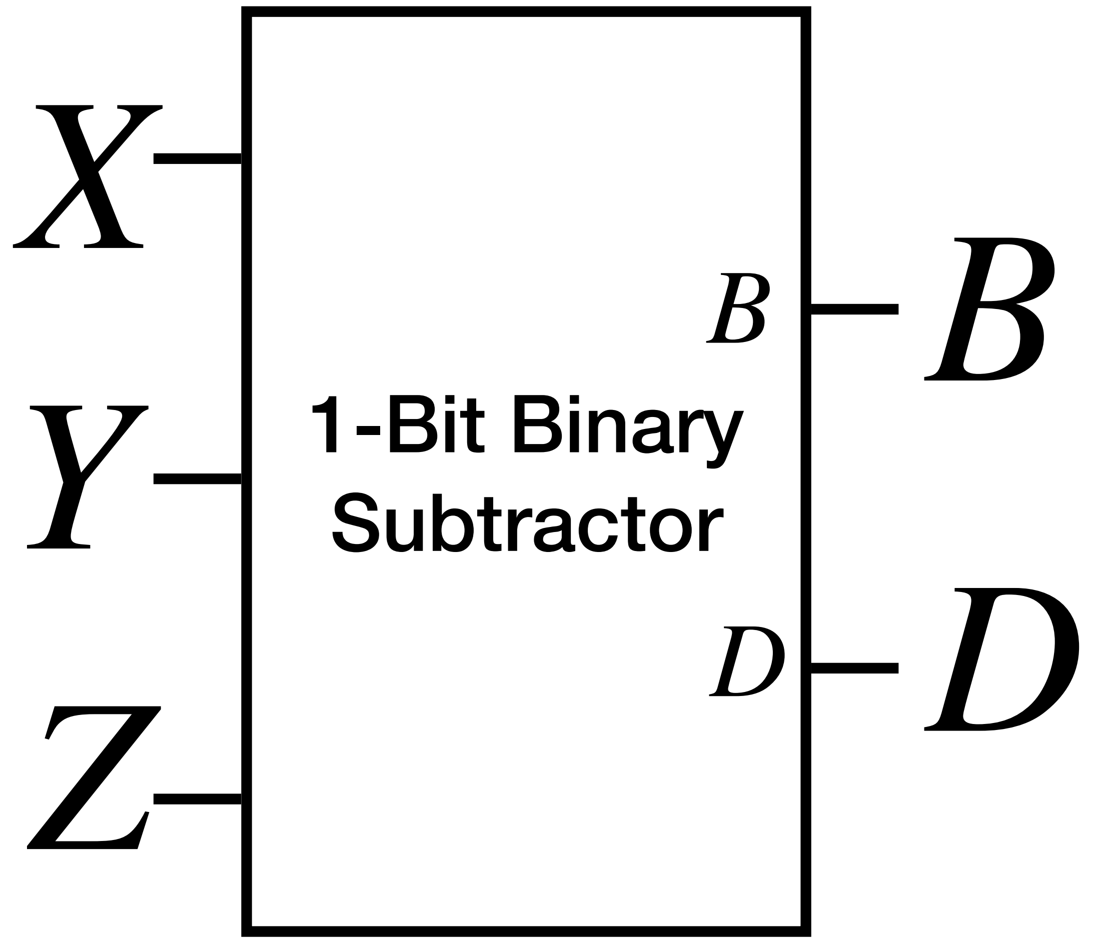
Input
Output

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
  - $B = \Sigma m(1,2,3,7)$
  - $D = \Sigma m(1,2,4,7)$



P1
Subtraction

Unsigned Binary Subtraction

Technology
- 1 bit Unsigned Subtractor

- Input: Minuend and Subtrahend
  Previous borrow

- Output: Last borrow, difference

	B Z
Borrows	000110
Minuend X	10110
Subtrahend Y	$-$ 10011
Difference D	00011

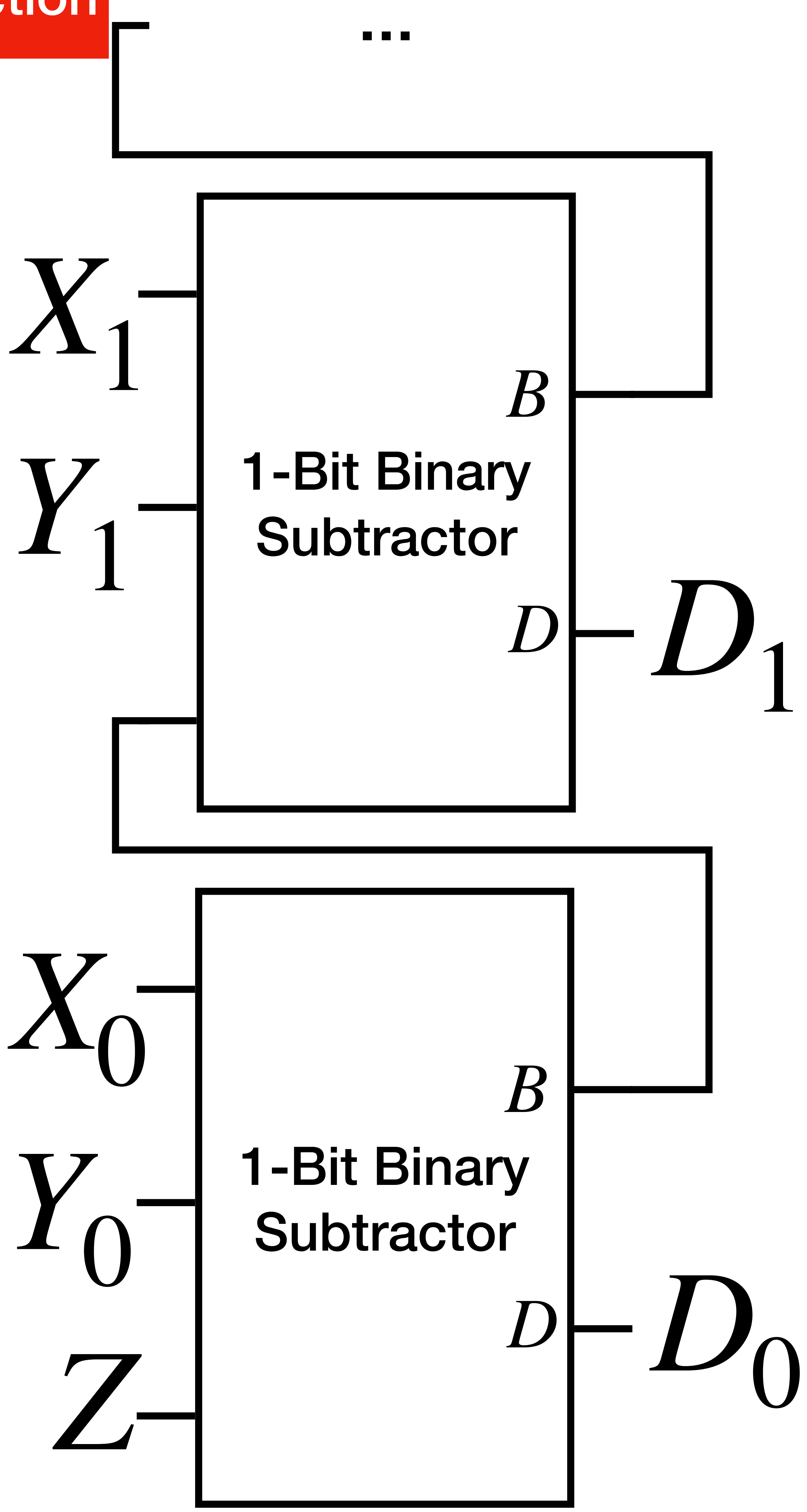
Input
Output

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

Summary

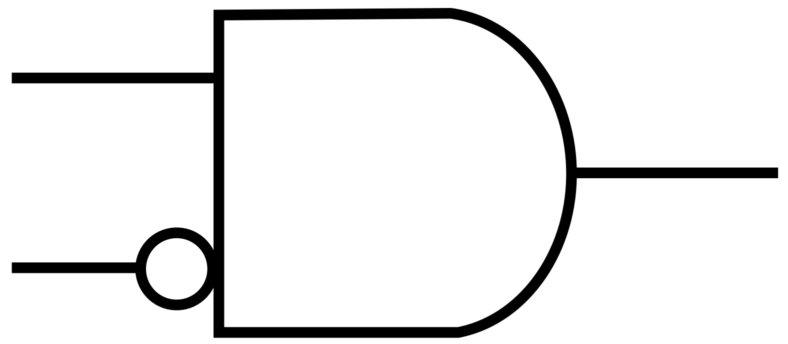
What is HDL

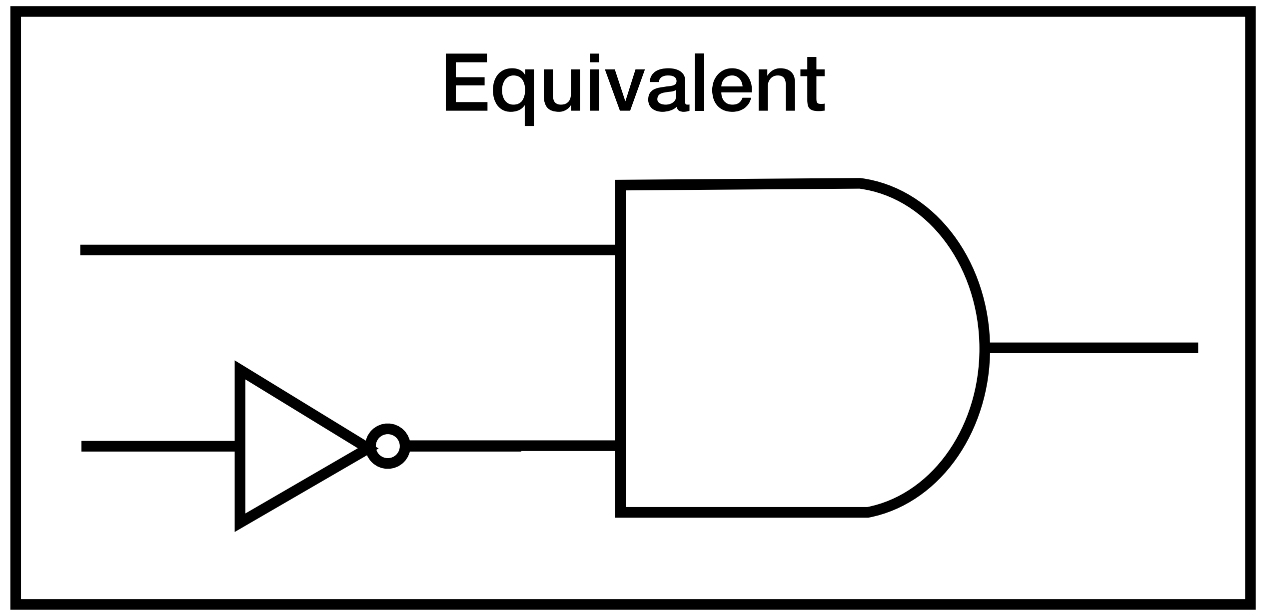
- Designing complex circuits using logic circuit diagrams is inefficient
- Hardware Description Language
  - Like programming language, describes hardware structures and behaviours
  - More efficient
  - Common languages
    - Verilog
    - VHDL

Concept

P2
VHDL

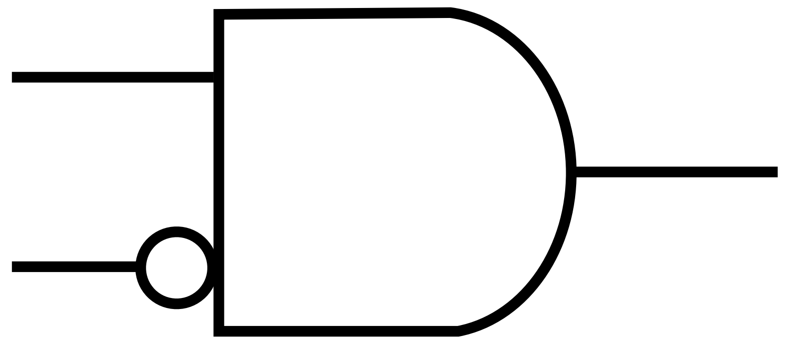
Creating a AND1INV model

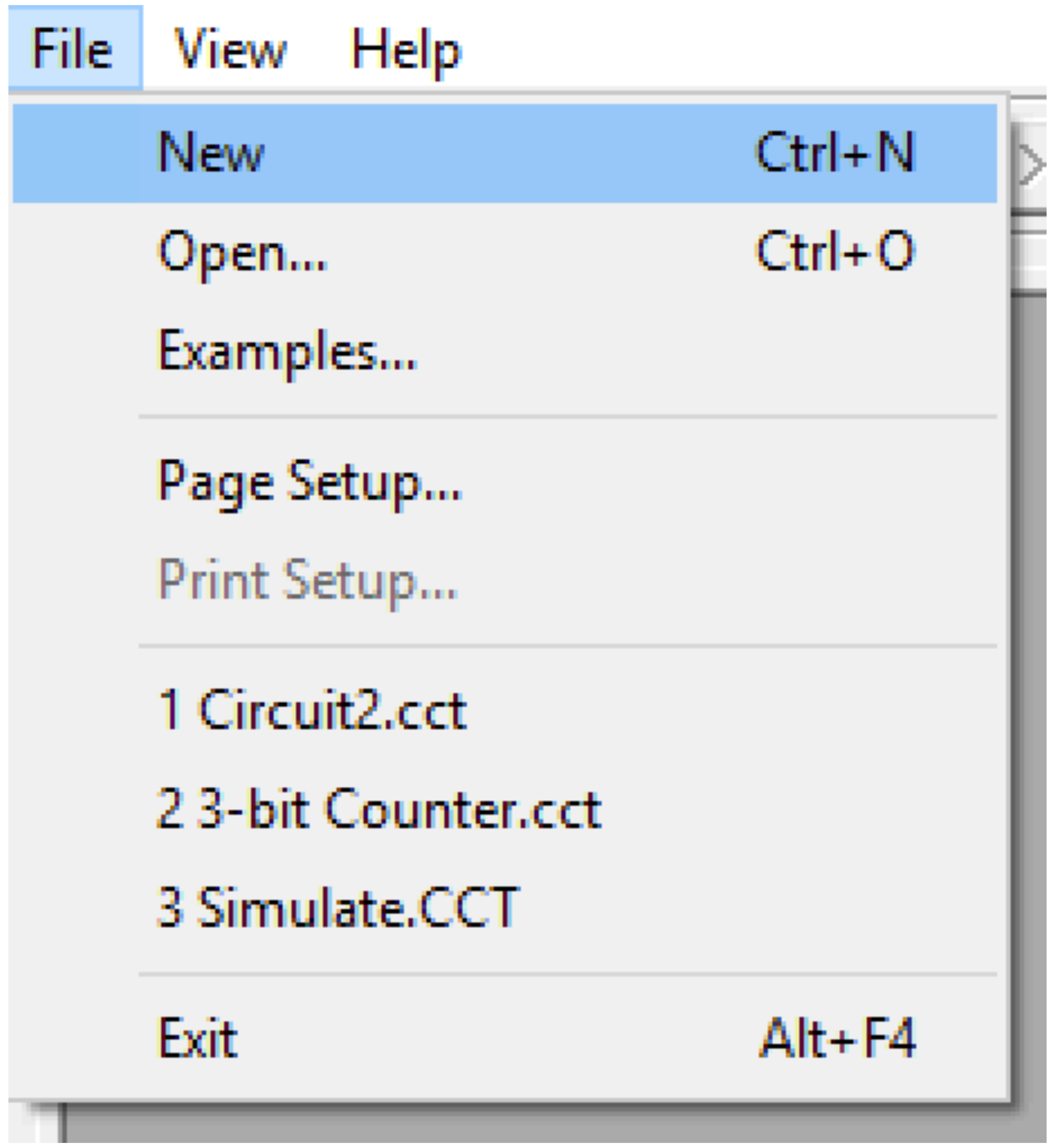


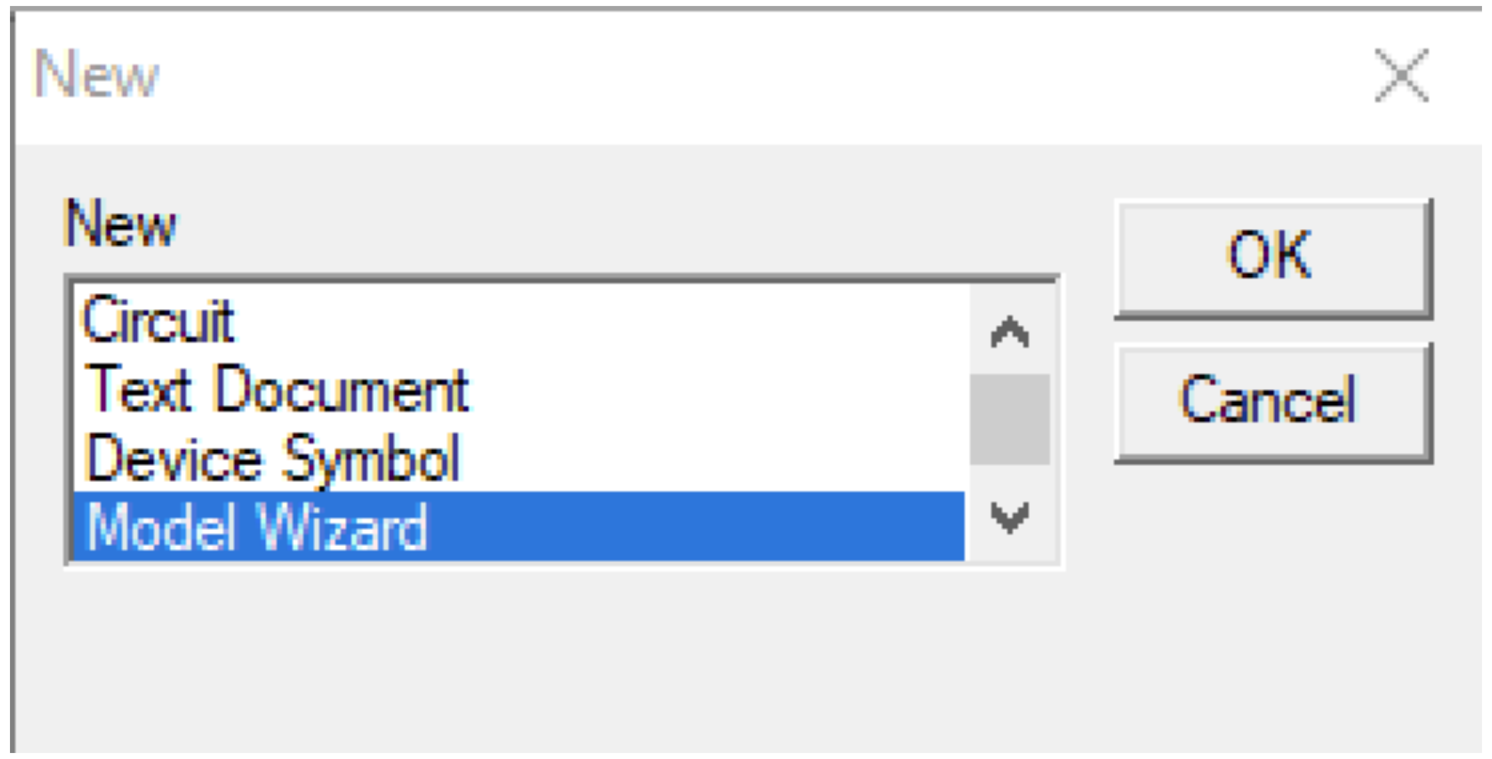


Tutorial

P2 VHDL Creating a AND1INV model





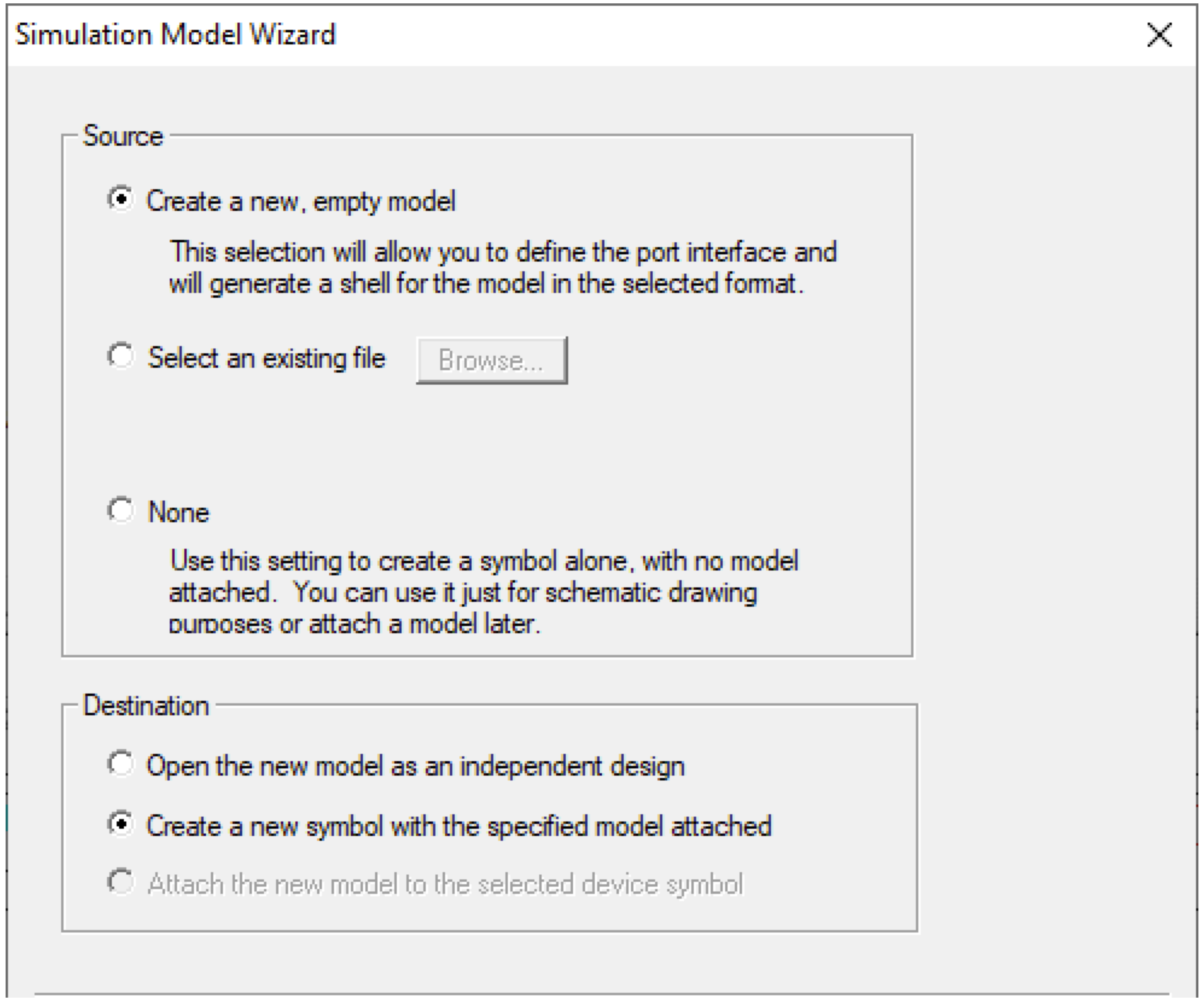


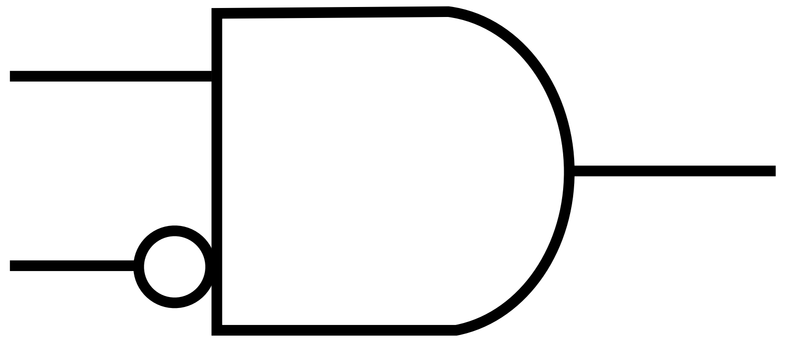
1. Go to the `File` menu, select `New` command. Select `Model Wizard` and click `OK`

Tutorial

P2
VHDL

Creating a AND1INV model



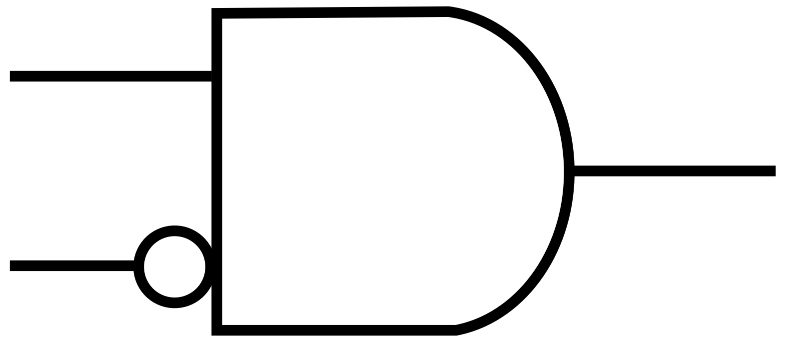


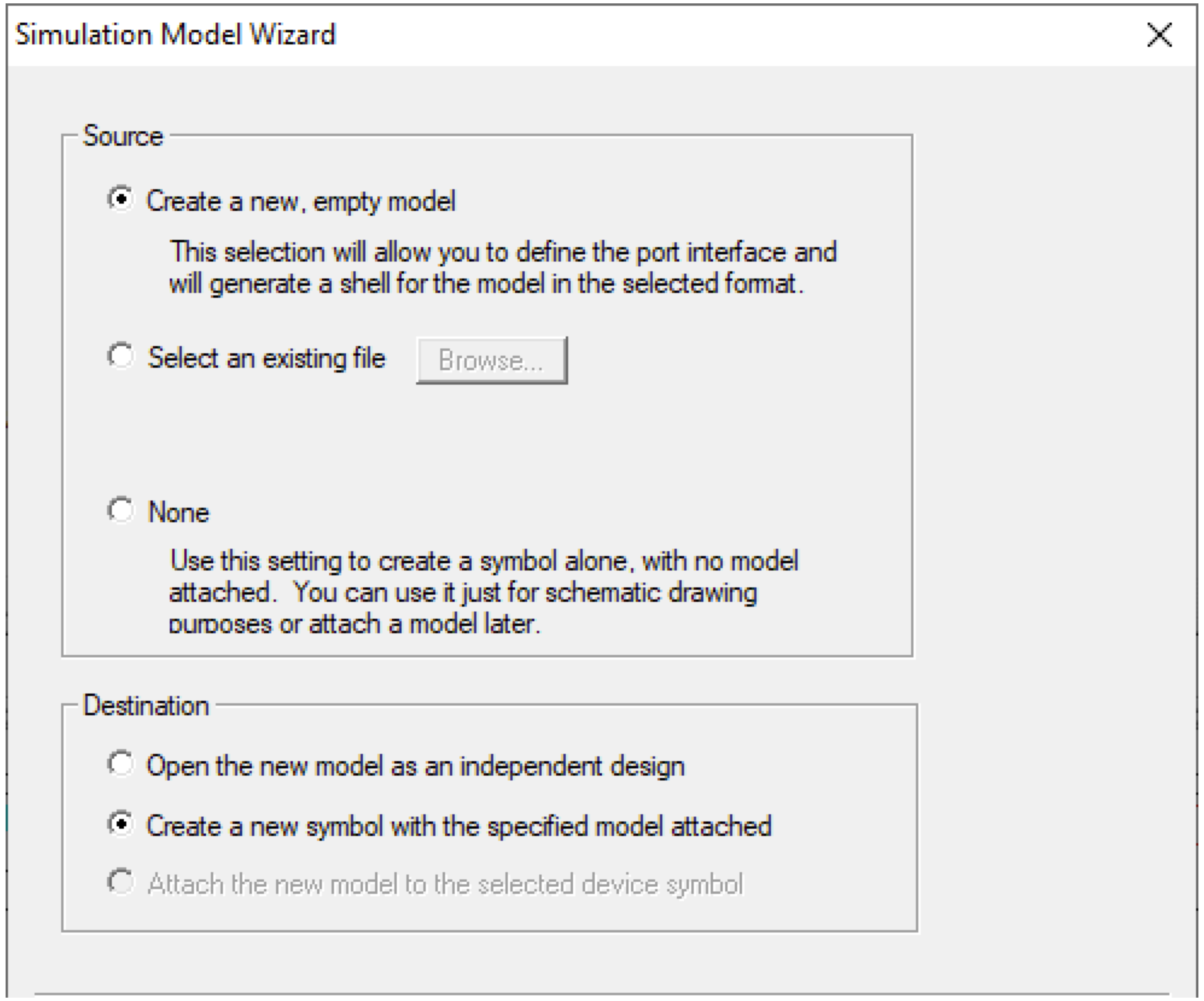
2. **Source:** `Create a new, empty model`, **Destination:** `Create a new symbol with the specified model attached`. Click `Next`

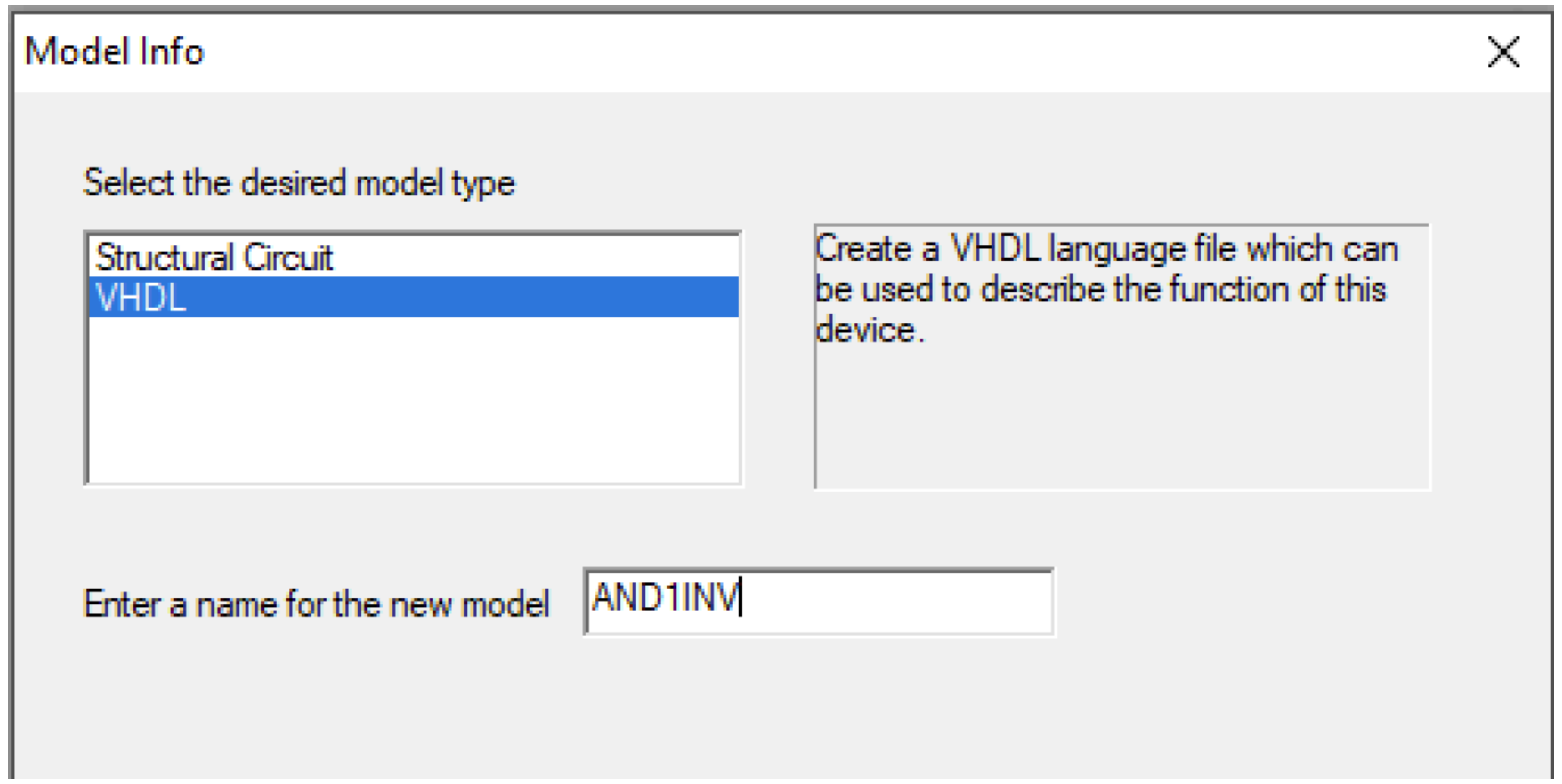
Tutorial

P2
VHDL

Creating a AND1INV model





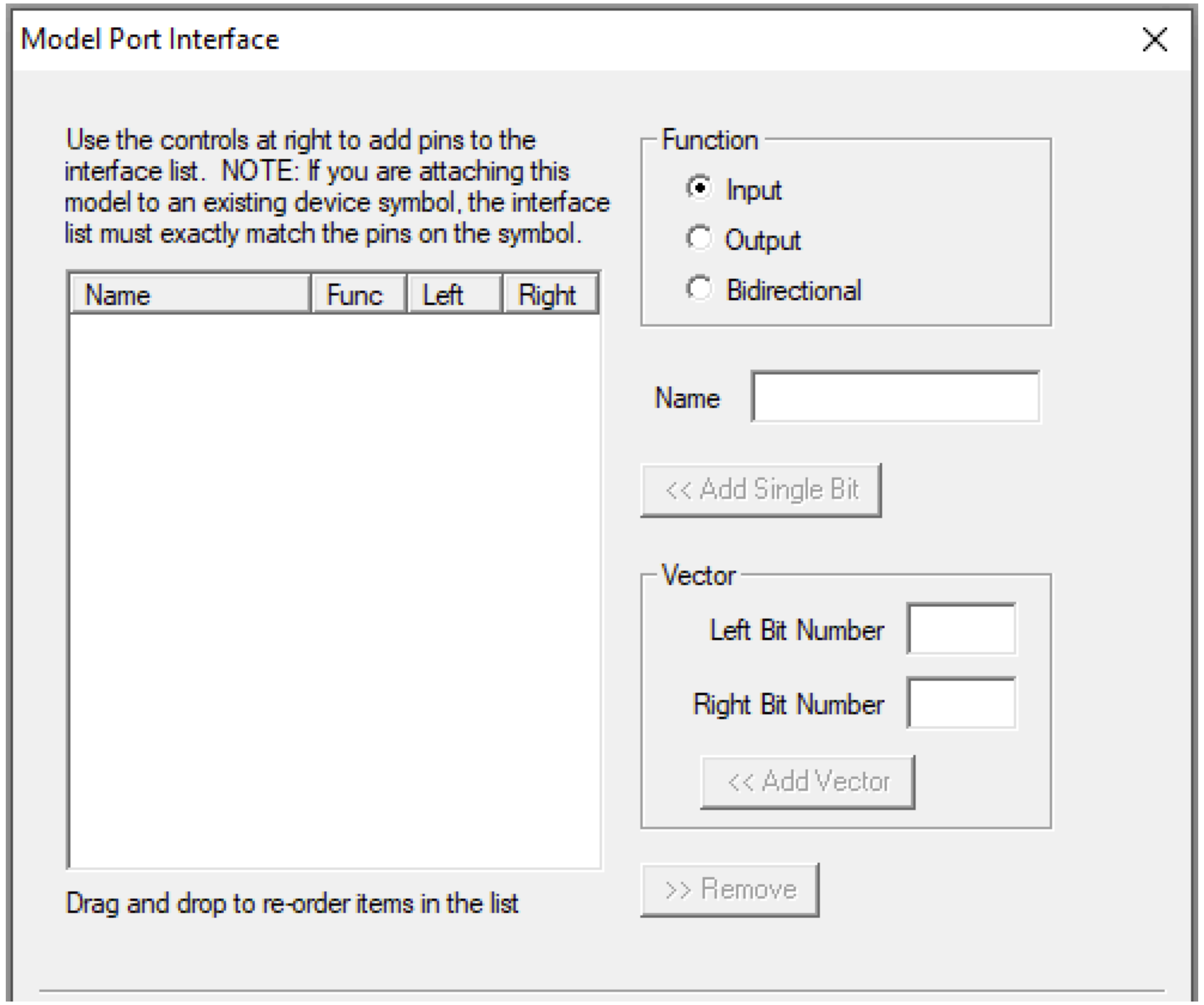


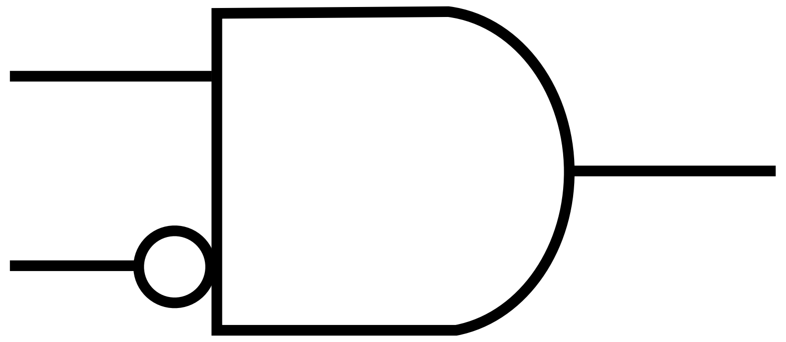
Tutorial

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

Creating a AND1INV model

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



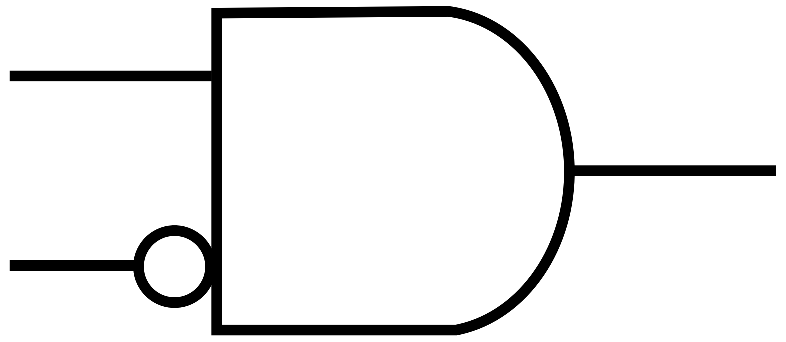


3. Type the `Name` and select the `Function` accordingly, then press `Add Single Bit`, click `Finish` to create the model file.

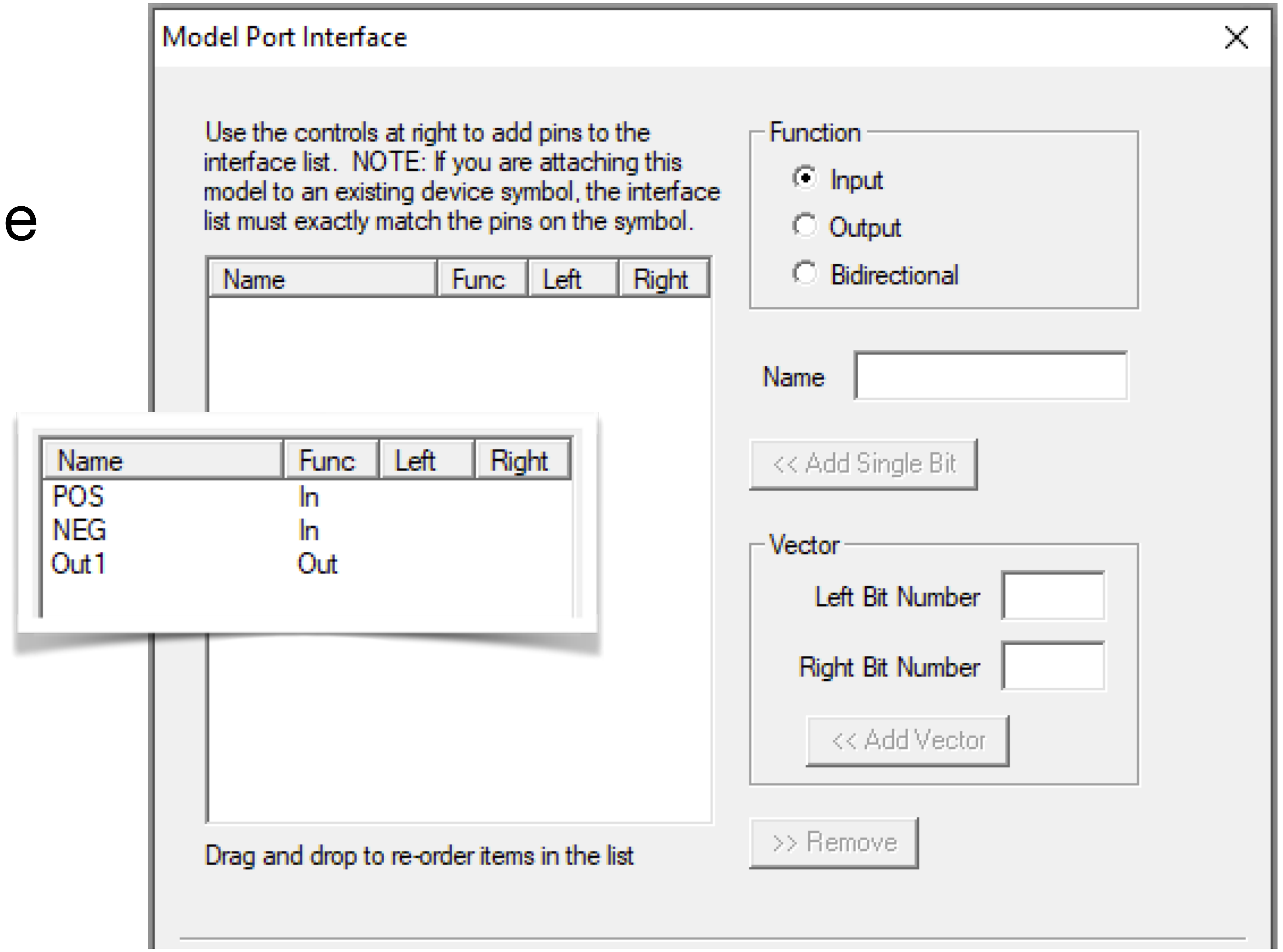
Tutorial

P2
VHDL

Creating a AND1INV model



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

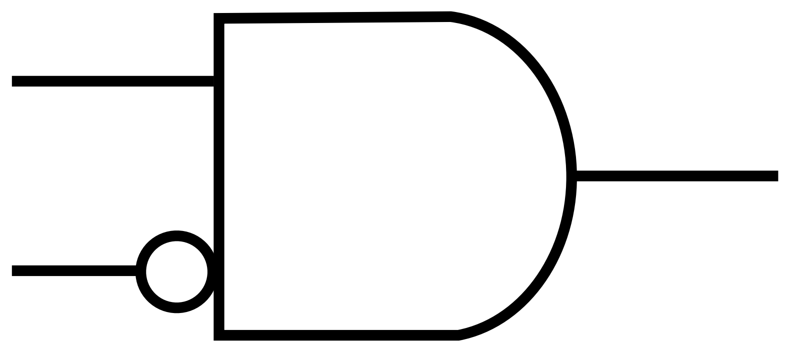


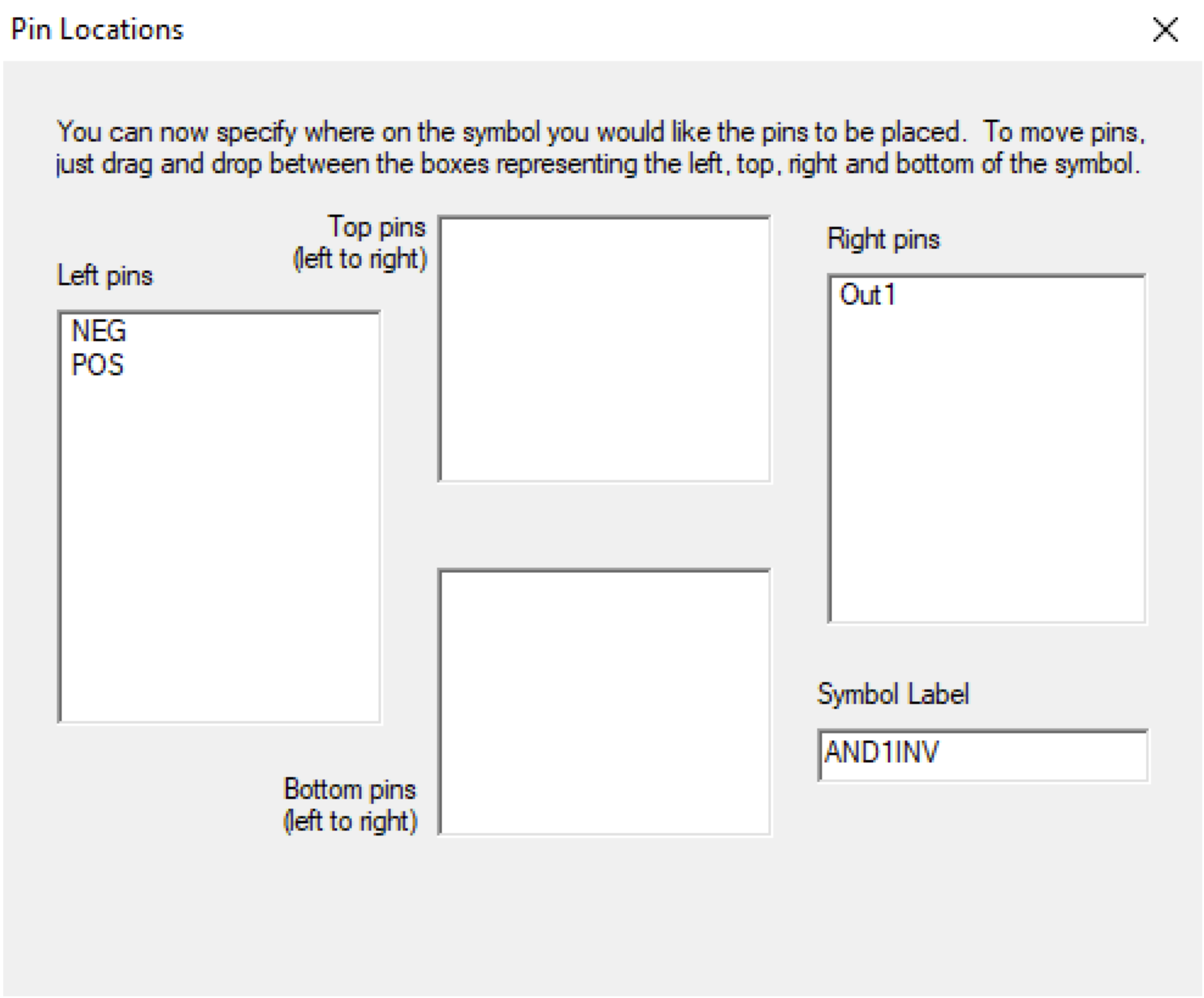
Tutorial

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

P2
VHDL

Creating a AND1INV model

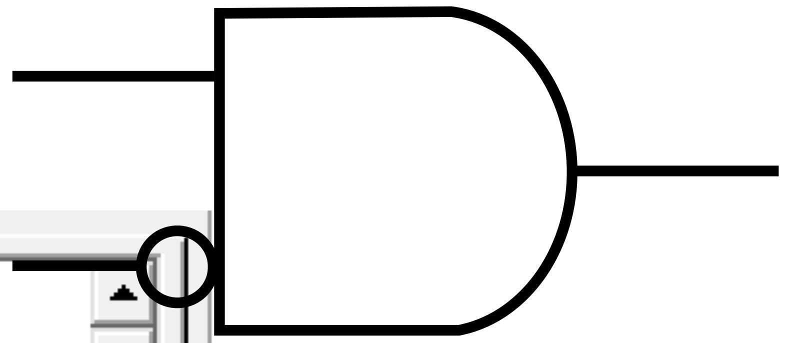




4. The programme will ask you for `Pin Location` assignment. Just click `Next`.

Tutorial

Creating a AND1INV model

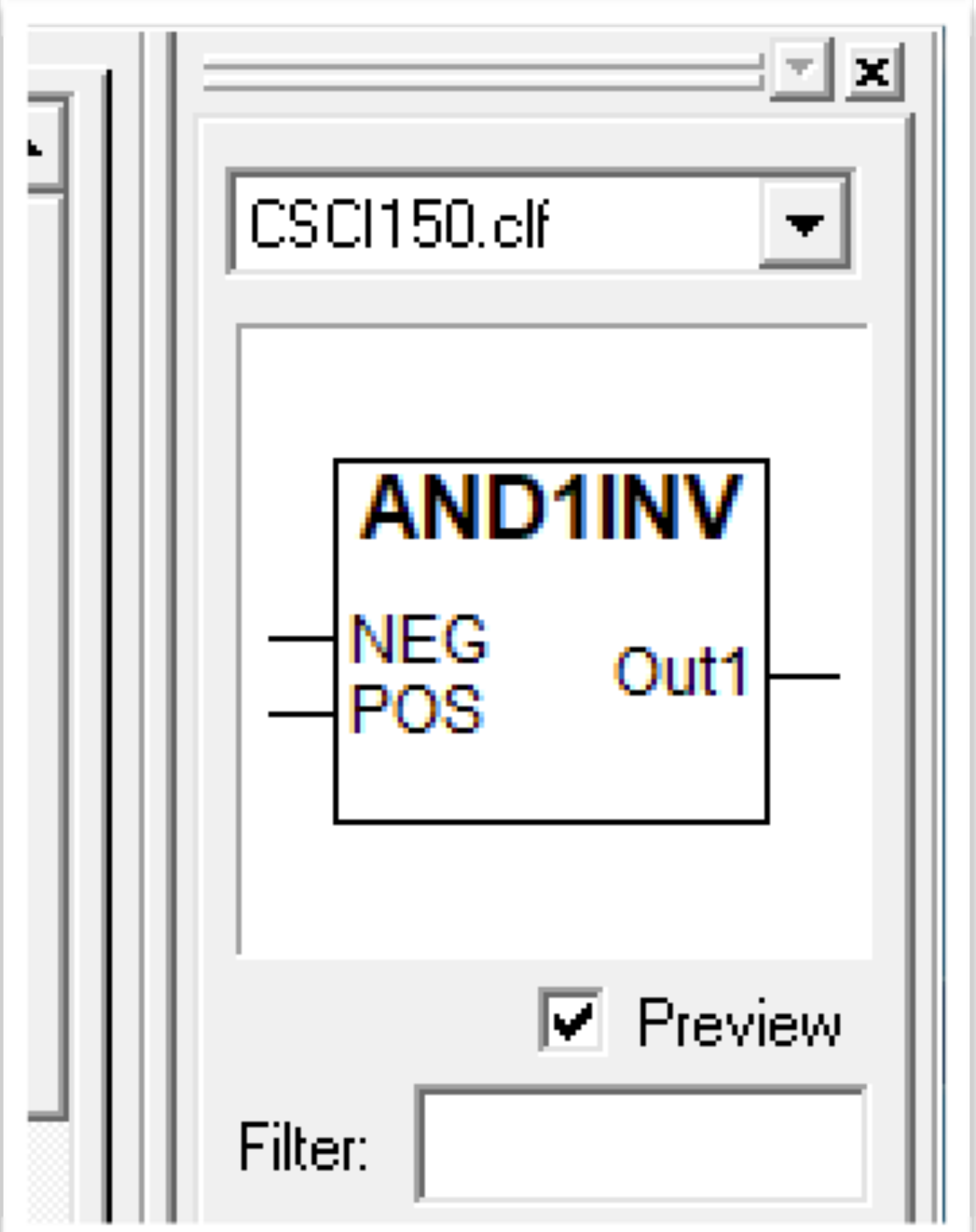


```
library IEEE;
use IEEE.std_logic_1164.all;

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

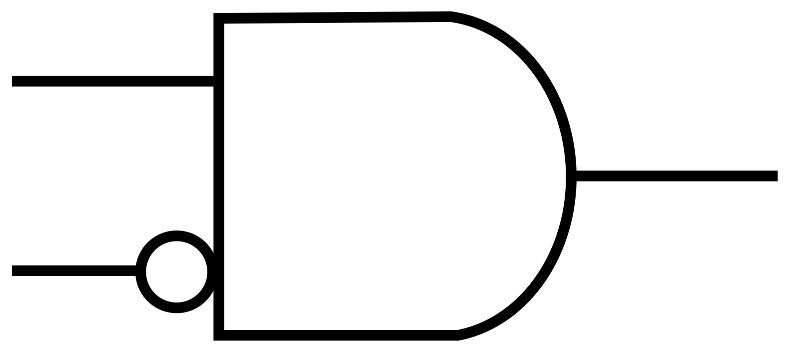
AND1INV.dwv



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;
```

AND1INV.dwv

`OUT1 <= POS AND NOT NEG AFTER 1NS;`

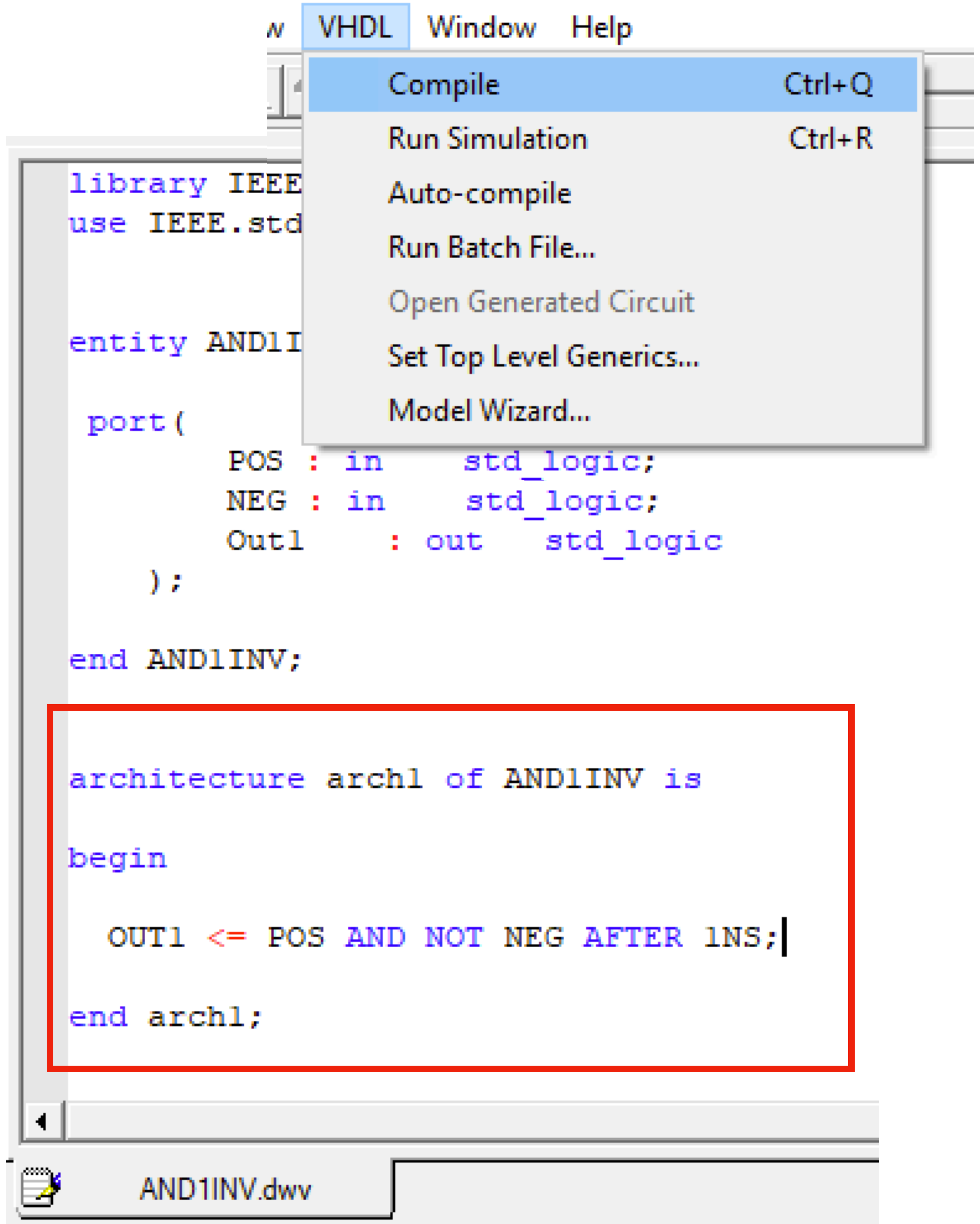
- `<=` — **Transferring**
- `AND`, `NOT` — **Boolean Operators**
- `AFTER`, `1NS` — **Things do not happen simultaneously**

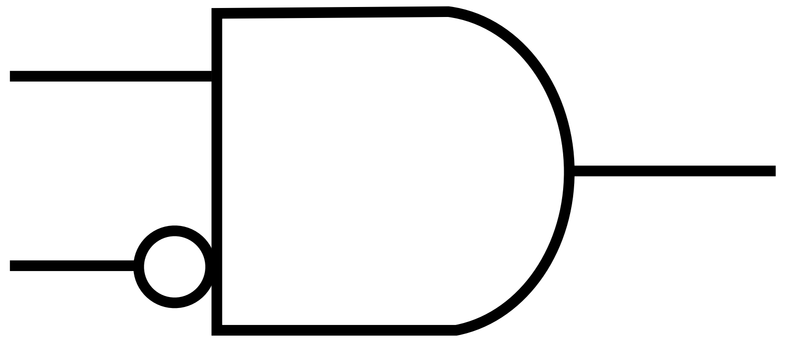
5. **Type:** `OUT1 <= POS AND NOT NEG AFTER 1NS;` **This is the Boolean description of the** `OUT1` **port**

Tutorial

P2
VHDL

Creating a AND1INV model





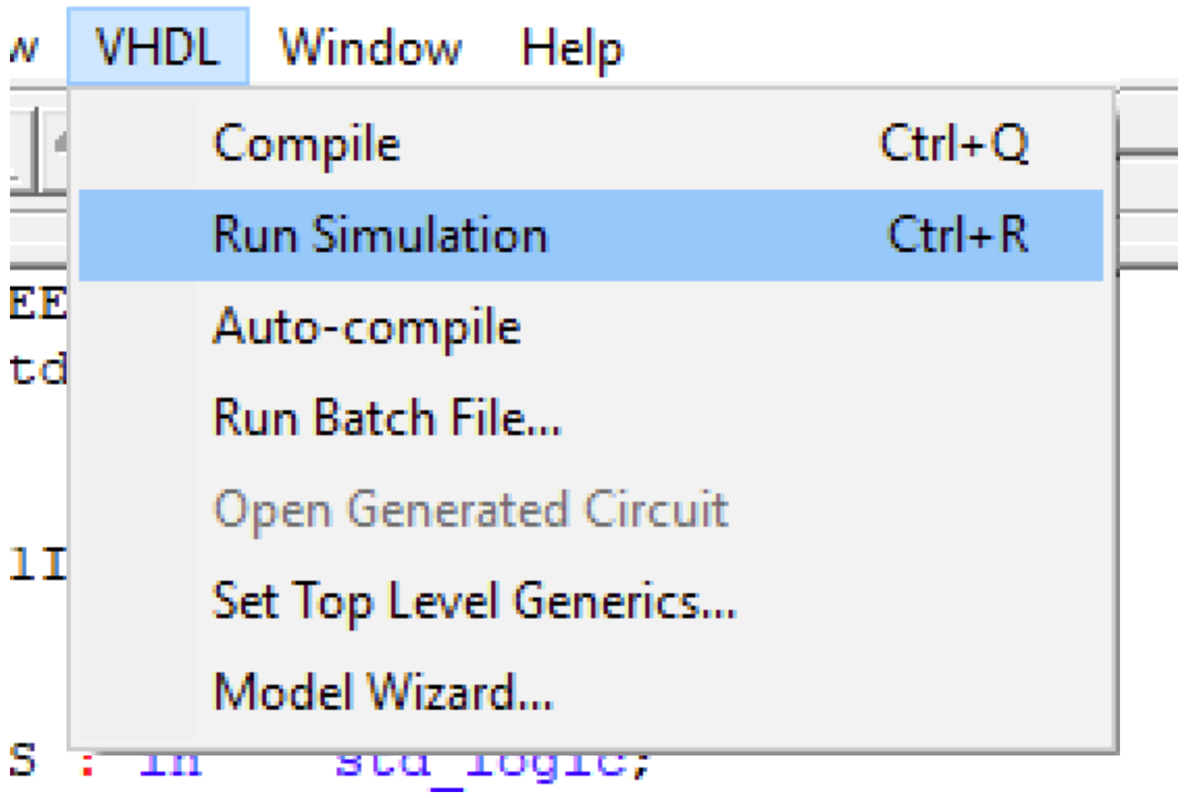
`OUT1 <= POS AND NOT NEG AFTER 1NS;`

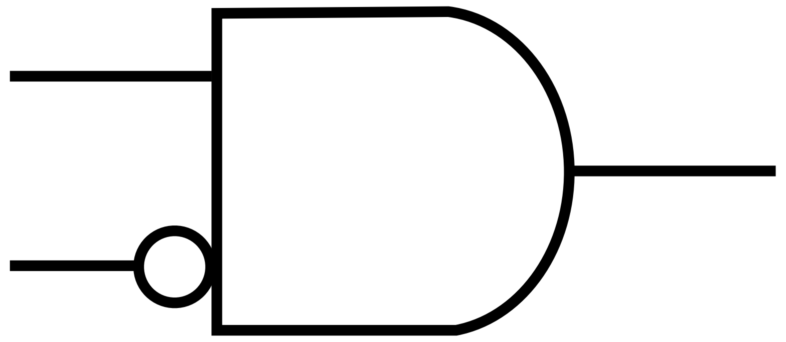
- `<=` : **Transferring**
- `AND`, `NOT` : **Boolean Operators**
- `AFTER`, `1NS` : **Things do not happen simultaneously**

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

Tutorial

Run Simulation





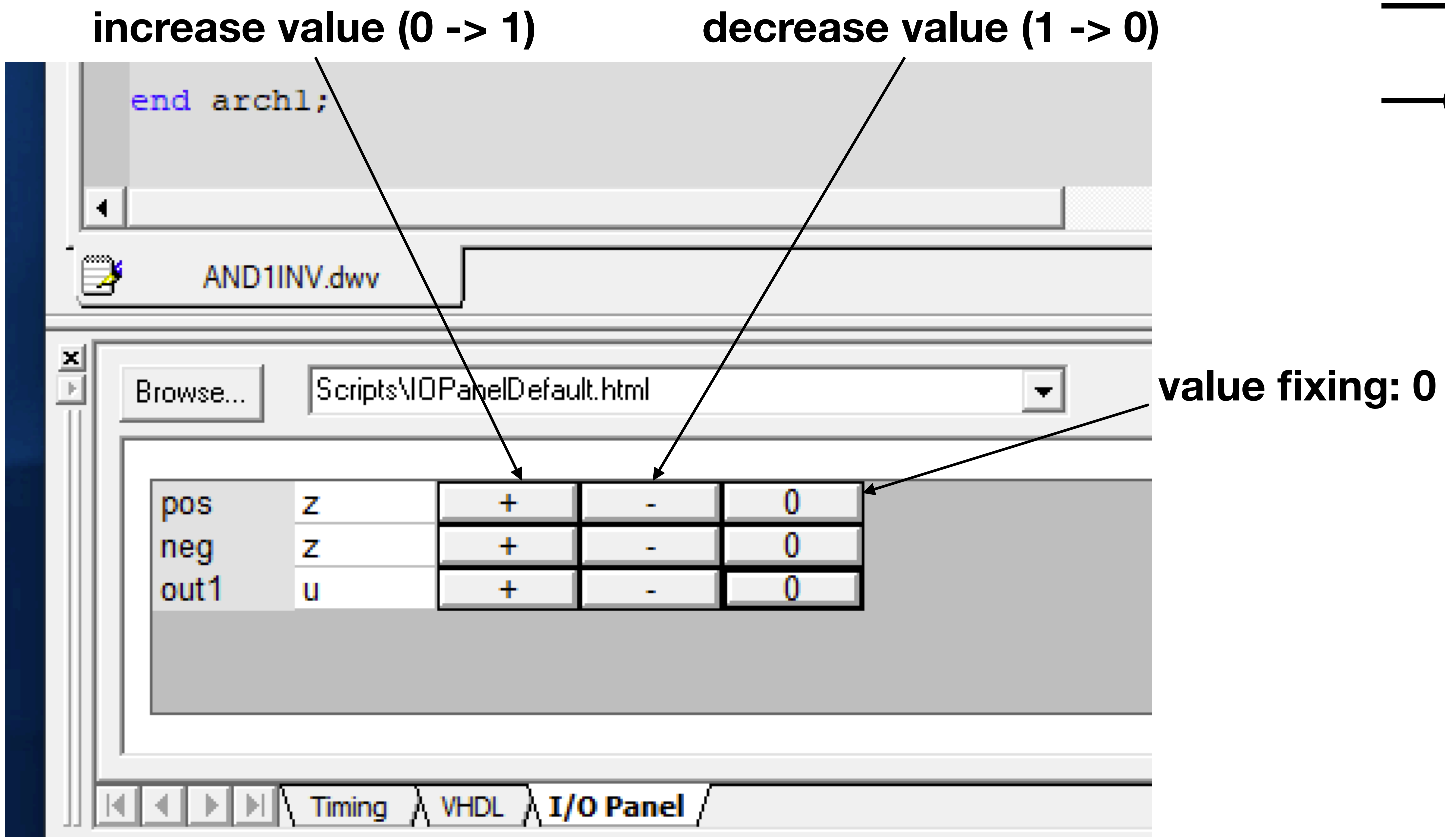
I/O Page

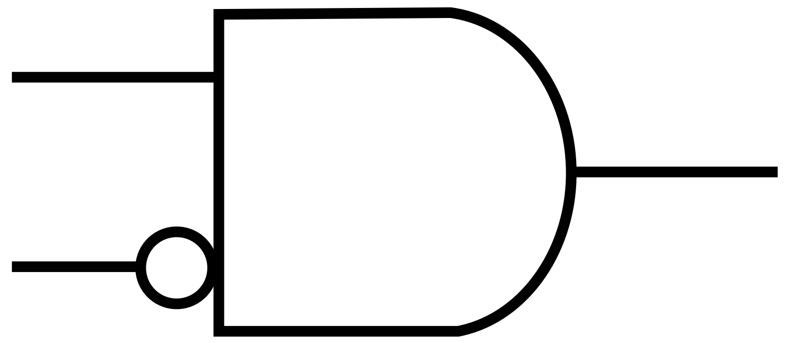
Run

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

Tutorial

Run Simulation

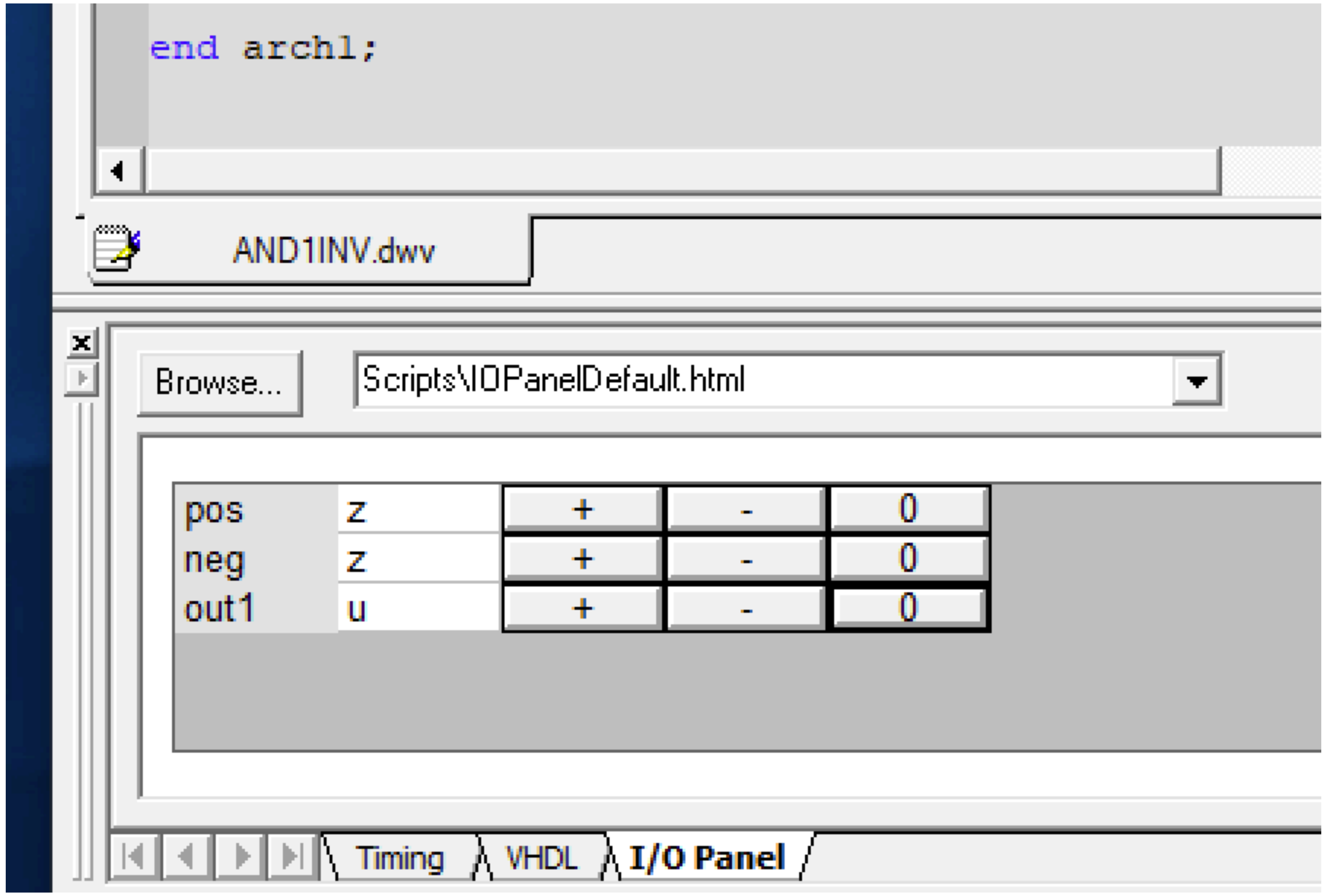


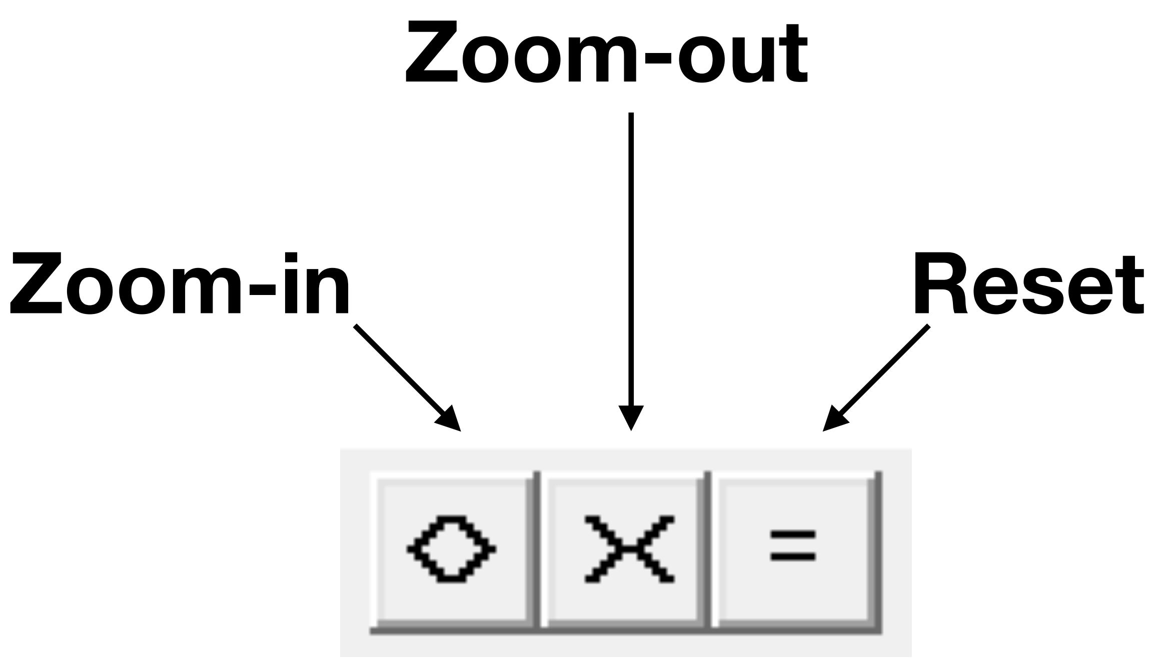


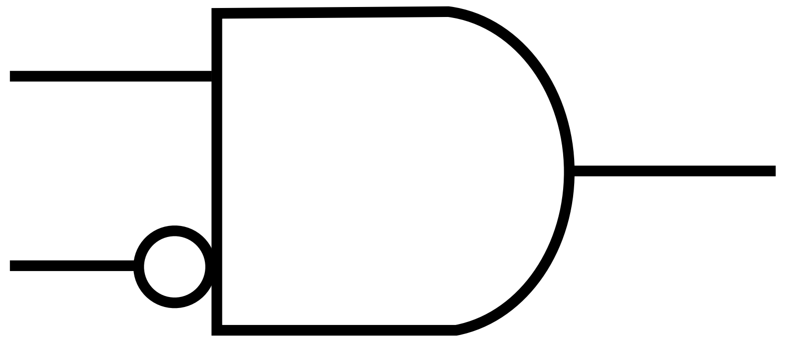
7. Play with the buttons in the I/O panel

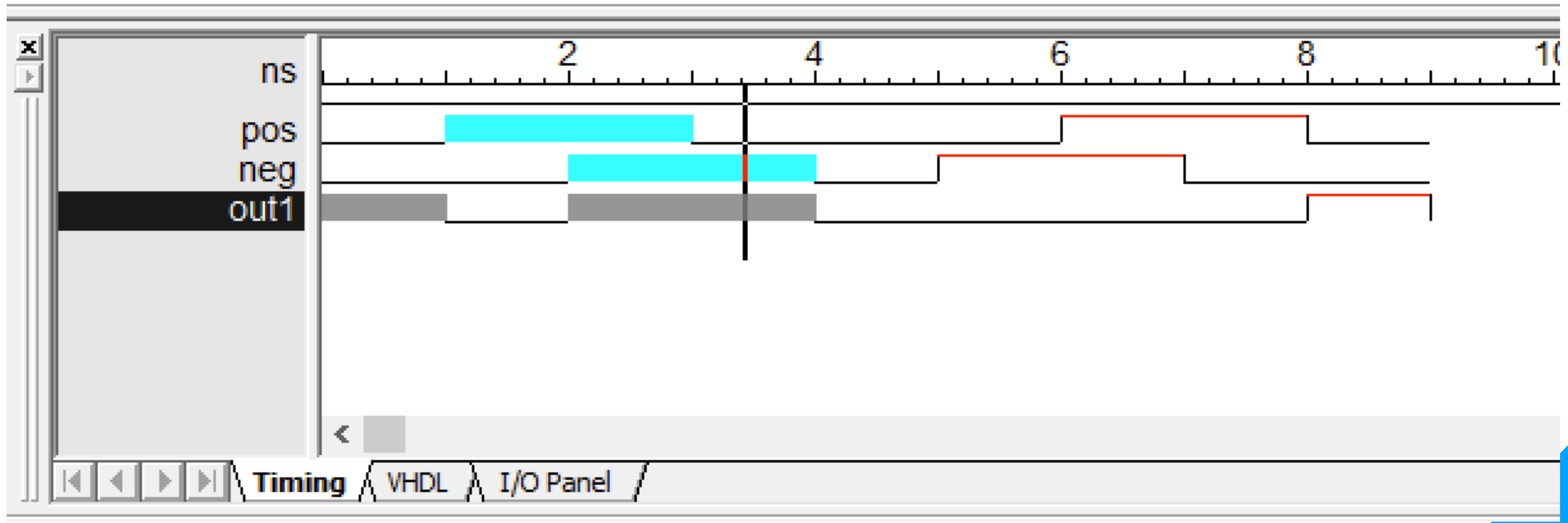
P2
VHDL

Run Simulation







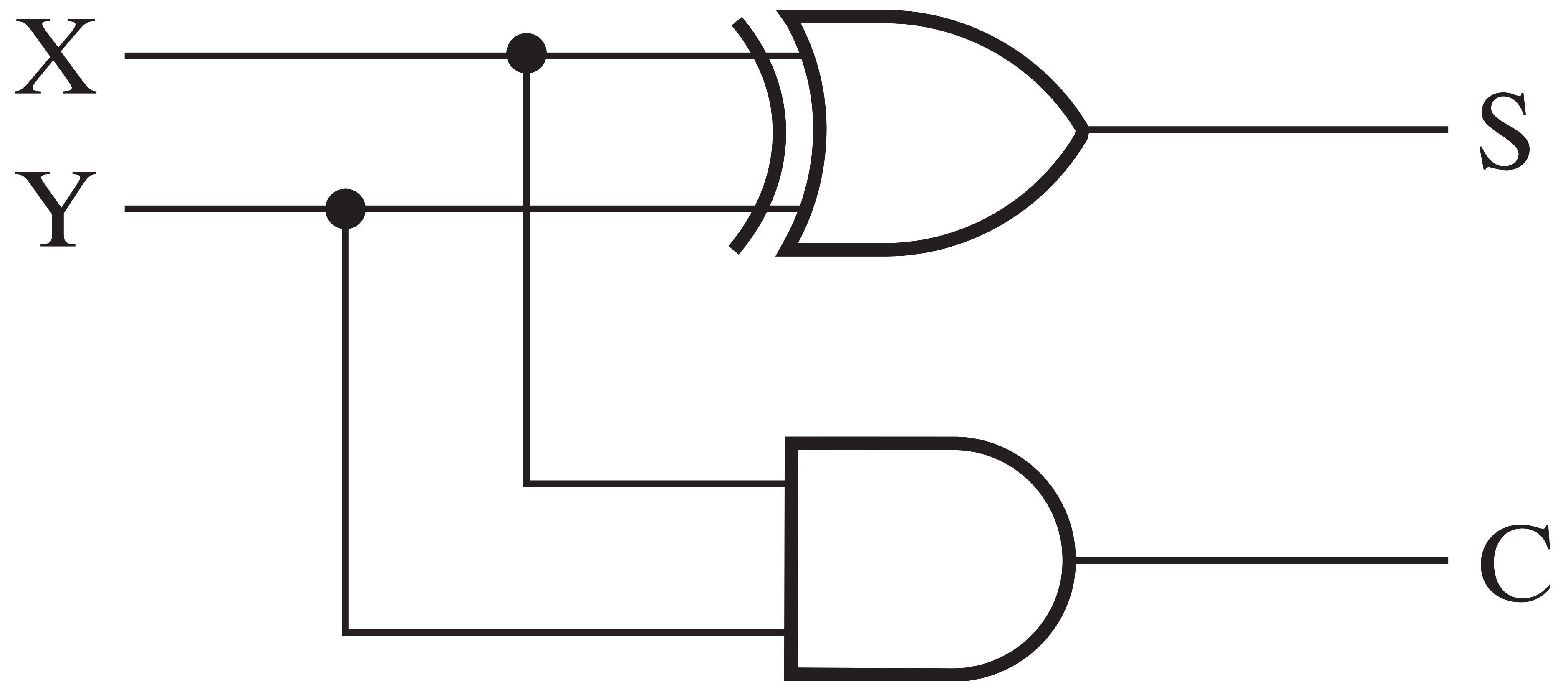


8. Changes are reflected in the Timing Diagram. Use Zoom panel to Zoom In and Out

Tutorial

Exe1: 1-bit Half Adder

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

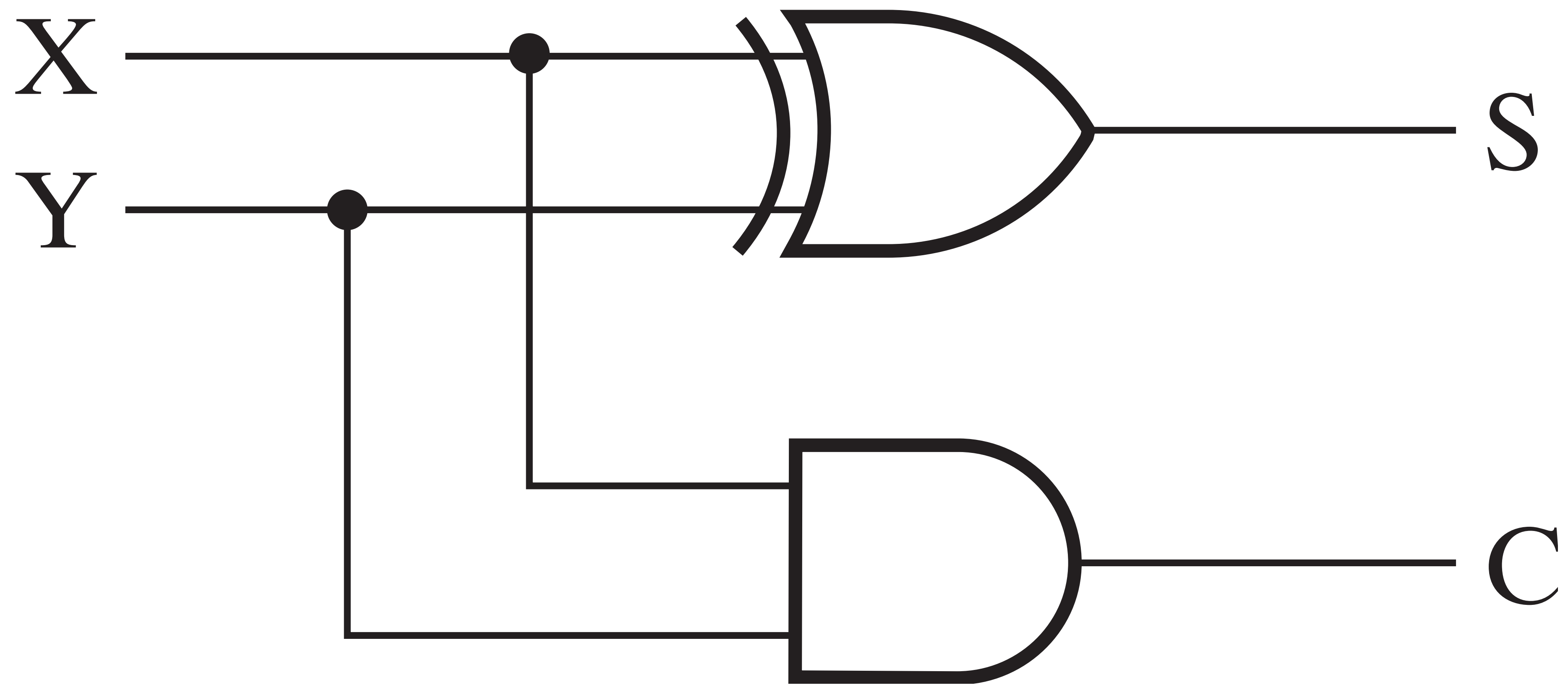


Practice

P2
VHDL

Exe1: 1-bit Half Adder

```
architecture arch1 of HalfAdder is
begin
    S <= X XOR Y;
    C <= X AND Y;
end arch1;
```



Practice