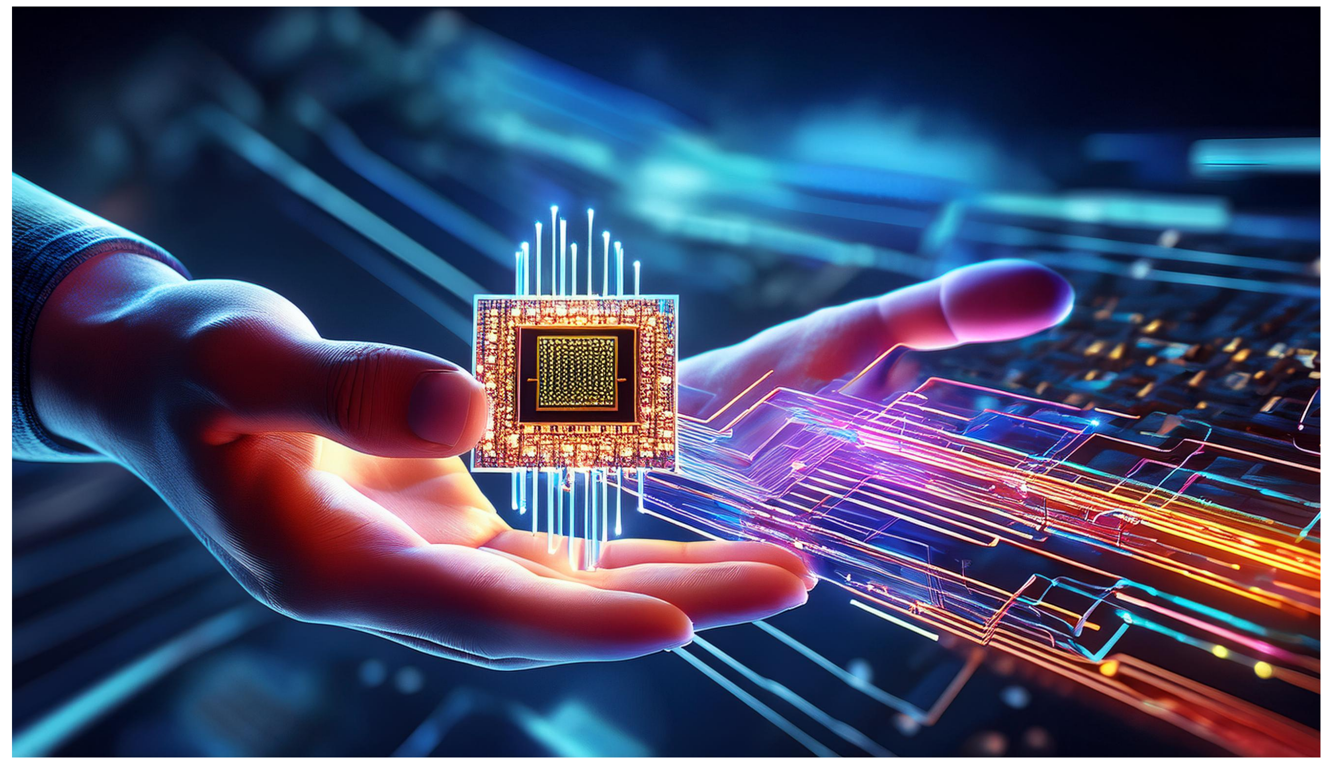
CSCI 250
Introduction to Computer Organisation
Lecture 1: Beyond Integer Arithmetics III



Jetic Gū

2024 Fall Semester (S3)

Overview

- Focus: Course Introduction
- Architecture: Logical Circuits
- Textbook: LW Chapter 7
- Core Ideas:
  1. VHDL, Binary Adder
  2. Lab 1 Part 2: Adder-Subtractor

VHSIC Hardware Description Language

What is HDL

- Programming Languages: e.g. Python, C, C++
  - Compiles/Interprets to machine code
  - Executed sequentially by a CPU
- Hardware Description Language: VHDL, Verilog
  - Describes hardware logic, how gates are connected
  - Loaded onto FPGA board, fully parallel (because it's a real circuit)

P1
VHDL

HDL IDE Platforms

- AMD Xilinx
    - [Chipset] Spartan 6-: ISE Suite
    - [Chipset] Spartan 7+: Vivado
- Intel Altera FPGA: Quartus Prime
- This is the industry standard, not as easy to get into
- Future CSCI250? For now, we’ll use LogicWorks

Technical

P1
VHDL

Previous: Register Transfer Operations (VHDL Syntax)

	Operator	Example
Assignment	`<=`	`ax` **`<=`** `12h`
Reg. Transfer	`<=`	`ax` **`<=`** `bx`
Addition	`+`	`ax` **`+`** `bx`
Subtraction	`-`	`ax` **`-`** `bx`
Shift Left	`sll`	`ax` **`sll`** `2`
Shift Right	`srl`	`ax` **`srl`** `2`

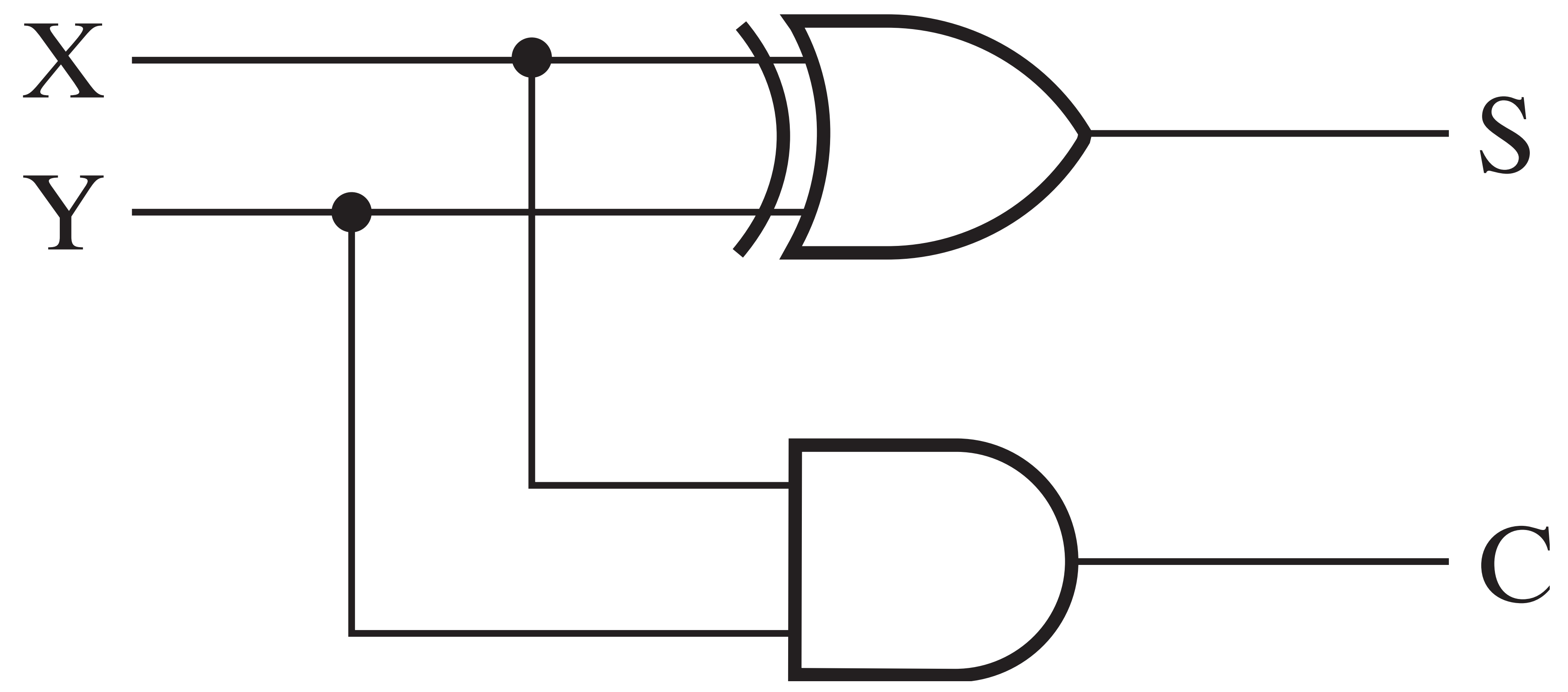
	Operator	Example
Bitwise AND	`and`	`ax` **`and`** `bx`
Bitwise OR	`or`	`ax` **`or`** `bx`
Bitwise NOT	`not`	**`not`** `ax`
Bitwise XOR	`xor`	`ax` **`xor`** `bx`
Vectors	`ax(3 down to 0)`	`ax(3 down to 0)`
Concatenate	`&`	`ax(7 down to 4)` **`&`**`ax(3 down to 0)`

Review

P1
VHDL

Previous: 1-bit Half Adder

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

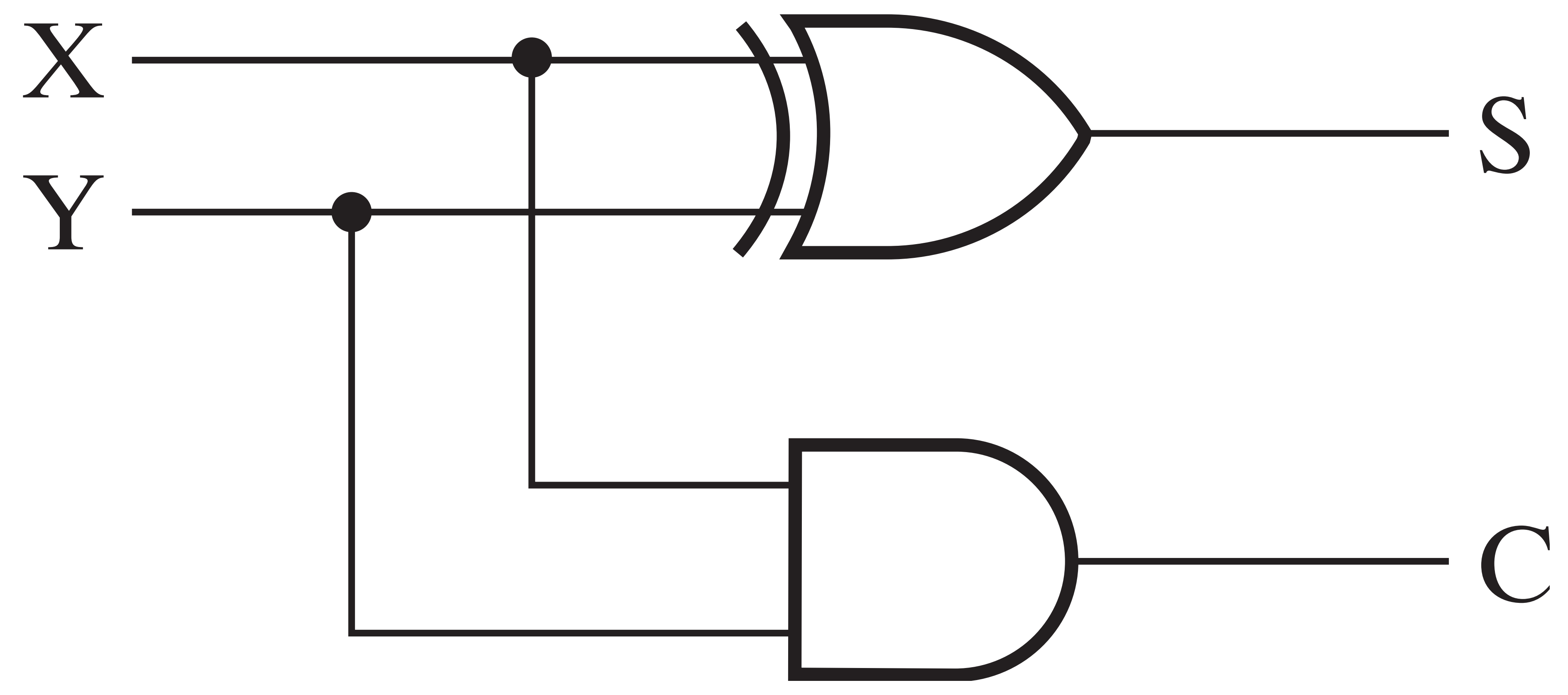


Practice

P1
VHDL

Previous: 1-bit Half Adder

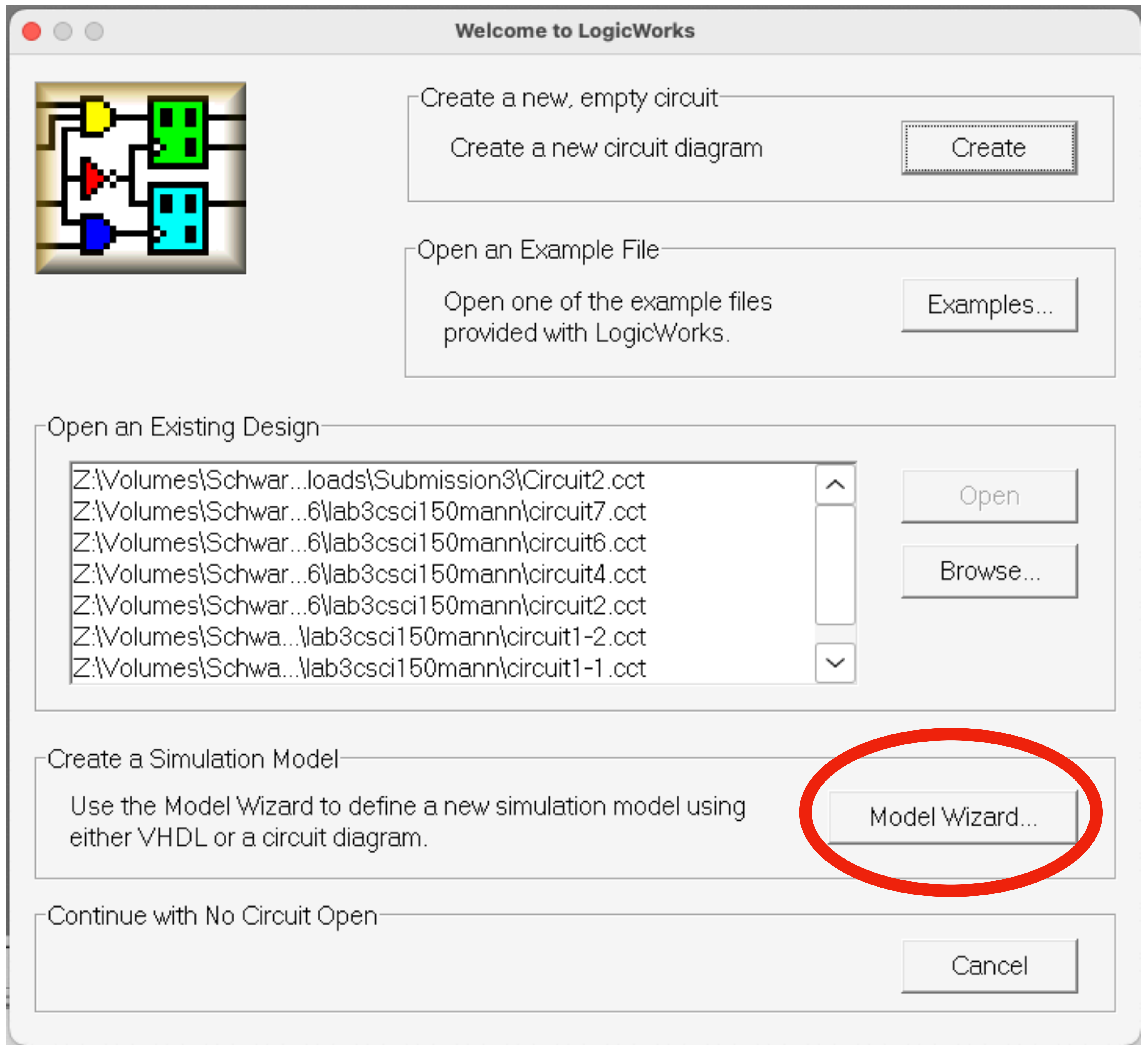
```
architecture arch1 of HalfAdder is

begin

    S <= X XOR Y;

    C <= X AND Y;

end arch1;
```

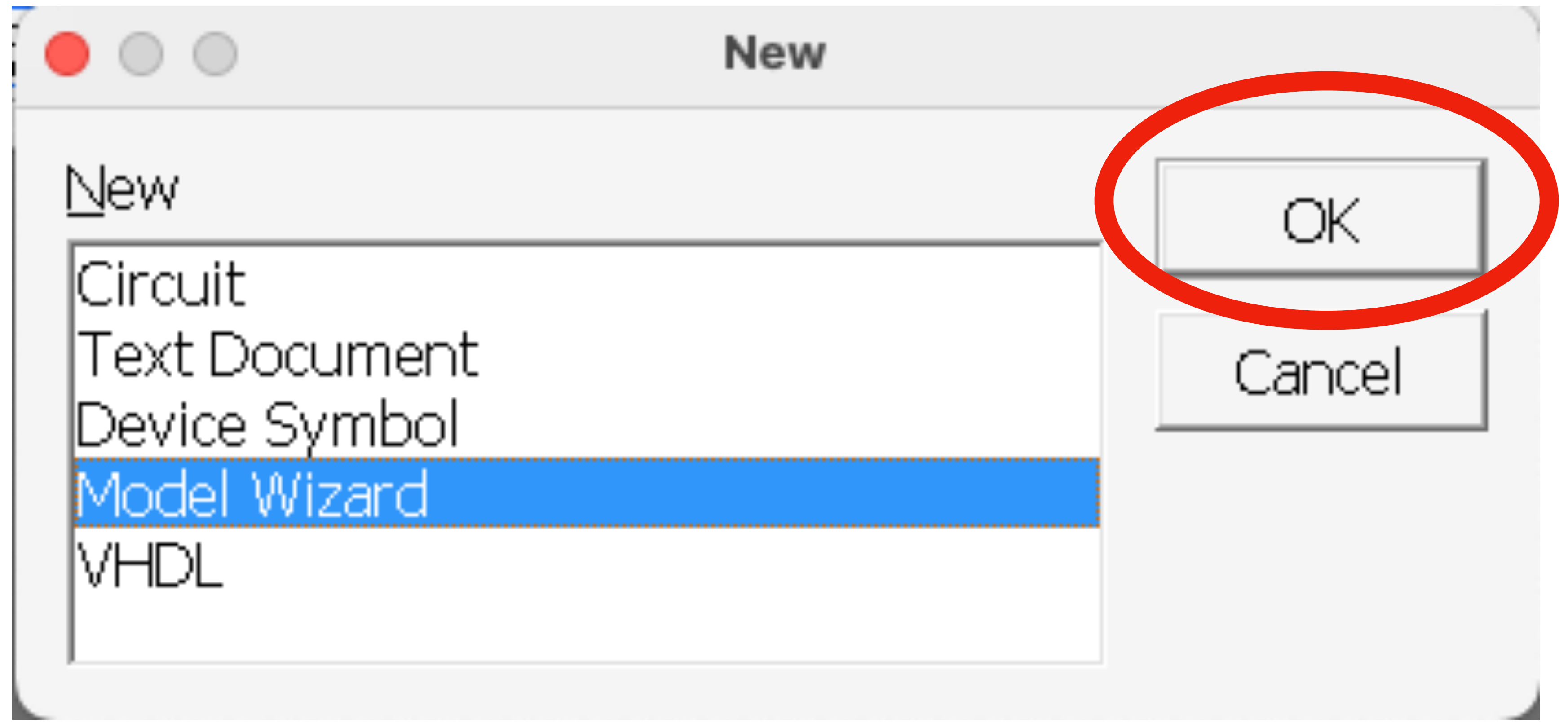


Practice

P1
VHDL

1bit Binary Adder


Welcome to LogicWorks
Create a new, empty circuit
Create a new circuit diagram
Create
Open an Example File
Open one of the example files provided with LogicWorks.
Examples...
Open an Existing Design
Z:\Volumes\Schwar...loads\Submission3\Circuit2.cct
Z:\Volumes\Schwar...6\lab3csci150mann\circuit7.cct
Z:\Volumes\Schwar...6\lab3csci150mann\circuit6.cct
Z:\Volumes\Schwar...6\lab3csci150mann\circuit4.cct
Z:\Volumes\Schwar...6\lab3csci150mann\circuit2.cct
Z:\Volumes\Schwa...\lab3csci150mann\circuit1-2.cct
Z:\Volumes\Schwa...\lab3csci150mann\circuit1-1.cct
Open
Browse...
Create a Simulation Model
Use the Model Wizard to define a new simulation model using either VHDL or a circuit diagram.
Model Wizard...
Continue with No Circuit Open
Cancel

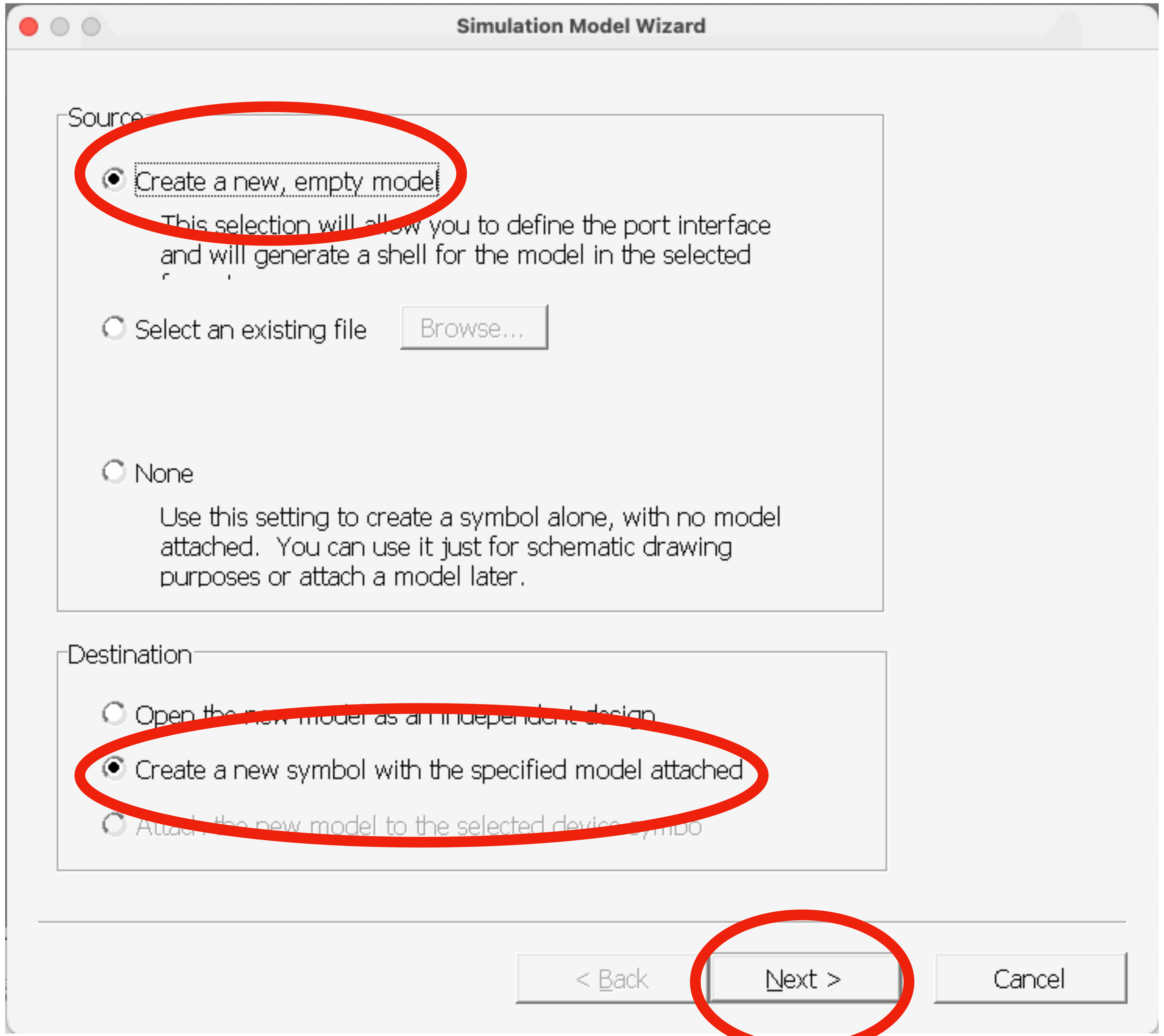


Tutorial 1

1. **Select** `Model Wizard...` **from** `Welcome`**, or from** `File->New`

P1
VHDL

1bit Binary Adder


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

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]

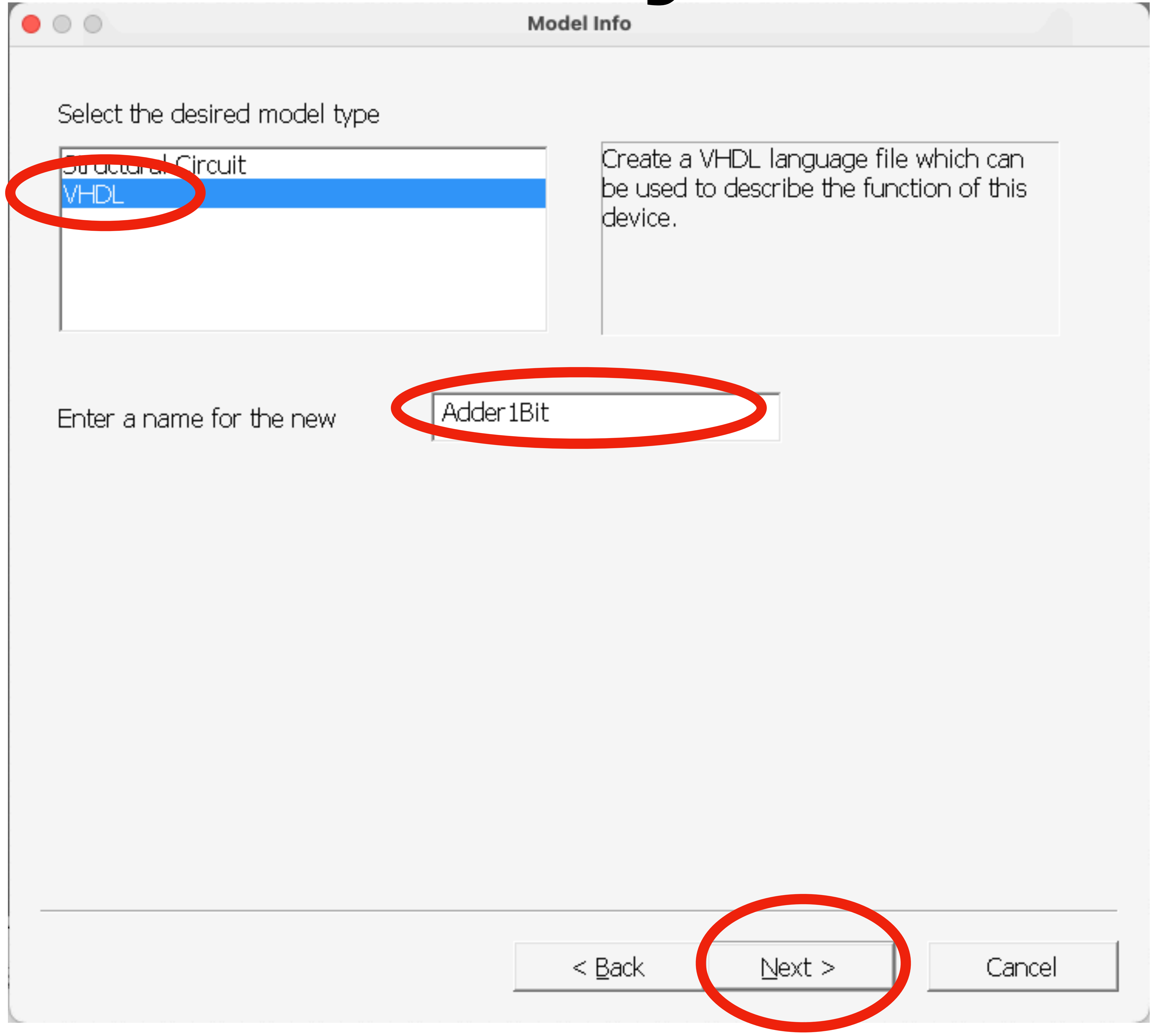
Attach the new model to the selected device symbo

2. Select `Create a new, empty model`; Select `Create a new symbol with the specified model attached`; Select `Next`

Tutorial 1

P1
VHDL

1bit Binary Adder



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

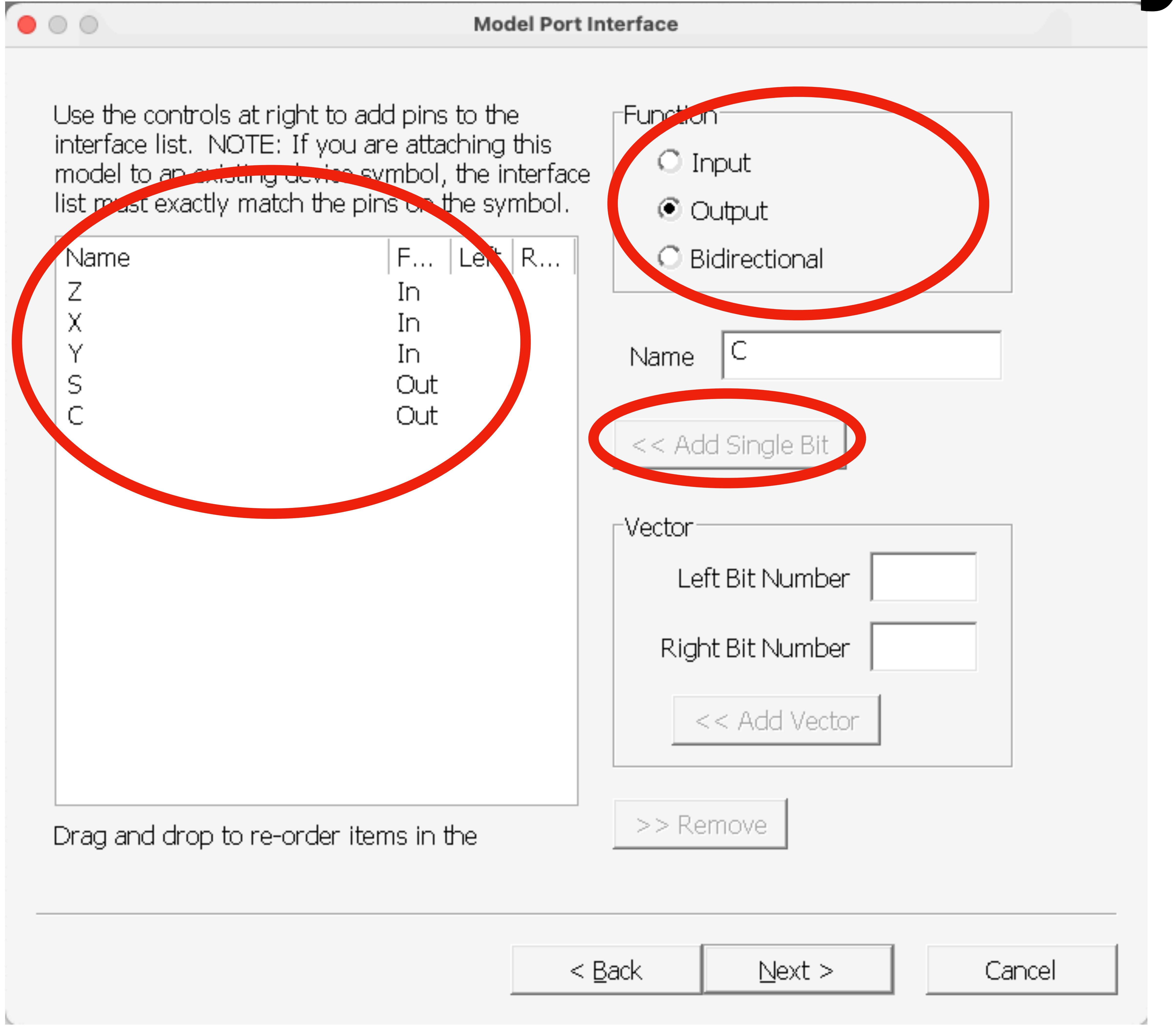
Adder1Bit

3. Select `VHDL`; Type in name `Adder1Bit`, the name cannot contain whitespace; Select `Next`

Tutorial 1

P1
VHDL

1bit Binary Adder


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	F...	Left	R...
Z	In		
X	In		
Y	In		
S	Out		
C	Out		

Function
- ( ) Input
- (●) Output
- ( ) Bidirectional

Name: C

Drag and drop to re-order items in the

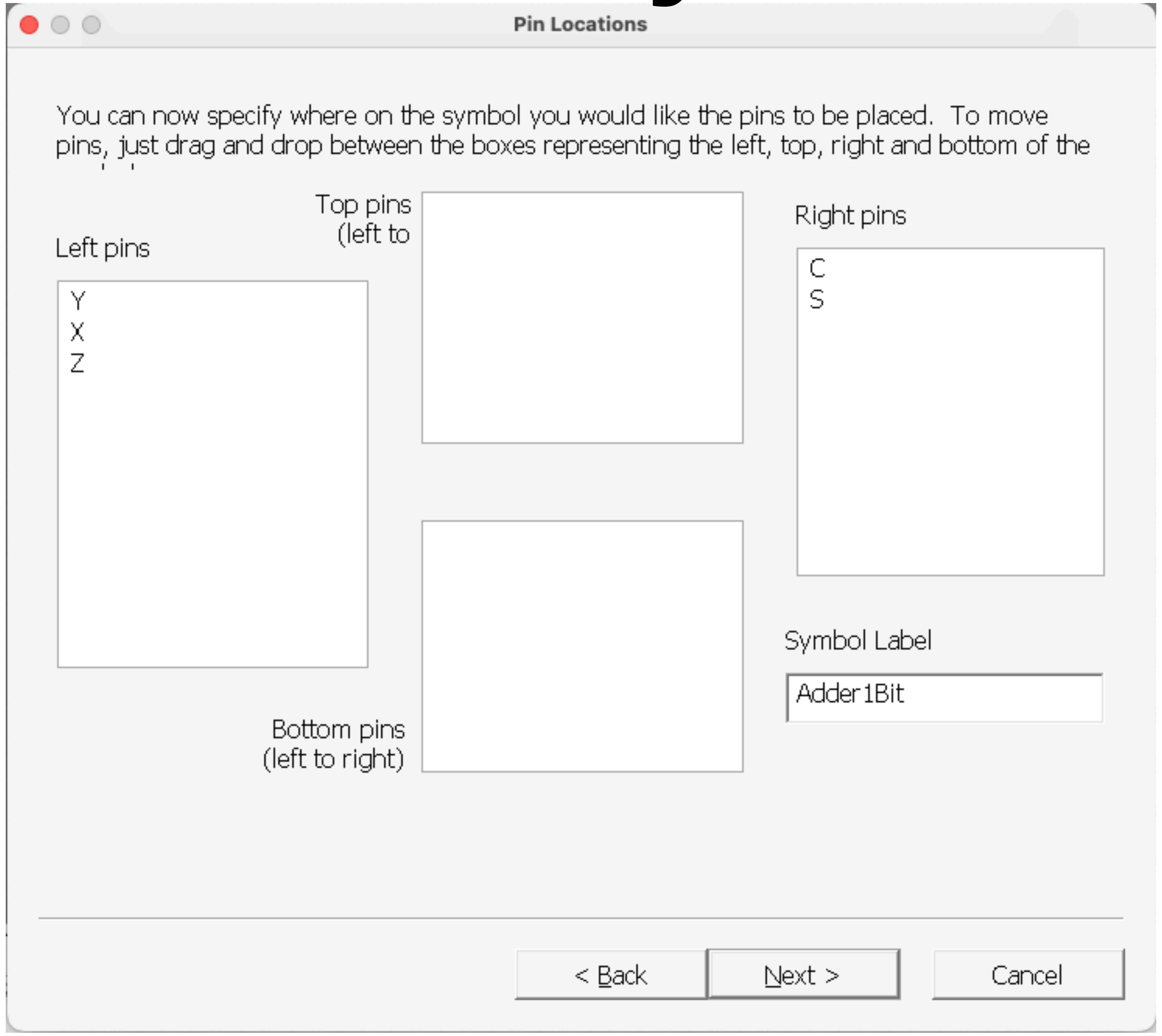
Name	Func	Left	Right
X	In		
Y	In		
Z	In		
S	Out		
C	Out		

Tutorial 1

4. Use `Function`, `Name`, and `<< Add Single Bit` to include `XYZSC` in the list of pins; Select `Next`

P1
VHDL

1bit Binary Adder


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

Left pins
Y
X
Z

Top pins
(left to

Right pins
C
S

Bottom pins
(left to right)

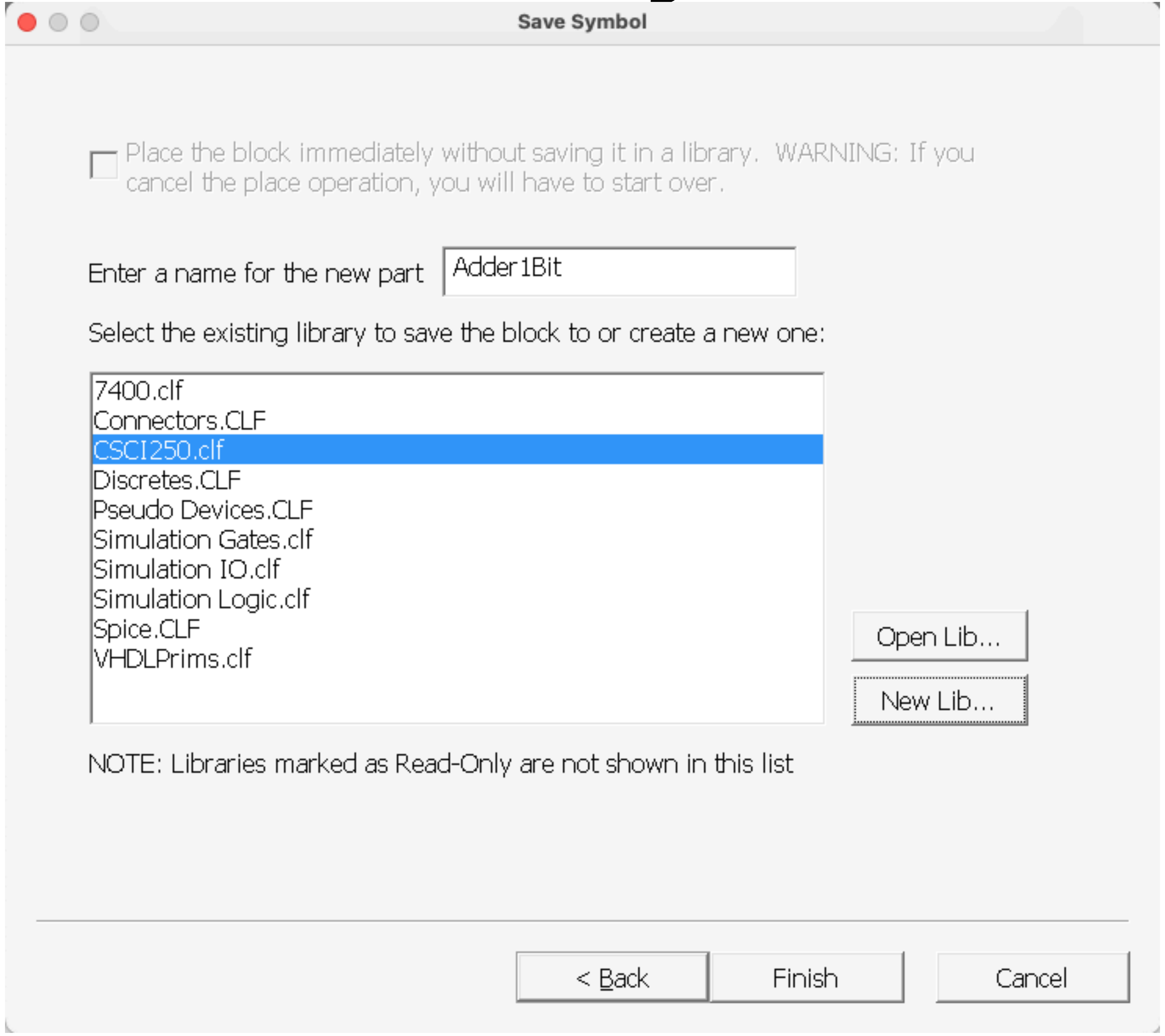
Symbol Label
Adder1Bit

5. Select `Next`

Tutorial 1

P1
VHDL

1bit Binary Adder


Save Symbol

☐ Place the block immediately without saving it in a library. WARNING: If you cancel the place operation, you will have to start over.

Enter a name for the new part: Adder1Bit

Select the existing library to save the block to or create a new one:

7400.clf
Connectors.CLF
CSCI250.clf
Discretes.CLF
Pseudo Devices.CLF
Simulation Gates.clf
Simulation IO.clf
Simulation Logic.clf
Spice.CLF
VHDLPrims.clf

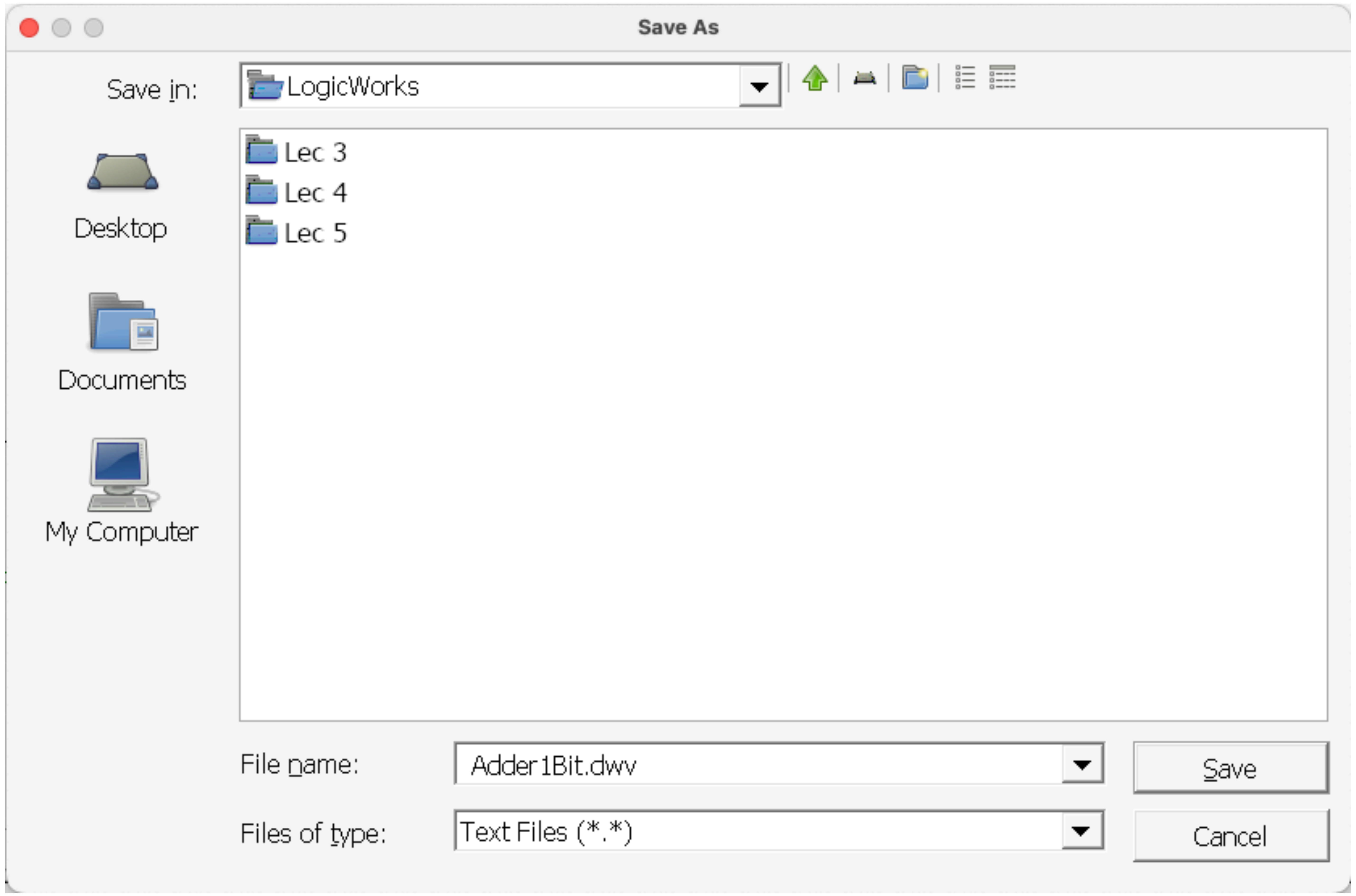
NOTE: Libraries marked as Read-Only are not shown in this list

6. Create a library called `CSCI250`; Select `Finish`

Tutorial 1

P1
VHDL

1bit Binary Adder



File name: Adder1Bit.dwv

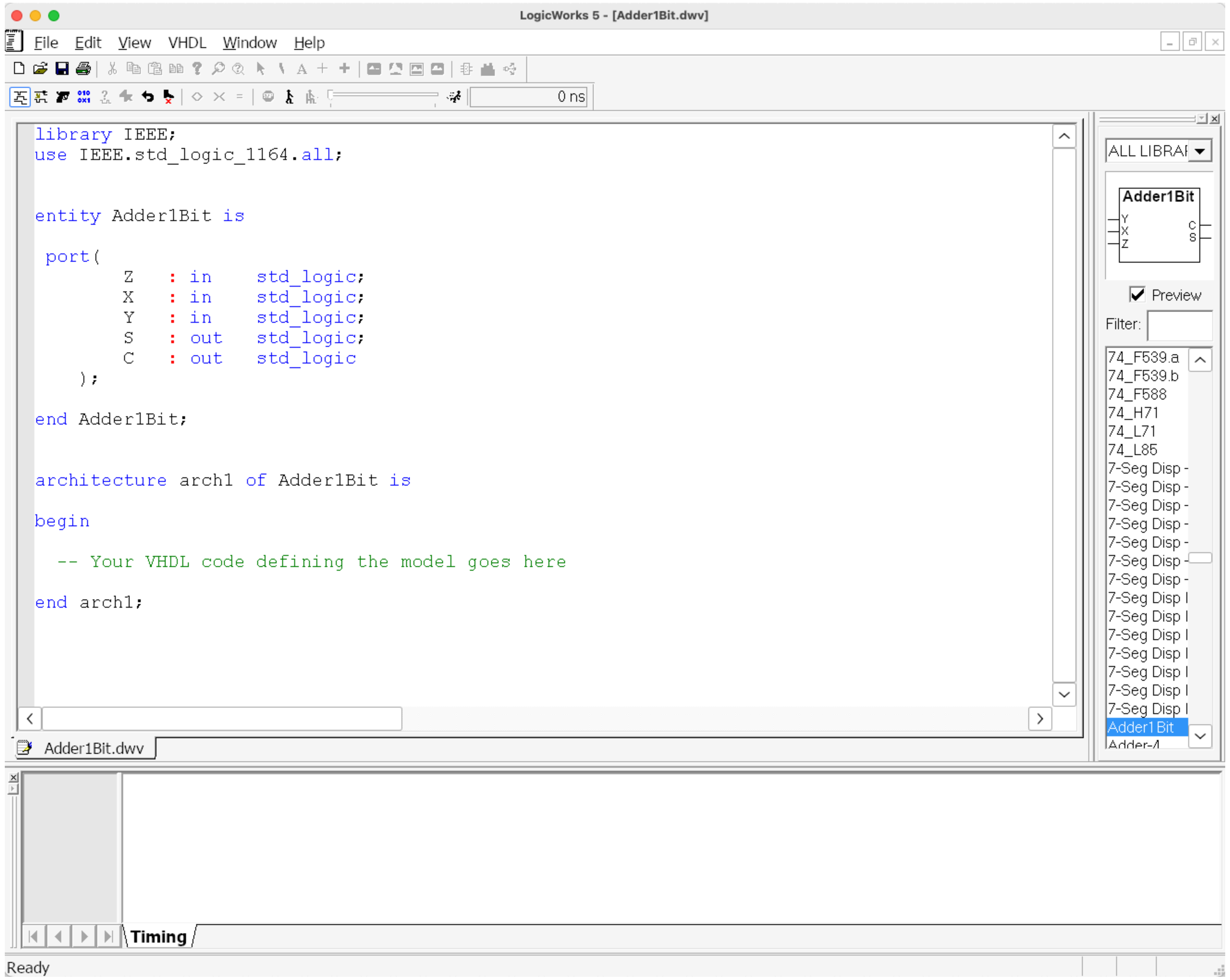
Files of type: Text Files (*.*)

7. **Save** `Adder1Bit.dwv`;

Tutorial 1

P1
VHDL

1bit Binary Adder


LogicWorks 5 - [Adder1Bit.dwv]

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


entity Adder1Bit is

 port(
        Z   : in    std_logic;
        X   : in    std_logic;
        Y   : in    std_logic;
        S   : out   std_logic;
        C   : out   std_logic
    );

end Adder1Bit;


architecture arch1 of Adder1Bit is

begin

  -- Your VHDL code defining the model goes here

end arch1;
```

8. This is what your VHDL code looks like

Tutorial 1

P1
VHDL

1bit Binary Adder

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

entity Adder1Bit is

 port(
       Z    : in std_logic;
       X    : in std_logic;
       Y    : in std_logic;
       S    : out    std_logic;
       C    : out    std_logic
  );

end Adder1Bit;


architecture arch1 of Adder1Bit is

begin

  -- Your VHDL code defining the model goes here

end arch1;
```

- In VHDL, every expressions should end with semi-colon unless otherwise required
- This is the library bit, just like `#include <...>` or `import` from C++/Python

Tutorial 1

8. This is what your VHDL code looks like

1bit Binary Adder

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

entity Adder1Bit is

 port(
        Z    : in std_logic;
        X    : in std_logic;
        Y    : in std_logic;
        S    : out    std_logic;
        C    : out    std_logic
   );

end Adder1Bit;

architecture arch1 of Adder1Bit is

begin

  -- Your VHDL code defining the model goes here

end arch1;
```

- This is where you define your design entity
- A design entity can be a **chip**, a **board**, or a single transistor
  We'll mostly concentrate on **chips/ boards**
- This part here defines the **interface** (I/O) of your component
- You do **NOT** need to modify this

1bit Binary Adder

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


entity Adder1Bit is

 port(
       Z    : in std_logic;
       X    : in std_logic;
       Y    : in std_logic;
       S    : out    std_logic;
       C    : out    std_logic
   );

end Adder1Bit;

architecture arch1 of Adder1Bit is

begin

  -- Your VHDL code defining the model goes here

end arch1;
```

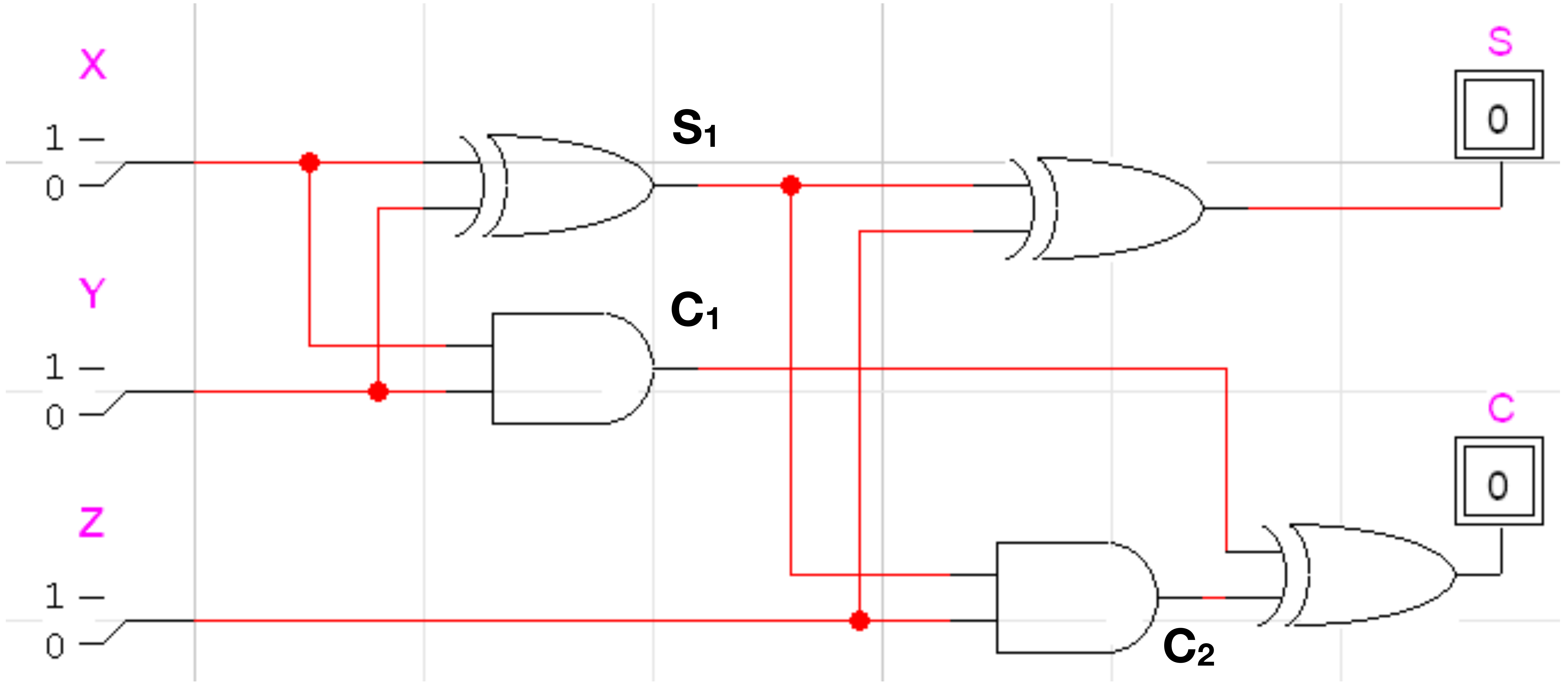
- **Concurrent Statement**
  Concurrent means parallel, there's no execution order, everything happens all at once

- This is where you will start coding

- `arch1` here is a label for this specific design of `Adder1Bit`. There might be multiple architectures that share the same IO. Important? Not to us as of right now

Tutorial 1

1bit Binary Adder



- This is a 1bit binary full adder

$S_1 = X \oplus Y; S = S_1 \oplus Z; C_1 = XY; C_2 = S_1Z; C = C_1 \oplus C_2$

P1
VHDL

1bit Binary Adder

$$S_1 = X \oplus Y; S = S_1 \oplus Z; C_1 = XY; C_2 = S_1Z; C = C_1 \oplus C_2$$

```
architecture arch1 of
Adder1Bit is

signal s1, c1, c2: std_logic;

begin

  s1 <= (x xor y);
  s <= s1 xor z;
  c1 <= x and y;
  c2 <= z and s1;
  c <= c1 xor c2;

end arch1;
```

- **Temporary labels**
  Declared before begin, variables that are neither Input nor Output
  - Use `signal`, datatype `std_logic`;
    This is for a single bit
- **Expressions**
  - Same syntax as we discussed in Register Microoperations, but in this case all labels are single bits

Tutorial 1

P1
VHDL

1bit Binary Adder

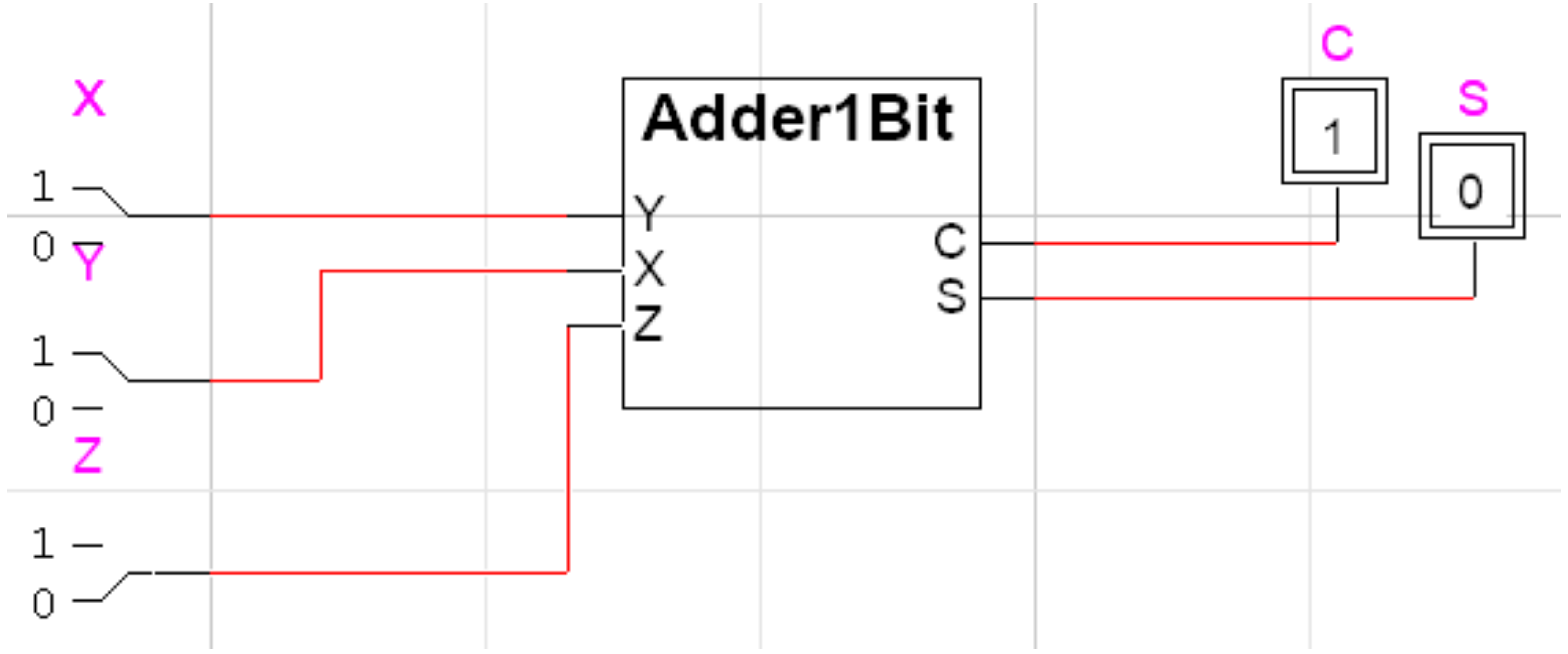
$$S_1 = X \oplus Y; S = S_1 \oplus Z; C_1 = XY; C_2 = S_1 Z; C = C_1 \oplus C_2$$

```
architecture arch1 of
Adder1Bit is

signal s1, c1, c2: std_logic;

begin

  s1 <= (x xor y);
  s <= s1 xor z;
  c1 <= x and y;
  c2 <= z and s1;
  c <= c1 xor c2;

end arch1;
```

- **Temporary labels**
  Declared before begin, variables that are neither Input nor Output
  - Use `signal`, datatype `std_logic`;
    This is for a single bit
- **Expressions**
  - Same syntax as we discussed in Register Microoperations, but in this case all labels are single bits

Tutorial 1

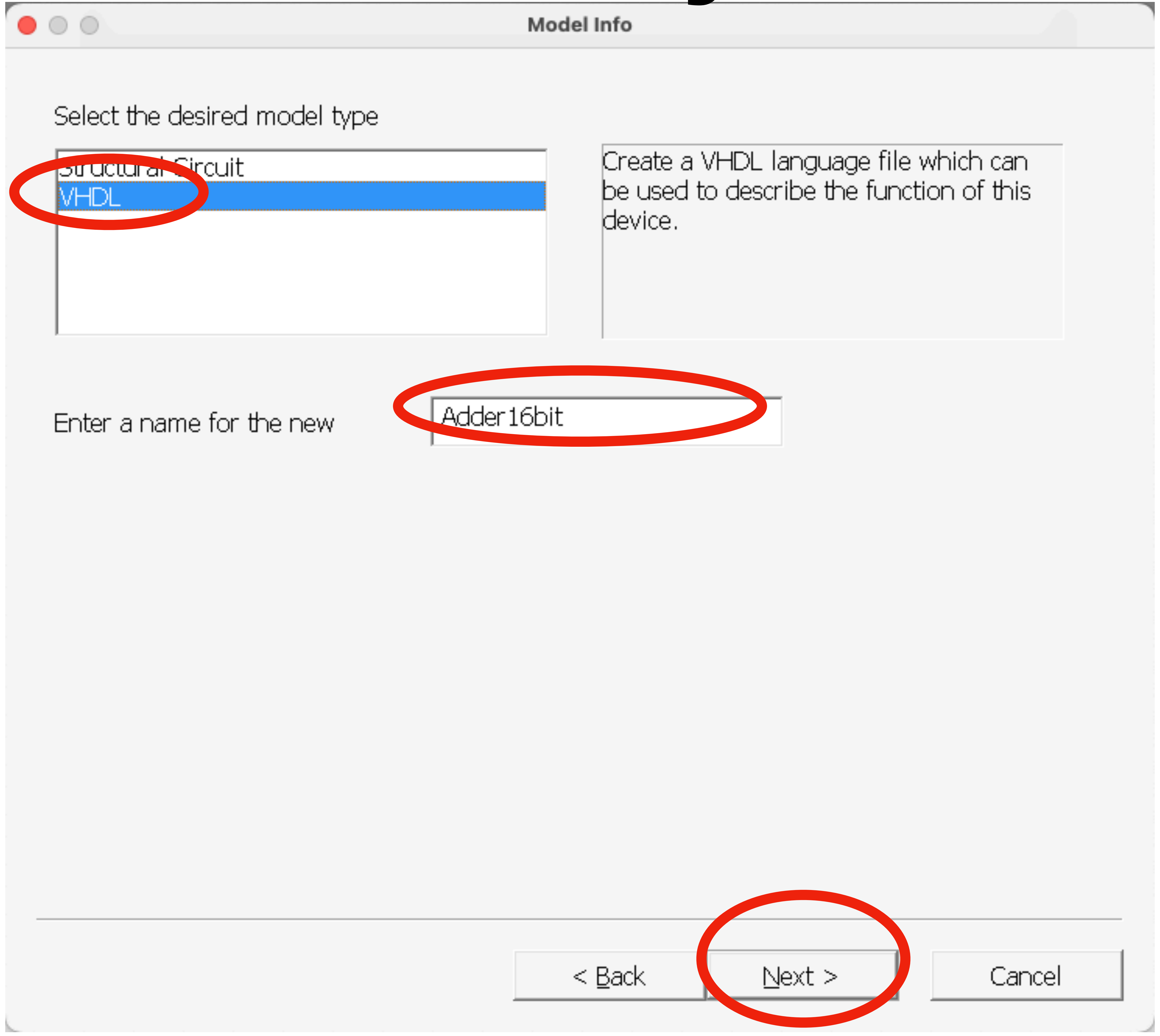
1bit Binary Adder



- Simulation: use the implemented component as just any other component

VHDL in LogicWorks
Buses

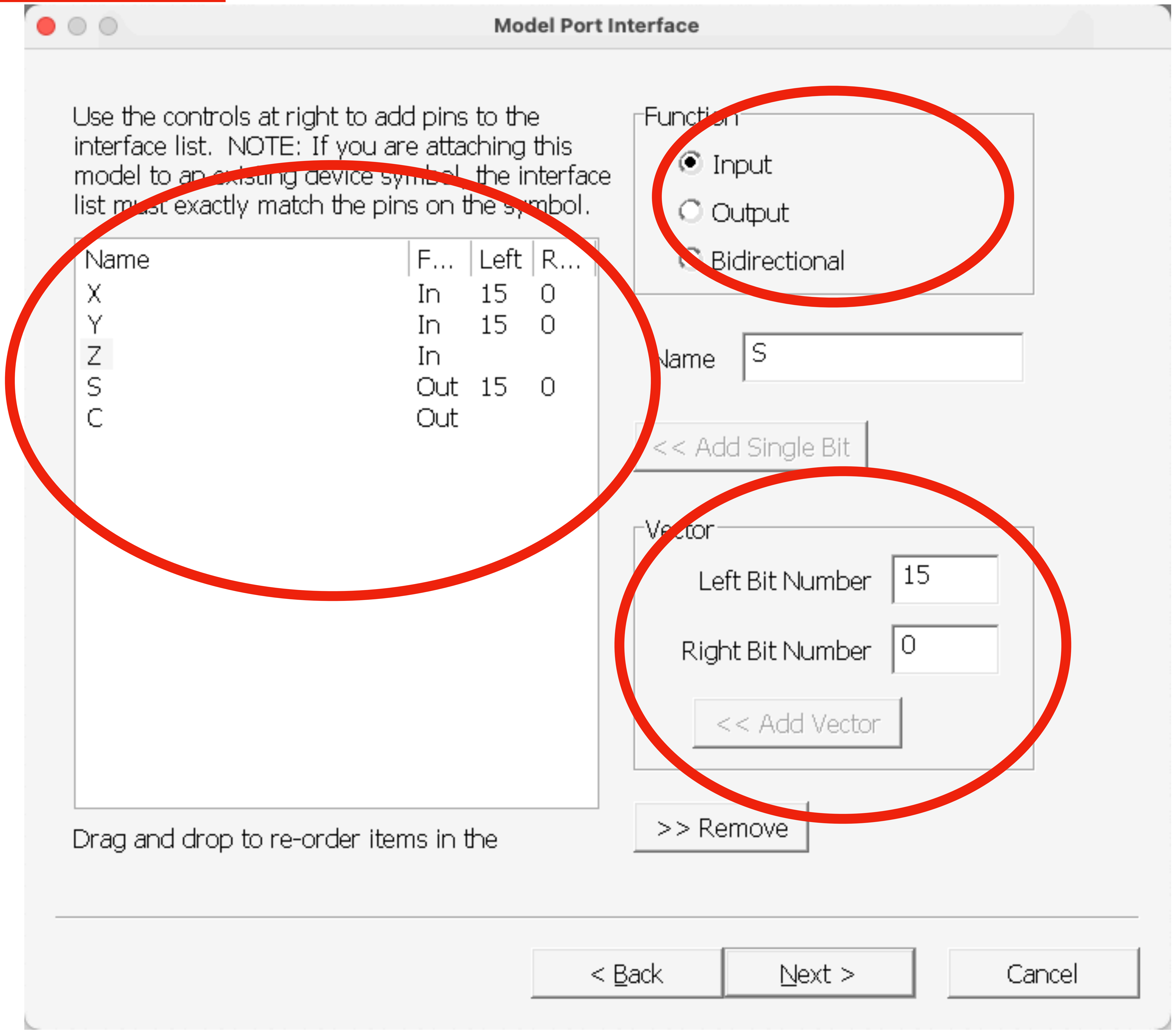
16bit Binary Adder


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
Adder16bit
< Back
Next >
Cancel

1. Select `VHDL`; Type in name `Adder16Bit`, the name cannot contain whitespace; Select `Next`

P2
Buses

16bit Binary Adder


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	F...	Left	R...
X	In	15	0
Y	In	15	0
Z	In		
S	Out	15	0
C	Out		

Function: [✓ Input] Output Bidirectional

Name: S

Vector
Left Bit Number: 15
Right Bit Number: 0

Drag and drop to re-order items in the

Name	Func	Left	Right
X	In	15	0
Y	In	15	0
Z	In		
S	Out	15	0
C	Out		

Tutorial 2

2. Use `Function`, `Name`, and `<< Add Vector` **in addition to single bits to include** `XYZSC` **in the list of pins/buses; Select** `Next`

P1
VHDL

16bit Binary Adder

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


entity Adder16bit is

 port(
       Z   : in std_logic;
       Y   : in std_logic_vector(15 downto 0);
       X   : in std_logic_vector(15 downto 0);
       C   : out std_logic;
       S   : out std_logic_vector(15 downto 0)
  );

end Adder16bit;


architecture arch1 of Adder16bit is

begin

  -- Your VHDL code defining the model goes here

end arch1;
```

- Notice the difference
  - `std_logic` is for single bits
  - `std_logic_vector` is for buses
- How can we design the adder?
  - Use Addition from register microoperations!

3. This is what your VHDL code looks like

Tutorial 2

P1
VHDL

16bit Binary Adder

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


...


architecture arch1 of Adder16bit
is

begin

  S <= X + Y + Z;

end arch1;
```

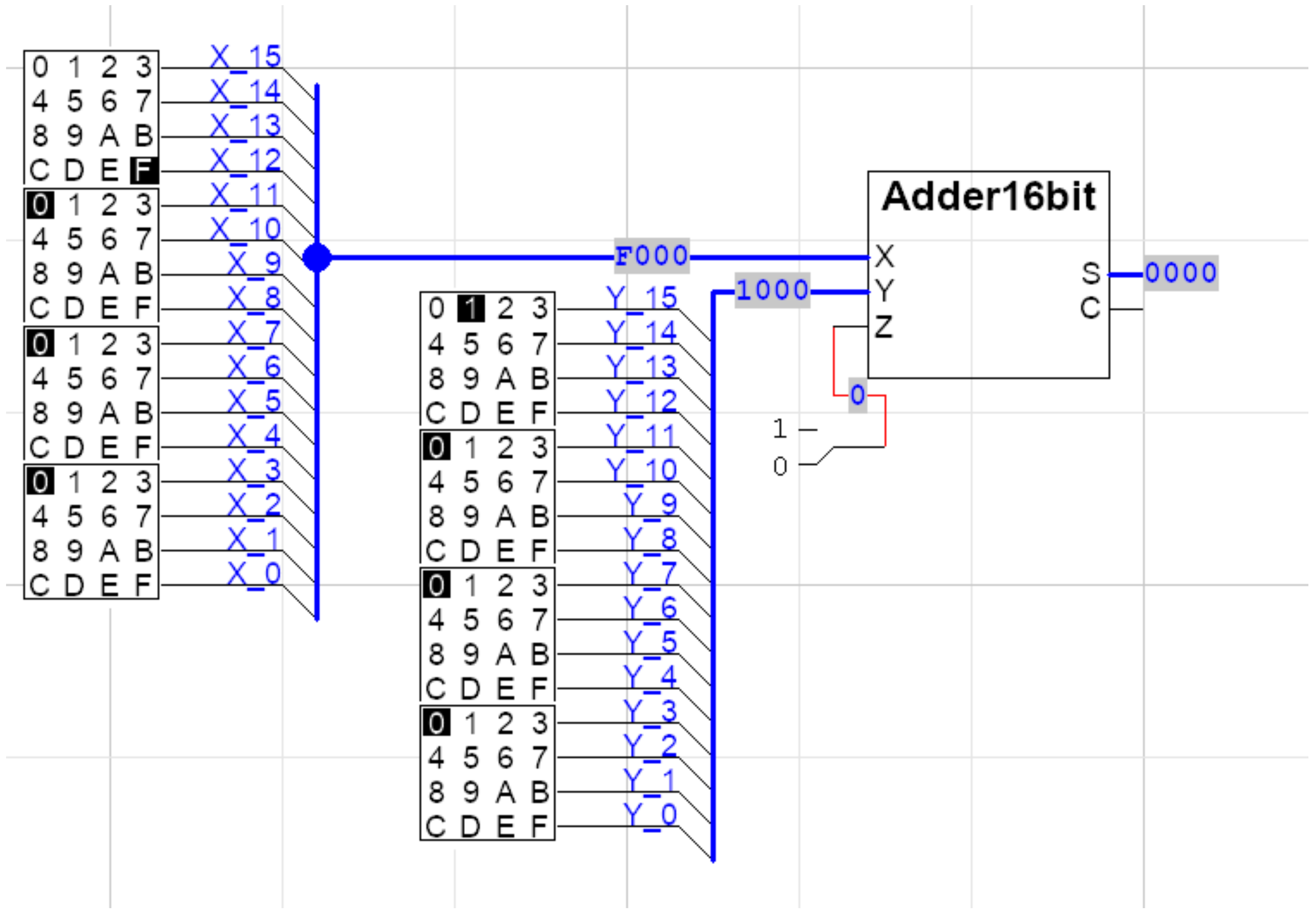
- Add a Package from Library
- How can we design the adder?
  - Use Addition from register microoperations!

4. Add `std_logic_arith.all` from IEEE library; Use addition in your code

Tutorial 2

16bit Binary Adder

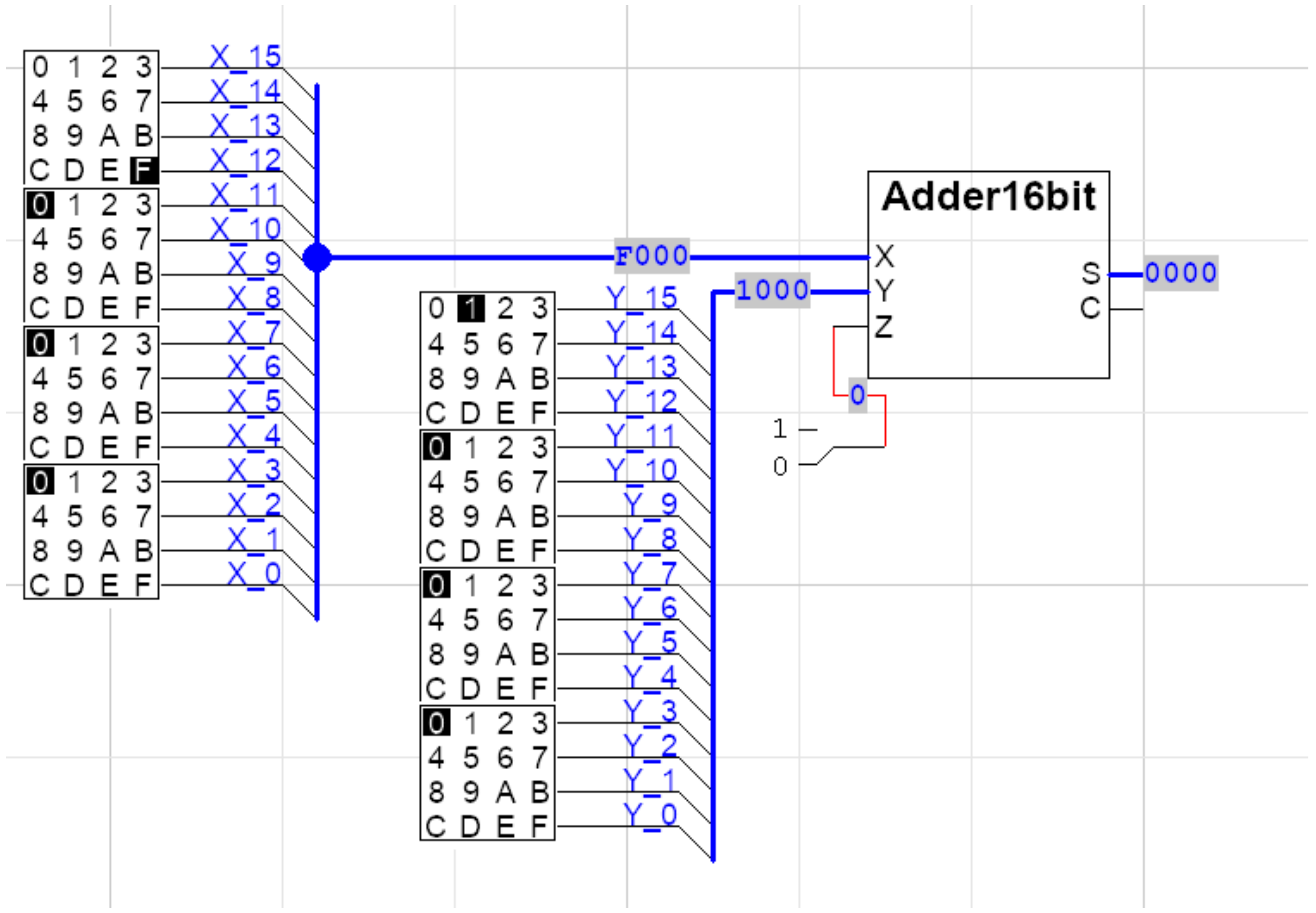
- For all VHDL vectors, the corresponding buses must have matching names
- E.g. `X` bus in Adder16bit should have bus `X_0..15`. Don't forget the underscore.
- There is a bug with the IO panel, I am investigating it
- Notice that `C` output doesn't work now. Solution?



P1
VHDL

16bit Binary Adder

- For all VHDL vectors, the corresponding buses must have matching names
- E.g. `X` bus in Adder16bit should have bus `X_0..15`. Don't forget the underscore.
- There is a bug with the IO panel, I am investigating it
- Notice that `C` output doesn't work now. Solution? (Hint: use concatenation & vector)



Tutorial 2

LAB 1 Part 2
A VHDL Exercise

LAB 1 Part 2
A VHDL Exercise

- Task 1: Implement `Adder16bit.dwv`, save it in `CSCI250.clf`
  - Find a way to make `C` output the correct value
  - You must show `Adder16bit` working in `circuit1.cct`

Name	Func	Left	Right
X	In	15	0
Y	In	15	0
Z	In		
S	Out	15	0
C	Out		

LAB 1 Part 2
A VHDL Exercise

- Task 2: Implement `AddSub16bit.dwv`, save it in `CSCI250.clf`
  - This is an adder subtractor. AS is short for notAdd/Sub
  - You must show `AddSub16bit` working in `circuit2.cct`

Name	Func	Left	Right
X	In	15	0
Y	In	15	0
AS	In		
O	Out	15	0
C	Out		

Concept