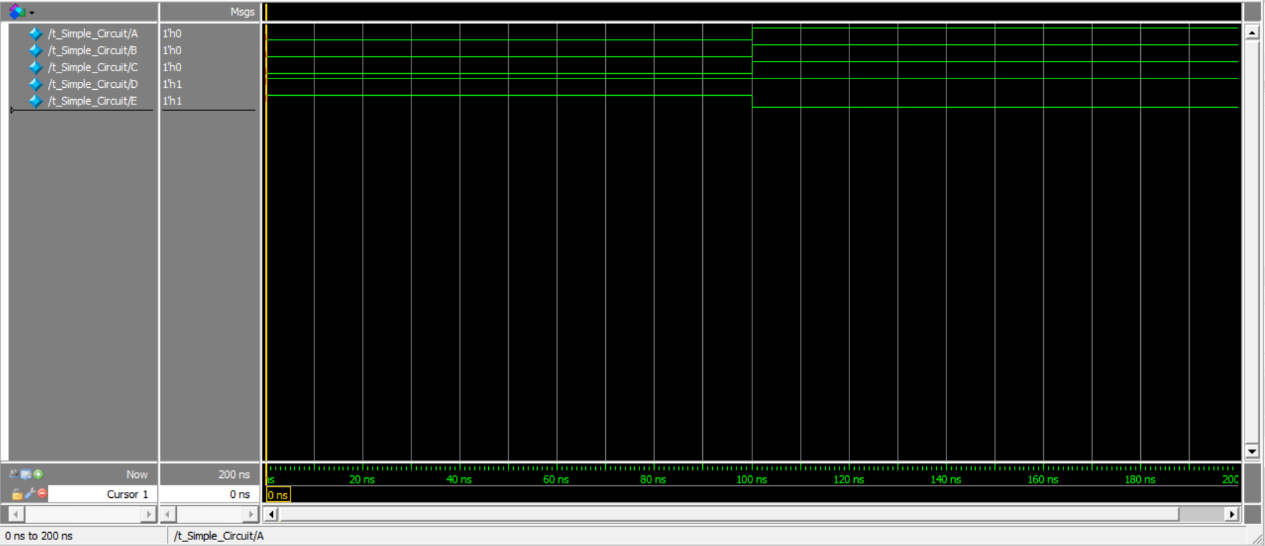
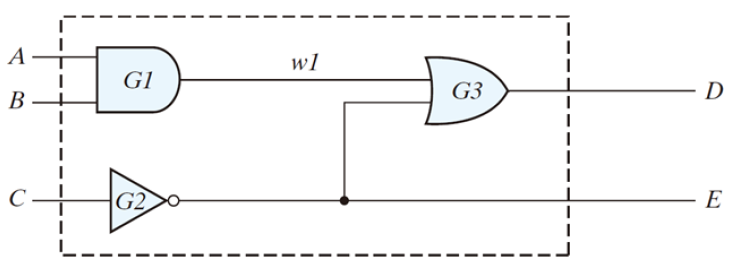
1. A (a)

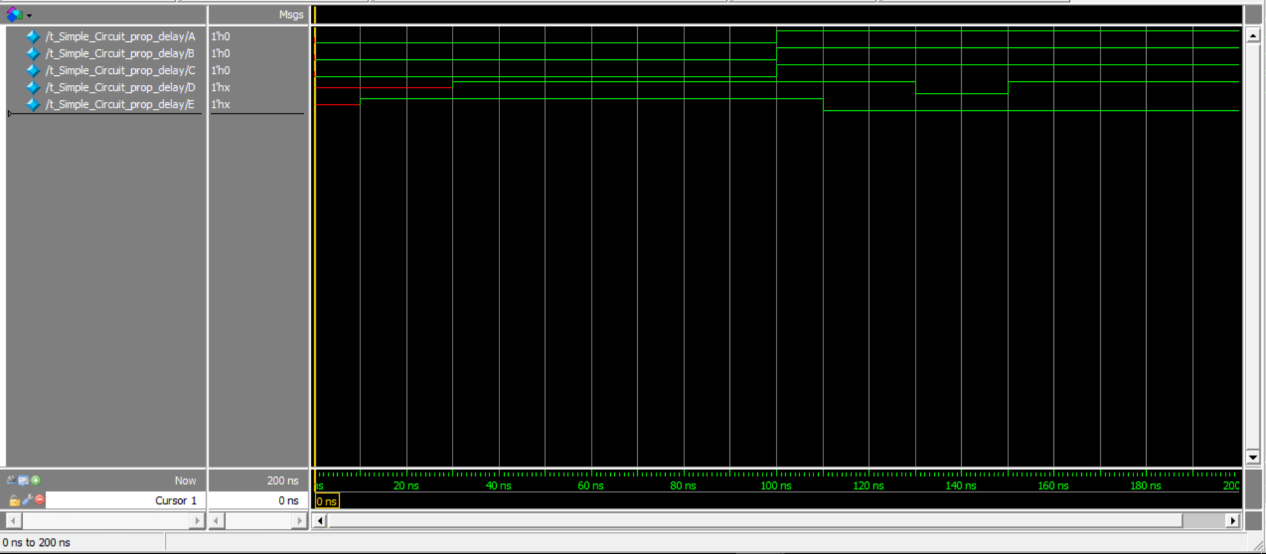
Simple\_Circuit波形圖



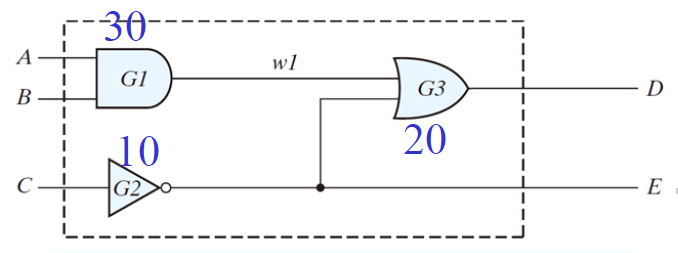
Simple\_Circuit電路圖



Simple\_Circuit\_prop\_delay波形圖



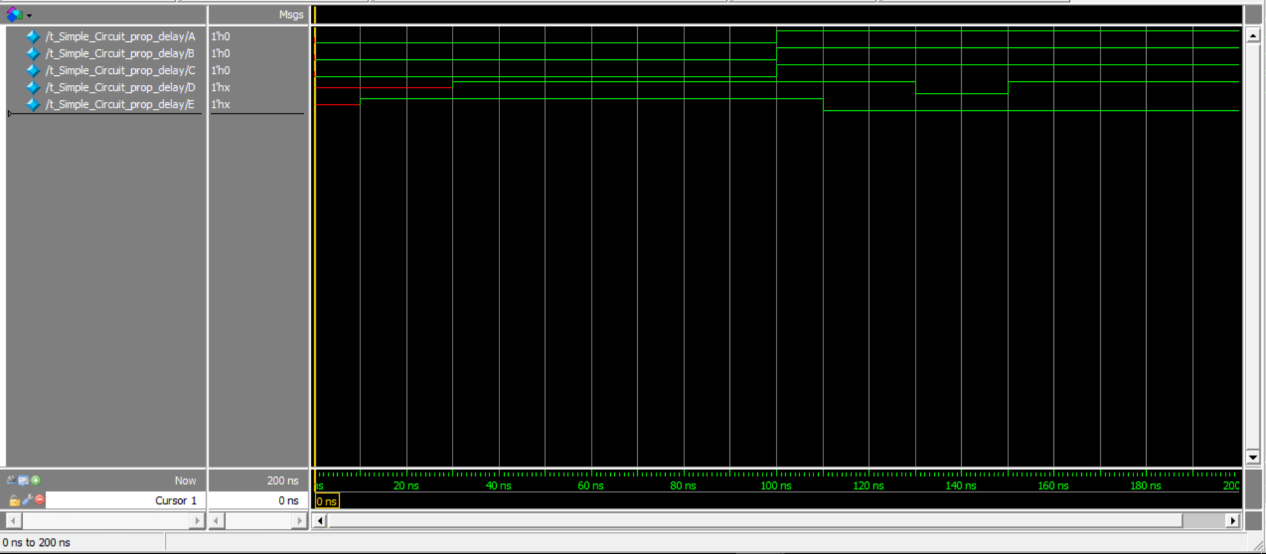
Simple\_Circuit\_prop\_delay電路圖



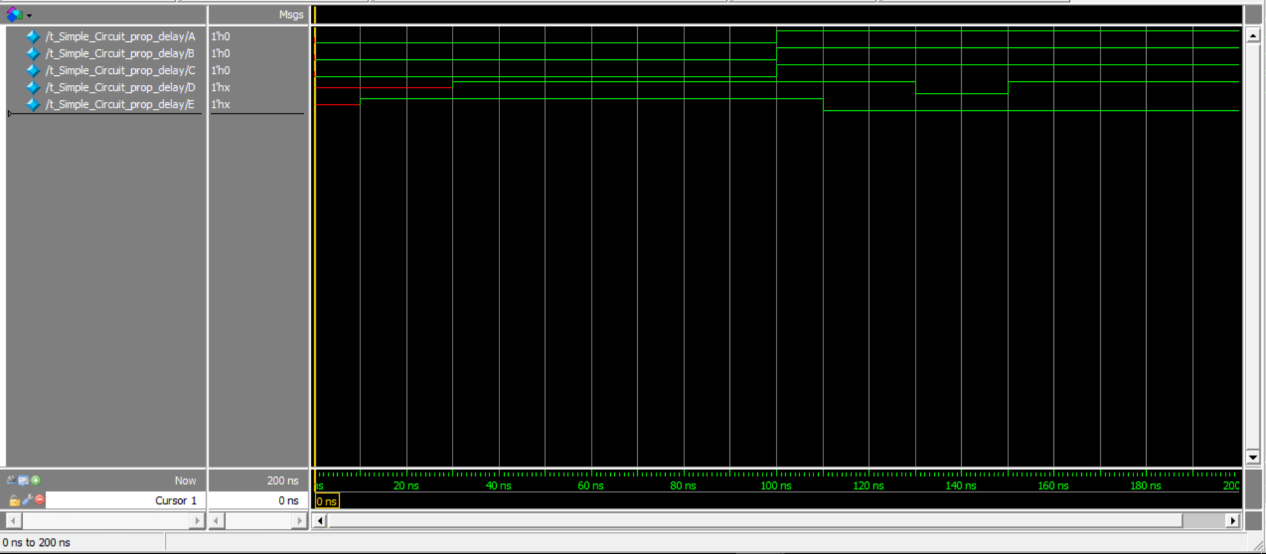
比較有無gate delay的波形圖差別。發現有gate delay時，一開始輸入(A, B, C) = (0, 0, 0)，但輸出並不會立即改變，因為G2有10ns的延遲，所以E在10ns時才能確定正確值為1。而在30ns時D能確定為1，是因為E在10ns時確定值之後，再經過G3延遲20ns。在100ns時改變輸入為(1, 1, 1)，再經過10ns後E才確定為0，再經過20ns後w1確定為1。但D變為0，因為E和w1先前的值經過G3所得的值為0，而再經過20ns後w1和E經過G3輸出1。

1. A (b)

Simple\_Circuit\_prop\_delay波形圖(修改前)



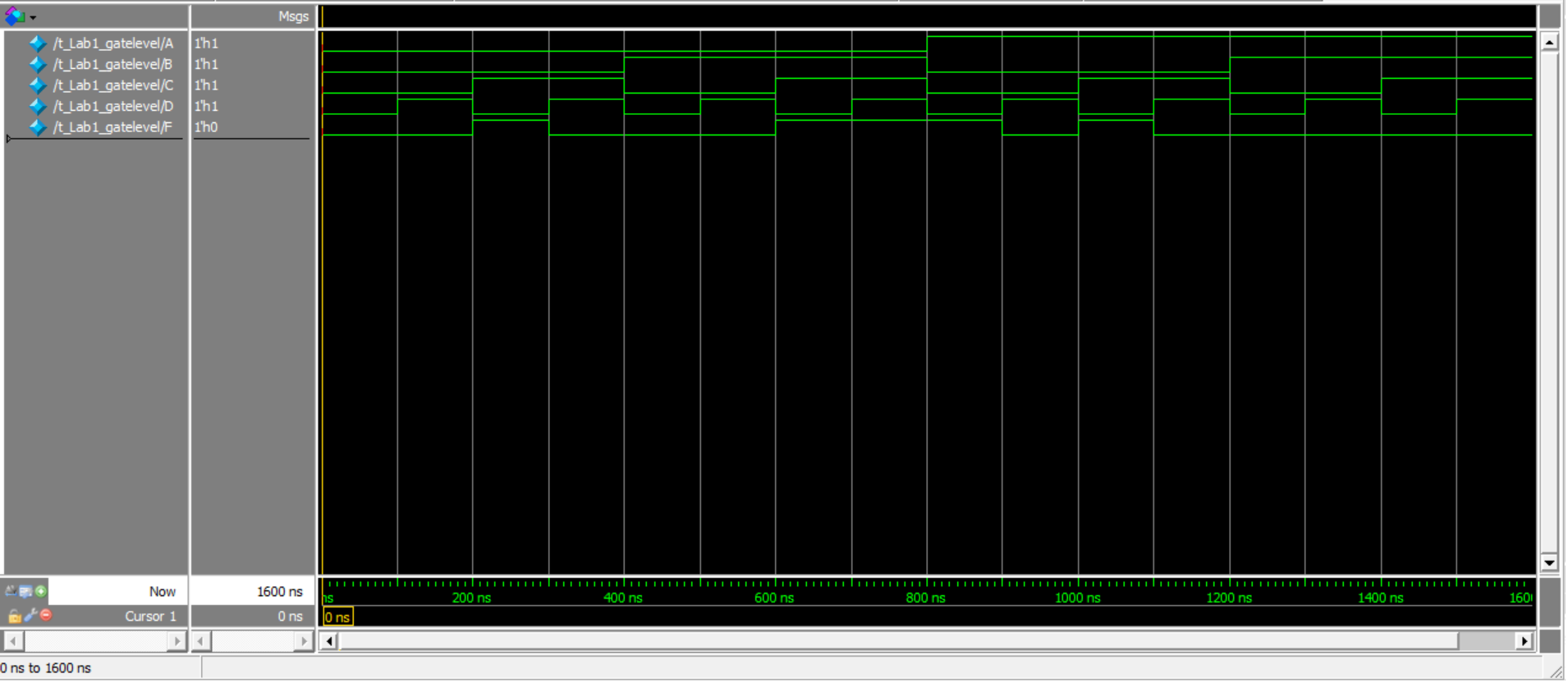
Simple\_Circuit\_prop\_delay波形圖(修改後)



若將not與or兩行敘述互換並不影響模擬出來的波形圖，因為Verilog為一種硬體描述語言，是用來描述一個電路的特性，並非一般程式語言是程序式程式以逐步執行的方式執行。

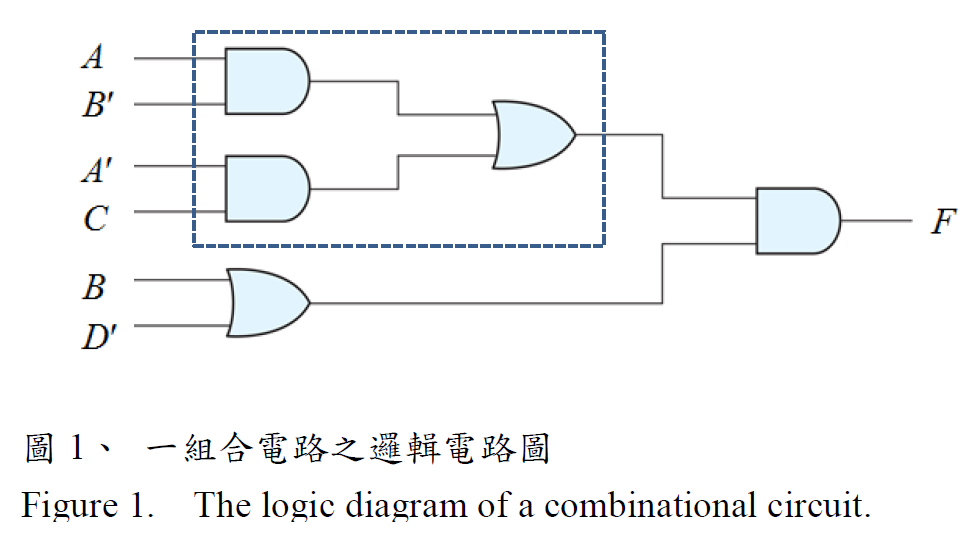
1. B (a)

Lab1\_gatelevel波形圖



輸入為(A, B, C, D)，輸出為F。以波形圖來看在輸入組合為(0, 0, 1, 0)、(0, 1, 1, 0)、(0, 1, 1, 1)、(1, 0, 0, 0)、(1, 0, 1, 0)的情況下輸出為1，若以(A, B, C, D)的順序作為minterm的編號，則此波形圖可表示為。

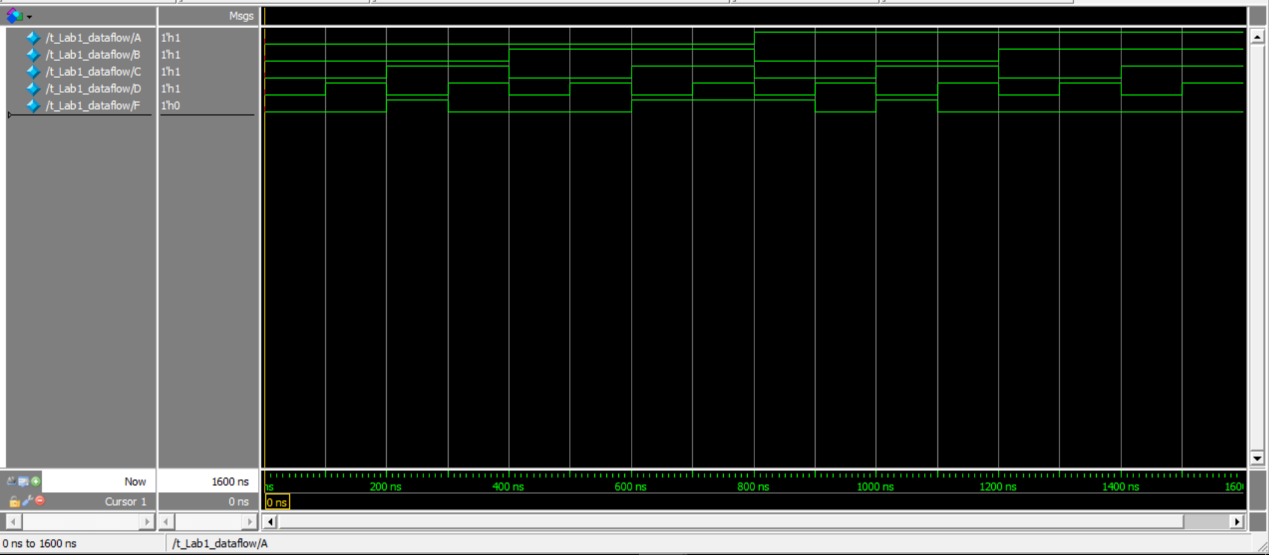
模擬的電路圖



輸出F = (AB’ + A’C)(B + D’) = A’BC + AB’D’ + A’CD’，輸入(A, B, C, D) = (0, 1, 1, -)、(1, 0, -, 0)、(0, -, 1, 0)時輸出為1，則所對應的sum of minterm也是，所以模擬的波形圖正確。

1. B (b)

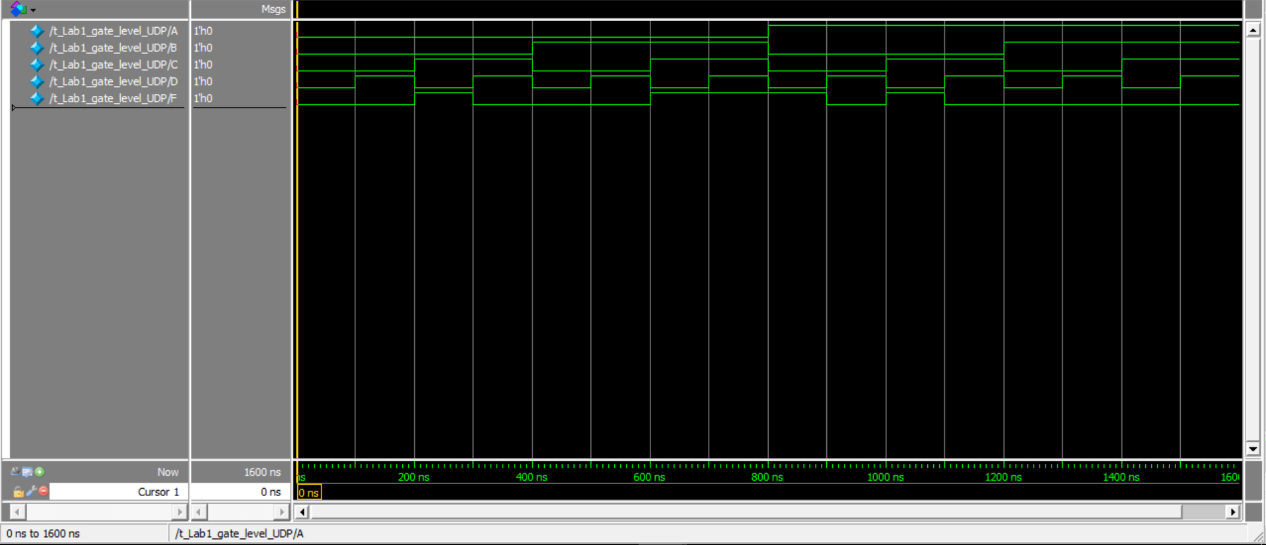
Lab1\_dataflow波形圖



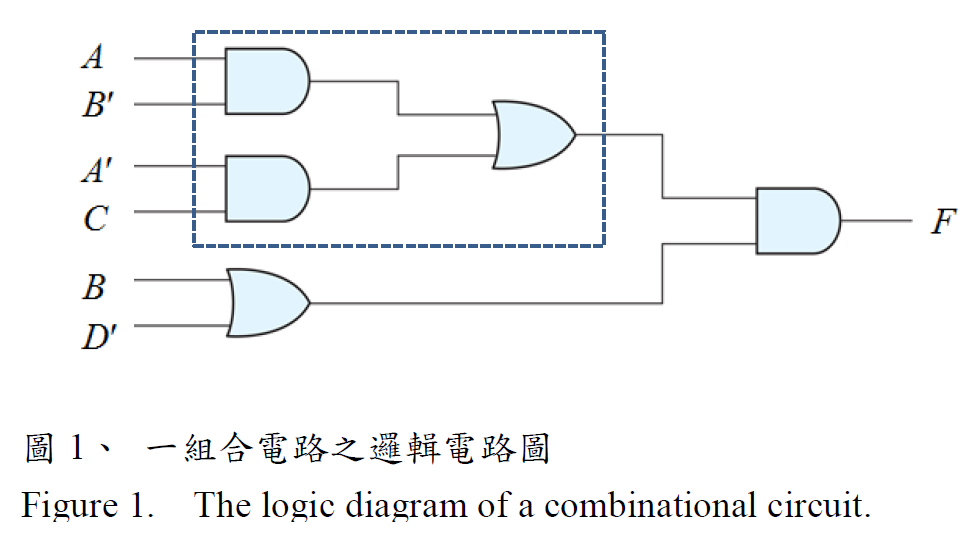
波形圖同B (a)，所以此波形圖可表示為。模擬的電路圖寫成布林函數的形式為F(A, B, C, D) = ((A && !B) || (!A && C)) && (B || !D)化簡後求出sum of product得!A && B && C || A && !B && !D || !A && C && !D，所對應的sum of minterm為與模擬的波形圖所得結果相符，故正確。

1. B (c)

Lab1\_gate\_level\_UDP波形圖



波形圖同B (a)，所以此波形圖可表示為。



首先將圖1中虛線框內之電路撰寫成user-defined primitive，tmp = AB’ + A’C真值表為

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | C | AB’ | A’C | tmp |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 |

然後利用設計好的user-defined primitive兜出完整的電路，F = tmp(B + D’)真值表為

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | tmp | B + D’ | F |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 |

由真值表可得與波形圖所顯示之結果相同，所以利用user-defined primitive設計出來的電路正確。

1. 在這次的作業中，我學到了如何把一張邏輯電路圖以gate-level modeling、dataflow modeling和自定user-defined primitive的方式實作，並撰寫測試模組去測試設計出來電路的正確性。但在編譯執行及畫出波形圖的過程中，我覺得ModelSim所提供的介面不是很好操作使用，反而是利用Icarus Verilog和GTKWave來編譯執行及顯示波形圖較為簡潔易懂，僅需下一兩行指令即可完成模擬和顯示波形圖。(使用說明參考[如何在Mac OS X上安裝Verilog環境](http://easonchang.logdown.com/posts/649863))目前遇到的問題是不知道有沒有可以直接列出真值表所有可能的語法來生成測試模組，如此可以增加程式可讀性也降低了打錯輸入測資的可能。另外最近讀到一篇文章[MIT生物工程學家研發出控制細菌的程式語言](https://panx.asia/archives/46286)，覺得Verilog的應用真的非常廣泛而且可以利用電路是同時運作的特性來加快求解問題的速度。