數位邏輯實習

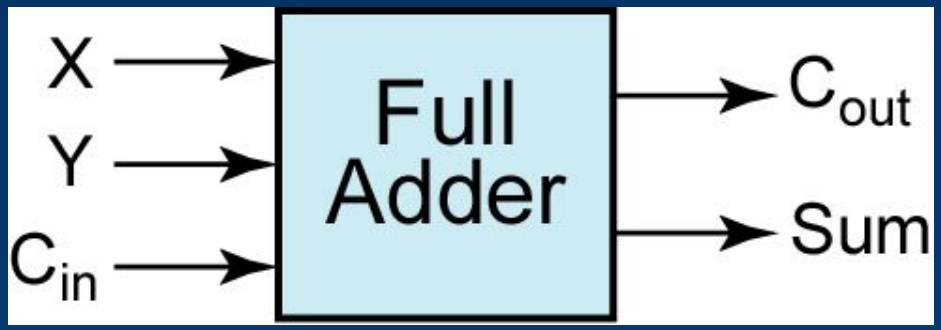
VHDL硬體描述語言(I)

國立台北科技大學
電機工程系
吳昭正

Outline

- Digital產生的VHDL程式碼
- VHDL語法介紹
- GHDL編譯器
- Digital實作VHDL
- 作業題

Digital產生的VHDL程式碼(1/4)



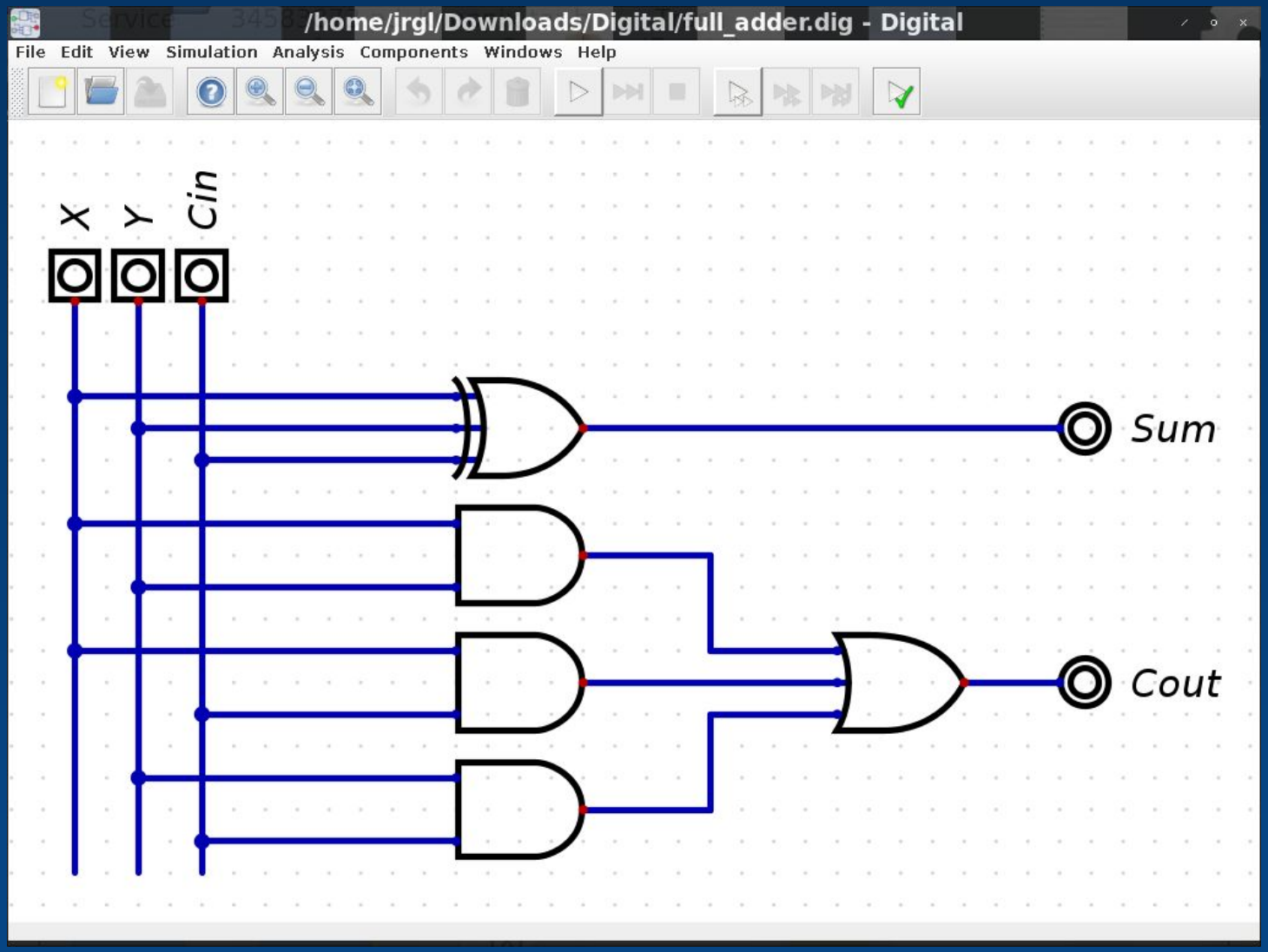
X	Y	C_{in}	C_{out}	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Digital產生的VHDL程式碼(2/4)

$$
\begin{aligned}
Sum &= X'Y'C_{in} + X'YC'_{in} + XY'C'_{in} + XYC_{in} \\
&= X'(Y'C_{in} + YC'_{in}) + X(Y'C'_{in} + YC_{in}) \\
&= X'(Y \oplus C_{in}) + X(Y \oplus C_{in})' = X \oplus Y \oplus C_{in}
\end{aligned}
$$

$$
\begin{aligned}
C_{out} &= X'YC_{in} + XY'C_{in} + XYC'_{in} + XYC_{in} \\
&= (X'YC_{in} + XYC_{in}) + (XY'C_{in} + XYC_{in}) + (XYC'_{in} + XYC_{in}) \\
&= YC_{in} + XC_{in} + XY
\end{aligned}
$$

Digital產生的VHDL程式碼(3/4)



Digital產生的VHDL程式碼(4/4)

```
-- generated by Digital. Don't modify this file!
-- Any changes will be lost if this file is regenerated.

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

entity main is
  port (
    X: in std_logic;
    Y: in std_logic;
    Cin: in std_logic;
    Sum: out std_logic;
    Cout: out std_logic);
end main;
```

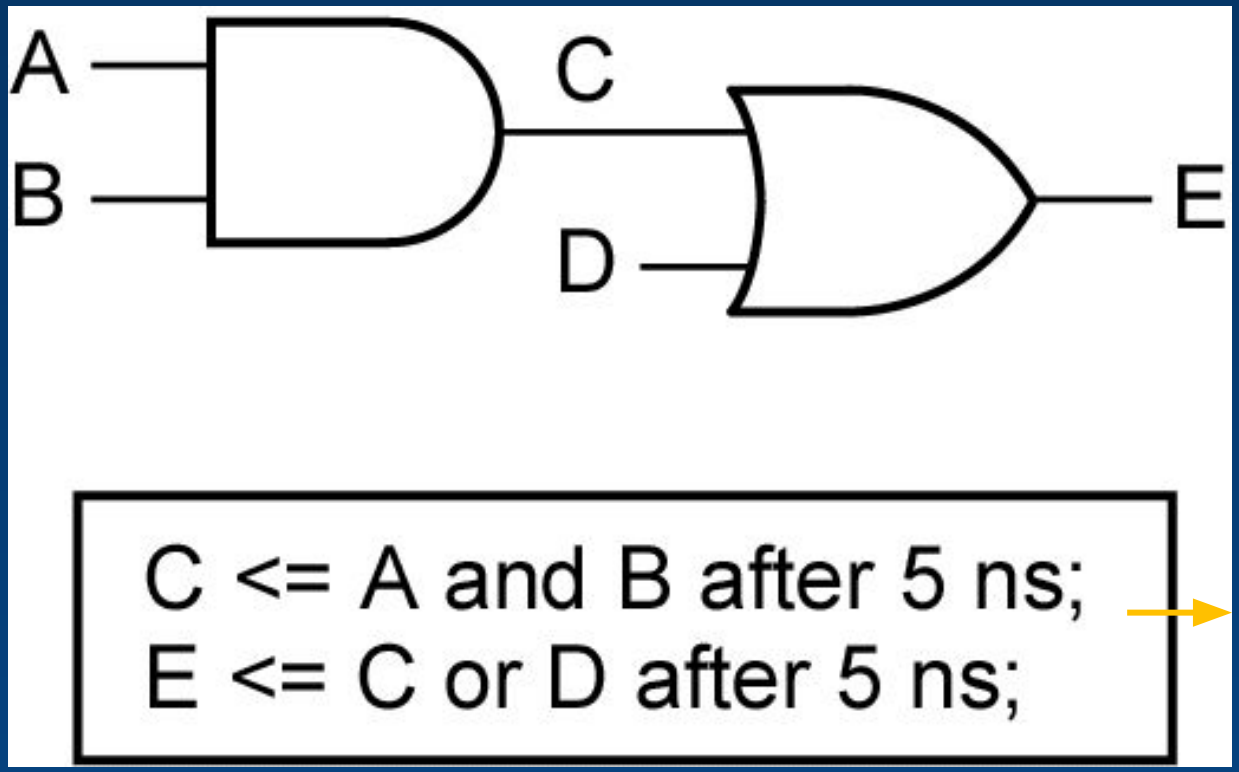
```
architecture Behavioral of main is
begin
  Sum <= (X XOR Y XOR Cin);
  Cout <= ((X AND Y) OR (X AND Cin) OR (Y AND Cin));
end Behavioral;
```

VHDL語法介紹(1/19)

- A VHDL (Very high speed integrated circuits Hardware Description Language) signal is used to describe a signal in a physical system.
- The symbol “<=“ is the signal assignment operator which indicates that the value computed on the right-hand side is assigned to the signal on the left side.
- VHDL中變數名稱大小寫視為一樣，舉例來說：CASE與case兩個變數為一樣。
- VHDL中的註解以”--”作為開頭，撰寫程式請養成好習慣，每個部分均加上註解。

Section 10.1 (p. 286-287)

VHDL語法介紹(2/19)



前後與執行順序無關

Figure 10-1: Gate Circuit

A *dataflow* description of the circuit where it is assumed that each gate has a 5-ns propagation delay.

VHDL語法介紹(3/19)

When simulated, the first statement will be evaluated any time A or B changes, and the second statement will be evaluated any time C or D changes.

Suppose that initially A = 1, and B = C = D = E = 0. If B changes to 1 at time 0, C will change to 1 at time = 5ns. Then, E will change to 1 at time = 10ns.

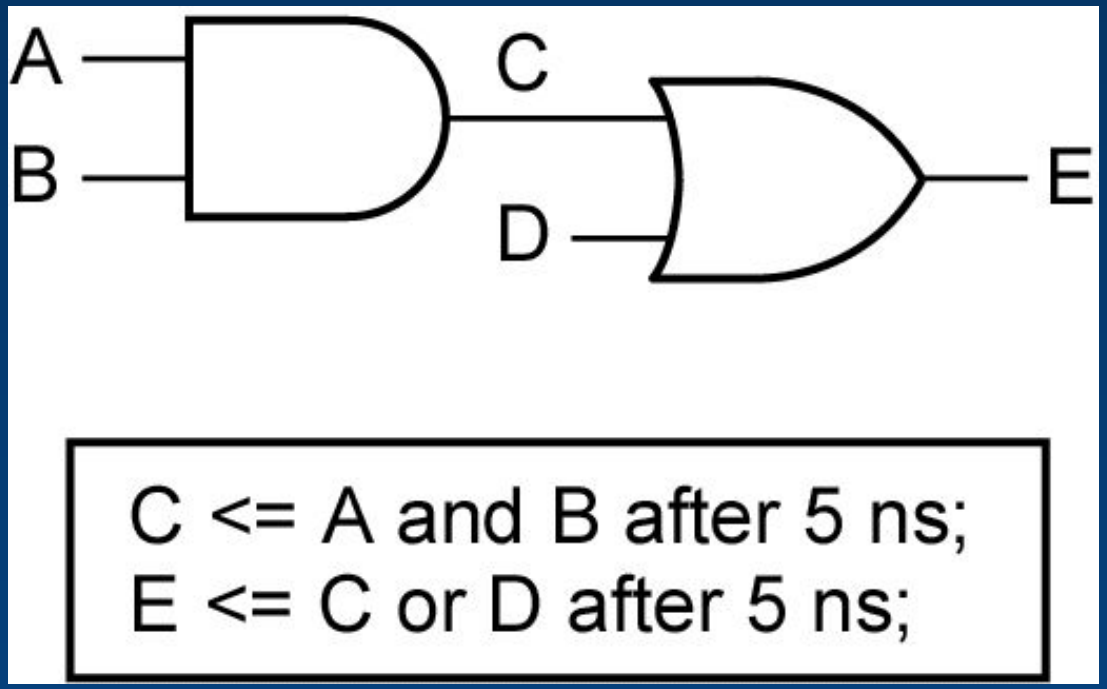


Figure 10-1: Gate Circuit

VHDL語法介紹(4/19)

A signal assignment statement has the form:

signal_name <= expression [**after** delay];

Brackets indicate "**after** delay" is optional. If omitted, an infinitesimal Δ (delta) delay is assumed.

Given the statements: E <= C **or** D;

C <= A **and** B;

If A = '1', B = C = D = '0', and B changes to '1' at time = 1, the second statement executes, and C changes to '1' at time 1+Δ. The first statement then executes, and E changes to '1' at time 1+2Δ.

VHDL語法介紹(5/19)

Even if a VHDL program has no explicit loops, concurrent statements may execute repeatedly as if they were in a loop.

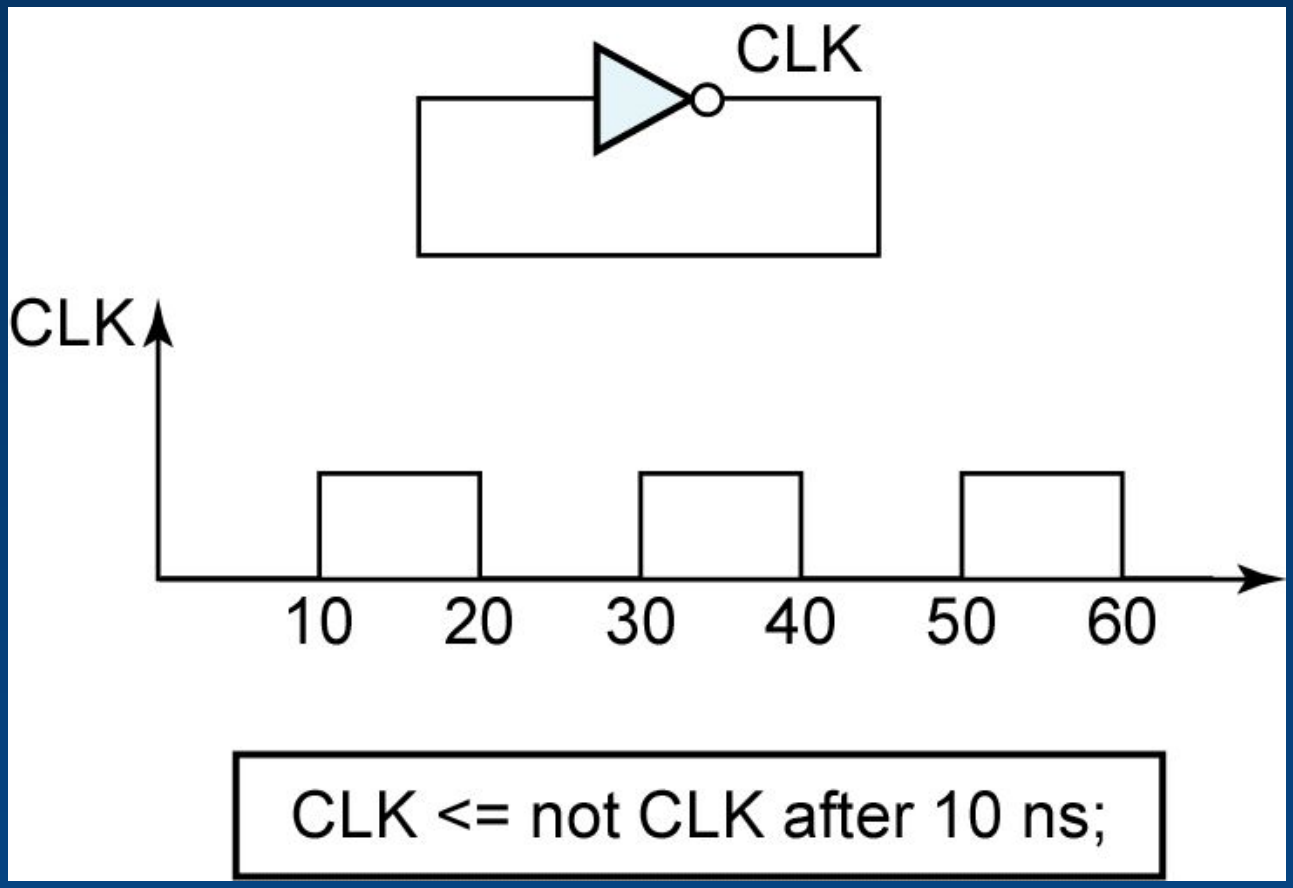
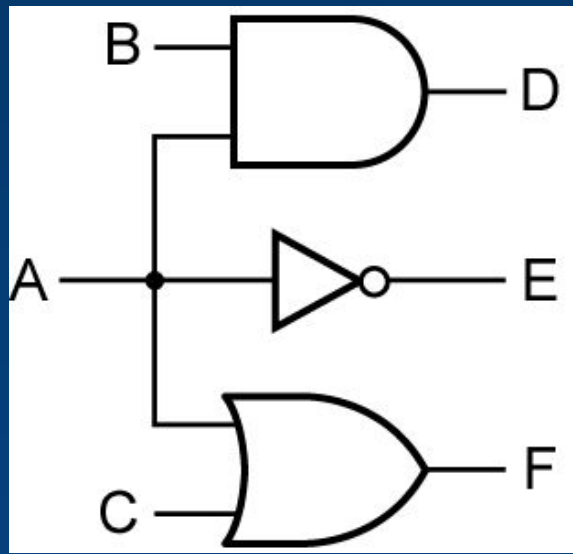


Figure 10-2: Inverter with Feedback

VHDL語法介紹(6/19)




```
-- when A changes, these concurrent
-- statements all execute at the same time

D <= A and B after 2 ns;

E <= not A after 1 ns;

F <= A or C after 3 ns;
```

Figure 10-3: Three Gates with a Common Input and Different Delays

VHDL語法介紹(7/19)

Inertial Delays

VHDL provides two kinds of delay models – inertial delays and transport delays. A signal assignment statement of the form

signal_name <= expression **after** delay;

assumes an inertial delay by default. A device with an inertial delay of D time units delays an input signal by D; however, input changes that occur less than D time units apart are filtered out and do not appear at the output.

Section 10.1 (p. 289)

VHDL語法介紹(8/19)

Transport Delays

A device with a transport (ideal) delay of D time units delays any input signal by D. Unlike an inertial delay, all signal changes pass through without any filtering. The following VHDL statement models a transport delay:

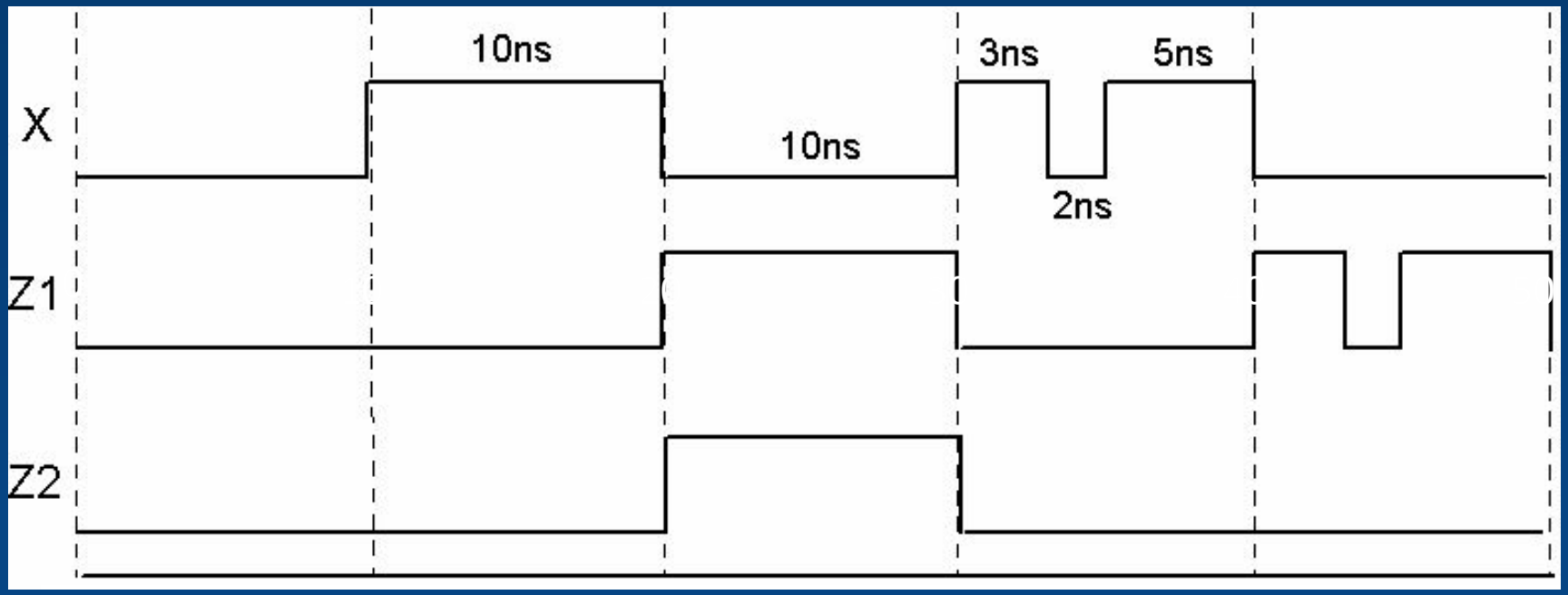
signal_name <= **transport** expression **after** delay;

The keyword **transport** is required; otherwise, the default inertial delay is assumed.

VHDL語法介紹(9/19)

Inertial and Transport Delays

Z_1 <= **transport** X **after** 10 ns;
Z_2 <= X after 10 ns; -- inertial delay



VHDL語法介紹(10/19)

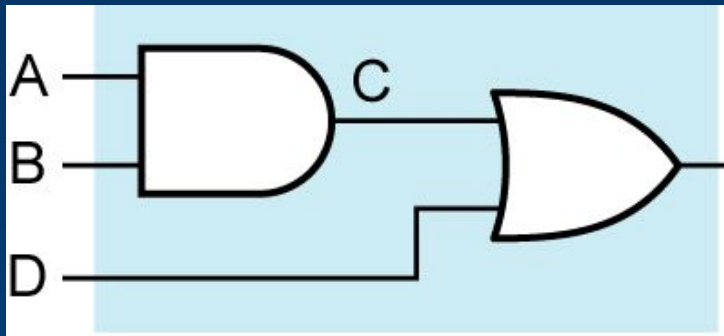
VHDL Modules

To write a complete VHDL module, we must declare all of the input and output signals using an **entity** declaration, and then specify the internal operation of the module using an **architecture** declaration.

Section 10.3 (p. 292)

VHDL語法介紹(11/19)

VHDL Modules




```
entity two_gates is
    port (A,B,D: in bit; E: out bit);
end two_gates;
architecture gates of two_gates is
    signal C: bit;
begin
    C <= A and B; -- concurrent
    E <= C or D; -- statements
end gates;
```

Figure 10-8: VHDL Module with Two Gates

VHDL語法介紹(12/19)

VHDL Modules

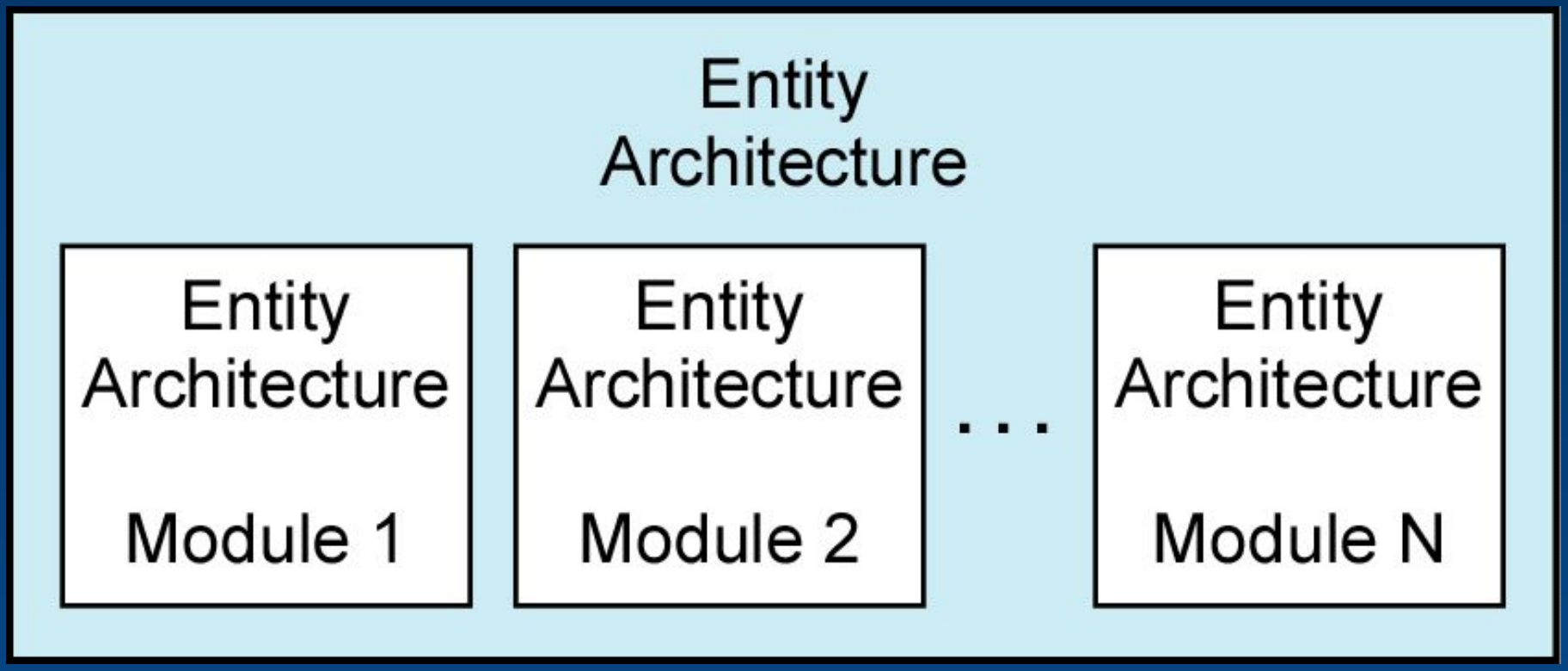


Figure 10-9: VHDL Program Structure

VHDL語法介紹(13/19)

VHDL Modules

Each entity declaration includes a list of interface signals that can be used to connect to other modules or to the outside world. We will use entity declarations of the form:

```
entity entity-name is
 [port(interface-signal-declaration);]
end [entity] [entity-name];
```

The items enclosed in brackets are optional. The interface-signal-declaration normally has the following form:

```
list-of-interface-signals: mode type [: = initial-value]
{; list-of-interface-signals: mode type [: = initial-value]};
```

VHDL語法介紹(14/19)

VHDL Modules

Here is an example of a port declaration that indicates A and B are input signals of type integer that are initially set to 2, and C and D are output signals of type bit that are initialized by default to ‘0’:

port(A, B: **in** integer : = 2; C, D: **out** bit);

VHDL語法介紹(15/19)

VHDL Modules

Associated with each entity is one or more architecture declarations of the form

```
architecture architecture-name of entity-name is
  [declarations]
begin
  architecture body
end [architecture] [architecture-name];
```

In the declarations section, we can declare signals and components that are used within the architecture. The architecture body contains statements that describe the operation of the module.

VHDL語法介紹(16/19)

Signals and Constants

Input and output signals for a module are declared in a port. Signals internal to a module are declared at the start of an architecture, before **begin**, and can be used only within that architecture. Port signals have an associated mode (usually in or out), but signals do not.

A signal used within an architecture must be declared either in a port or in the declaration section of an architecture, but it cannot be declared in both places.

VHDL語法介紹(17/19)

Signals and Constants

```
signal list_of_signal_names: type_name [constraint] [:= initial_value];
signal A, B, C: bit_vector(3 downto 0):= "1111";
signal E, F: integer range 0 to 15;
```

```
constant constant_name: type_name [constraint] [:= constant_value];
constant limit : integer := 17;
constant delay1 : time := 5 ns;
```

Section 10.4 (p. 297)

資料型態(1/5)
Pre-defined VHDL Types

bit	'0' or '1'
boolean	FALSE or TRUE
integer	an integer in the range $-(2^{31}-1)$ to $+(2^{31}-1)$ (some implementations support a wider range)
positive	an integer in the range 1 to $2^{31}-1$ (positive integers)
natural	an integer in the range 0 to $2^{31}-1$ (positive integers and zero)
real	floating-point number in the range −1.0E38 to + 1.0E38
character	any legal VHDL character including upper- and lower case letters, digits, and special characters; each printable character must be enclosed in single quotes, e.g., 'd', '7', '+'
time	an integer with units fs, ps, ns, us, ms, sec, min, or hr

Section 10.4 (p. 298)

資料型態(2/5)
IEEE Standard Logic

Use of two-valued logic (bits and bit vectors) is generally not adequate for simulation of digital systems. In addition to ‘0’ and ‘1’, values of ‘Z’ (high-impedance or no connection), ‘X’ (unknown), and ‘U’ (uninitialized) are frequently used in digital system simulation. The IEEE standard 1164 defines a std_logic type that actually has nine values:

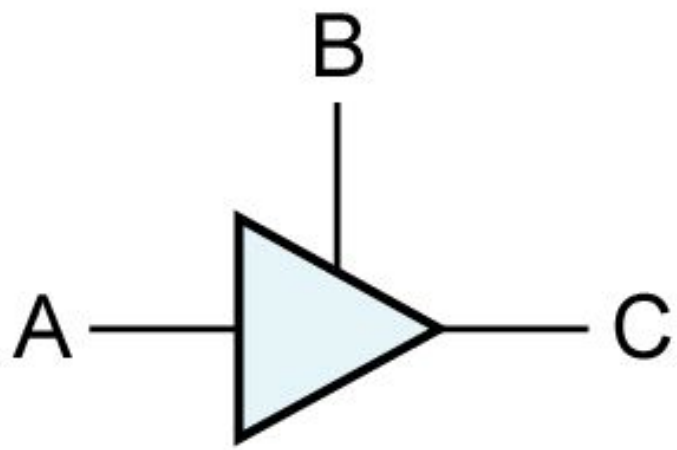
‘U’, ‘X’, ‘0’, ‘1’, ‘Z’, ‘W’, ‘L’, ‘H’, ‘-’

(We will only use the values ‘U’, ‘X’, ‘0’, ‘1’, and ‘Z’.)

Section 10.8 (p. 304)

資料型態(3/5)

IEEE Standard Logic




```
signal A,B,C: std_logic;
-------------------------------
C <= A when B = '1' else 'Z';
```

Figure 10-16: Tri-State Buffer

資料型態(4/5)

IEEE Standard Logic

If a std_logic signal is assigned two different values, VHDL automatically calls a resolution function to determine the outcome.

	S2				
S1	U	X	0	1	Z
U	U	U	U	U	U
X	U	X	X	X	X
0	U	X	0	X	0
1	U	X	X	1	1
Z	U	X	0	1	Z

Figure 10-18: Resolution Function for Two Signals

資料型態(5/5)

VHDL Operators

Predefined VHDL operators can be grouped into seven classes:

1. binary logical operators: **and or nand nor xor xnor**
2. relational operators: = /= < <= > >=
3. shift operators: **sll srl sla sra rol ror**
4. adding operators: + − & (concatenation)
5. unary sign operators: + −
6. multiplying operators: * / **mod rem**
7. miscellaneous operators: **not abs** **

When parentheses are not used, operators in class 7 have the highest precedence and are applied first, followed by class 6, then class 5, etc.

Section 10.6 (p. 301)

VHDL語法介紹(18/19)

Library and use

To access components and functions within a package requires a **library** statement and a **use** statement. This statement allows your design to access the BITLIB:

library BITLIB;

This statement allows your design to use the entire bit_pack package:

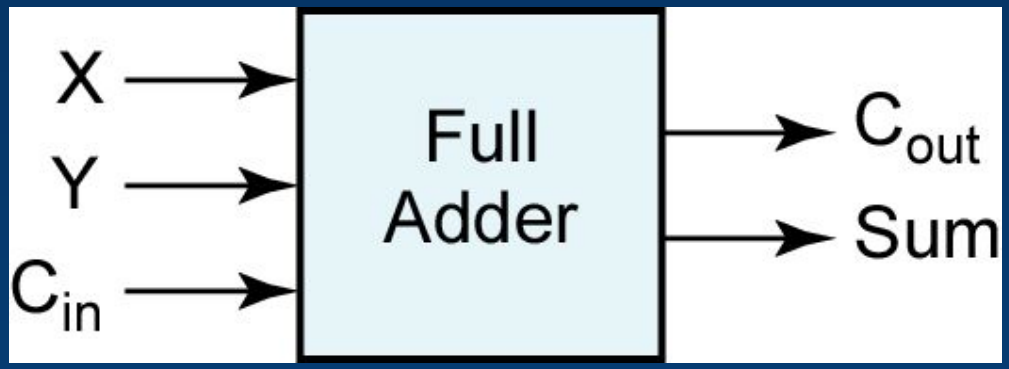
use BITLIB.bit_pack.**all**;

This statement allows your design to use just the Nor2 component:

use BITLIB.bit_pack.Nor2;

VHDL語法介紹(19/19)

Full Adder




```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

entity FullAdder is
  port (
    X, Y, Cin: in std_logic;
    Sum, Cout: out std_logic);
end FullAdder;

architecture Equations of FullAdder is
begin
      -- concurrent assignment statements
      Sum  <= X xor Y xor Cin;
      Cout <= (X and Y) or (X and Cin) or (Y and Cin);
end Equations;
```

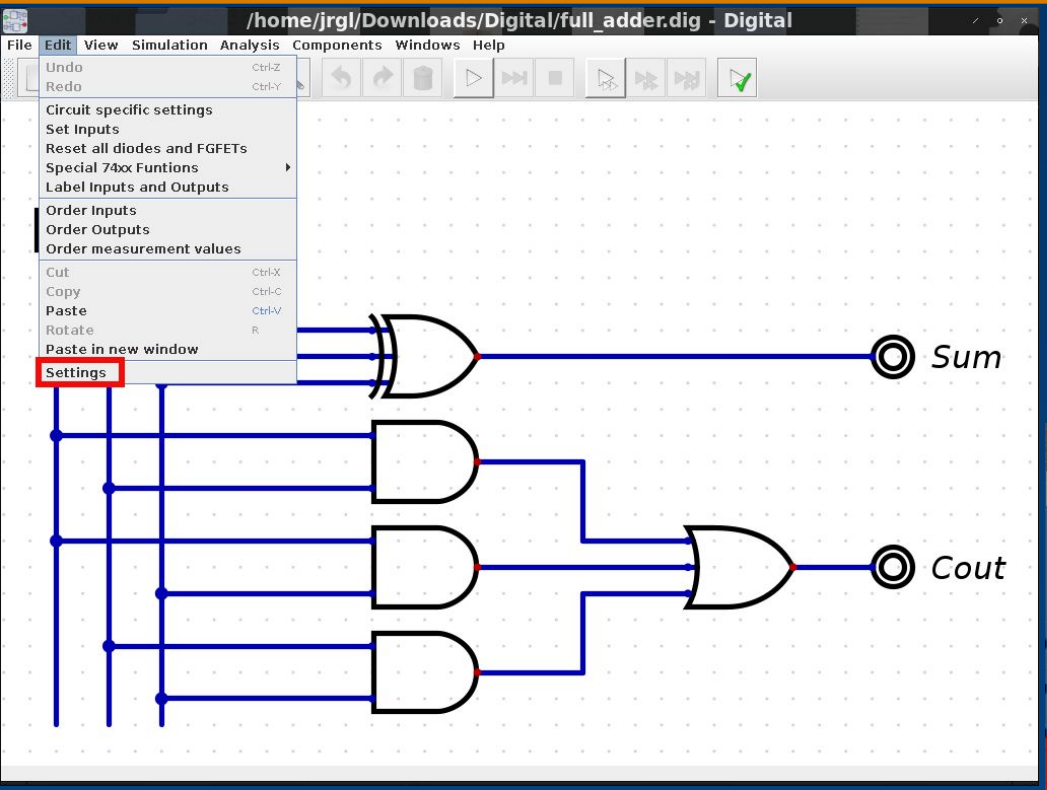
GHDL編譯器(1/2)

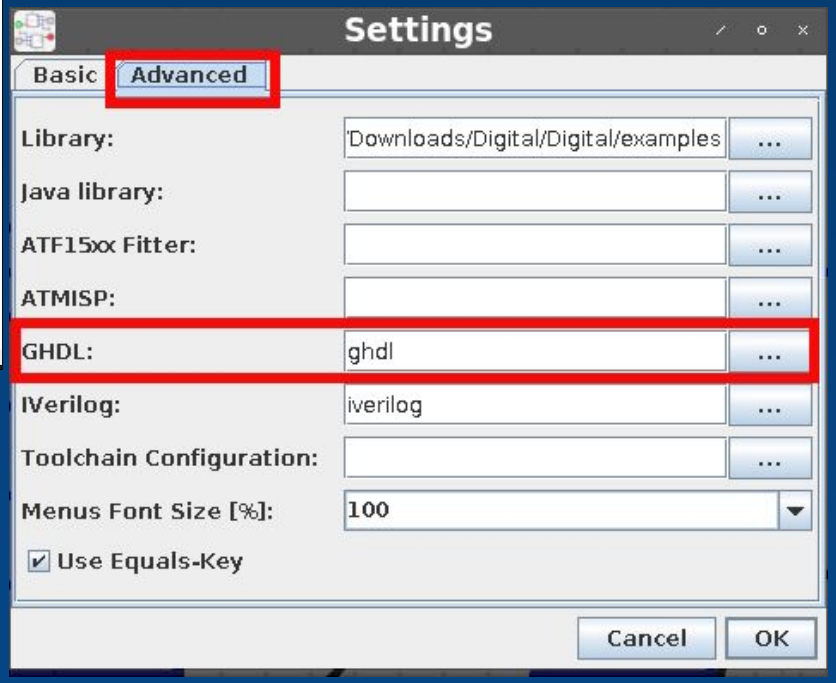
- GHDL is an open-source simulator for the VHDL language. GHDL allows you to compile and execute your VHDL code directly in your PC.
- It could be downloaded from the following URL:

https://github.com/ghdl/ghdl/releases

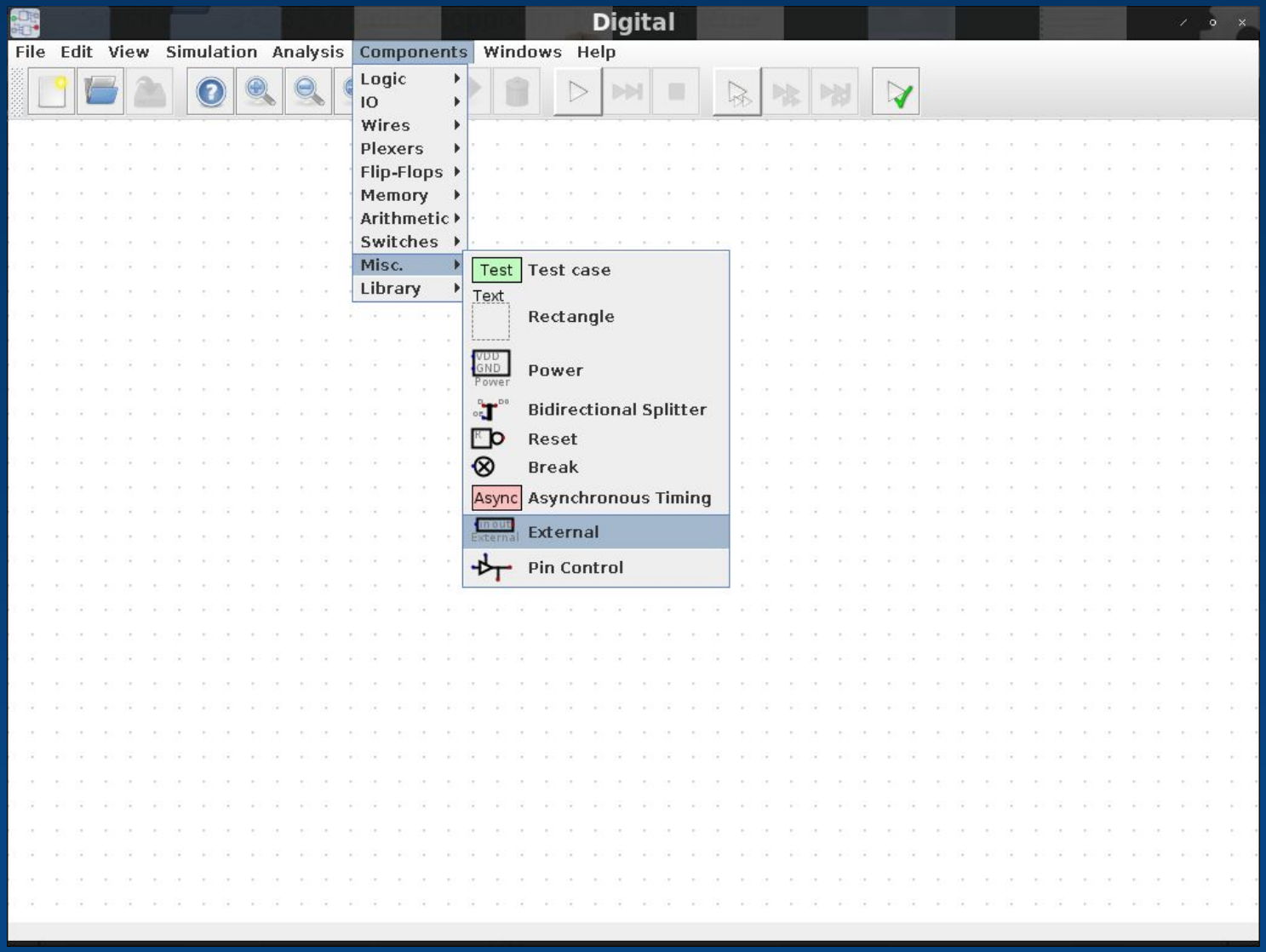
ghdl-0.37-buster-mcode-gpl.src.tgz	3.85 MB
ghdl-0.37-buster-mcode-gpl.tgz	2.68 MB
ghdl-0.37-buster-mcode-synth.tgz	3.57 MB
ghdl-0.37-buster-mcode.tgz	3.01 MB
ghdl-0.37-fedora31-llvm.tgz	6.55 MB
ghdl-0.37-fedora31-mcode.tgz	2.93 MB
ghdl-0.37-macosx-mcode.tgz	2.22 MB
ghdl-0.37-mingw32-mcode.zip	4.89 MB
ghdl-0.37-mingw64-llvm.zip	20 MB
ghdl-0.37-ubuntu16-llvm-3.9.tgz	6.51 MB
ghdl-0.37-ubuntu16-mcode.tgz	2.89 MB
ghdl.1	165 KB
Source code (zip)	
Source code (tar.gz)	

GHDL編譯器(2/2)

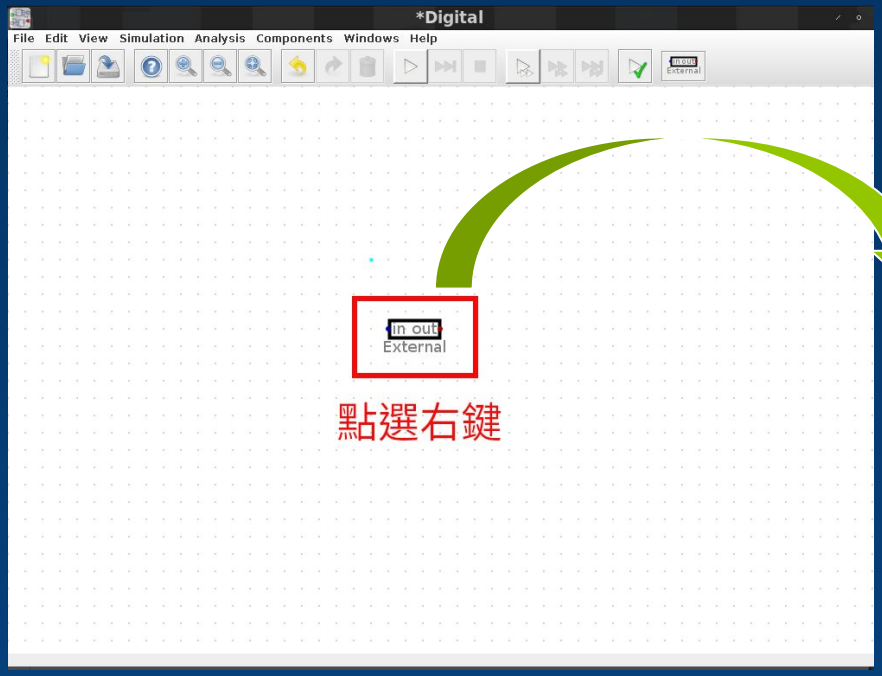

/home/jrgl/Downloads/Digital/full_adder.dig - Digital
File Edit View Simulation Analysis Components Windows Help
Undo Ctrl-Z
Redo Ctrl-Y
Circuit specific settings
Set Inputs
Reset all diodes and FGFETs
Special 74xx Funtions
Label Inputs and Outputs
Order Inputs
Order Outputs
Order measurement values
Cut Ctrl-X
Copy Ctrl-C
Paste Ctrl-V
Rotate R
Paste in new window
Settings
Sum
Cout

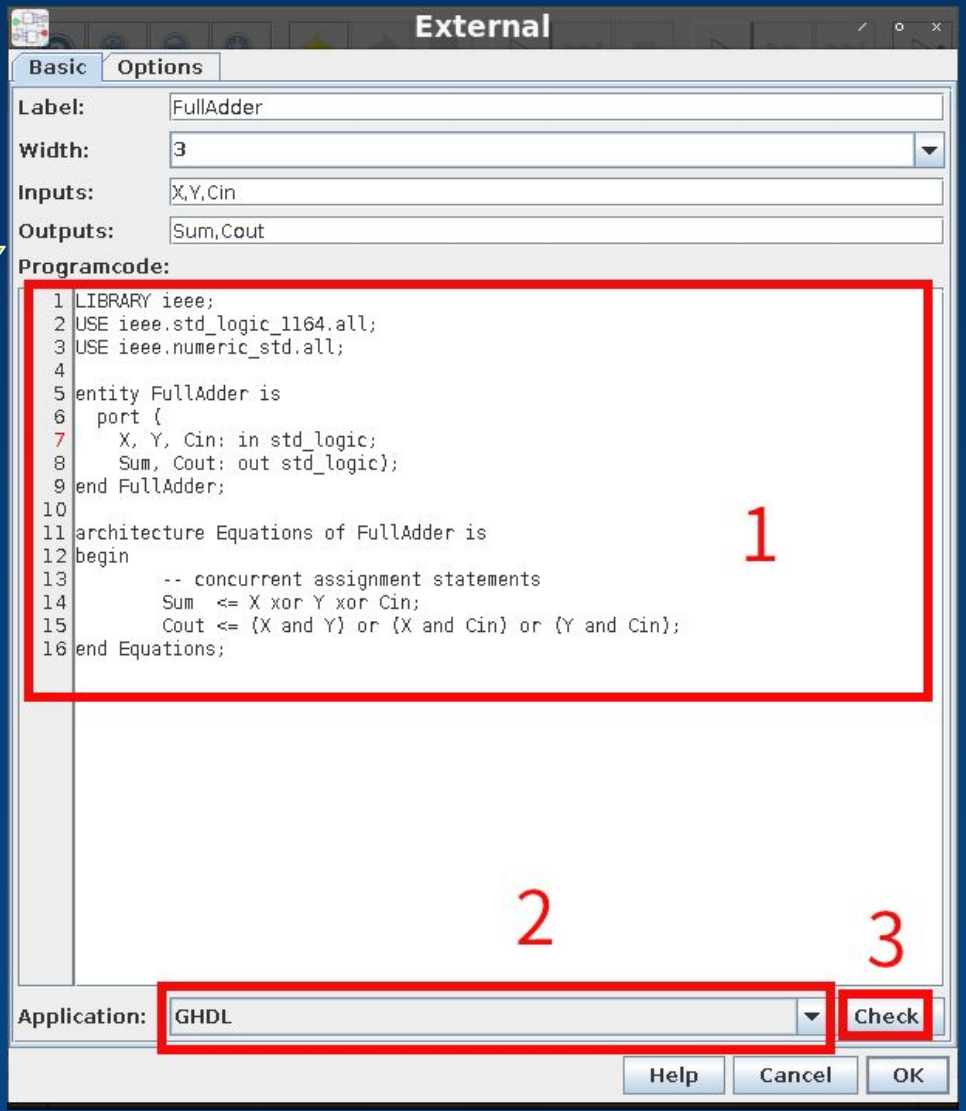

Settings
Basic
Advanced
Library:
Downloads/Digital/Digital/examples
Java library:
ATF15xx Fitter:
ATMISP:
GHDL:
ghdl
IVerilog:
iverilog
Toolchain Configuration:
Menus Font Size [%]:
100
Use Equals-Key
Cancel
OK

Digital實作VHDL(1/3)



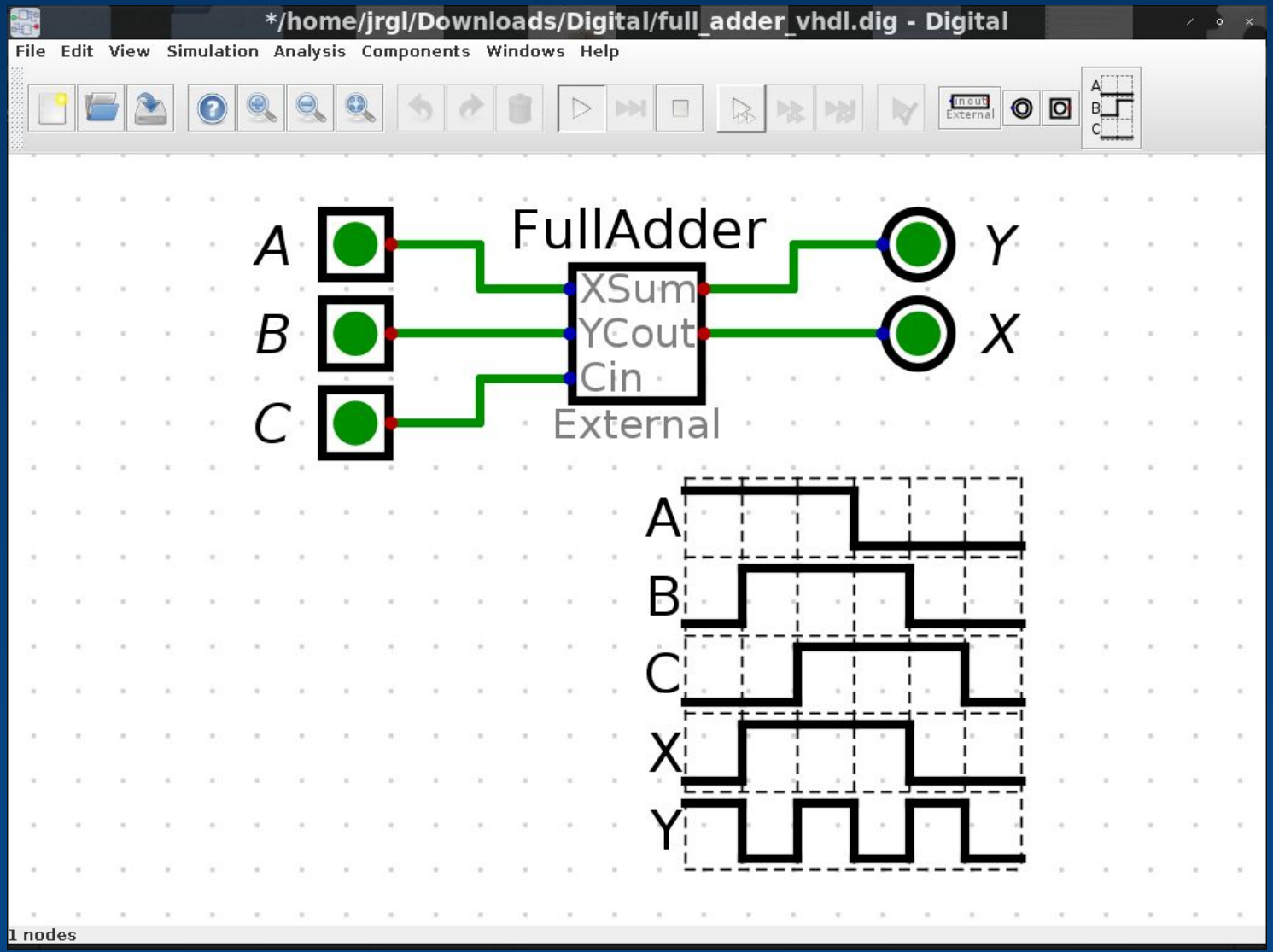
Digital實作VHDL(2/3)





1. 撰寫VHDL程式碼
2. 選取”GHDL”
3. 點選”Check”確定語法是否正確，並自動填入上方的參數。

Digital實作VHDL(3/3)



作業實作題目(1/4)

- 作業題1: 利用VHDL實作1位元的加減法器，將1位元加減法器串接為4位元加減法器。

作業實作題目(2/4)

0為加法運算、1為減法運算

A	B	C_{in}	Sub	C_{out}	Sum
0	1	0	0	0	1
1	0	1	0	1	0
1	1	1	0	1	1
1	0	1	1	0	0
0	1	0	1	1	1
1	1	1	1	1	1
0	1	1	1	1	0

完成以上的真值表，並推導C_{out}與Sum的布林代數式

作業實作題目(3/4)

- 作業題2: 利用VHDL實作Braille點字系統
- 本題請先導出最簡化的布林代數式, 之後用VHDL實作電路。

作業實作題目(4/4)

A	B	C	D	輸出
0	0	0	0	A
0	0	0	1	B
0	0	1	0	C
0	0	1	1	D
0	1	0	0	E
0	1	0	1	F
0	1	1	0	G
0	1	1	1	H

A	B	C	D	輸出
1	0	0	0	I
1	0	0	1	J
1	0	1	0	K
1	0	1	1	L
1	1	0	0	M
1	1	0	1	N
1	1	1	0	O
1	1	1	1	P