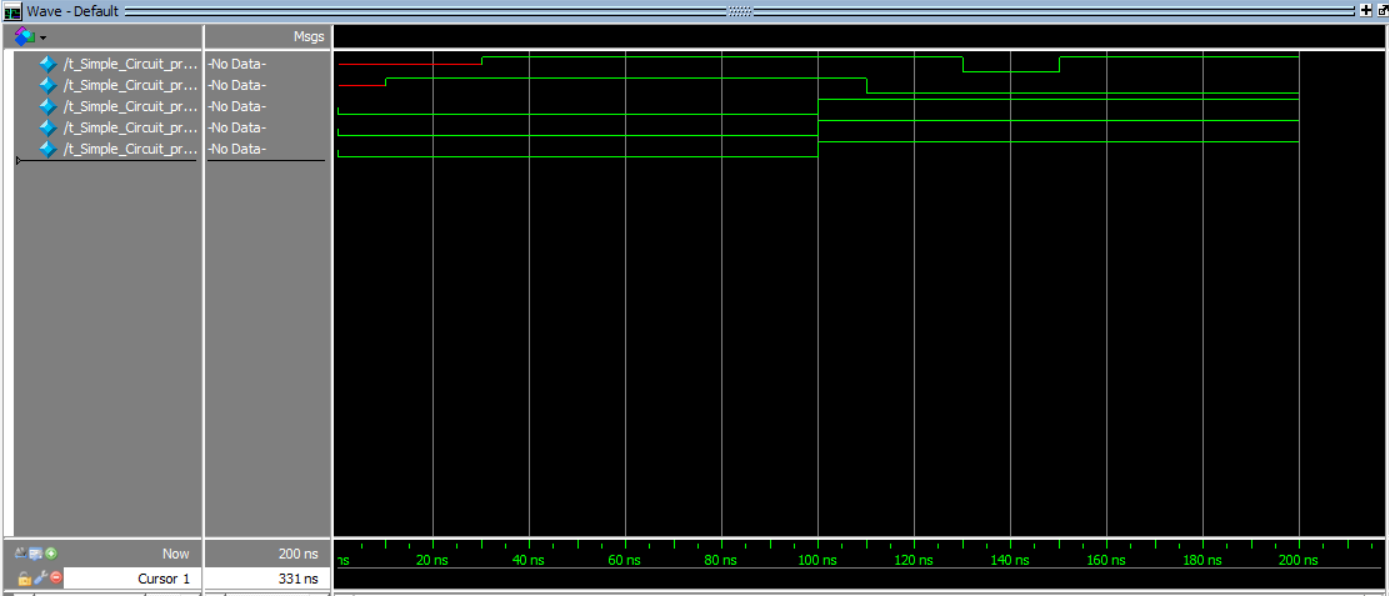
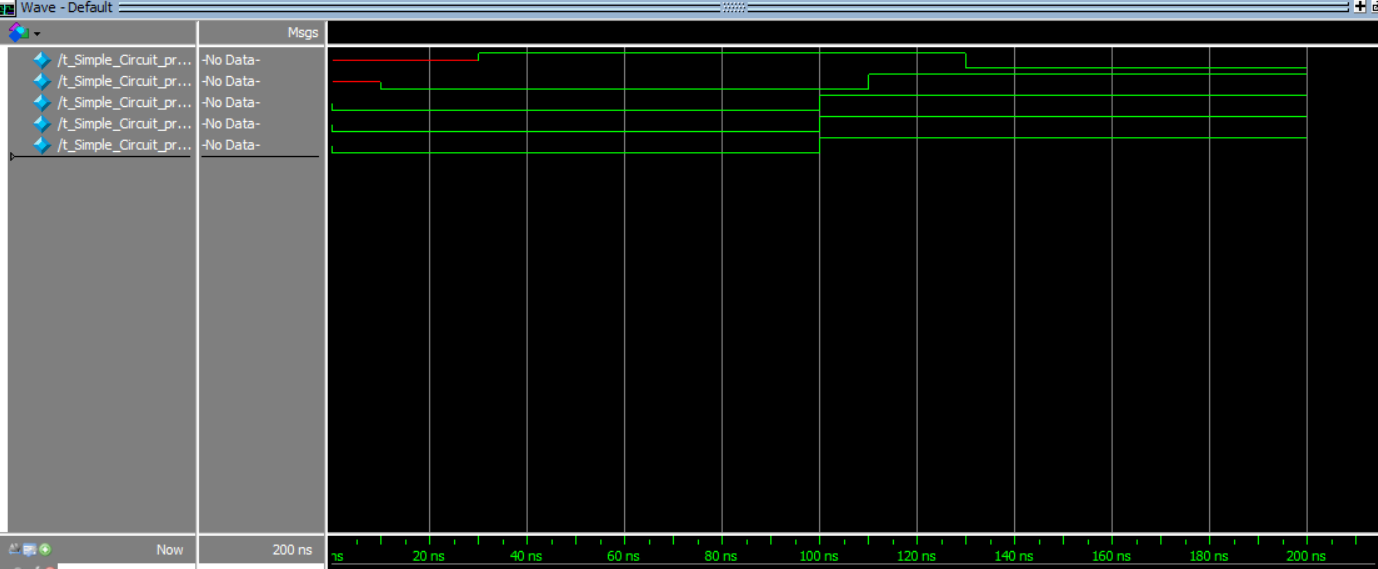
(1)



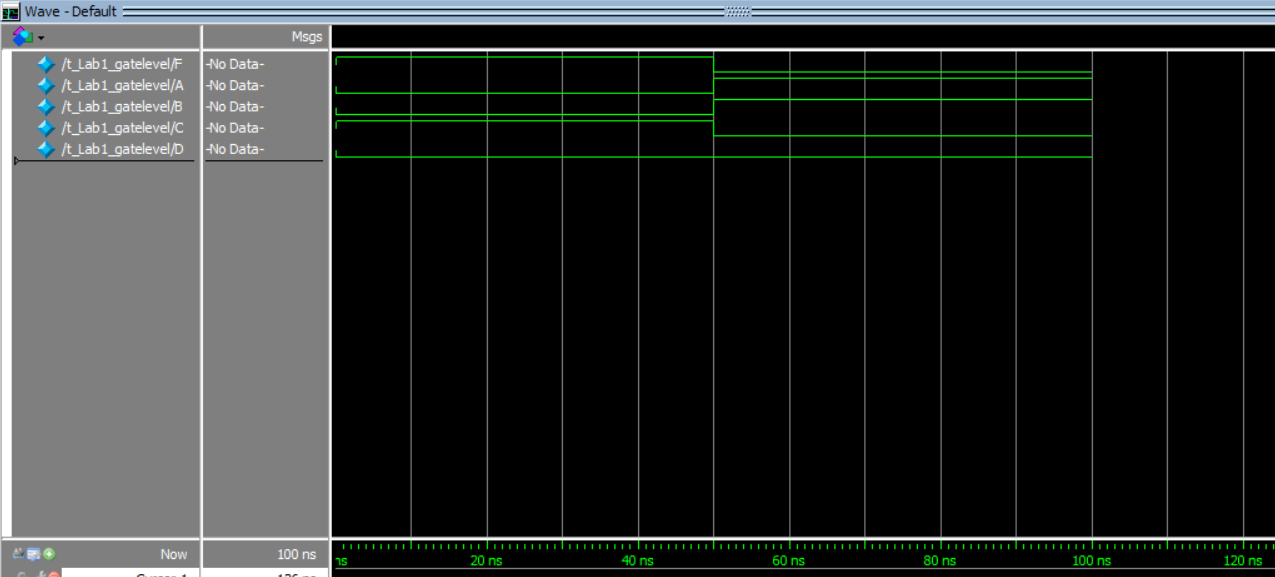
與Simple\_Circuit.v 之波形圖的差異在於D、E一開始的線條顏色是紅色的，代表此時電路因各邏輯閘設定的延誤時間，所以整個電路還在處理傳輸進來的資料，因此D、E尚未得到其應有的值。此外，因為各邏輯閘有延誤時間，所以D的圖形在130ns的時候數值由1變成了0，在150ns時，又從0回復到1，與Simple\_Circuit.v 之波形圖中D的數值一直是1有很大的改變，而且D、E開始有數值出現的時間也被往後延。

(2)



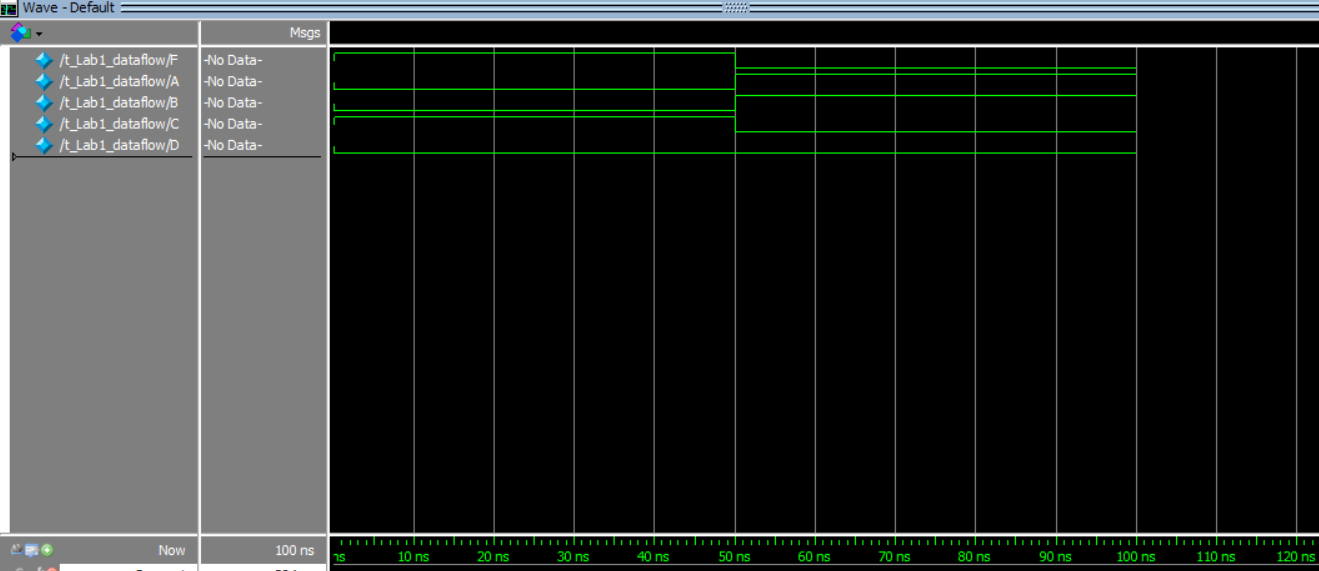
與2A(a)圖的差異在於把not和or中的指令對調，所以造成D的圖形從130ns開始就一直維持0直到結束，與2A(a)中D的波形圖有很大的差異，而E的圖形則是變成由0開始，在110ns時輸出值變成1，與2A(a)中E的圖形剛好形成互補。

(3)



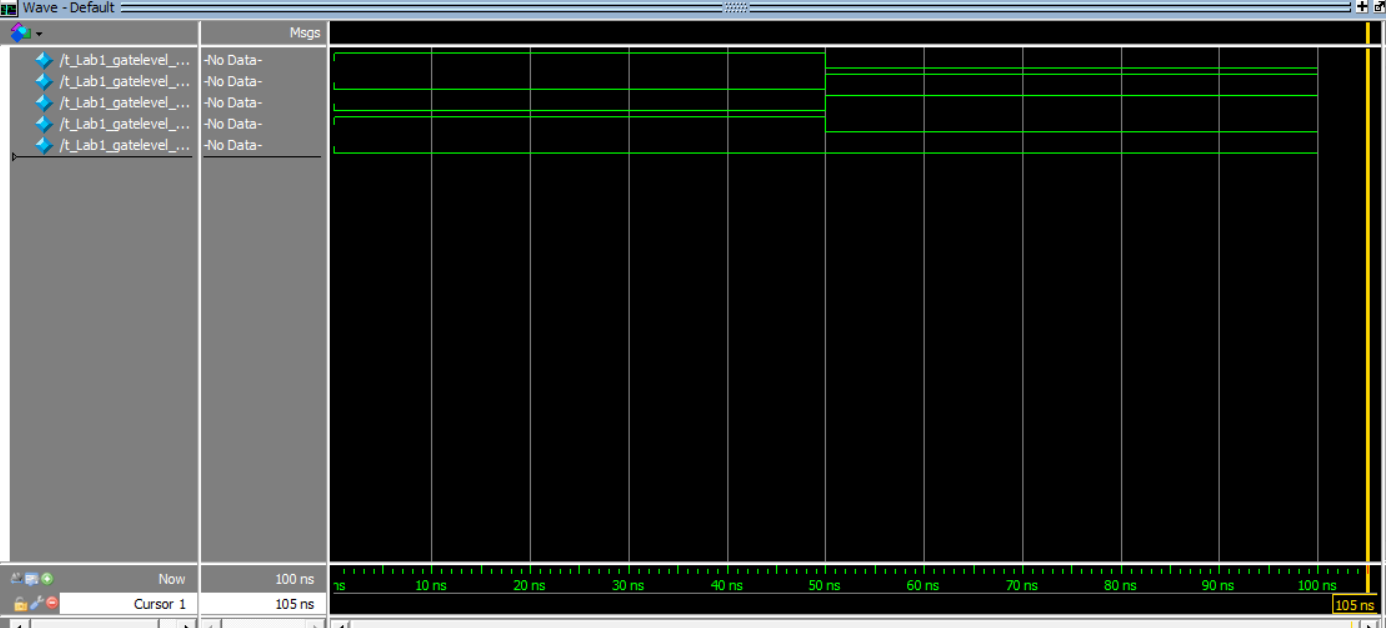
此模擬結果與我設定的數值完全符合，我在一開始設定A = 0、B = 0、C = 1、D = 0，所以F的值應為1，這個結果與我所預期的完全相符，我在50ns的時候將(A、B、C)的值改為(1、1、0)，而D的值維持不變，所以此時F的值應為0，此結果也符合波形圖顯示出的結果，所以此次模擬結果是正確的。

(4)



這次的測試值與上一個方法所用的值相同初始值為A = 0、B = 0、C = 1、D = 0，所以F的初始值為1與波形圖顯示的完全相符，在50ns時，我將A改為1，B也改為1，C改為0，而D則保持0不變，所以這時F的值就會變成0，此結果與波形圖顯示的相同，所以此次模擬結果是正確的。

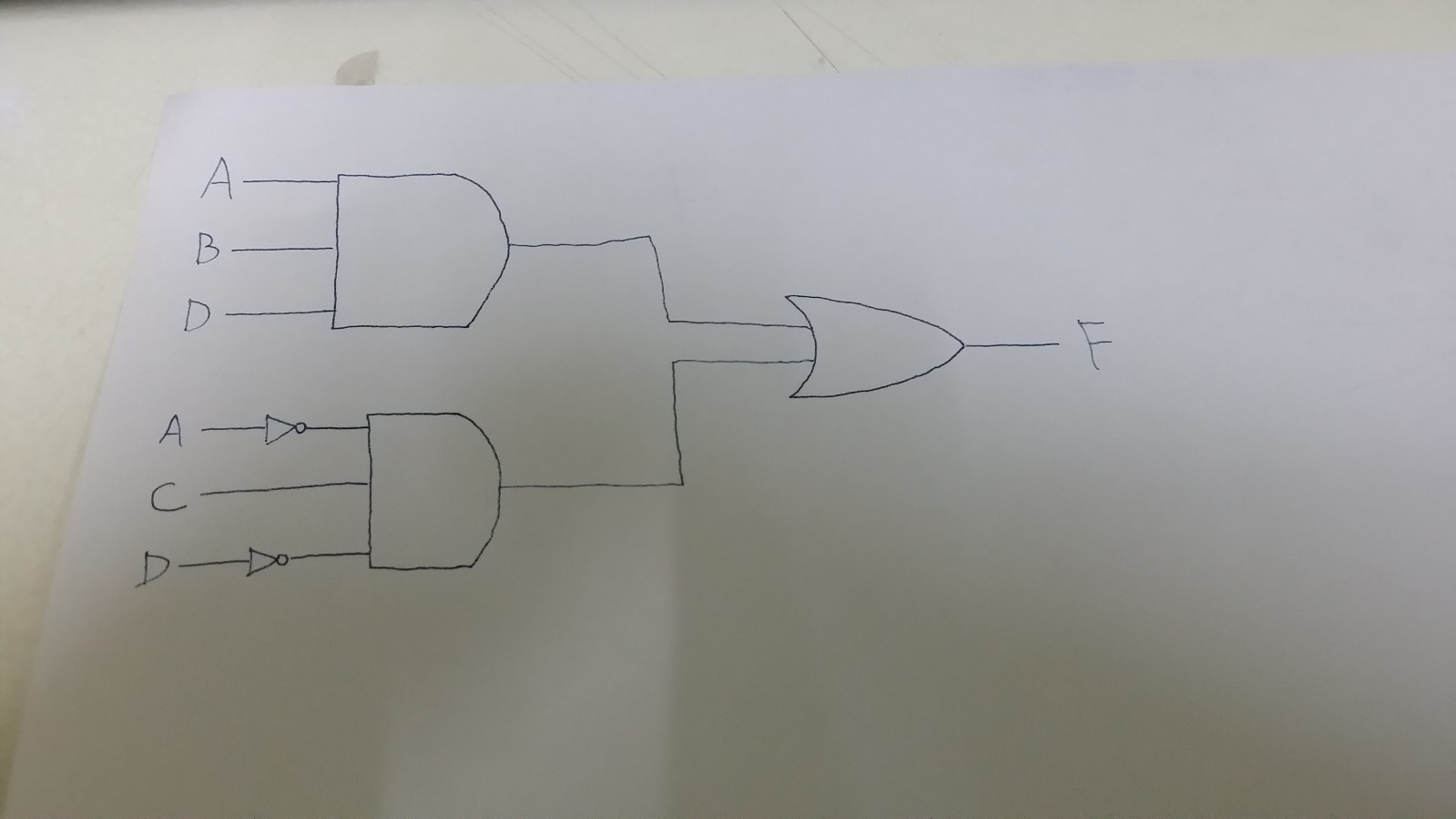
(5)



此次的測試值我也是採用(A、B、C、D)設為(0、0、1、0)的方式來進行測試，所以在此測試資料下，F一開始的值應為1，此預設與波形圖顯示的相同，在50ns時我將A、B、C改為1、1、0，而D維持不變，這時F的值就會變成0，直到模擬結束，此結果與波形圖所示相同，所以此次模擬結果是正確的。

(6)圖一的電路並非該函式GIC最少的操作，該函式還可以化簡為F = ABD + A’CD’，而化簡後的GIC為10。

用邏輯閘所畫出的電路圖:



(7)此次用三種方法寫同一個電路，我最喜歡的方式是用dataflow來寫，我覺得用這種方式寫比較直觀且輕鬆，不用像gatelevel要宣告到底用了多少不同的邏輯閘，也不用像UDP把所有變數的真值表寫出，而且用UDP的方式不小心還可能會打錯，造成之後模擬的時候出現錯誤，也不容易找出到底是哪個值不小心輸入錯了，而dataflow的方式只要一行就可以把函式所要表示的輸入完，實在是比另兩個方法好用且容易偵錯，程式也簡單易懂，只要一看就大致能理解此函是想表達什麼。