19.10.20 13:49

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

Jetic Gū

2020 Fall Semester (S3)

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

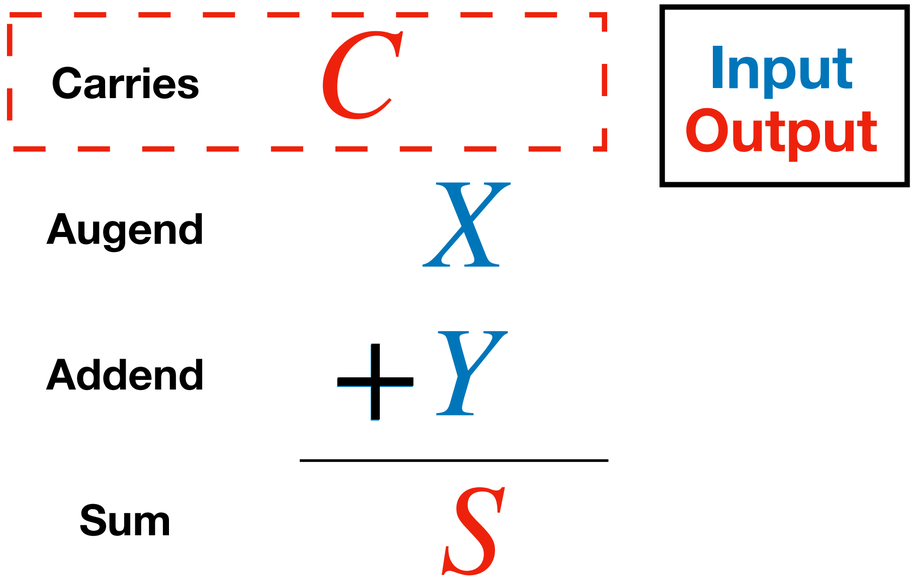
Review

Unsigned Binary Adder

Summary

1-bit Half Adder

- Half adder
  input X, Y
  output S, C



Review

P0
Binary Adder

1-bit Half Adder

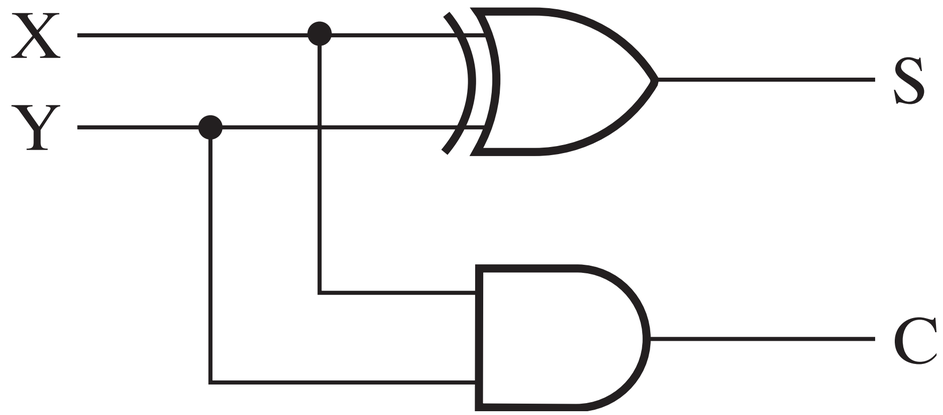
Carries C

Augend X

Addend $+ Y$

Sum S

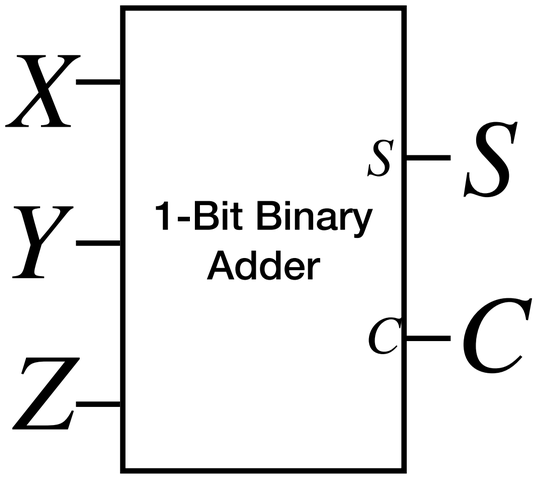
Input
Output

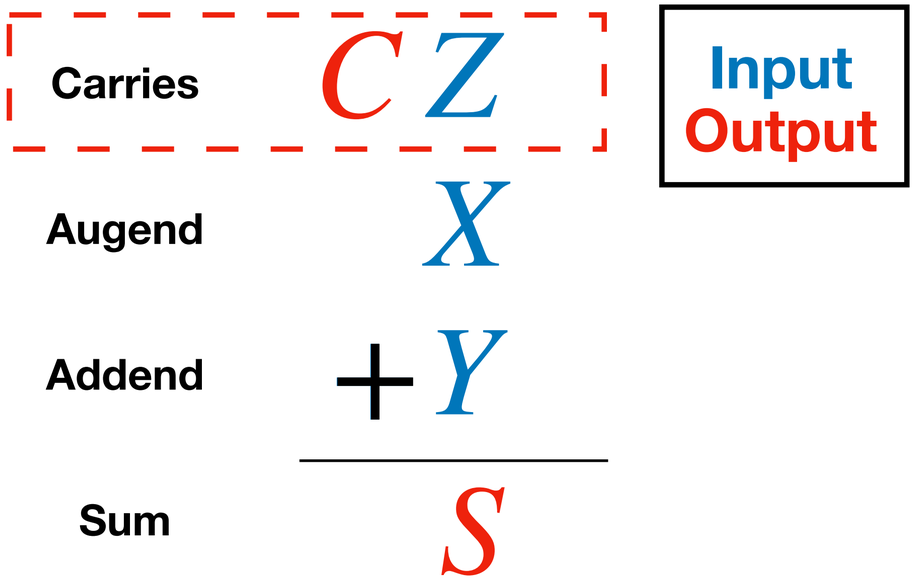


Review

1-bit Full Adder

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





Review

P0
Binary Adder

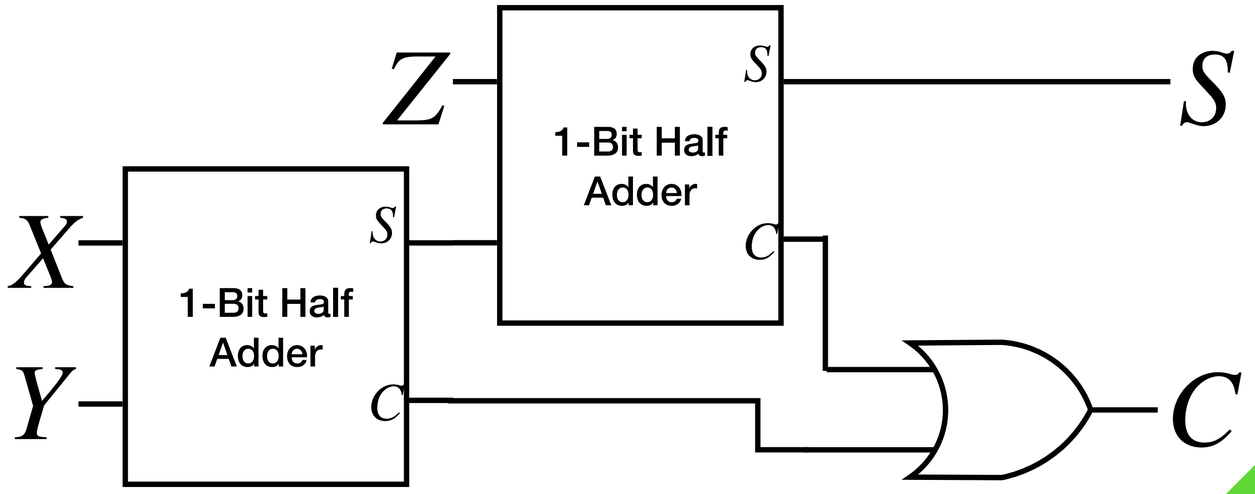
1-bit Full Adder

- 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''

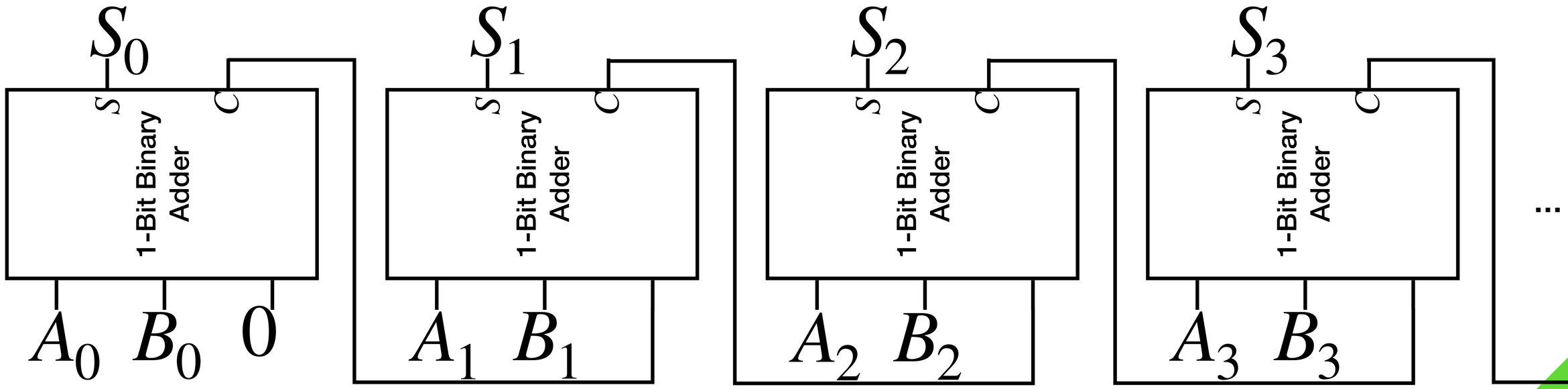
$$C = C' + C''$$



Review

P0
Binary Adder

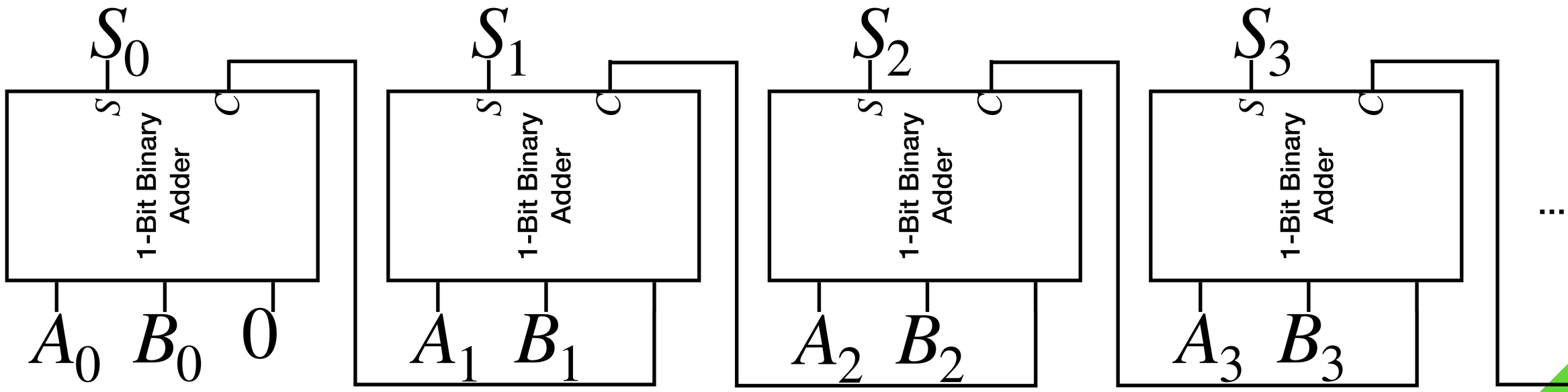
n-bit Full Adder



Review

n-bit Full Adder

- Ripple Carry Adder



Review

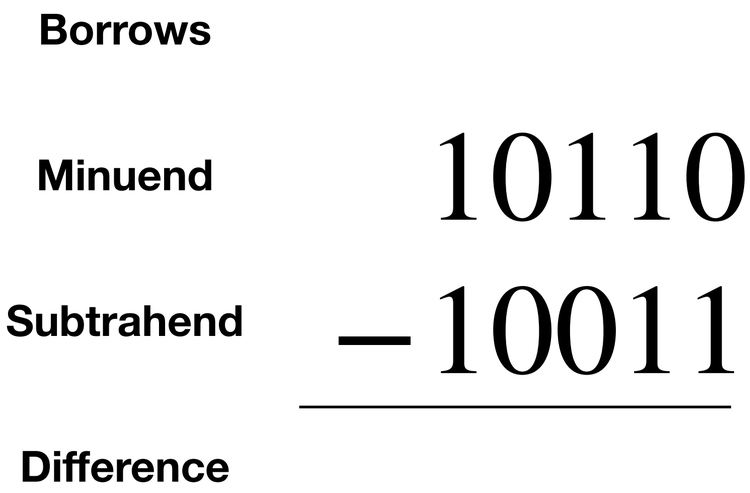
Unsigned Binary Subtraction I

Summary

P1
Subtraction

Unsigned Binary Subtraction

- Input: Minuend and Subtrahend
  Previous borrow
- Output: Last borrow, difference



Review

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	10
Minuend	10110
Subtrahend	−10011
Difference	1

Review

P1
Subtraction

Unsigned Binary Subtraction

- Input: Minuend and Subtrahend
  Previous borrow
- Output: Last borrow, difference

Borrows	110
Minuend	10110
Subtrahend	−10011
Difference	11

Review

P1
Subtraction

Unsigned Binary Subtraction

- Input: Minuend and Subtrahend
  Previous borrow
- Output: Last borrow, difference

Borrows	0110
Minuend	10110
Subtrahend	−10011
Difference	011

Review

P1
Subtraction

Unsigned Binary Subtraction

- Input: Minuend and Subtrahend
  Previous borrow
- Output: Last borrow, difference

Borrows	00110
Minuend	10110
Subtrahend	−10011
Difference	0011

Review

P1
Subtraction

Unsigned Binary Subtraction

- Input: Minuend and Subtrahend
  Previous borrow
- Output: Last borrow, difference

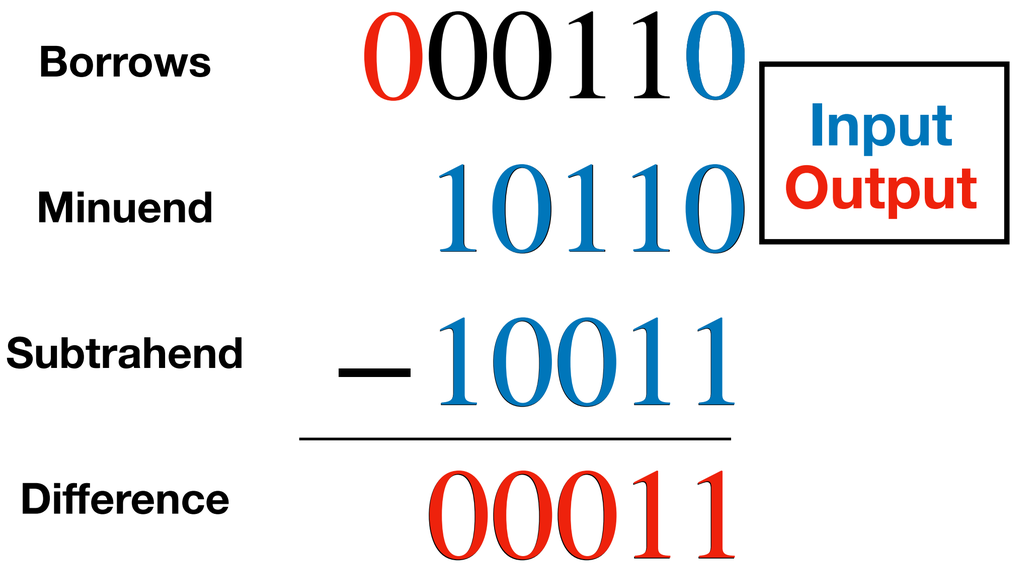
Borrows	000110
Minuend	10110
Subtrahend	−10011
Difference	00011

Review

P1
Subtraction

Unsigned Binary Subtraction

- Input: Minuend and Subtrahend
  Previous borrow
- Output: Last borrow, difference

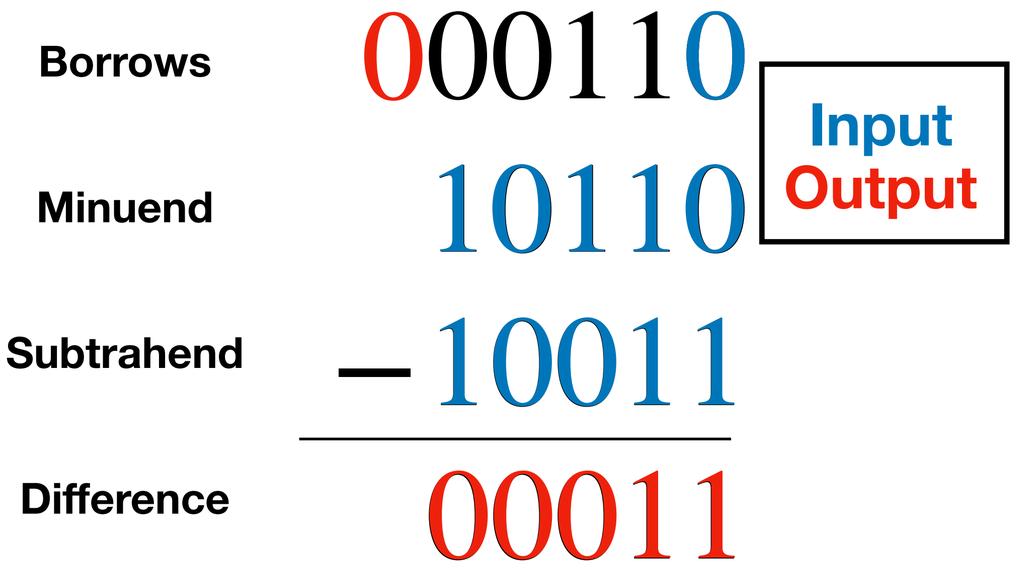


Review

P1
Subtraction

Unsigned Binary Subtraction

- Input: Minuend and Subtrahend
  Previous borrow
- Output: Last borrow, difference



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

Review

P1
Subtraction

Unsigned Binary Subtraction

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

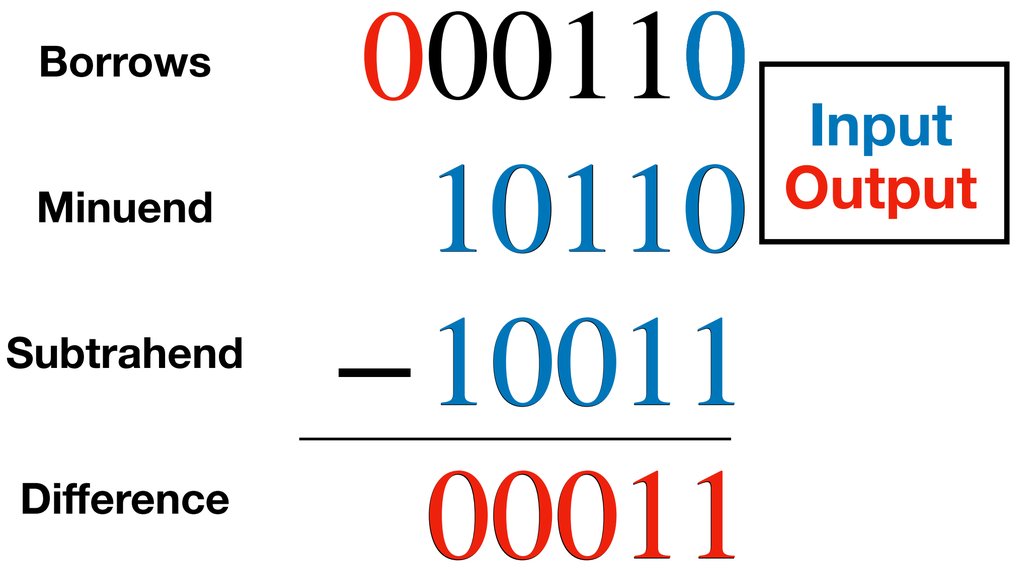
- We learned to perform subtraction, by subtracting the smaller number from the greater number

Review

P1
Subtraction

Unsigned Binary Subtraction

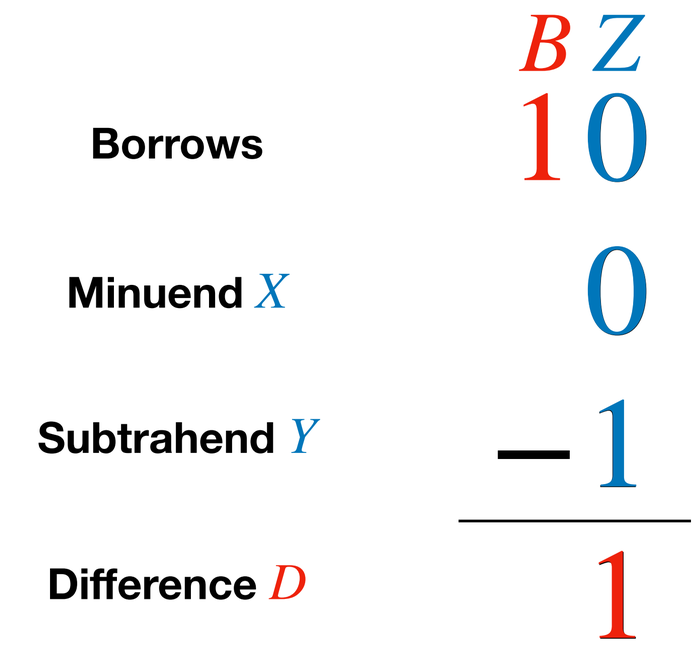
- Input: Minuend and Subtrahend
  Previous borrow
- Output: Last borrow, difference



Concept

Unsigned 1-bit Binary Subtraction

- Input: Minuend X and Subtrahend Y
  Previous borrow Z
- Output: Last borrow B, difference D



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

Borrows $B\ Z$ = $1\ 0$

Minuend X 0

Subtrahend Y $-\ 1$

Difference D 1

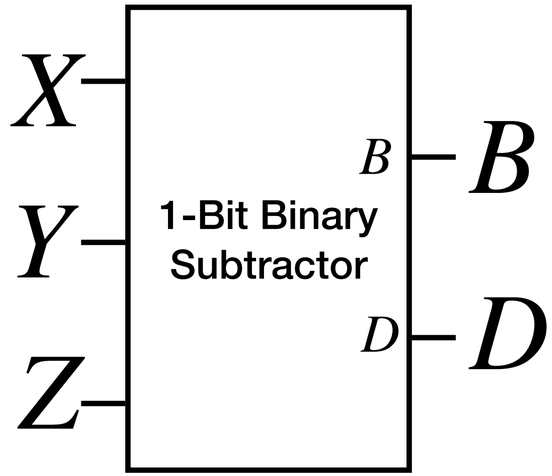
Input
Output

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

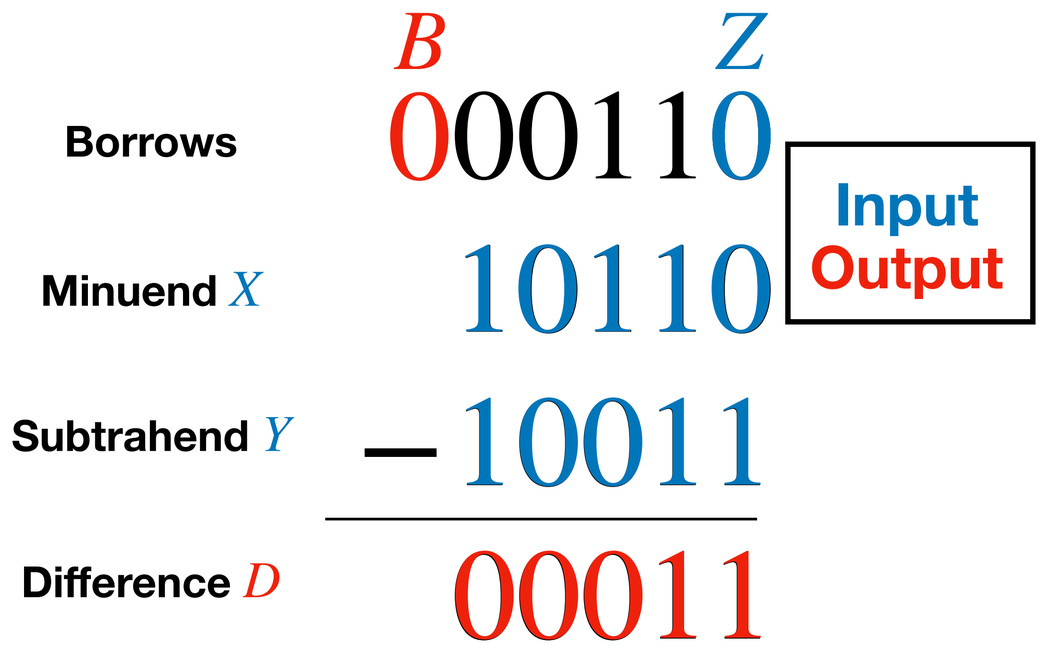


P1
Subtraction

Unsigned Binary Subtraction

Technology
- 1 bit Unsigned Subtractor

- Input: Minuend and Subtrahend
  Previous borrow
- Output: Last borrow, difference

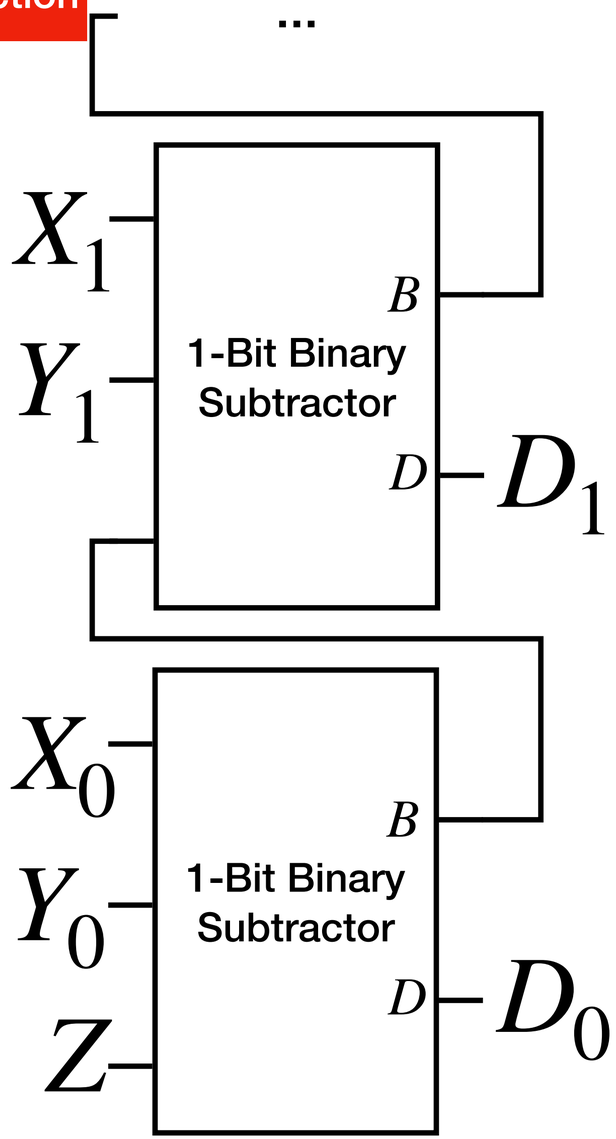


Concept

P1
Subtraction

Unsigned Binary Subtraction

Technology
- 1 bit Unsigned Subtractor



		B				Z
Borrows		0	0	0	1	1 0
Minuend $X_{0:n-1}$		1	0	1	1	0
Subtrahend $Y_{0:n-1}$	−	1	0	0	1	1
Difference $D_{0:n-1}$		0	0	0	1	1

Input
Output

Concept

Hardware Description Language

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

Summary

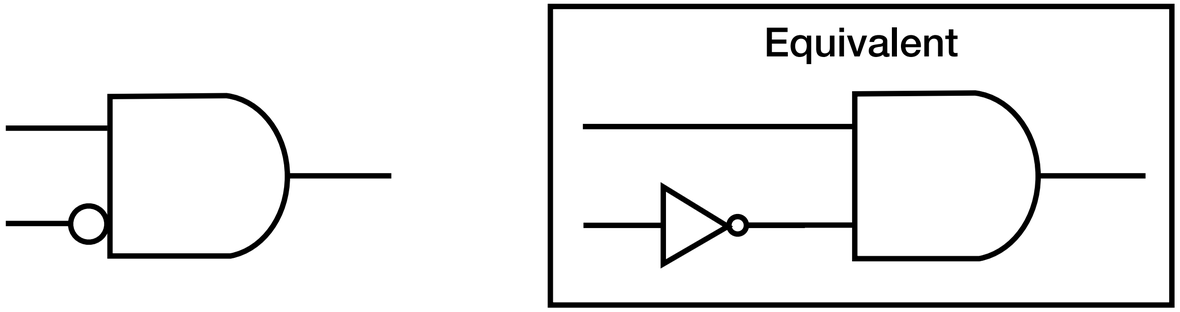
What is HDL

- 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

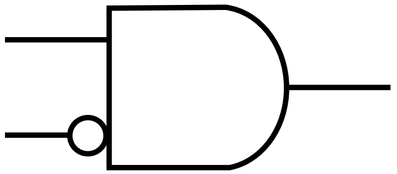
Creating a AND1INV model

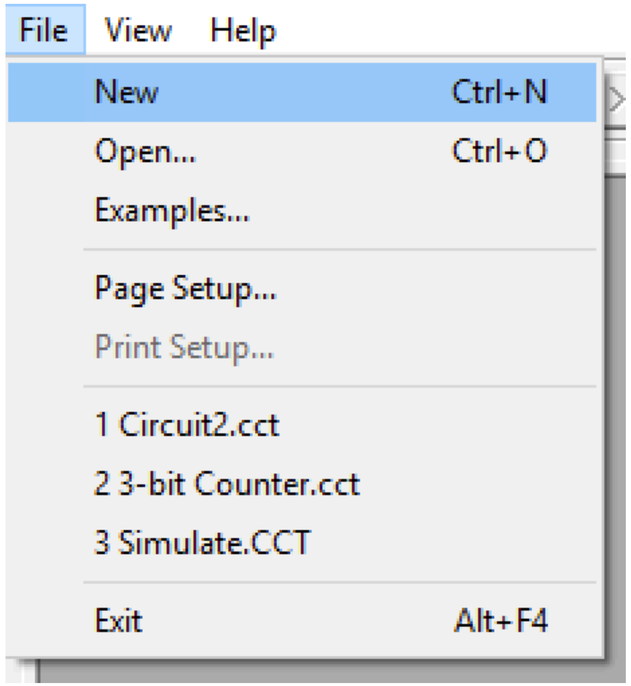


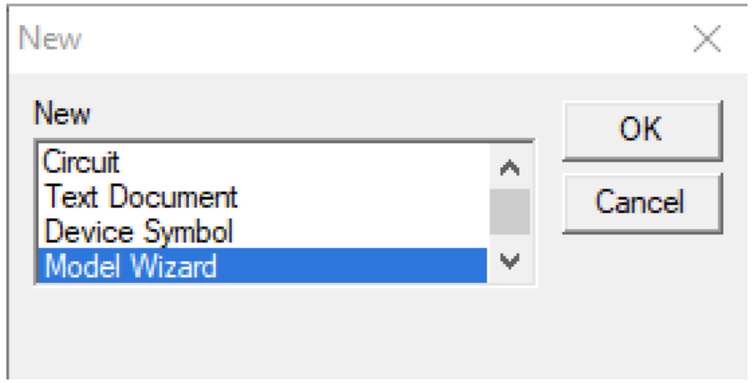
Tutorial

P2
VHDL

Creating a AND1INV model





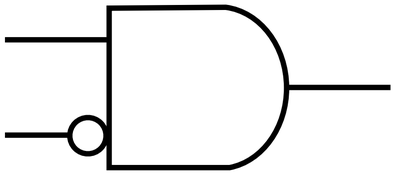


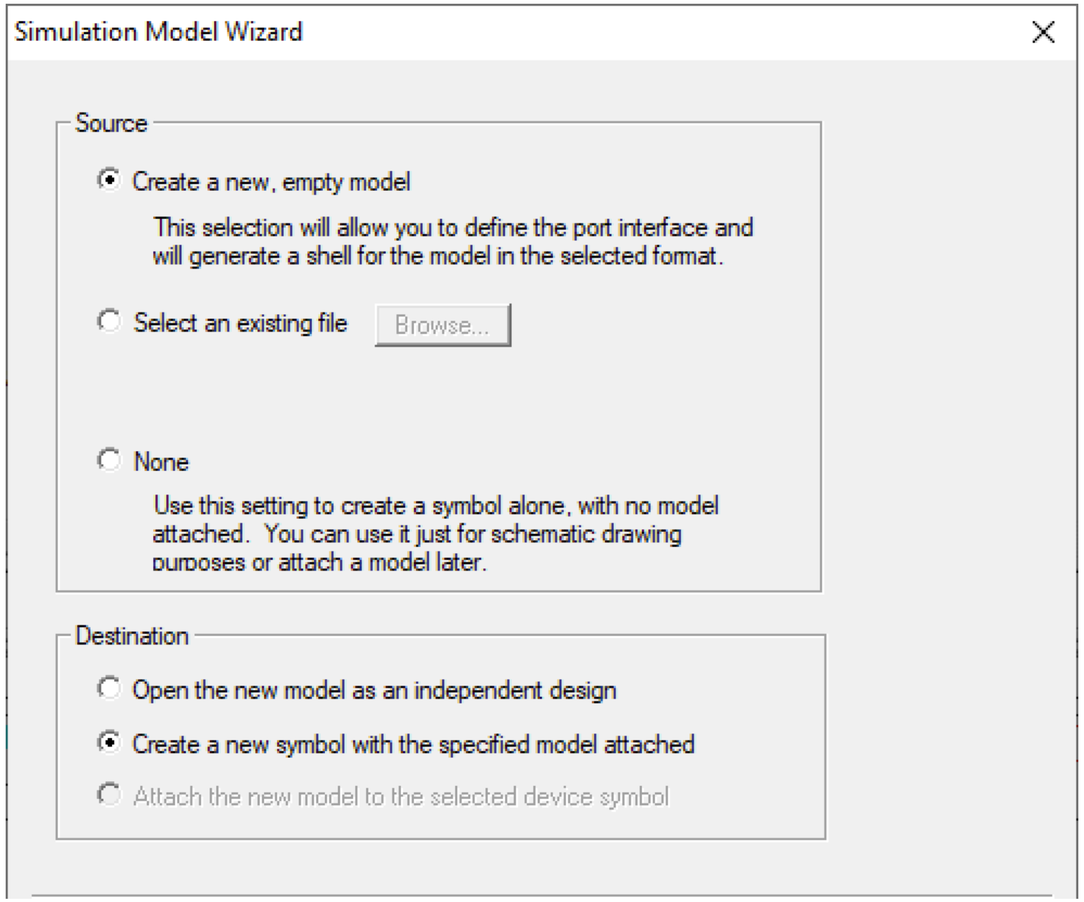
Tutorial

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

P2
VHDL

Creating a AND1INV model



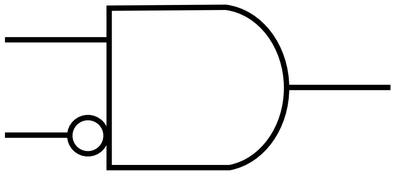


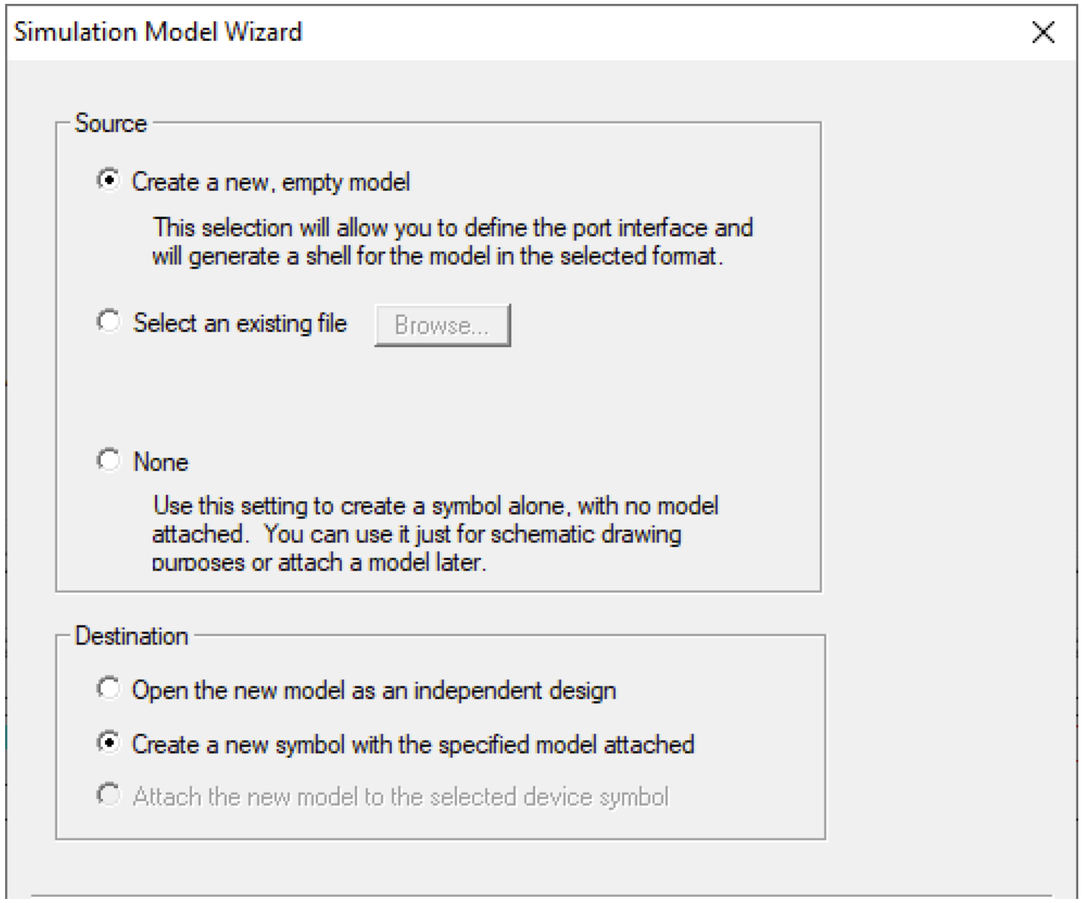
2. **Source:** `Create a new, empty model`, **Destination:** `Create a new symbol with the specified model attached`. Click `Next`

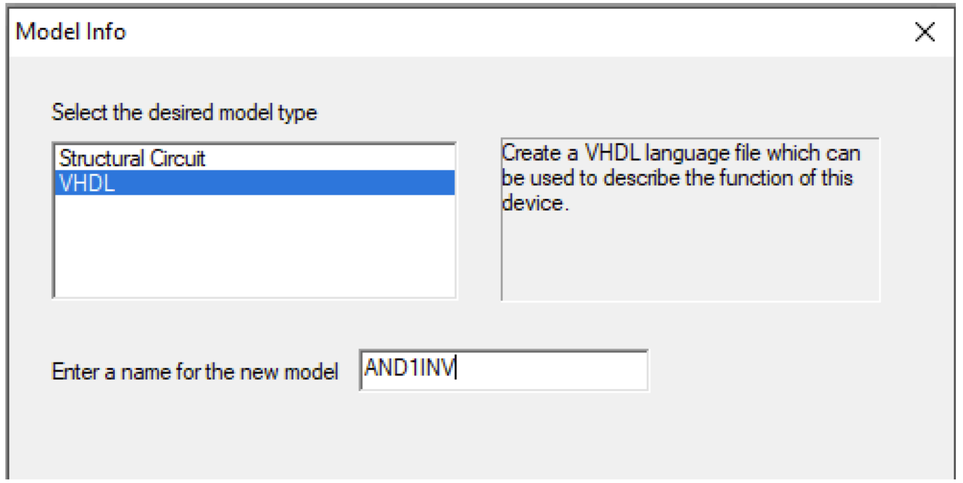
Tutorial

P2
VHDL

Creating a AND1INV model





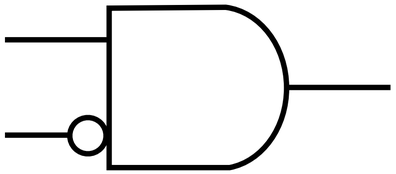


Tutorial

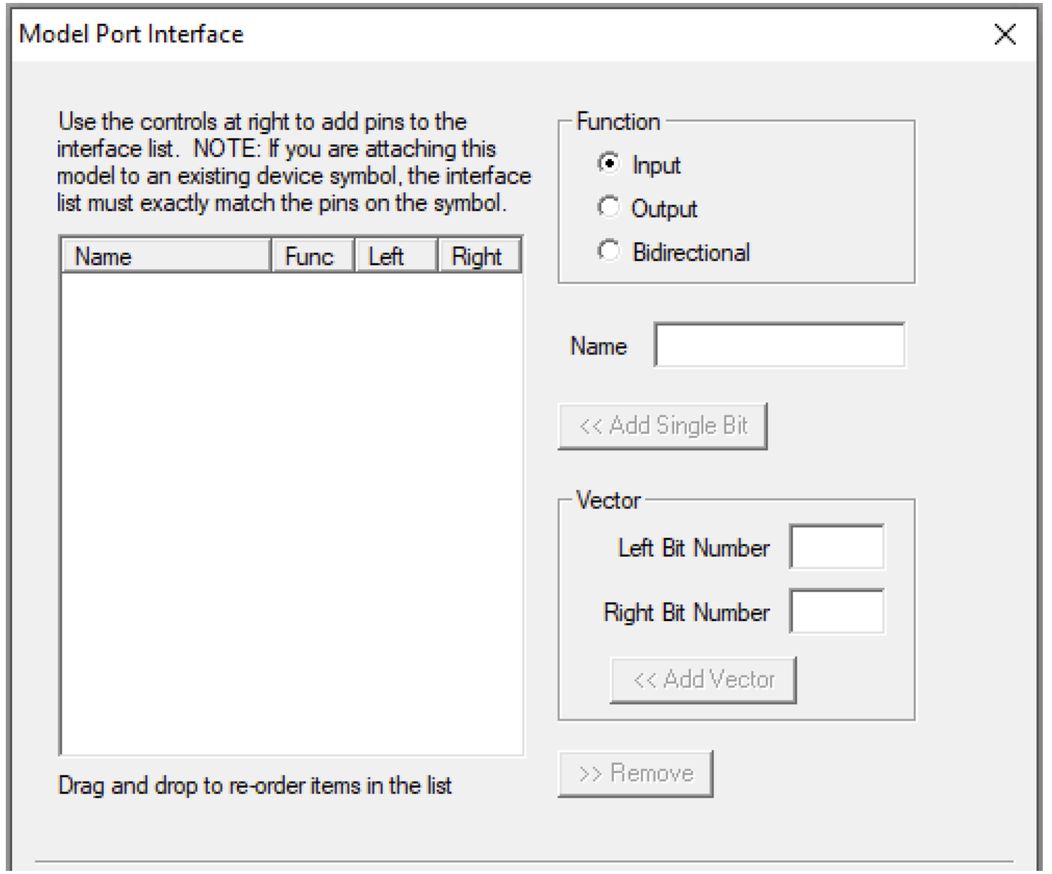
2. **Source:** `Create…`, **Destination:** `Open the…` . Click `Next` . Select `VHDL`, Enter name `AND1INV`.

P2
VHDL

Creating a AND1INV model



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

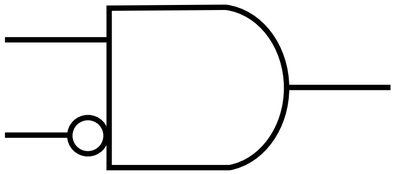


Tutorial

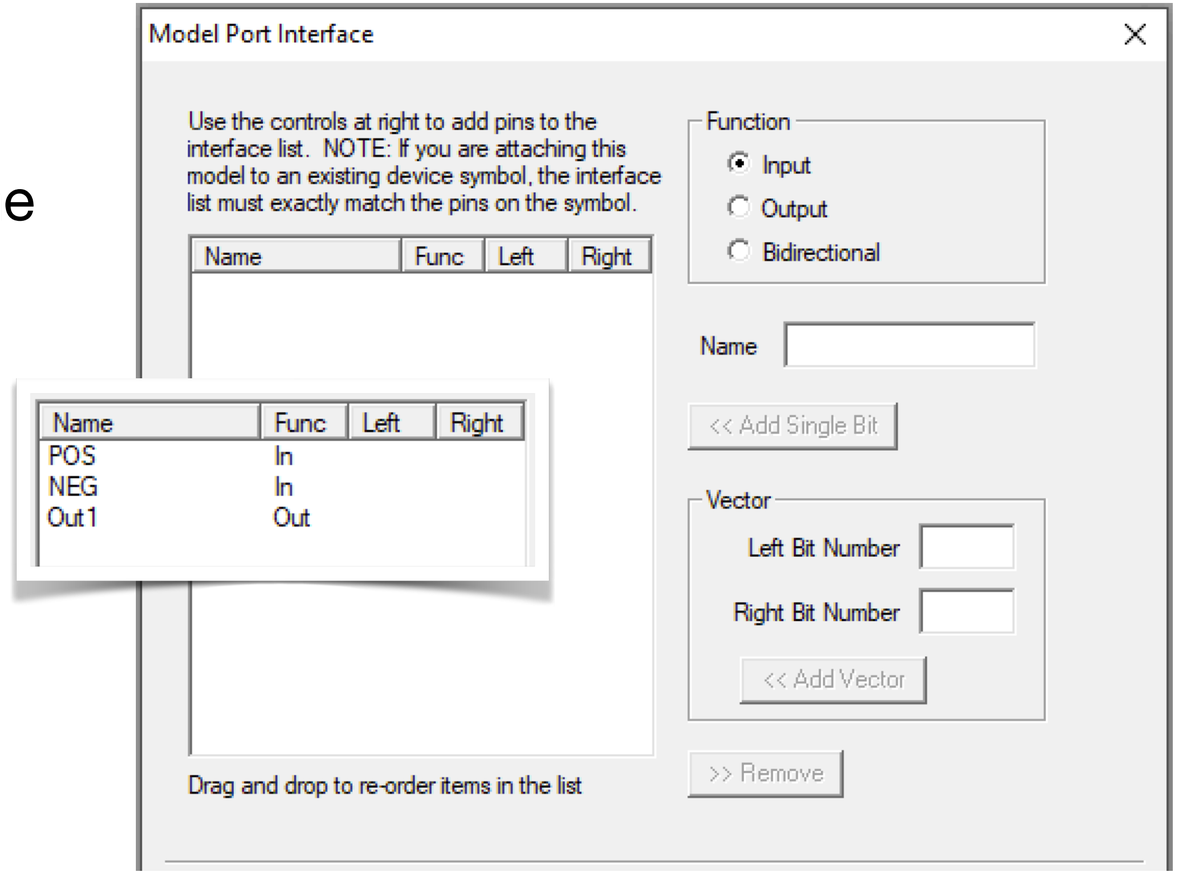
3. **Type the `Name` and select the `Function` accordingly, then press `Add Single Bit`, click `Finish` to create the model file.**

P2
VHDL

Creating a AND1INV model



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

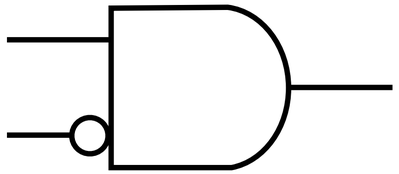


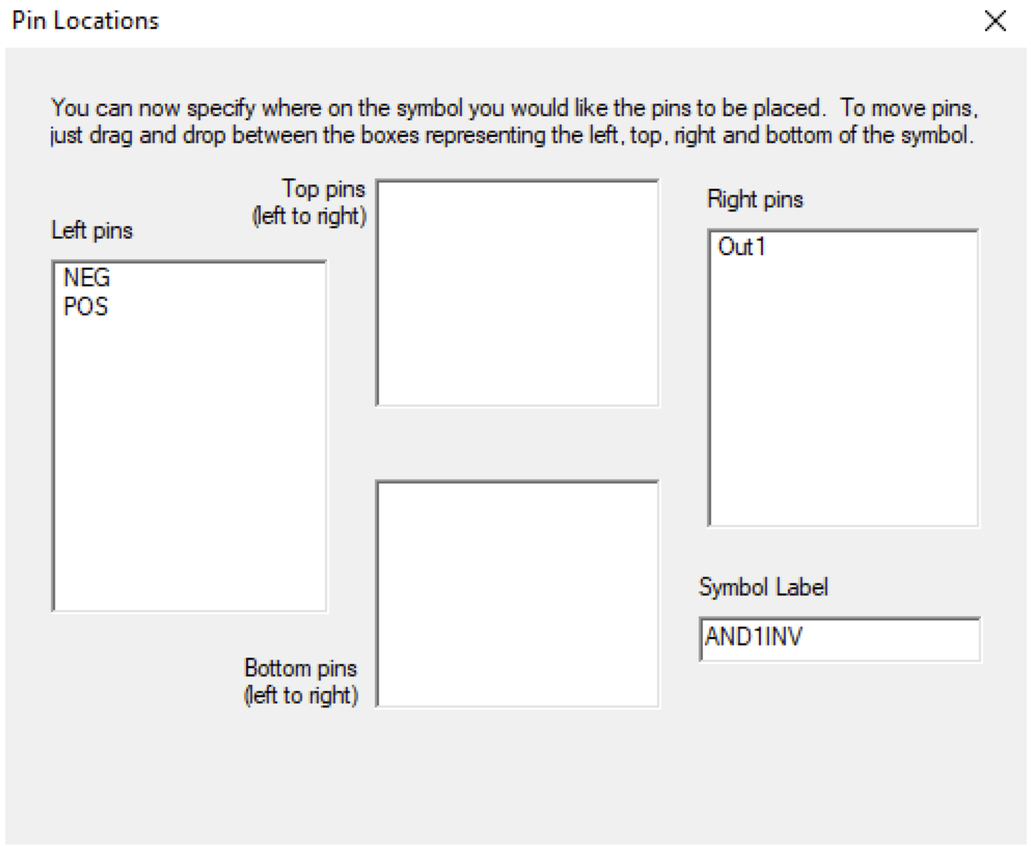
3. **Type the** `Name` **and select the** `Function` **accordingly, then press** `Add Single Bit`, click `Finish` to create the model file.

Tutorial

P2
VHDL

Creating a AND1INV model



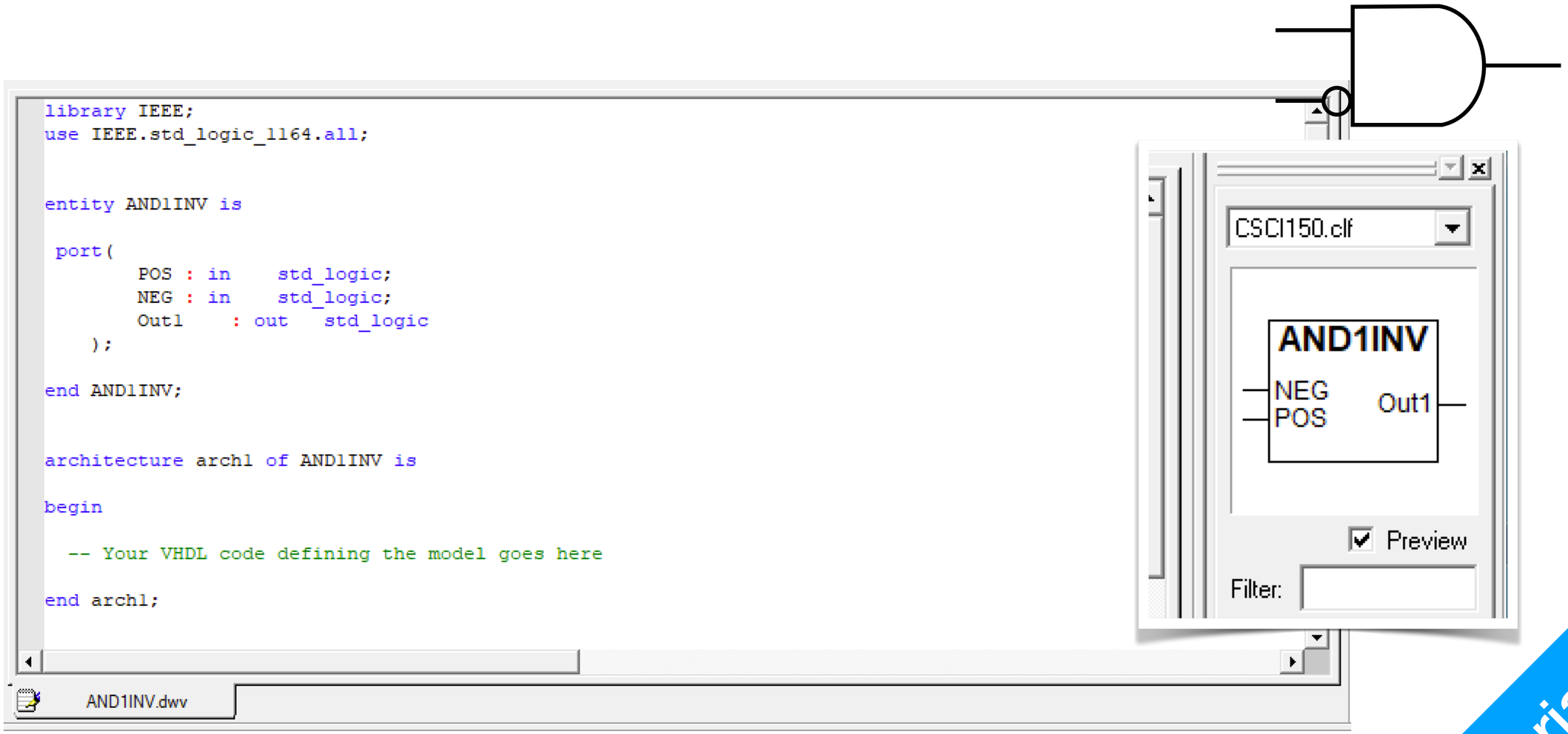


4. The programme will ask you for `Pin Location` assignment. Just click `Next`.

Tutorial

P2
VHDL

Creating a AND1INV model



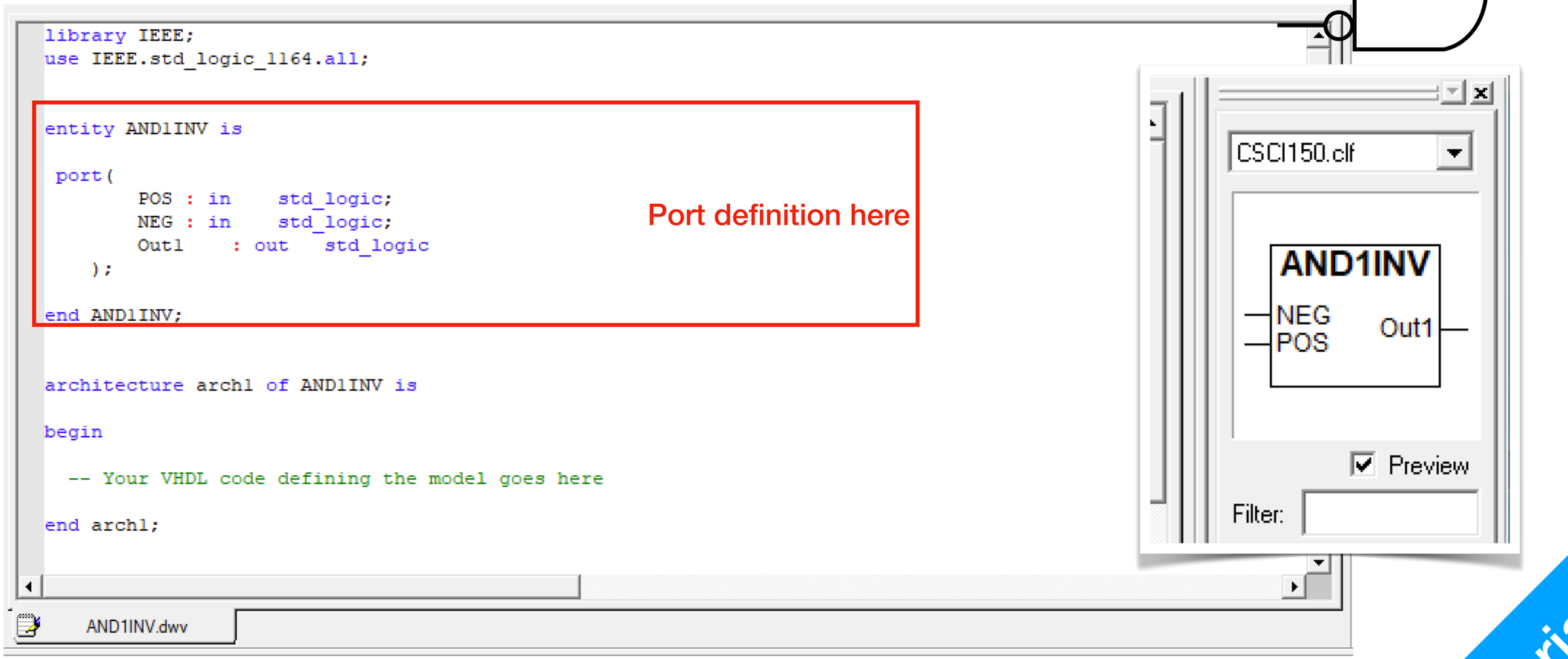
Tutorial

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

  -- Your VHDL code defining the model goes here

end arch1;
```

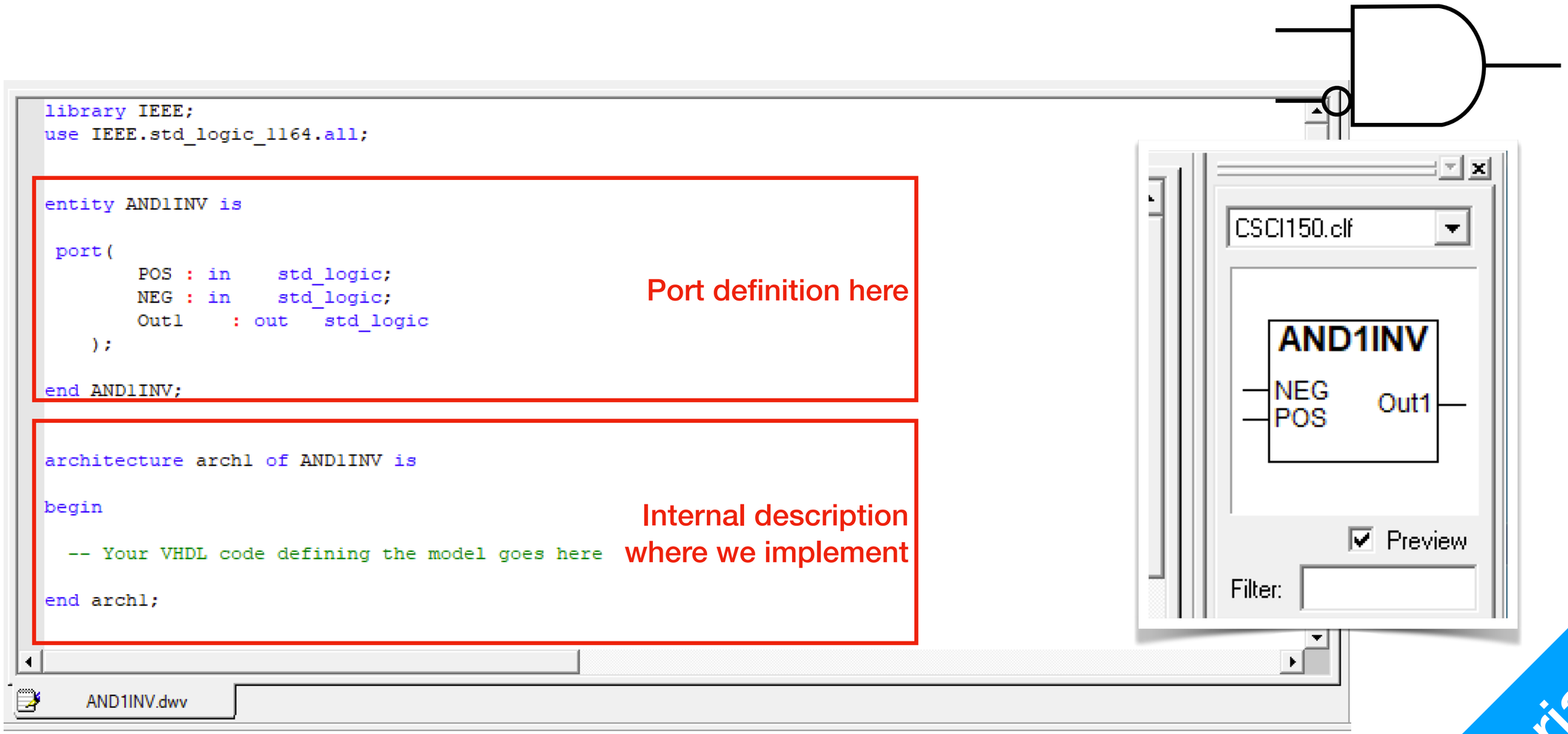
Port definition here



Tutorial

P2
VHDL

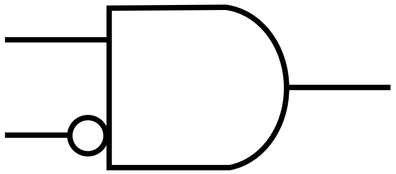
Creating a AND1INV model



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

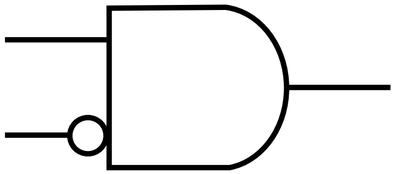
Boolean Operators

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_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

Boolean Operators

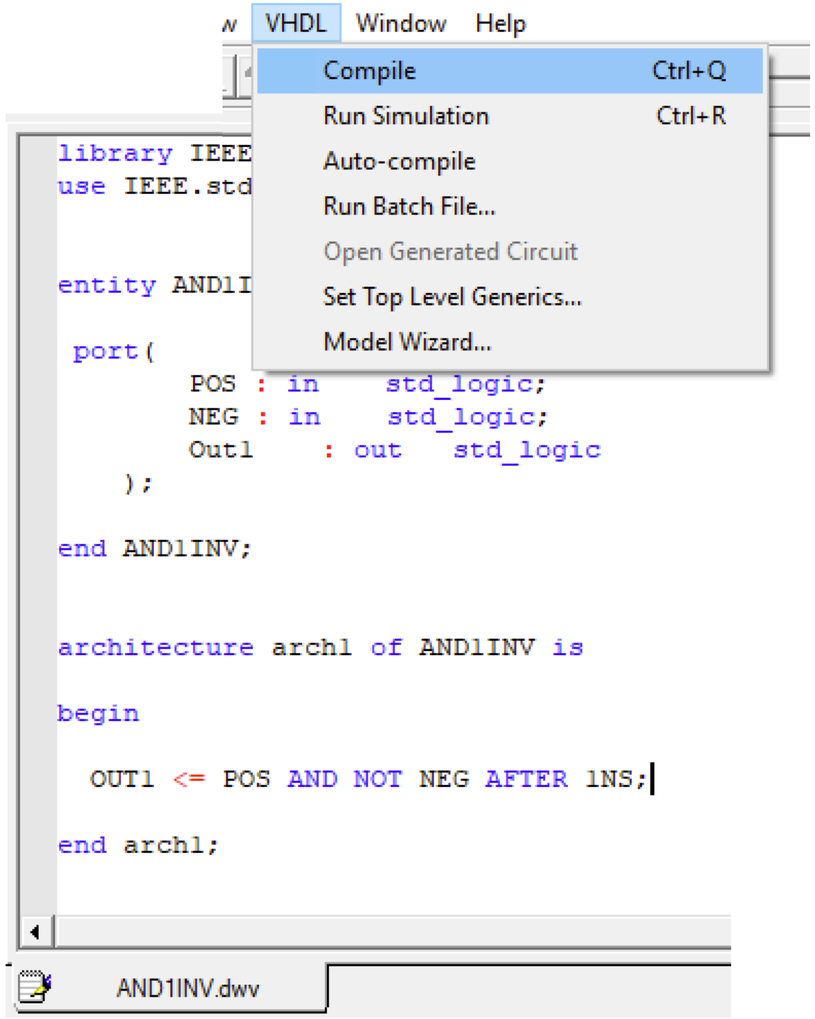
Things do not happen simultaneously

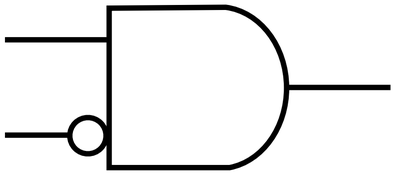
5. **Type:** `OUT1 <= POS AND NOT NEG AFTER 1NS;` **This is the Boolean description of the** `OUT1` **port**

Tutorial

P2
VHDL

Creating a AND1INV model





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

Transferring

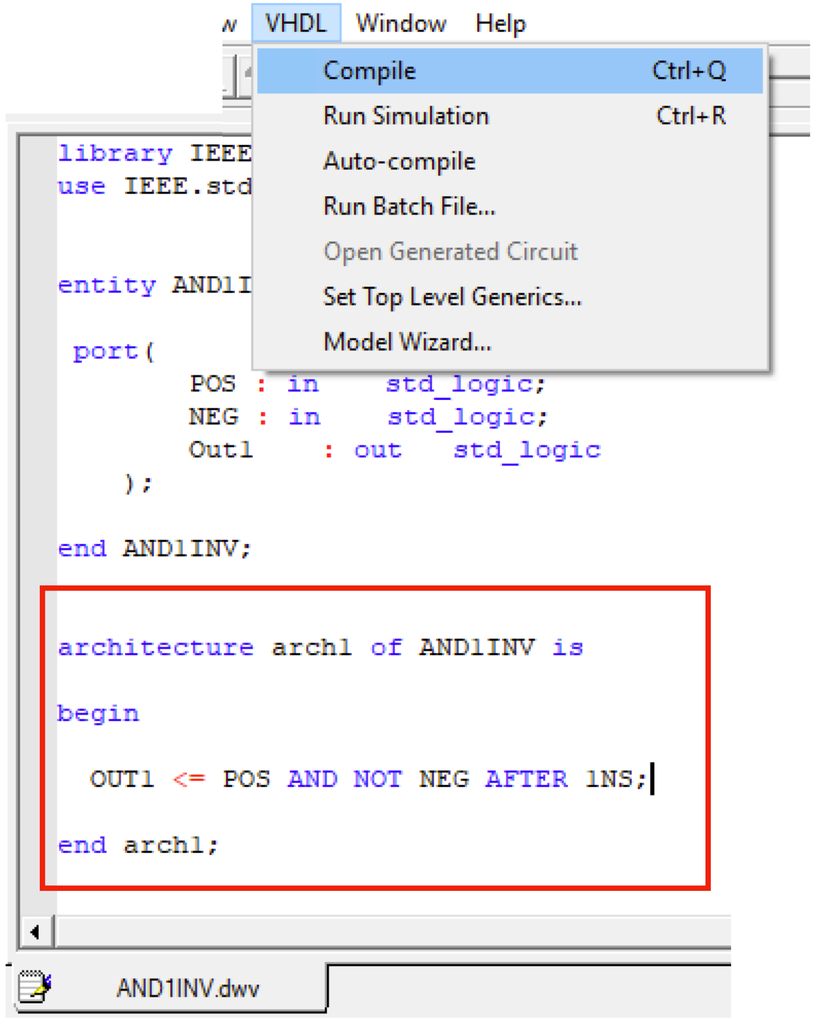
Boolean Operators

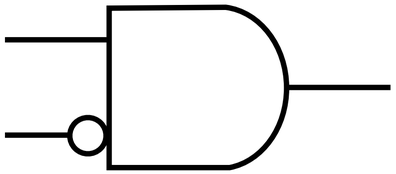
Things do not happen simultaneously

Tutorial

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

Creating a AND1INV model





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

Transferring

Boolean Operators

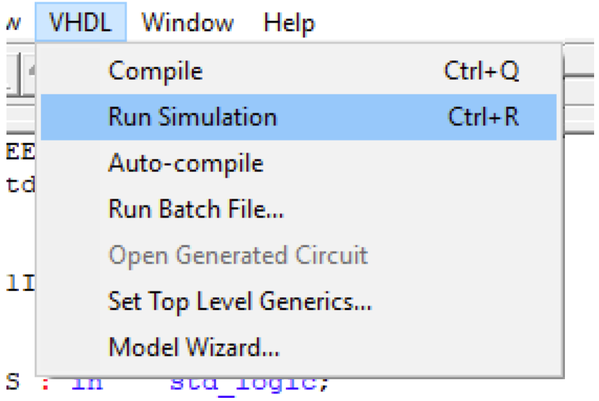
Things do not happen simultaneously

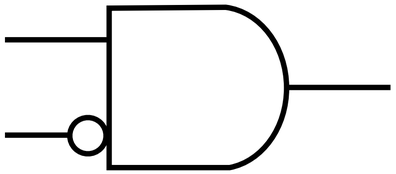
Tutorial

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

P2
VHDL

Run Simulation





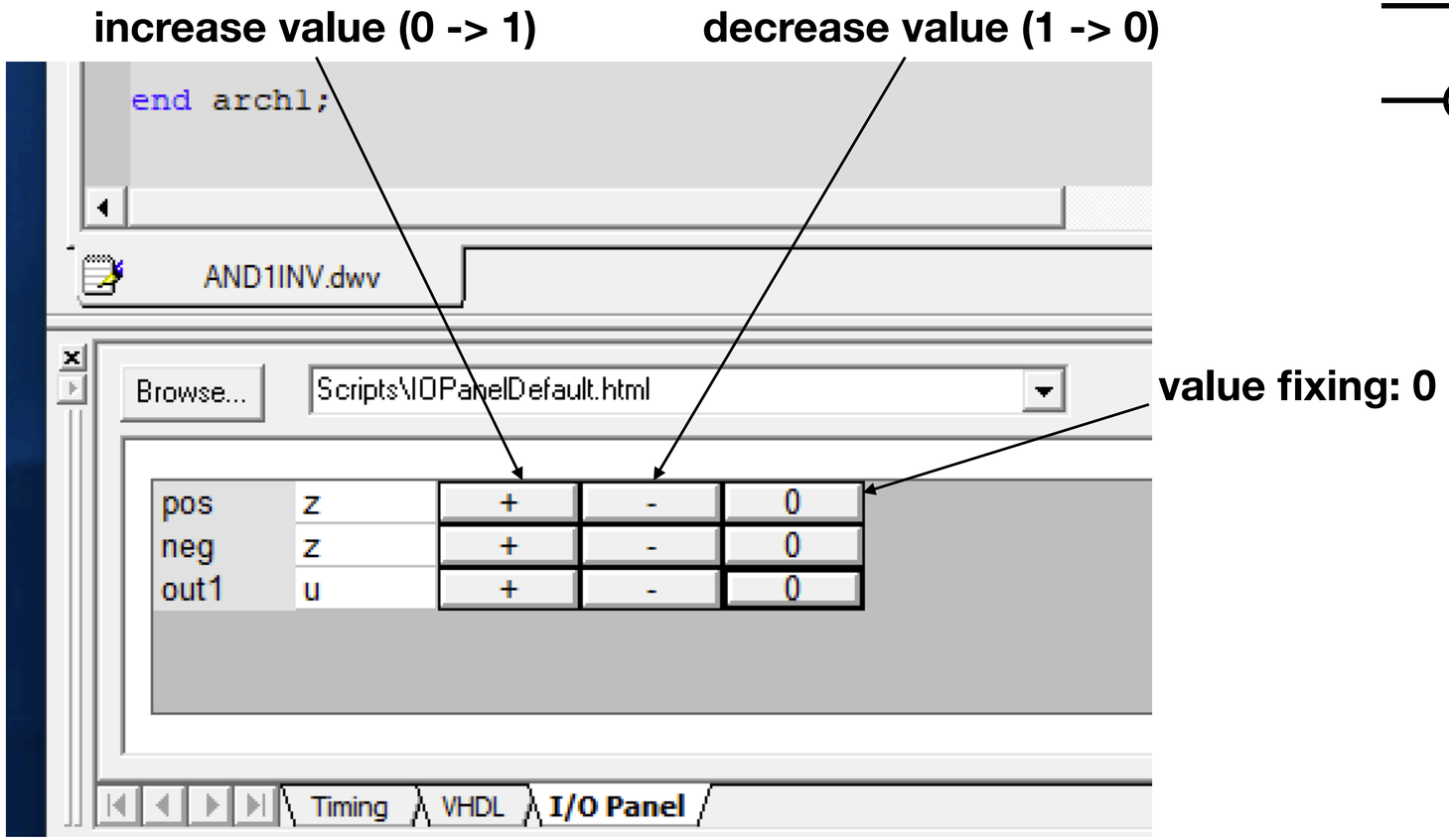
I/O Page

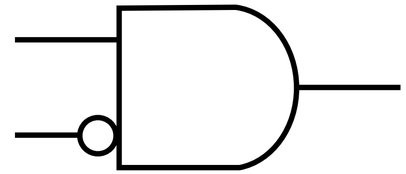
Run

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



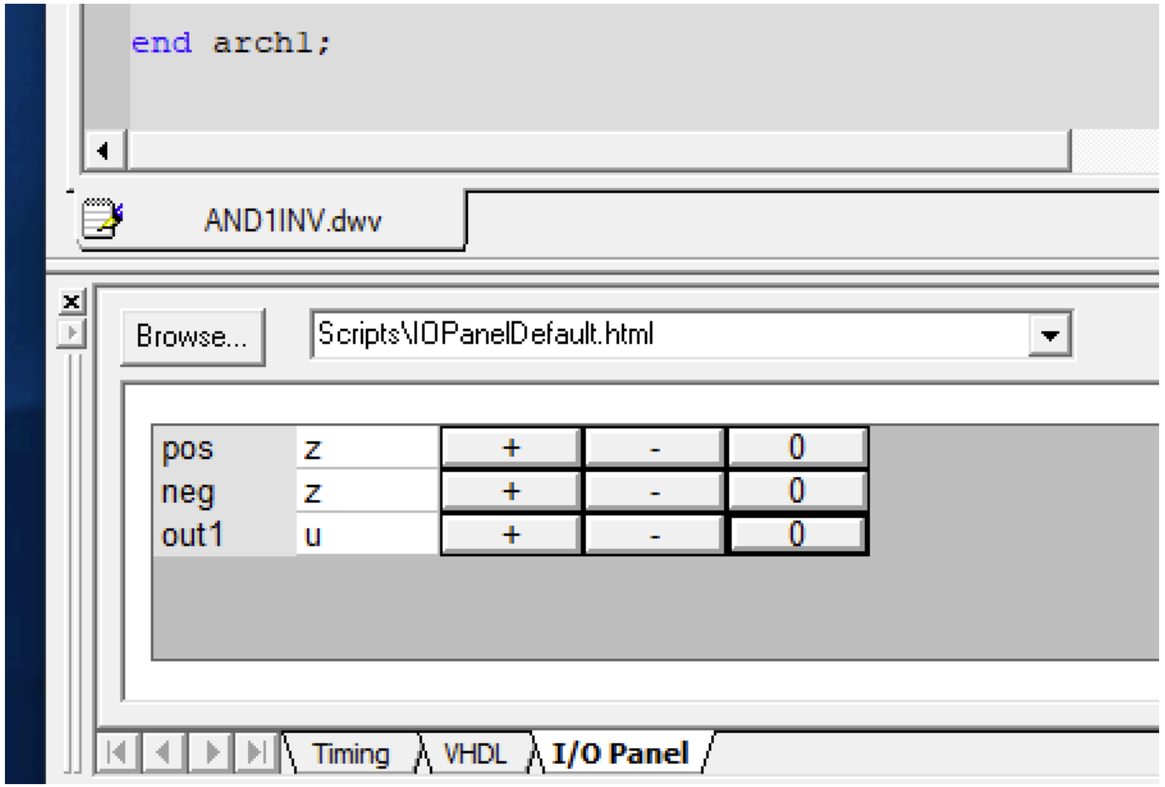


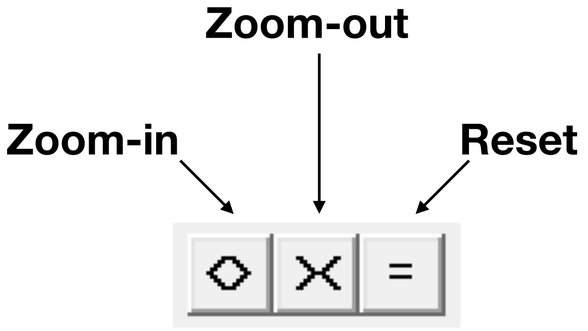
7. Play with the buttons in the I/O panel

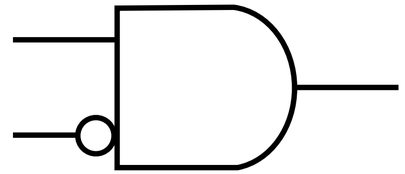
Tutorial

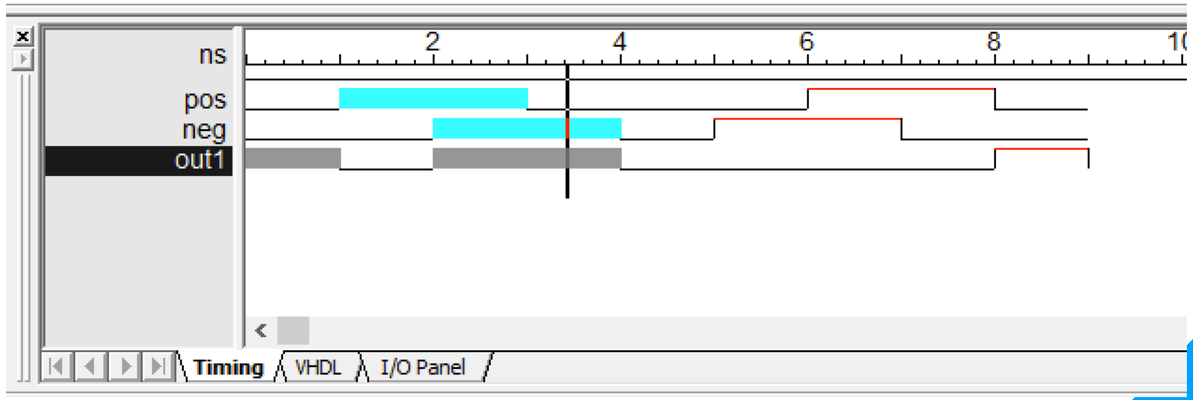
P2
VHDL

Run Simulation







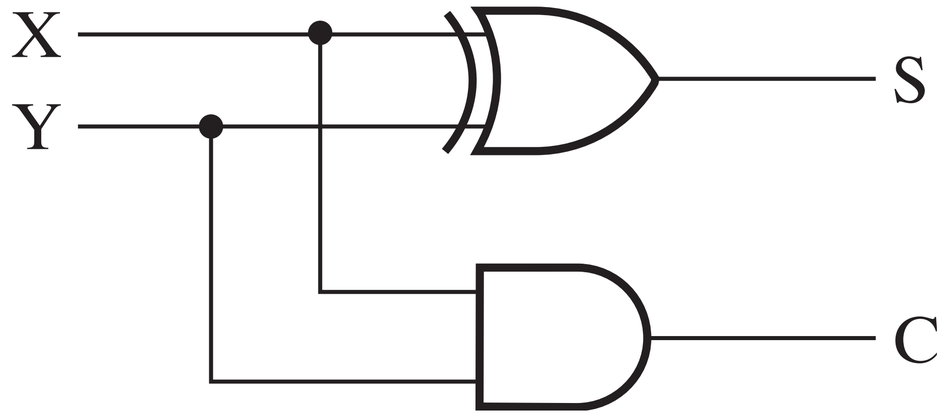


Tutorial

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`

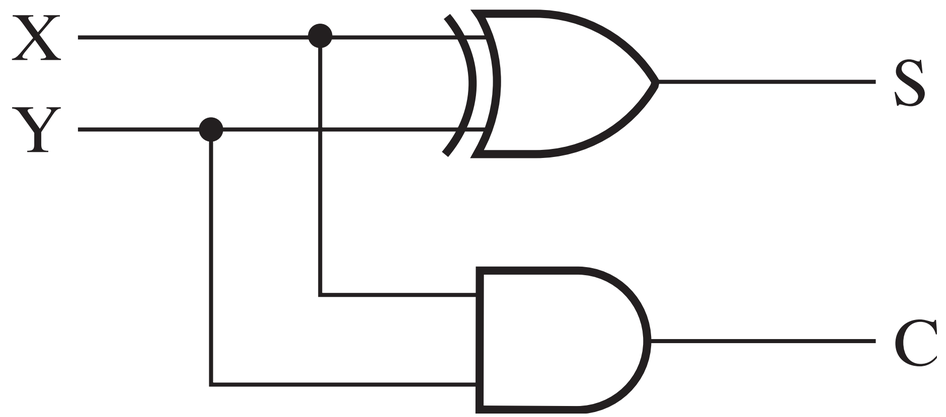


Practice

P2
VHDL

Exe1: 1-bit Half Adder

```
architecture arch1 of HalfAdder is

begin

     S <= X XOR Y;

     C <= X AND Y;

end arch1;
```



Practice