正確學會 改訂新版 ディジタル回路と Verilog HDL

Verilog 的16堂課

第 3 章

數位電路設計模擬

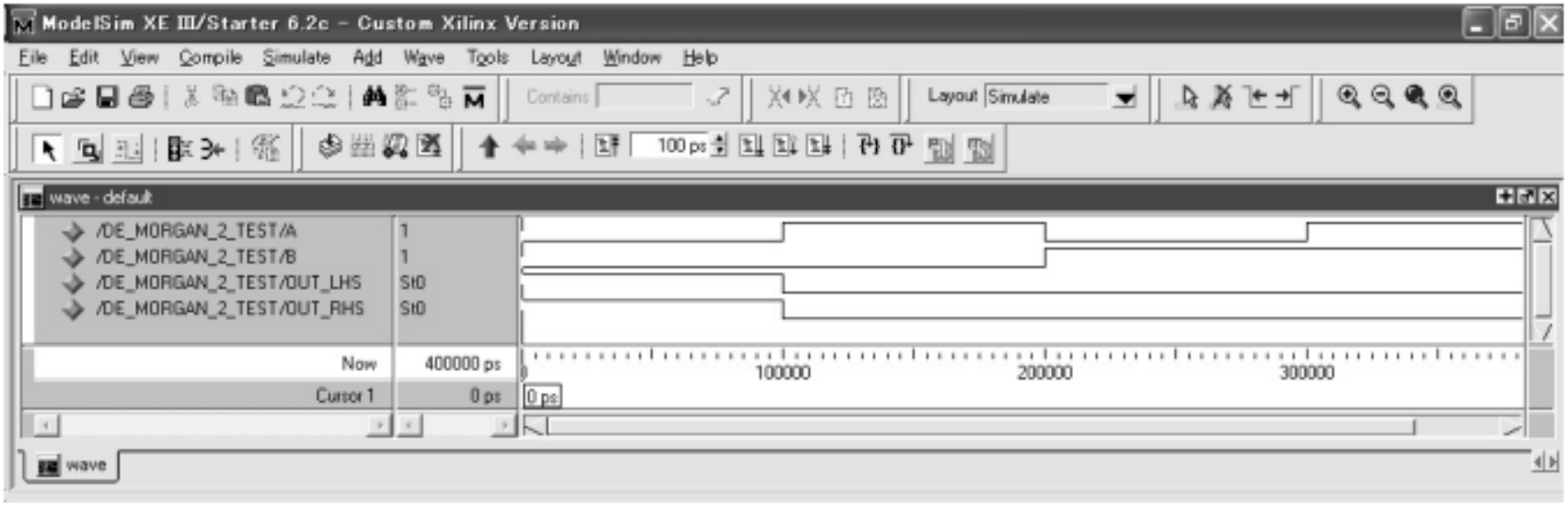
本章重點

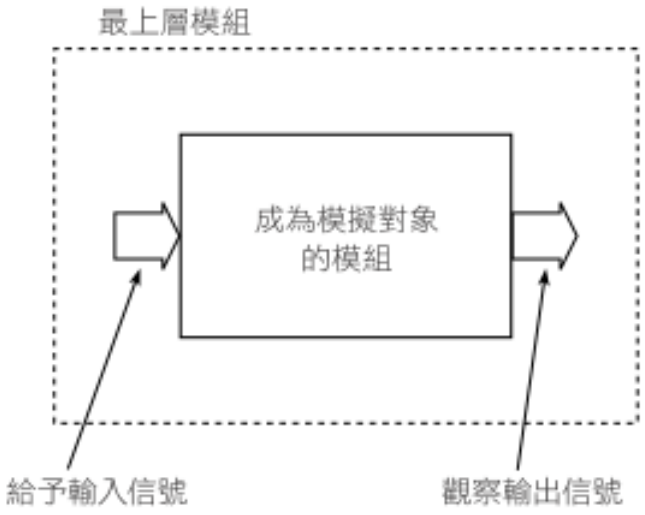
- 3.1 模擬
- 3.2 電路合成

3.1 模擬

- 模擬 (simulation)則是指「給予電路必須的輸入, 觀察其輸出的變化」

3.1.1 測試平台的設定





▲圖 3.1 測試平台的設定

3.1.2 宣告暫存器

- 要給予模擬電路的輸入信號, 要用保留字 reg 來做暫存器的宣告

3.1.3 宣告網絡

- 要觀察模擬電路的輸出信號, 要用保留字 wire 來做網絡的宣告

程式 3.1　為了模擬 2AND 電路所建的測試平台

```
第幾行
001   /*    AND_2_TEST      */
002   `timescale   1ns/1ns
003                // 在模擬狀況中, 每一個時間單位為 1ns
004   module AND_2_TEST;
005   reg   IN1, IN2;                    // 宣告暫存器
006   wire  OUT;                         // 宣告網絡
007         AND_2  AND_2    ( IN1, IN2, OUT );
008         initial begin
009                          IN1 = 0;  IN2 = 0;
010                #100      IN1 = 1;
011                #100      IN1 = 0;  IN2 = 1;
012                #100      IN1 = 1;
013                #200      $finish;
014         end
015   endmodule
```

測試平台的設定

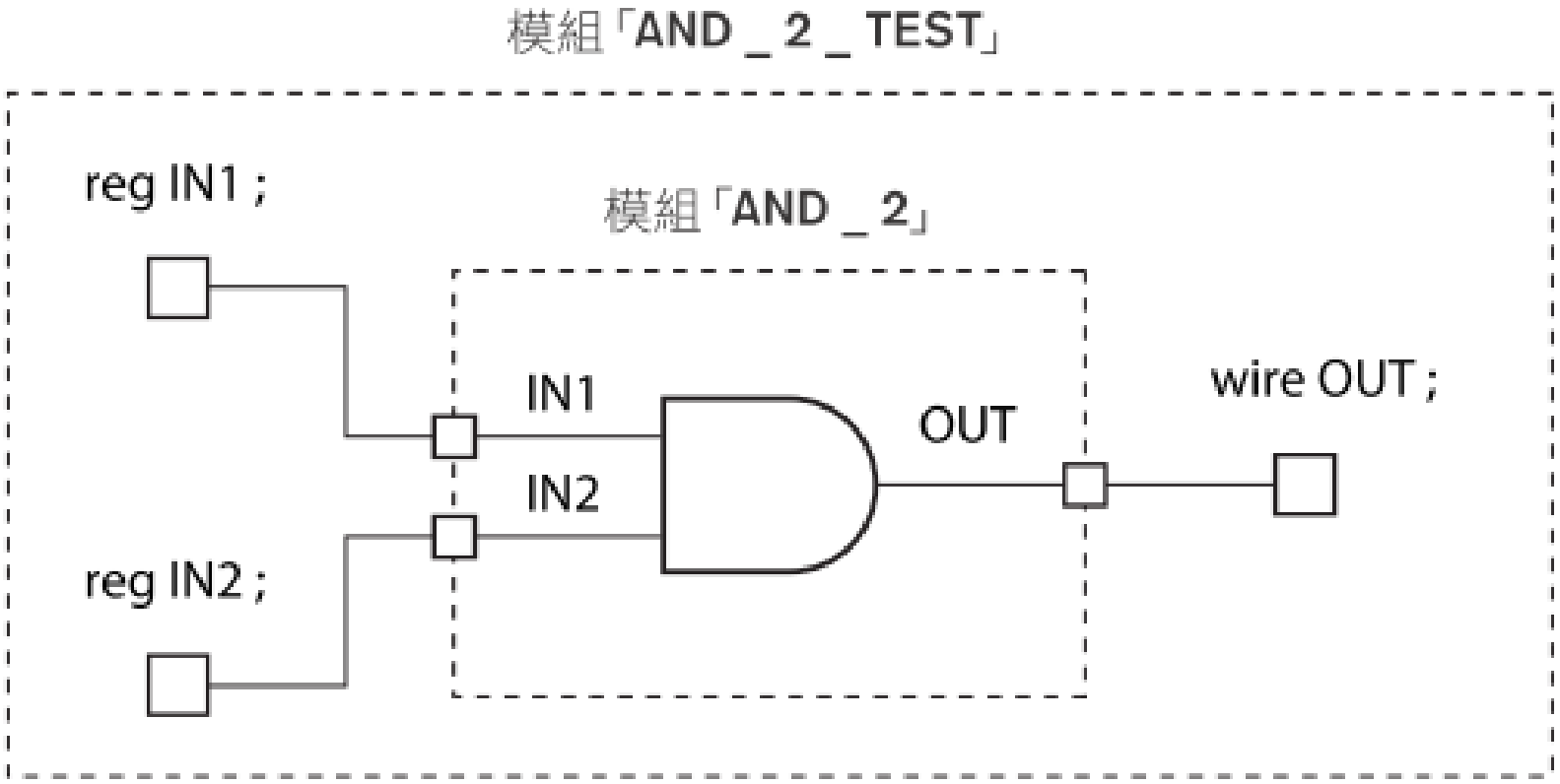
3.1.4 呼叫下層模組

- 透過測試平台呼叫要模擬對象的模組, 在 Verilog HDL 稱為模組的**簡碼化** (instance)

```
下層模組名    簡碼名稱          (參數群);
```

```
AND_2     AND_2     IN1, IN2, OUT
```

- 要呼叫下層模組,「呼叫端的網絡與下層模組的輸入輸出順序要一致, 這樣簡碼化才能成立」



▲ 圖 3.2　測試平台「AND _ 2 _ TEST」與下層模組「AND _ 2」的關係

3.1.5 根據順序與根據名稱做結合

- 信號的結合也可以利用名稱(信號名) 來做結合

```
. 定義端輸入輸出名稱 (要結合的信號)
```

- 定義端輸入輸出名稱 (要結合的信號) 也可寫成下面這樣會比較好懂：

```
下層模組的輸入輸出名稱 (上層模組的信號)
```

3.1.6 給下層模組輸入信號的波形定義

- initial 之後「begin-end」之間是描述輸入波形的變化, 如下：

```
initial          begin
IN_A = 0;        IN_B = 0;
                 #100   IN_A = 1;
                 #100   IN_A = 0;      IN_B = 1;
                 #100   IN_A = 1;
                 #200   $finish;
          end
```

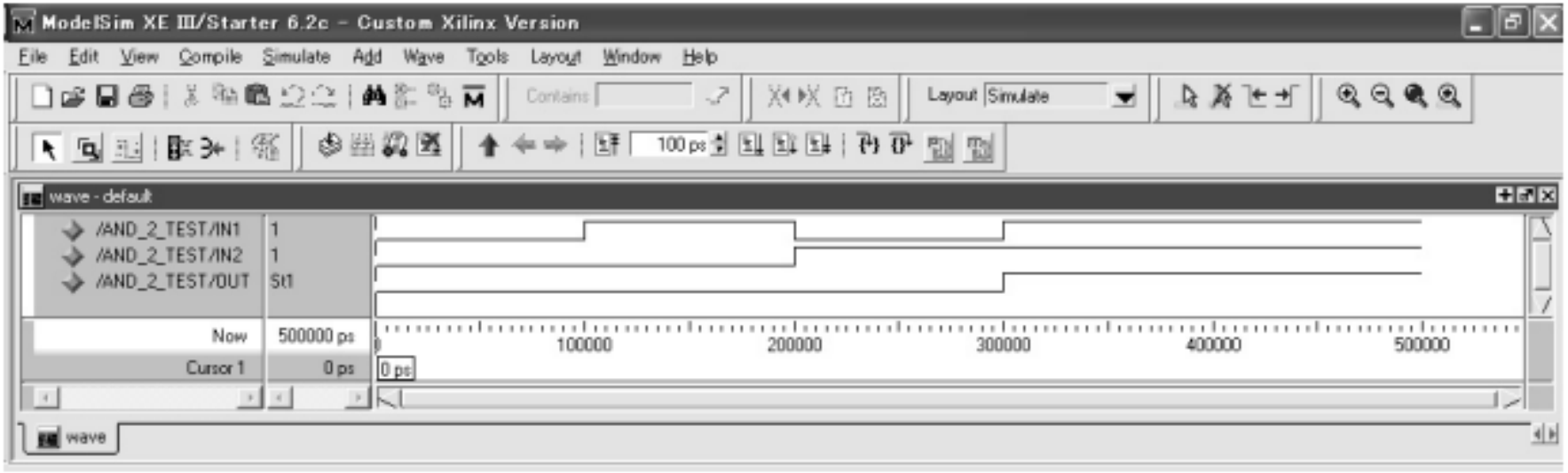
  - 一開始「IN1」= '0' ,「IN2」= '0'
  - 在 100[ns] 之後「IN1」= '1' ,「IN2」= '0'
  - 在 200[ns] 之後「 IN1」= '0' ,「IN2」= '1'
  - 在 300[ns] 之後「 IN1」= '1' ,「IN2」= '1'

- 在Verilog HDL 模擬器裡面負責指定時間的是編譯指示單元 **`timescale**
- 如：`timescale 1ns/1ns 表示時間單位被設定為 1 ns
- 在波型定義的最後，會寫 **SystemTask「$finish」**作為模擬的結尾
- '#' 表示延遲，所以「#100」就表示要延遲 100 個模擬時間單位

3.1.7 「2AND」電路的模擬結果

- 在做模擬的時候，要指定如下的參數：

Add Source	LST3_3.V、LST3_4.V
Run Time	500[ns]



▲圖 3.3 2AND 電路的模擬結果

(1) 輸入信號的變化

- 模擬結果會以時序圖表示；圖 3.3 從上而下分別是輸入信號「IN1」與「IN2」，輸出信號則為「OUT」
- 模擬結果的每一個信號都是在模組裡面的 reg 及 wire 所宣告

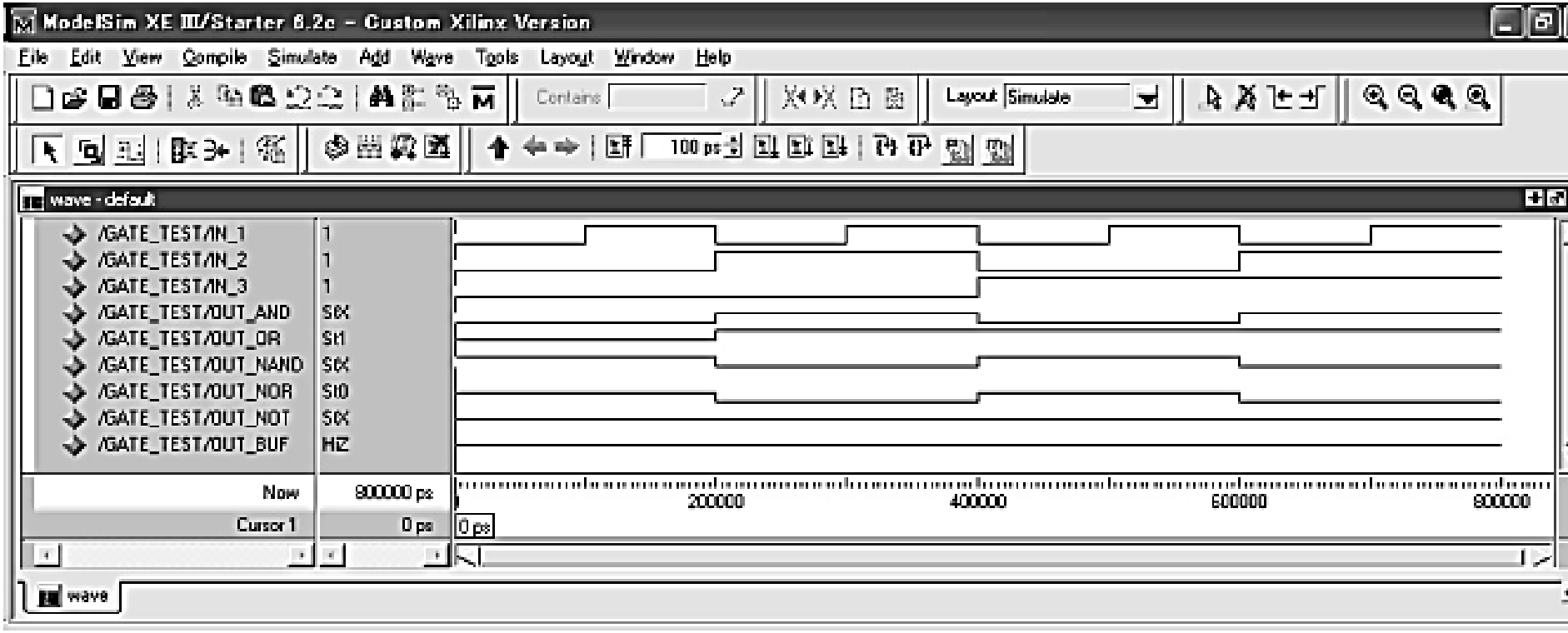
(2) 輸出信號的變化

- 輸入信號會影響輸出信號的變化
- 請觀察圖3.32AND電路模擬結果,只有輸入信號「IN1」與「IN2」同時為 '1' 的時候，輸出信號「OUT」才會為 '1'

(3) 波形表示方式

- 模擬結果的波形表示方式會根據不同的模擬器而有所不同
- 有時需要在測試平台就指定好信號名稱與順序的模擬器

3.1.8 其他電路的模擬結果



▲ 圖 3.4　基本邏輯電路的模擬結果

- 透過 parameter 來定義「STEP」等於「100」；往後出現在程式裡面的「STEP」就等同於「100」
- 為了電路合成的效率，我們可以把下層模組直接簡碼化為「GATE」
- 透過模組「GATE」呼叫複數下層模組，再根據測試平台的 reg 所宣告的信號跟下層模組做結合，下層模組的輸出信號再根據 wire 定義結合起來，以達到簡碼化的目的。

3.2 電路合成

- 模擬結果正確的程式, 不保證經過電路合成工具之後也會有正確結果
- 可以正確合成電路的描述稱為 RTL (Register Transfer Level) 描述，不一定可以正確合成電路的則稱為行為層 (behavior level) 描述 (behavior, 行為)

- 若不一定可以正確合成電路, 那行為層的存在目的為何？
  - 將整個大電路分割為各個小部分，只要針對該部分的程式的動作做確認即可。此種方式稱為綜合 (Top-down) 設計
  - 各部分開發時若進度不一, 落後的部分就先用行為層來開發； 其他的部分就先用 RTL 做整體的模擬。混用行為層與 RTL 的模擬方式稱為混合層模擬 (mix level simulation)
- 電路合成的對象並不是測試平台, 而是程式中的模擬