Verilog 硬體描述語言

Verilog HDL

A Guide to Digital Design and Synthesis

IEEE 1364-2001 Compliant

第2章 階層模組的觀念

2.1 設計方法

- 在數位系統設計中有兩種基本的方法:由下而上(bottom-up)與由上而下(top-down)。
- 由上而下是先定義好設計最終的目標，再於細部的分析構成最終目標所需的各個部份，再由各部份繼續向下細分，一直到不能再劃分出最基本的元件為止。

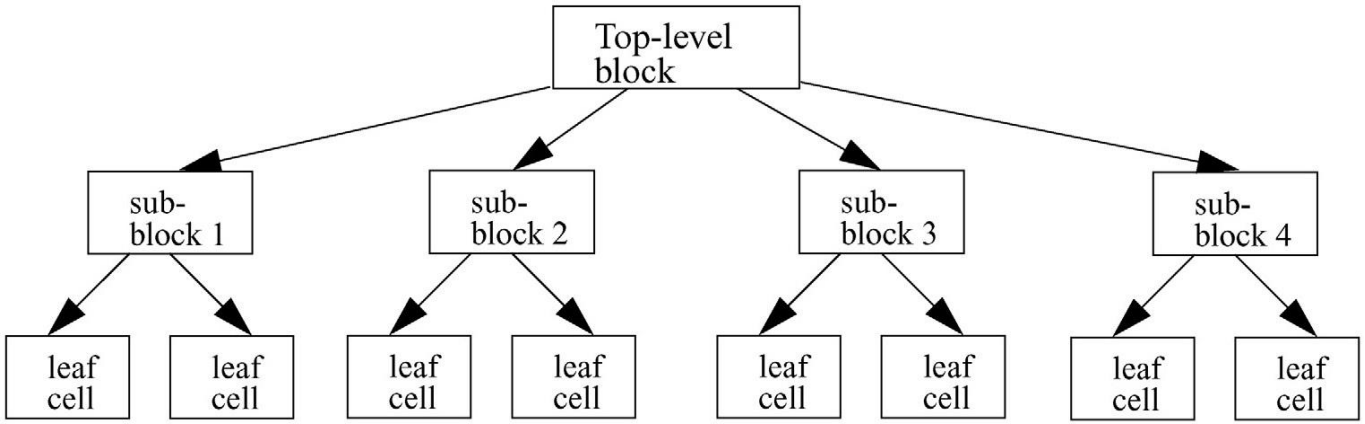


圖2-1　上而下的設計方法

2.1 設計方法

- 由下而上則是先找出可得到的基本元件，再由基本的元件一步步的向上組成更大的元件，直到完成設計的目標為止。

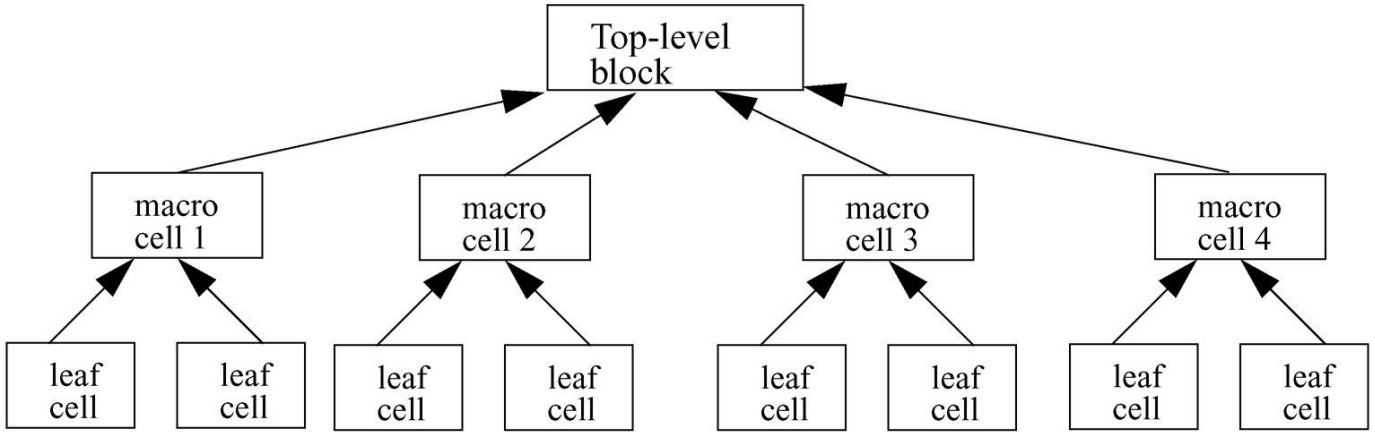


圖2-2　下而上的設計方法

2.1 設計方法

- 通常，在實際設計時是將這兩種方法混合使用。
- 首先，由系統設計師指定最終的目標，然後一方面由數位設計者，將最終的目標依照其功能細分成較小部份，一直到基本的元件為止。另一方面則由線路設計者，設計基本元件的電路，並做最佳化的處理。
- 這樣，當邏輯設計者完成細分的流程圖時，線路設計者也同時使用電晶體，將基本元件的函數庫設計完成，稱為兩種方法的交會點。
- 這樣，我們就可以使用設計完成的基本元件來組成最終的元件。

2.2 4位元漣波進位計數器

- 圖2-3是一個由負緣觸發的T型正反器(T_FF)所組成的漣波進位計數器(Ripple Carry Counter)。

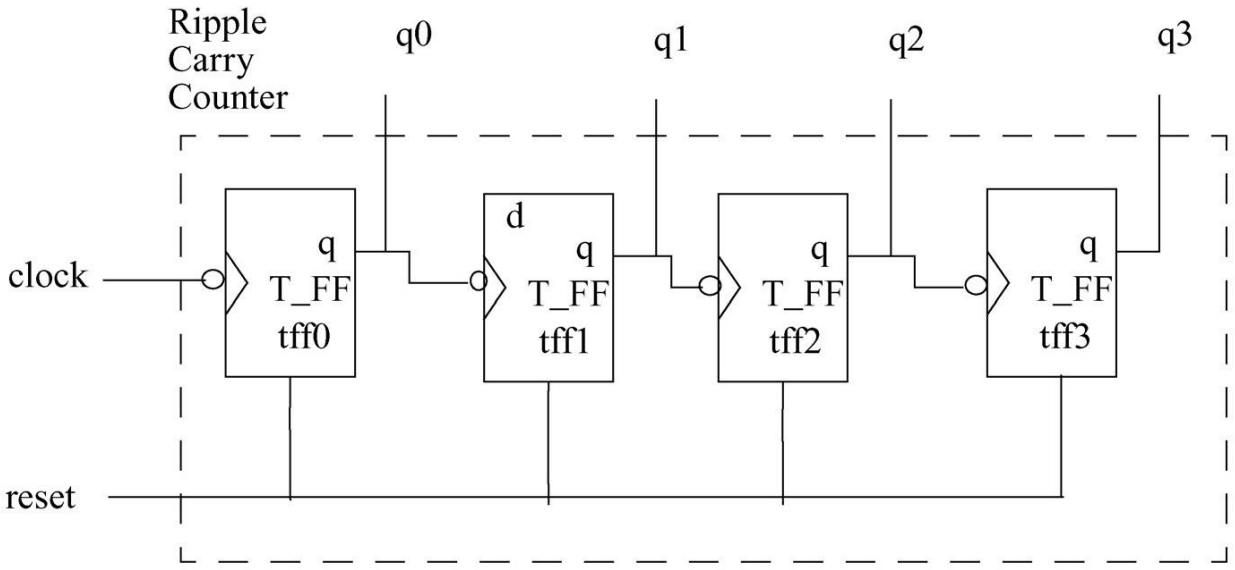


圖2-3　漣波進位計數器

2.2 4位元漣波進位計數器

- 圖2-3當中的每個T型正反器，是由一個負緣觸發的D型正反器，和一個反閘所組成，如圖2-4所示。
- 此處所用的D型正反器假設無q_bar的輸出訊號。

reset	q_n	q_{n+1}
1	1	0
1	0	0
0	0	1
0	1	0
0	0	0

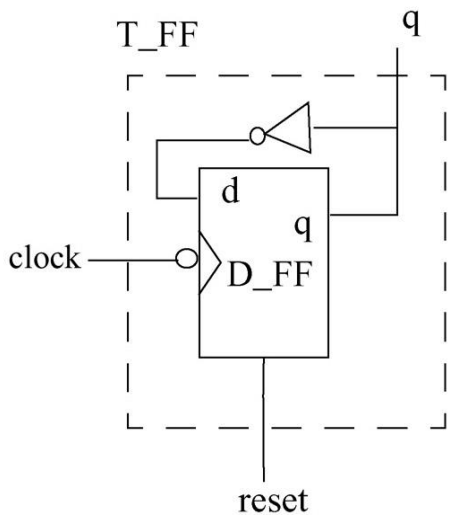


圖2-4　T型正反器

2.2 4位元漣波進位計數器

- 漣波進位計數器就由這些已建好的元件來層層的組裝起來，如圖2-5所示。
- 由下而上(bottom-up)與由上而下(top-down)的設計方法正好相反，混合使用則是同時進行。

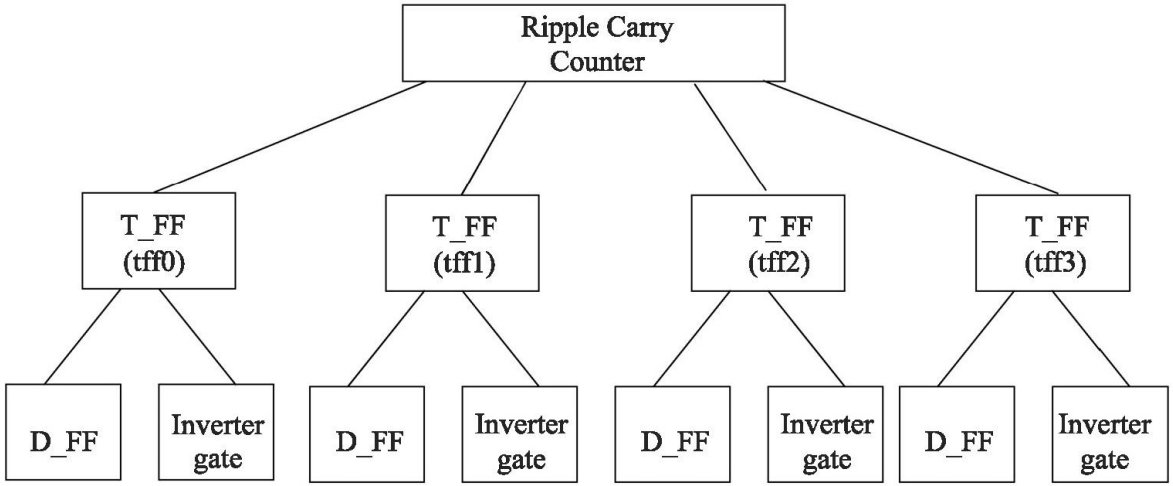


圖2-5　漣波進位計數器的階層化架構

2.3 模組(Modules)

- 現在將這種階層觀念放入Verilog中，並在其中提供一種模組(module)的架構。
- 模組是Verilog的基本組成元件，一個模組可以是一個基本元件，或是由其他的模組所組合而成。
- 我們只需藉由連接模組間的介面、輸入與輸出，就可以設計出所需要的元件，而不需要去考慮模組內部的詳細線路。在圖2-5當中，D型正反器或T型正反器就可以當成一個模組。

```
module  T_FF(q, clock, reset);
    .
    .
     <functionality of T-flipflop>
    .
endmodule
```

2.3 模組(Modules)

- Verilog可以用來描述功能，或是描述架構的方式來描述一個電路。
- 對於一個模組內部的敘述，在Verilog中有四種不同的層次，設計者可以依不同的需要，使用不同層次來描述模組的功能或內部線路。
- 一個模組外部所表現出來的功能，不受內部描述所用層次的影響。
- 對一整個線路而言，每個模組內部的詳細構造，是隱藏而無法得知的。

2.3 模組(Modules)

四個層次的敘述

- 行為模式演算法層次(Behavioral or algorithmic level)
  - 在此最高層次，我們只需考慮模組功能或演算法，不須考慮詳細的硬體線路
- 資料處理層次(Dataflow level)
  - 在此層次必須要指明資料處理的方式，設計者要說明資料如何在暫存器中儲存與傳送
- 邏輯閘層次(Gate level)
  - 模組是由邏輯閘連接而成，設計工作如同描繪邏輯閘的設計方式
- 低階交換層次(Switch level)
  - 在此最低層次，線路是由開關與儲存點所組成，此層次設計者必須了解電晶體元件特性

2.3 模組(Modules)

- 在實際的設計中，我們可以四個層次混和使用。
- 結合了行為層次與資料處理層次，且能做合成處理的暫存器轉移層次(RTL)，這種方式在業界中被廣泛的使用。
- 雖然四個層次可以混和使用，但設計完成後大部份的模組之描述層次，皆被邏輯閘層次所取代。
- 一般而言，對於越高階層次的敘述，其修改的空間就越大，且無關於製程技術，越低階則相反。
- 在低階的設計中，通常小部份的修改，所牽動的影響是可觀的。
- 高階與低階的差異好像C語言和組合語言，C有較大可程式化與可攜性，組合語言就相反。

2.4 別　名

- 從模組創造出物件的過程稱為模組取別名，而從模組中所造出的物件，則稱為此模組的別名。
- **例題2-1 模組的別名**

```
// 定義一個名為漣波進位計數器的最上層模組，並在內部取用四個 T 型正反
// 器的別名，其連接方法如 2-2 節中所述。
module ripple_carry_counter(q,clk,reset);

output [3:0] q;     // 輸出與輸入訊號的宣告
input clk,reset;    // 輸出與輸入訊號的宣告

// 創造四個 T 型正反器的別名，並且每個都具有自己的名稱，並輸入一組
// 訊號，這四個別名都是從 T 型正反器的模組中建立出來的。
T_FF tff0(q[0],clk,reset);
T_FF tff1(q[1],q[0],reset);
T_FF tff2(q[2],q[1],reset);
T_FF tff3(q[3],q[2],reset);

endmodule

// 定義 T 型正反器模組，並在內部引用 D 型正反器模組的別名，其構造如
// 圖 2-4 所示。
module T_FF(q,clk,reset);

// 定義輸出與輸入訊號
output q;
input clk , reset;
wire d;
D_FF dff0(q, d, clk, reset); // 取用 D 型正反器模組的別名，
                             // 並取名為 dff0。
not n1(d, q); // 反閘是一個在 Verilog 中提供的基本的元件，這在
              // 往後會解釋。

endmodule
```

2.4 別　名

- Verilog不允許巢狀的模組宣告，不可以在定義一個模組時，內部包含有另一對module和endmodule的宣告。
- 模組的定義與模組的別名是不同的。
- **例題2-2 非法的巢狀模組宣告**

```
// 定義一個名為 ripple_carry_counter 的模組，並在內部非法的宣告
// 一個 T 型正反器的模組。
module ripple_carry_counter(q,clk,reset);
output [3:0] q;
input  clk, reset;
   module T_FF(q,clock,reset); // 非法的巢狀模組宣告
   ...
   <module T_FF internals>
   ...
   endmodule // 結尾
endmodule
```

2.5 模　擬

- 設計區塊完成後，接下來就是測試的工作。
- 在Verilog中測試的工作是用一個觸發區塊(Stimulus Block)來驅動設計區塊，並檢查輸出結果是否符合要求?
- 將觸發區塊與設計區塊分開是較好的做法。
- 觸發區塊又稱為測試程式(test bench)，通常若要完整的測試設計區塊，需要有許多不同的測試程式才行。
- 兩種常見的觸發區塊
  - 在區塊內使用設計模組的別名，並直接驅動這些別名的輸入訊號，測試區塊為最上層的區塊
  - 將觸發區塊當成是訊號產生器一般，另建一個模組，在模組中使用設計模組及觸發模組的別名

2.5 模　擬

- 第一種:直接驅動設計區塊的輸入訊號clk與reset，檢查輸出訊號q，並將結果展示在螢幕上。

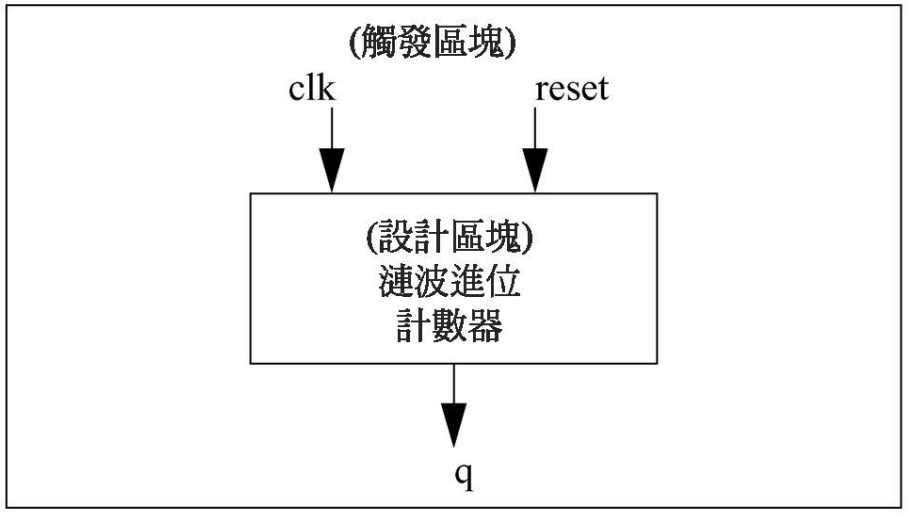


圖2-6 引用設計區塊的模擬區塊

2.5 模　擬

- 第二種:另建一個模組，在模組中使用設計模組及觸發模組的別名，並將觸發區塊的輸出訊號(d_clk與d_reset)輸入到設計區塊的輸入(clk與reset)，藉由c_q將結果展示在螢幕上。

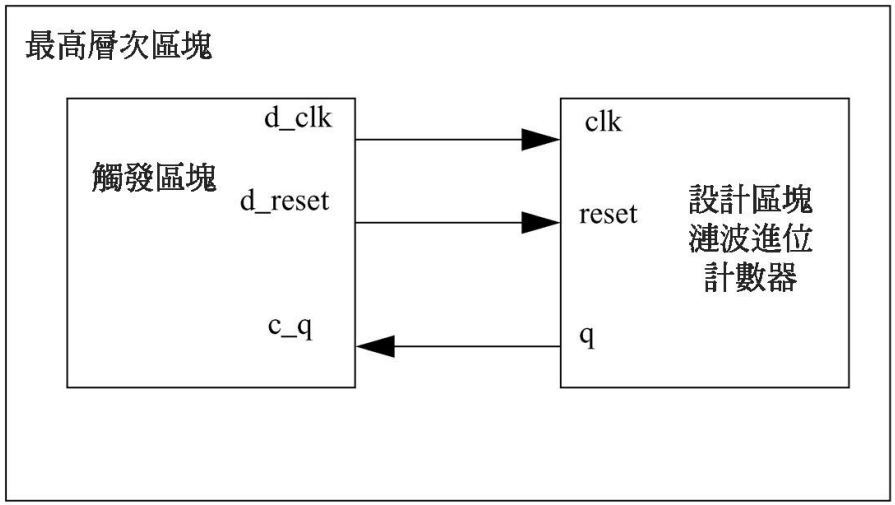


圖2-7　在一個多餘的最高層次區塊中引用設計區塊與模擬區塊

2.6 例　題

- 2.6.1 設計區塊
- 例題2-3 漣波進位計數器(由上而下)
- 先用 Verilog設計最上層的漣波進位計數區塊

```
module ripple_carry_counter(q,      clk,      reset);

output    [3:0]    q;
input      clk,    reset;

//  創造四個正反器的別名，並且每個都有自己的名稱，並輸入一組訊號，
    這四個別名都是從 T 型正反器的模組中建立出來的。

T_FF tff0(q[0] , clk , reset);
 T_FF tff1(q[1] , q[0] , reset);
 T_FF tff2(q[2] , q[1] , reset);
 T FF tff3(q[3] , q[2] , reset);

endmodule
```

2.6 例　題

- 2.6.1 設計區塊
- 例題2-4 T型正反器
- 再用四個T_FF模組的別名，下面定義T_FF模組

```
module T_FF(q, clk, reset);
output q;
input clk, reset;
wire d;
D_FF dff0(q, d, clk, reset);
not n1(d, q); // 反閘為 Verilog 中的主要元件
endmodule
```

2.6 例　題

- 2.6.1 設計區塊
- 例題2-5 D型正反器
- 因為T_FF模組使用D_FF模組的別名，下面定義D_FF模組

```
// 這裡的 D 型正反器的 reset 訊號是同步的。
module D_FF(q, d, clk, reset);

output q;
input d, clk, reset;
reg q;

// 下面有許多不認識的指令，在這裡先不要管他，只要專心於如何建立上
// 到下的設計方塊即可。
always @(posedge reset or negedge clk)
if(reset)
   q = 1'b0;
// 有同步 reset 訊號的 D 型正反器模組
else
   q = d;

endmodule
```

2.6 例　題

- **2.6.2 觸發區塊(Stimulus Block)**
- 本節將設計觸發區塊來檢驗計數器是否正常工作?我們將控制輸入訊號clk與reset來檢驗計數器的計數功能與重置功能， clk與reset的輸入波型如圖2-8所示。
- clk週期為10個時間單位，reset在0-15與195-205為1其餘為0，期望輸出q如下圖。

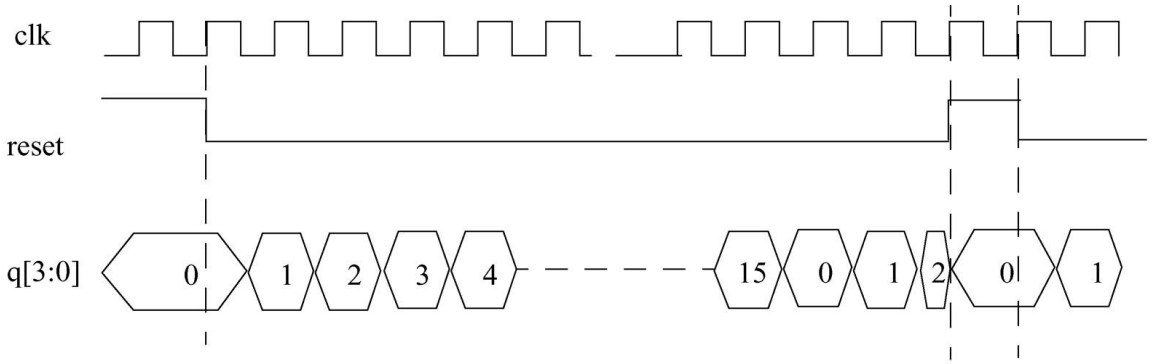


圖2-8　觸發與輸出波形

2.6 例　題

- 2.6.2 觸發區塊(Stimulus Block)
- 例題2-6 觸發區塊

```
module stimulus;

reg clk;
reg reset;
wire [3:0] q;

// 引用設計方塊
ripple_carry_counter r1(q, clk, reset);

// 控制驅動設計方塊的 clk 訊號週期為 10
initial
   clk = 1'b0;     // 將 clk 訊號設為 0
always
   #5 clk = ~clk; // clk 訊號每隔 5 個時間單位反相一次

// 控制驅動設計方塊的reset訊號，在時間 0-20 及 200-220 時，為 1 其
// 時段為 0。
```

```
initial
begin
  reset  =  1'b1;
  #15 reset = 1'b0;
  #180 reset = 1'b1;
  #10 reset = 1'b0;
  #20 $finish;   // 模擬結束
end

// 將結果輸出到螢幕上
initial
   $monitor($time,"Output q = %d",q);

endmodule
```

2.6 例　題

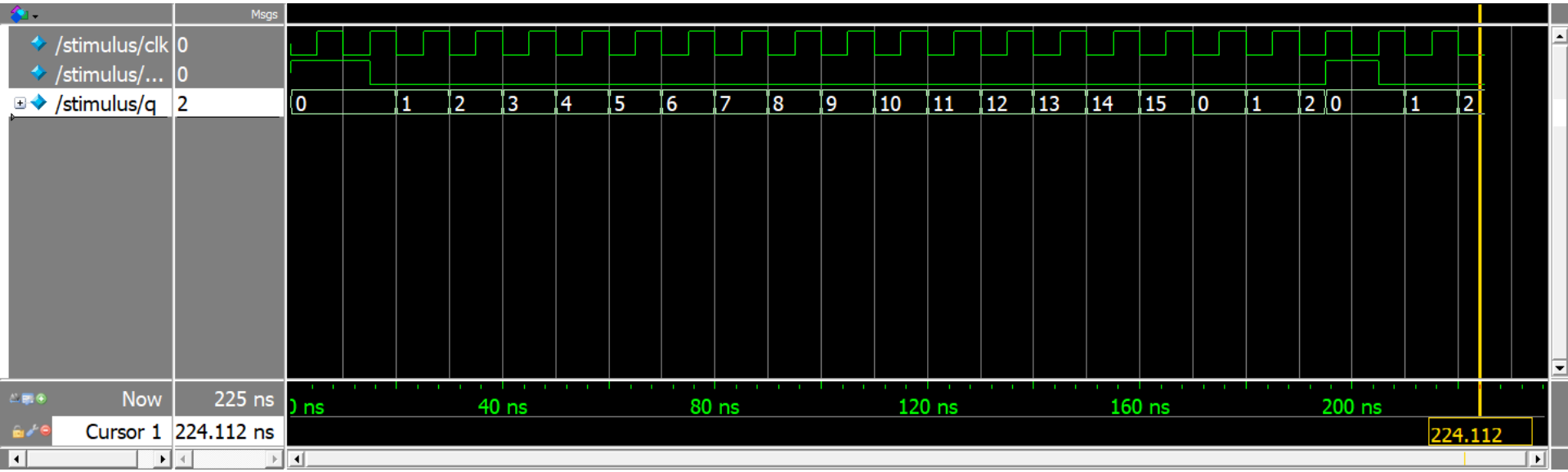
- 2.6.2 觸發區塊(Stimulus Block)
- 模擬結果

```
  0  Output q =  0
 20  Output q =  1
 30  Output q =  2
 40  Output q =  3
 50  Output q =  4
 60  Output q =  5
 70  Output q =  6
 80  Output q =  7
 90  Output q =  8
100  Output q =  9
110  Output q = 10
120  Output q = 11
130  Output q = 12
140  Output q = 13
150  Output q = 14
160  Output q = 15
170  Output q =  0
180  Output q =  1
190  Output q =  2
195  Output q =  0
210  Output q =  1
220  Output q =  2
```

2.6 例　題

2.1
2.2
2.3
2.4

- 2.6.2 觸發區塊(Stimulus Block)
- 模擬結果



2.8 習　題

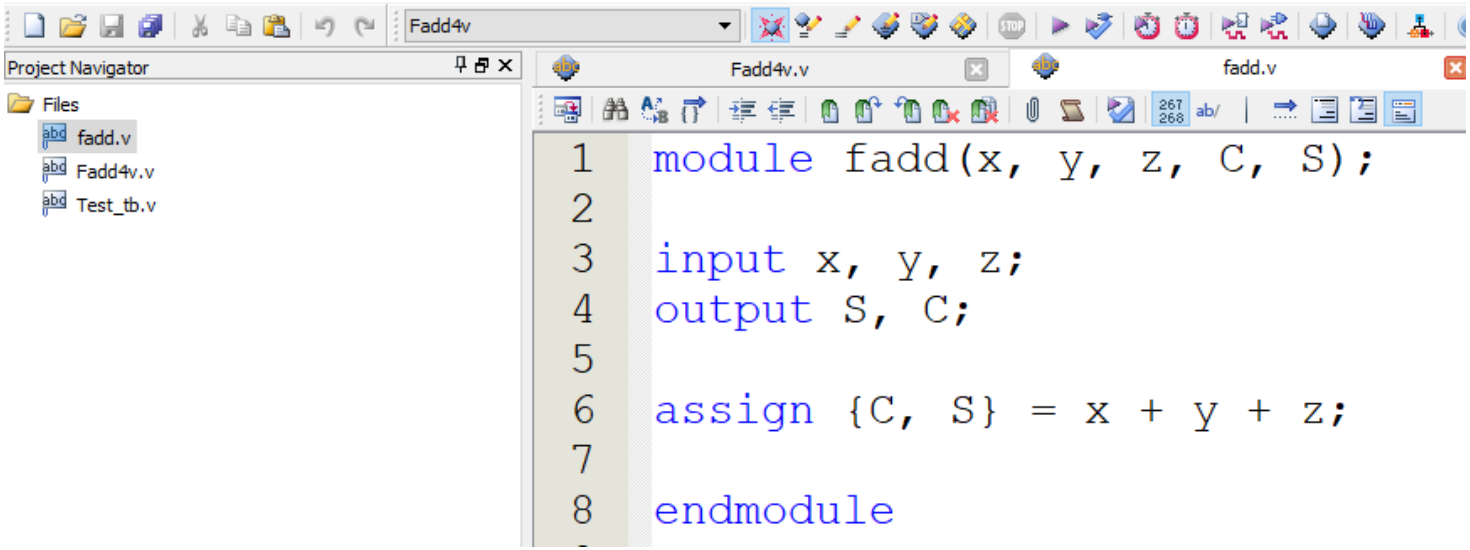
(習題一)撰寫Verilog電路程式專案 Fadd4v

一個4位元的漣波進位加法器(Ripple_Add)包含四個1位元的全加器:

- 定義一個全加器模組(fadd.v)
- 定義一個4位元的漣波進位加法器模組(Fadd4v.v)，在內部引用四個全加器模組的別名(fa0, fa1, fa2, fa3)
- 撰寫測試程式檔Test_tb.v進行模擬測試(RTL Simulation)
  (可在Quartus II 或 ModelSim-Altera直接模擬)
- End simulation time at 500 ns
- 資料可在上課投影片中查詢

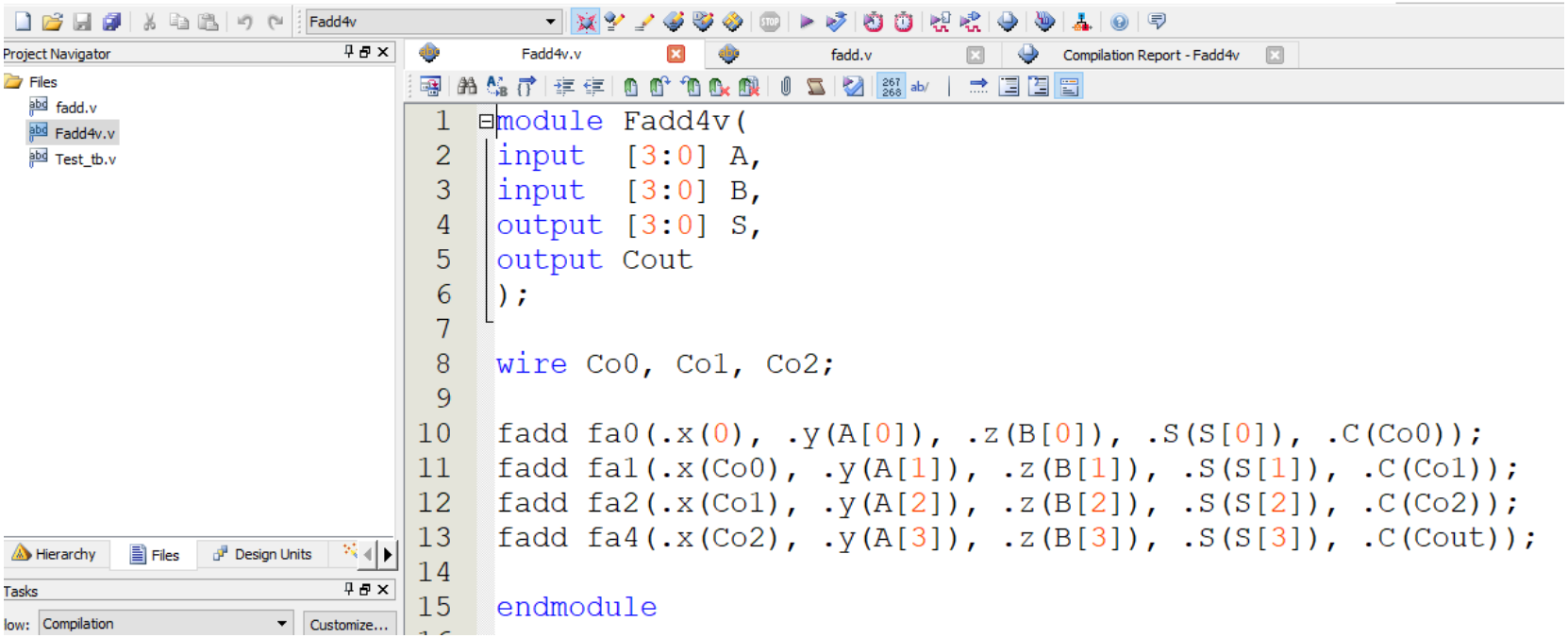
2.8 習　題

參考答案：fadd.v



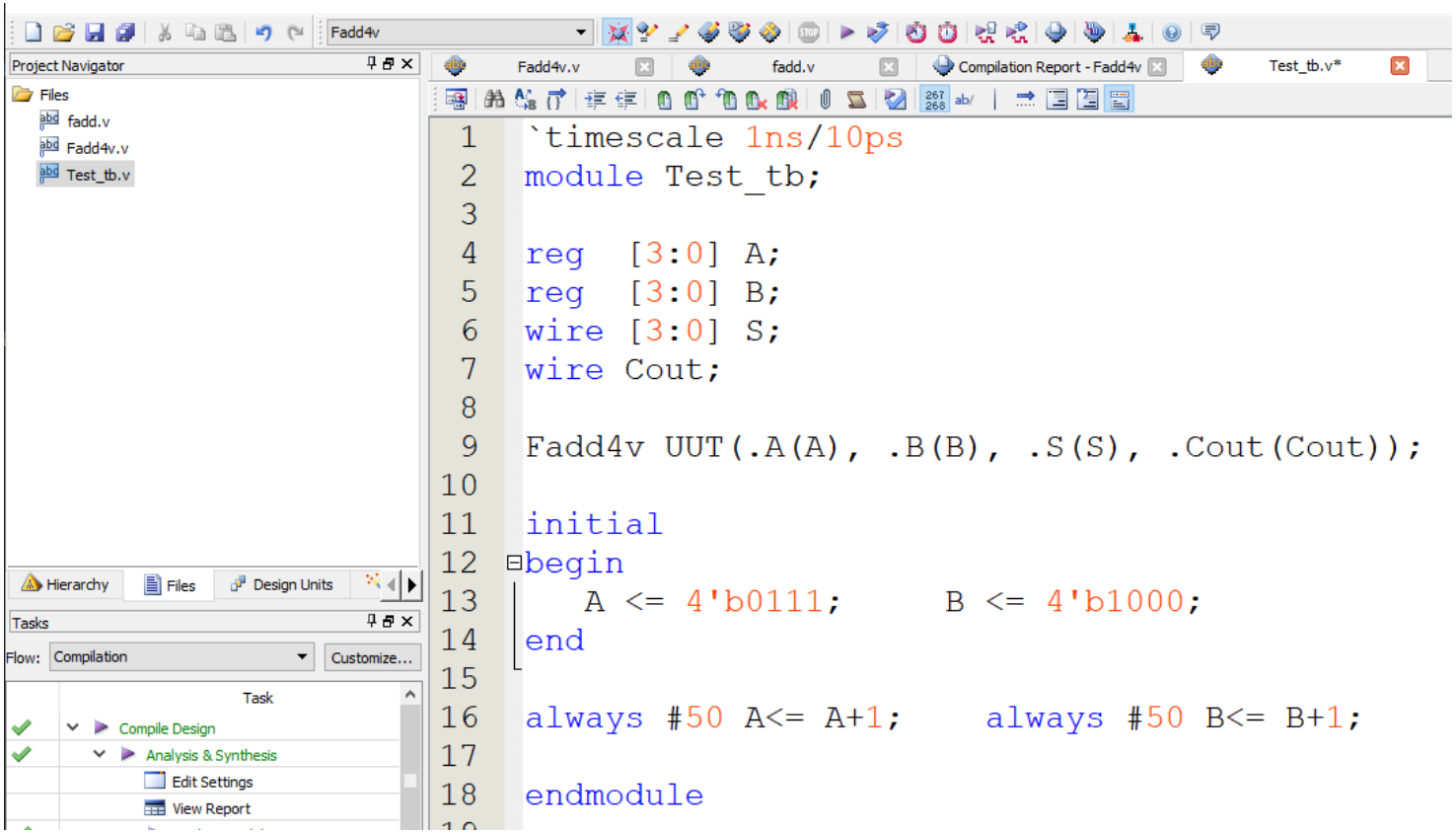
2.8 習　題

參考答案：Fadd4v.v



2.8 習　題

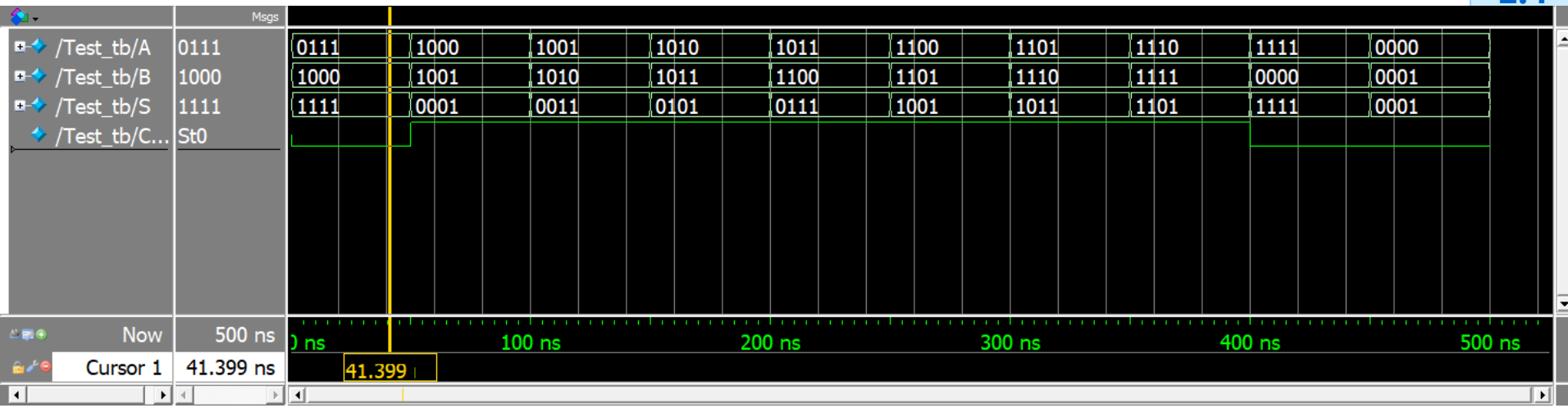
參考答案：Test_tb.v



2.8 習　題

2.1
2.2
2.3
2.4

參考答案：ModelSim 輸出圖



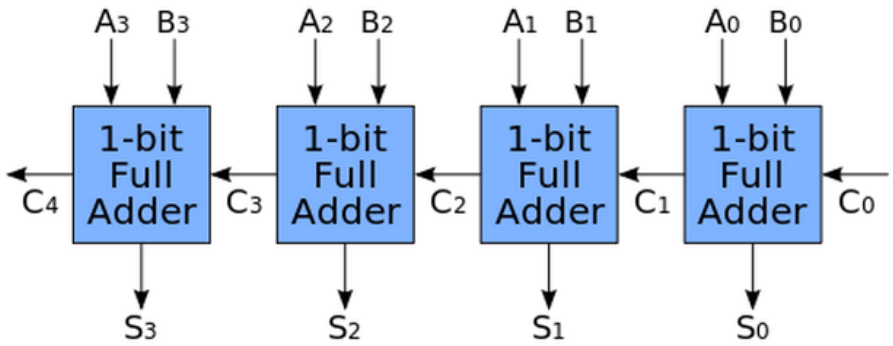
2.8 習　題

(習題二) 撰寫Quartus II電路繪圖專案 Fadd4b

- 定義一個全加器模組(fadd.v)，編譯無誤後先製作Symbol電路符號fadd備用
- 新增一個電路繪圖檔Fadd4b.bdf，在內部引用四個全加器模組 fadd symbol(如次頁圖)
- 使用之前之測試程式檔Test_tb.v進行模擬測試(Gate Level Simulation)，應得出相同結果

(只能在Quartus環境下的 ModelSim-Altera 模擬)

- End simulation time at 500 ns
- 資料可在上課投影片中查詢



2.8 習　題

(習題二) Quartus II電路繪圖專案

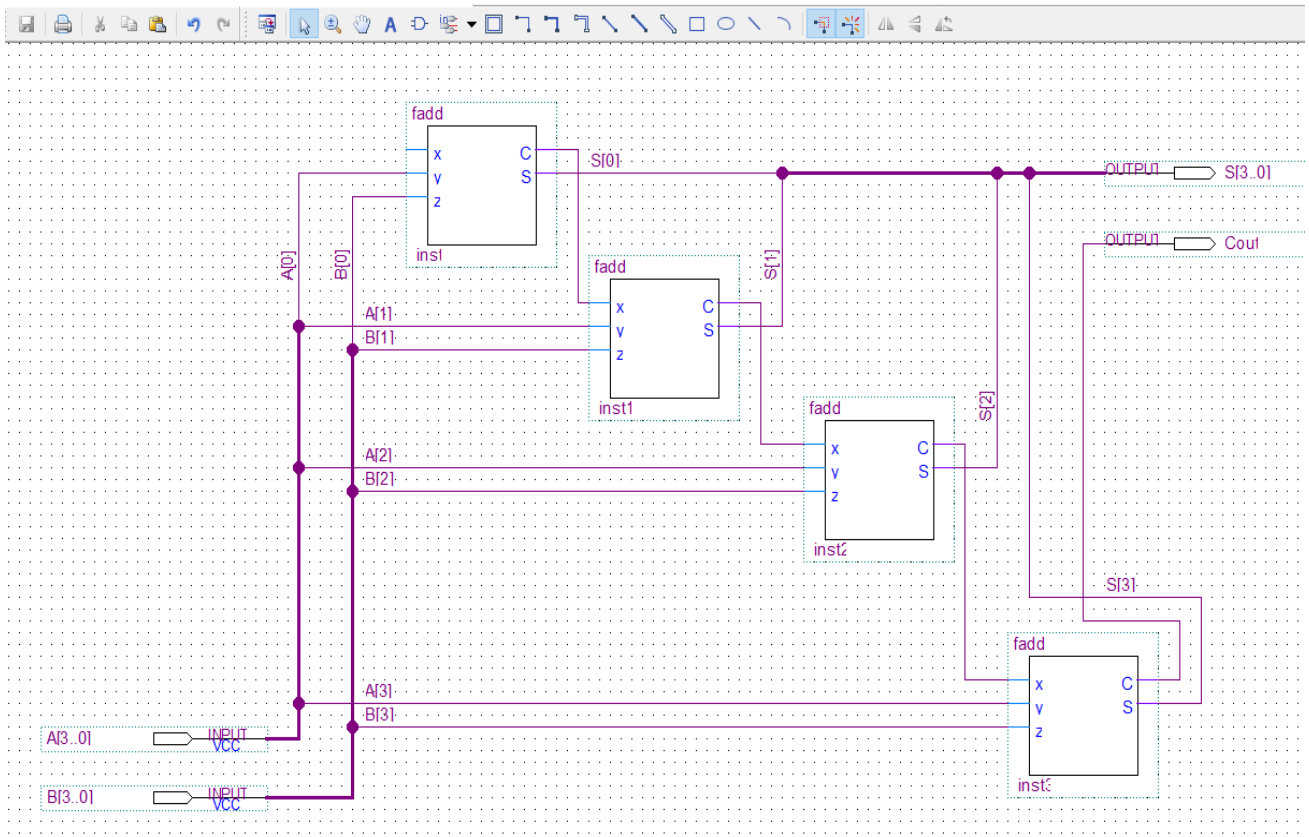
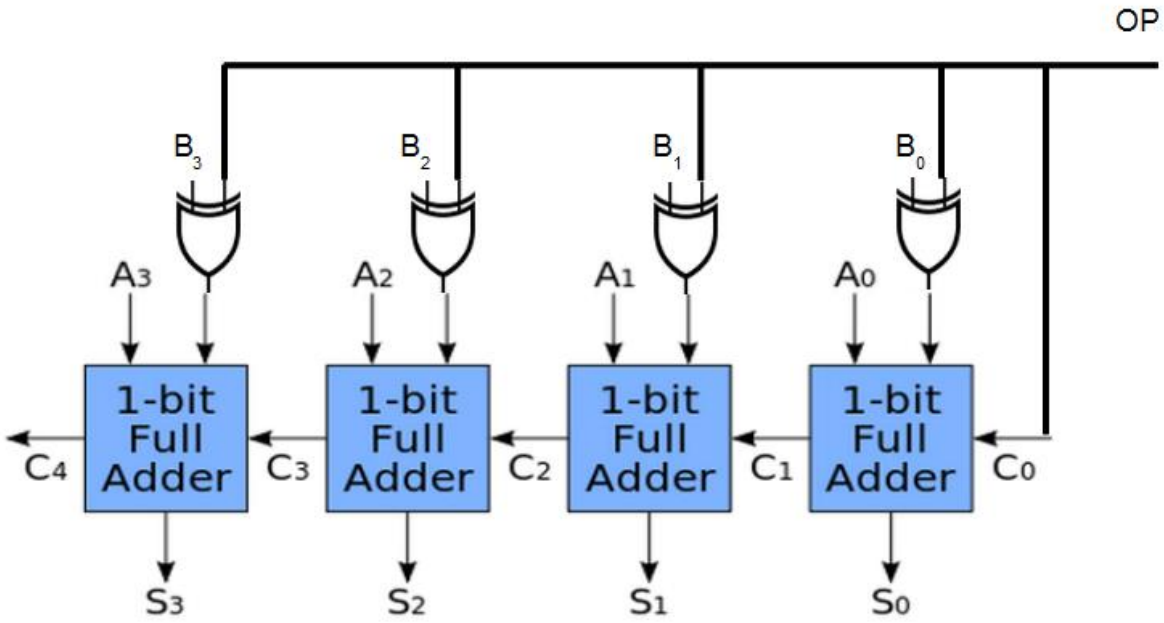


圖 3-53 使用 fadd symbol符號串接

2.8 習　題

(習題三) 設計一個4位元的漣波加/減法器模組

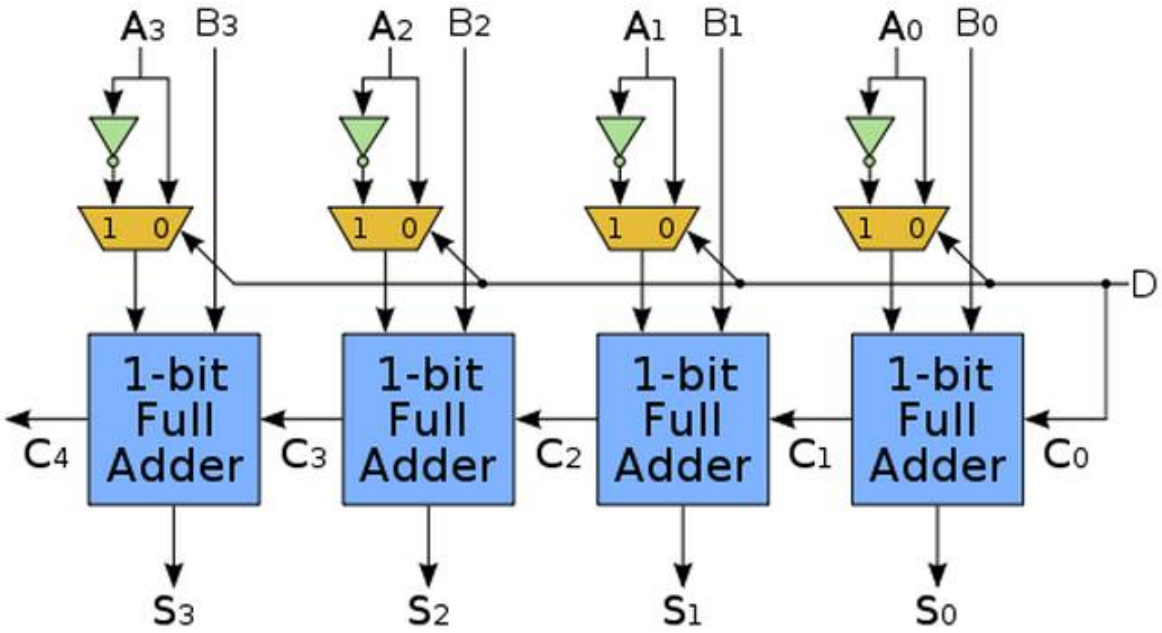
參考來源: Verilog (3) – 組合邏輯電路 (作者：陳鍾誠)



使用 XOR 控制的二進位加/減器電路

2.8 習　題

(習題三) 設計一個4位元的漣波加/減法器模組



採用 2 選 1 多工器控制的加減器電路