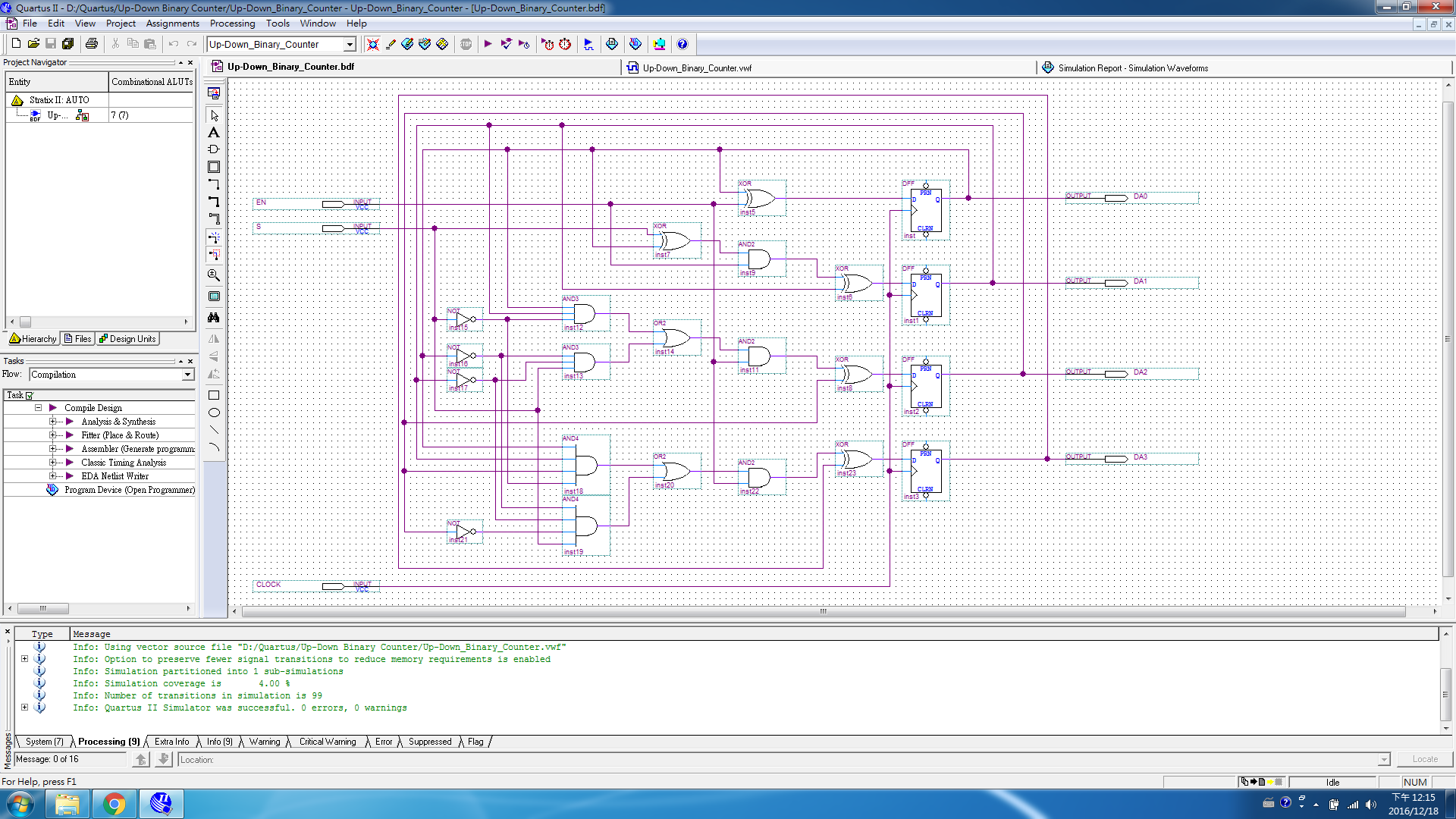
Title:

Up-Down Binary Counter



**系級 : 資工系 108 級**

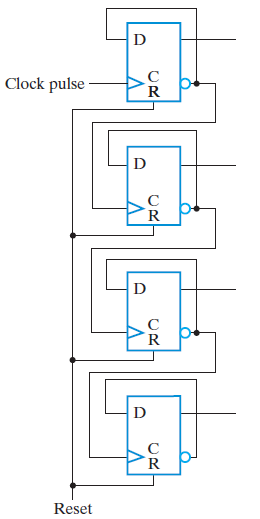
**學號 : A1045516**

**姓名 : 蔡湘俊**

**指導老師 : 潘欣泰 老師**

Abstract

Synchronous Binary Counter這一個電路是一個類似於一開始上課老師所教的Binary Ripple Counter這一個電路，彼此都可以實現Upward Counting以及Downward Counting的功能，而這兩種電路最大的差別在於Binary Ripple Counter這一個電路它是屬於非同步的設計方式，也就是說我Clock Pulse的觸發只會針對第一個D flip-flop，其餘的D flip-flop就由前一個D flip-flop的輸出來去做其下一個D flip-flop的Clock Pulse，下面這張圖就是Binary Ripple Counter的電路圖:



而反觀Synchronous Binary Counter，它的Clock的觸發就是採用同步的方法，那Synchronous Binary Counter有兩種設計型式，分別是Serial gating和Parallel gating這兩種型式，Serial gating的AND是採用串連在一起的方法，如下圖所示:



而Parallel gating的AND則是獨立開來的，如下圖所示:



利用Parallel gating的型式下去設計Synchronous Binary Counter最大的好處就在於說其只會有1個AND Delay，可是

Serial gating就會有4個AND Delay，就以執行速度來講Parallel gating會比Serial gating快很多，可是這兩種型式的電路的Counter的模式都只會有一種，也就是說一次只能夠實現Upward Counting或Downward Counting的其中一種功能，那為了讓其電路能夠同時有Upward Counting和Downward Counting這兩種功能就衍生出來今天作業要做的Up-Down Binary Counter這一個電路。

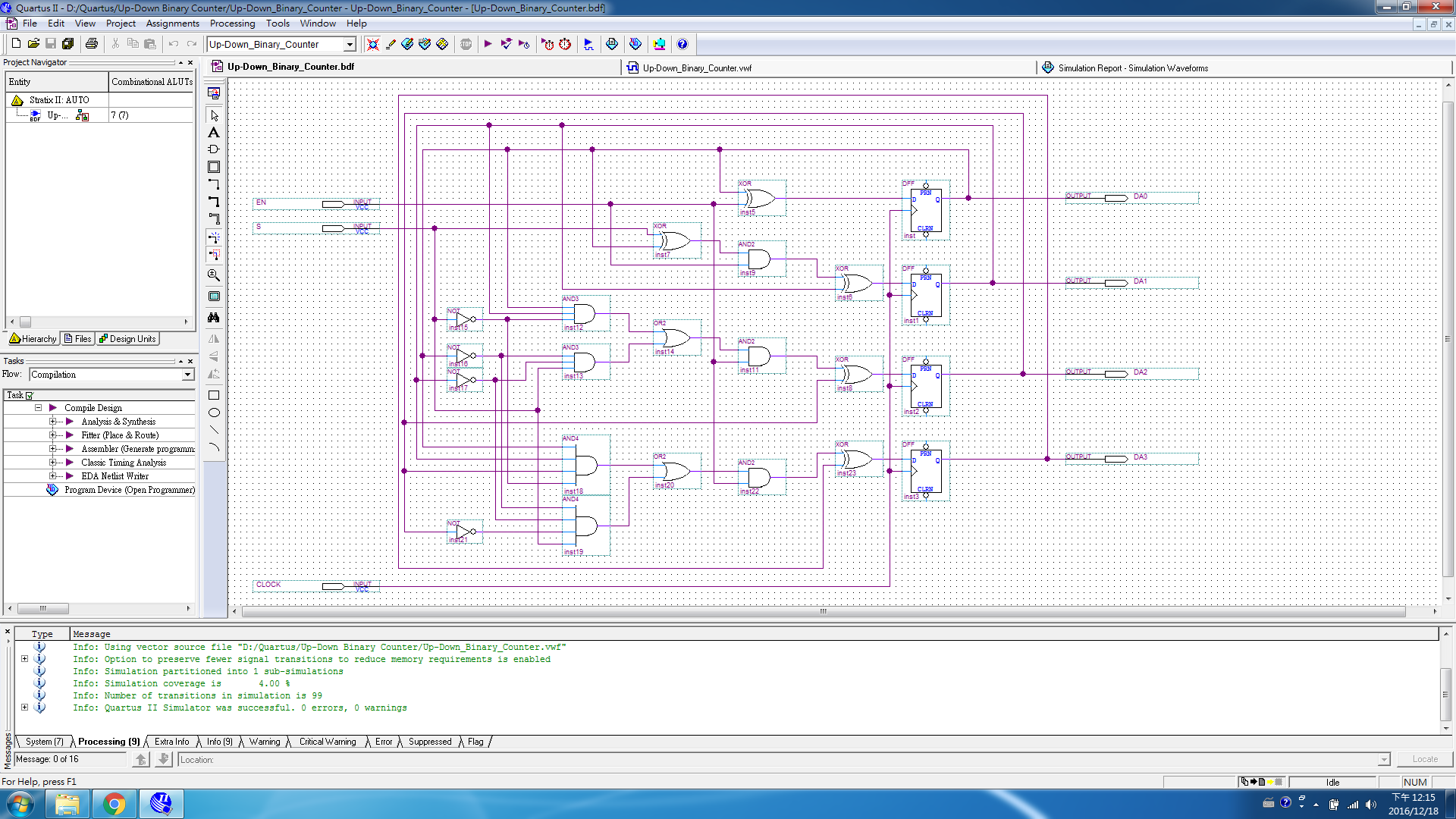
Content

1. methods:

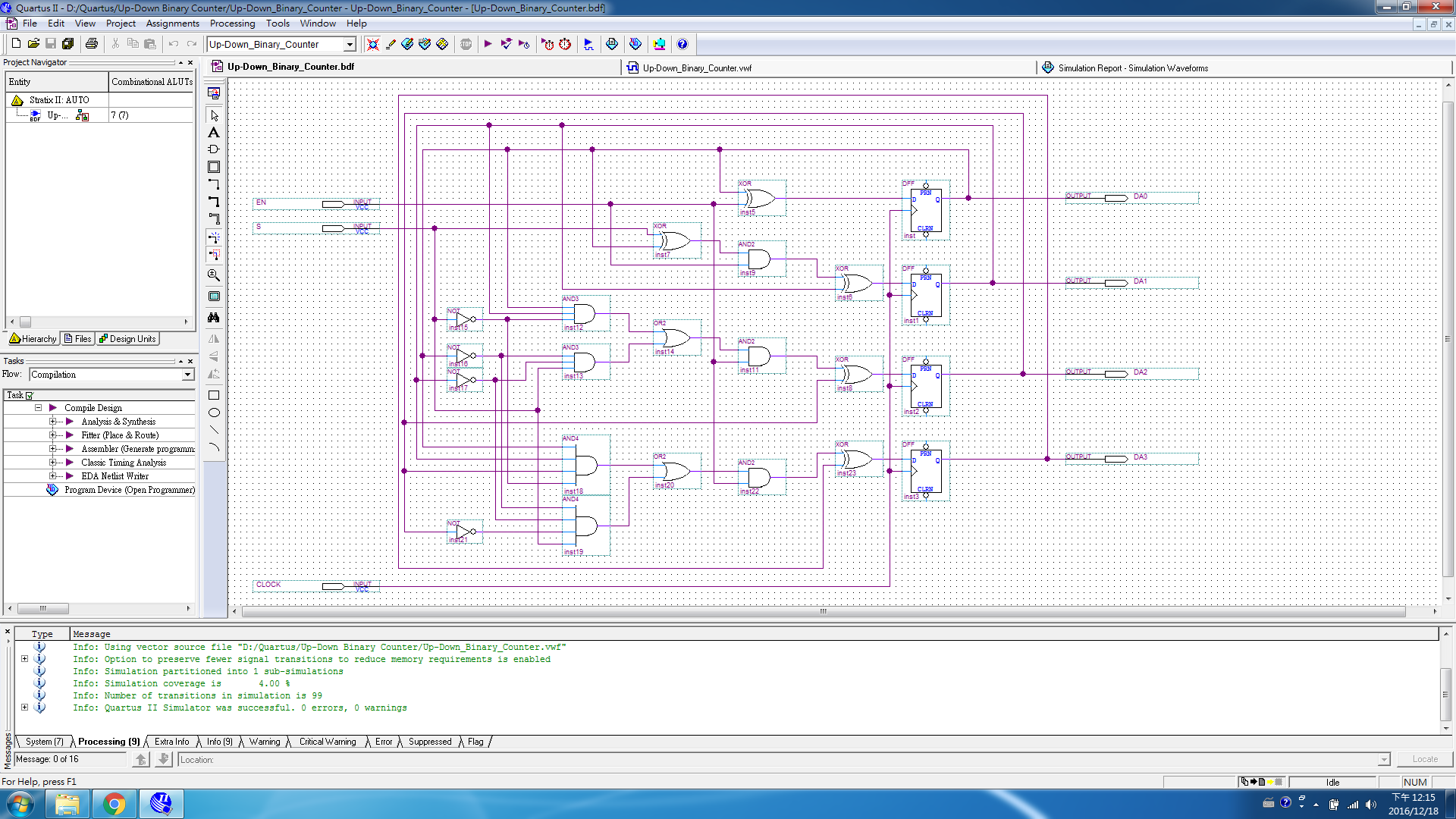
而設計的方式就是利用老師上課用的PPT所給的這四個Function來去做設計的動作，而這四個Function就如下圖所示:



在整個電路的設計當中我們必須用到3個Input、4個Output、4個D flip-flop、5個XOR、7個AND、2個OR以及4個NOT來去完成這一整個電路，而其中這一個Function當中的可以被簡化成這一種形式，也就是說這一個Function就變成，如此一來就能夠去簡化我們的電路，只是其餘3個Function就無法去做化簡的動作，那依序其化簡過後的4個Function所畫出來的電路如下圖所示:

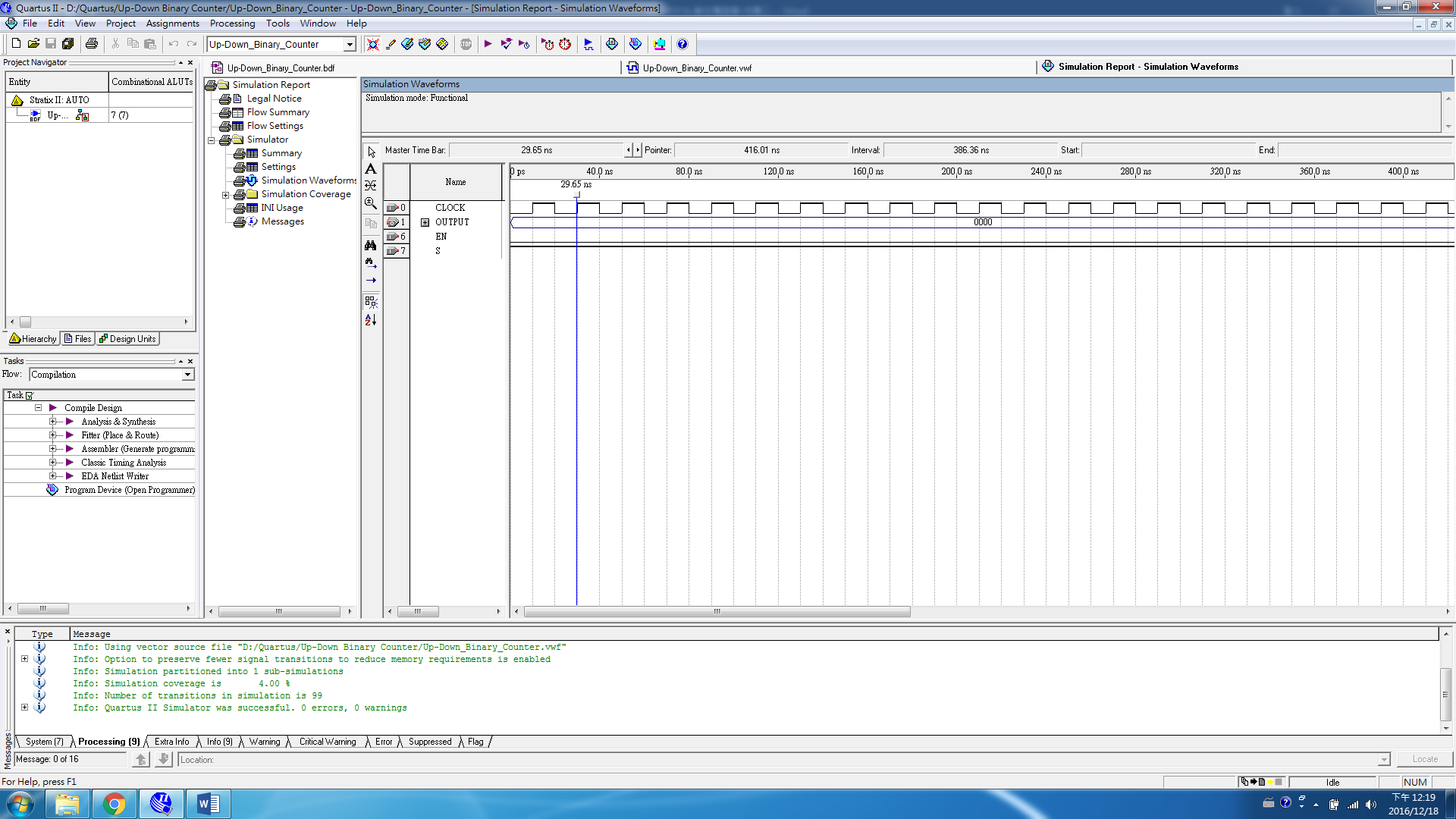


1. circuit diagram

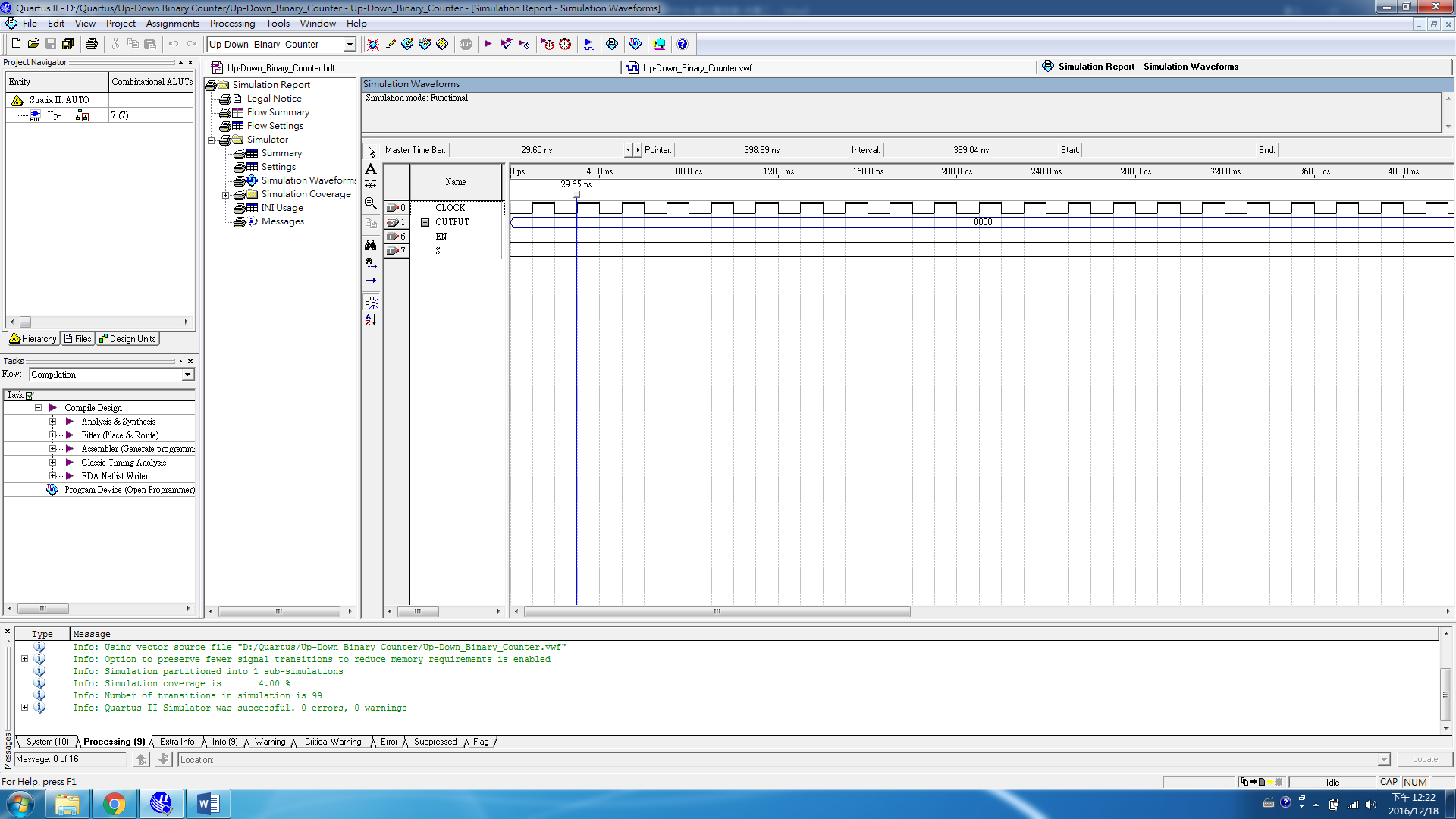


1. simulation results

當我EN = 0且S = 1的時候，其顯示結果如下圖所示:

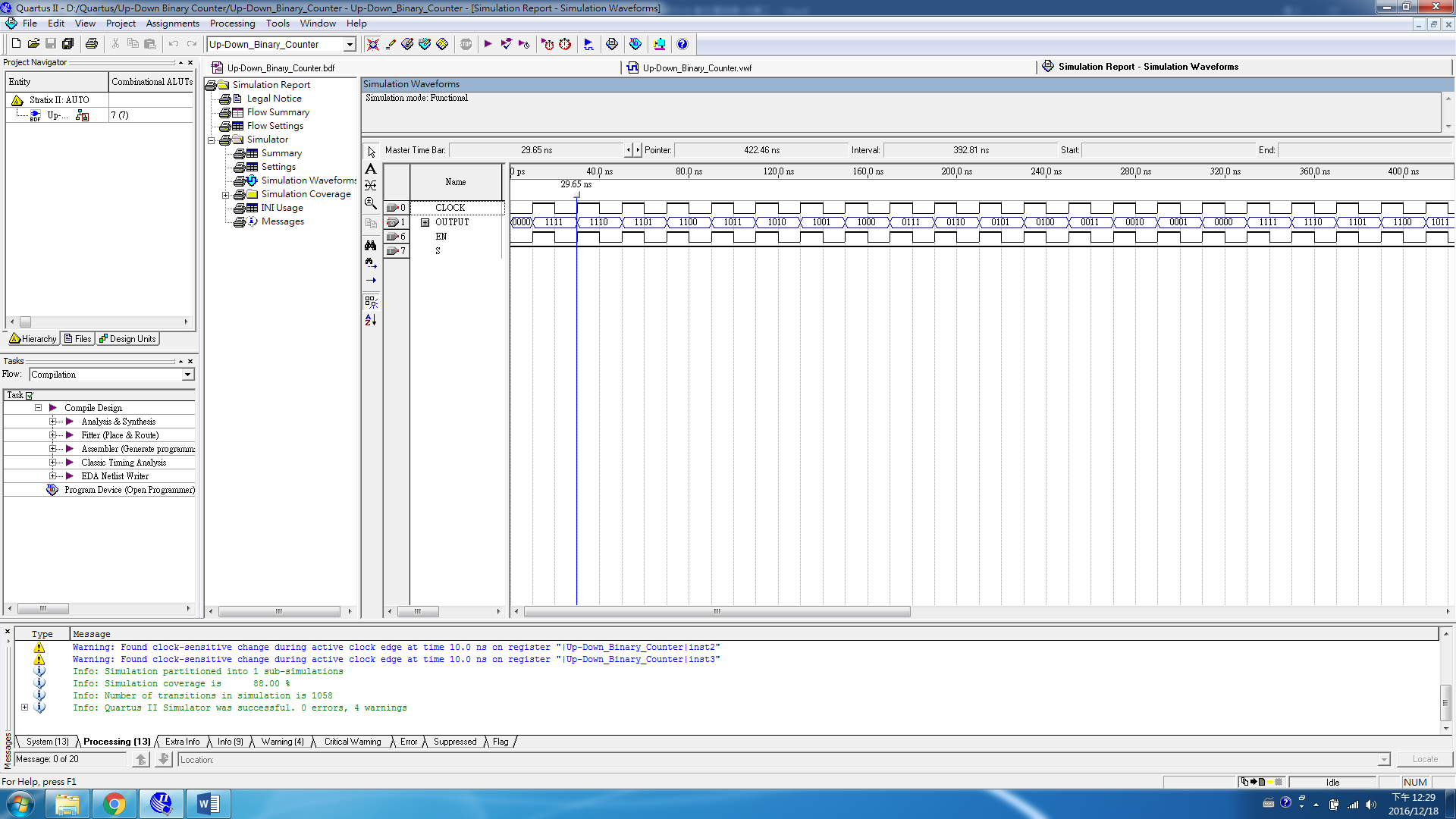


當我EN = 0且S = 0的時候，其顯示結果如下圖所示:



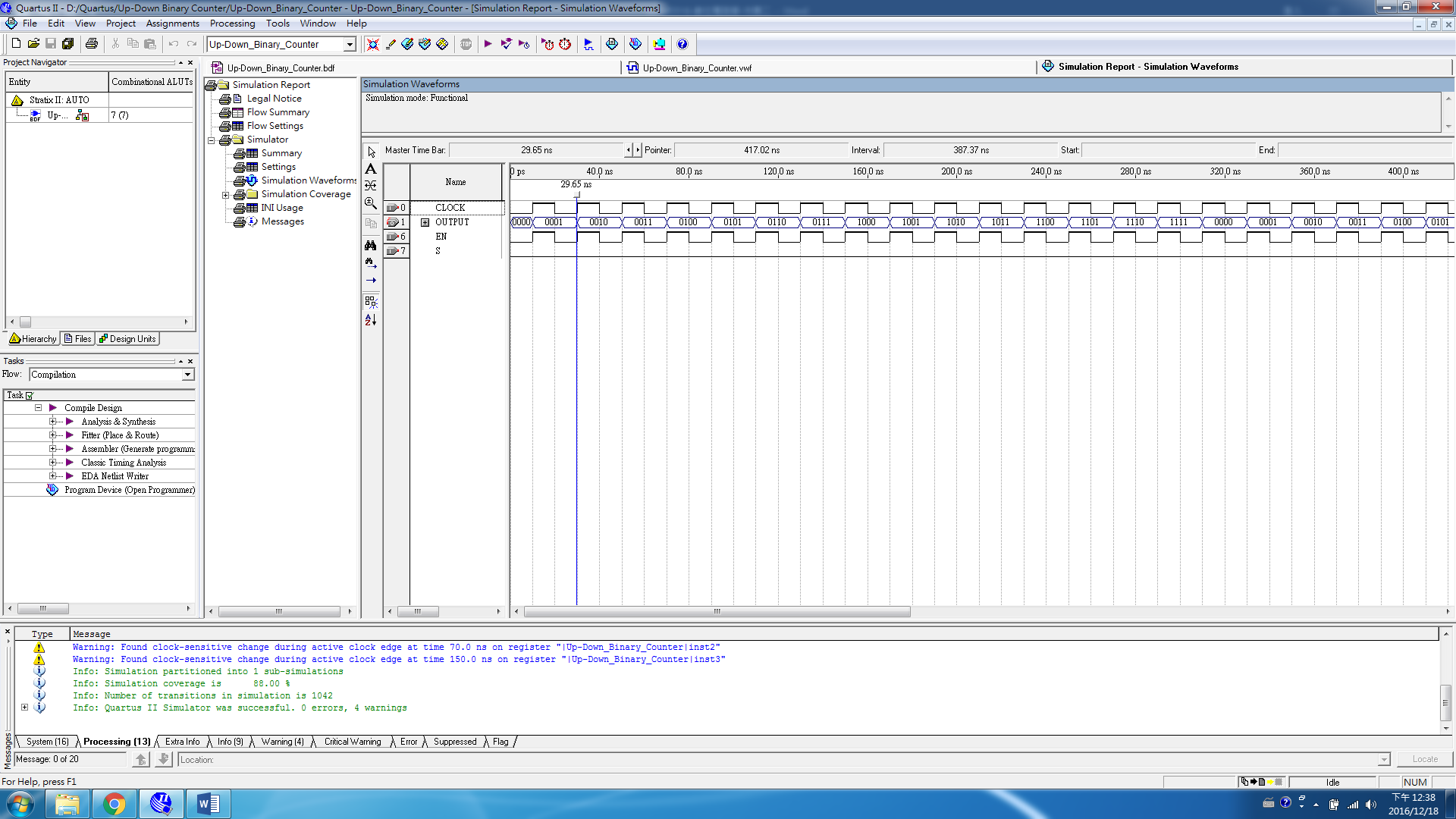
由上面這兩張圖的結果我們可以得出不論是S = 0(Upward Counting)還是S = 1(Downward Counting)，只要其EN = 0，就不會發生任何的動靜。

當我EN = 0和1的連續變化且S = 1的時候，其顯示結果如下圖所示:



由上圖可以看到說因為其電路當中的D flip-flop是positive edge trigger，所以每當到0變成1的瞬間其Counter就會產生動作，又因為此電路的設計是當S = 1的時候，就代表要執行Downward Counting功能的Counter，所以可以看到說其一路由1111一直Downward Counting到0000為止並且以每組1111到0000為一個循環。

當我EN = 0和1的連續變化且S = 0的時候，其顯示結果如下圖所示:



由上圖可以看到其顯示的結果剛好是前一張圖的反向結果，也就是說當S = 0的時候，就代表要執行Upward Counting功能的Counter，所以可以看到說其一路由0000一直Upward Counting到1111為止並且以每組0000到1111為一個循環。

1. Problems

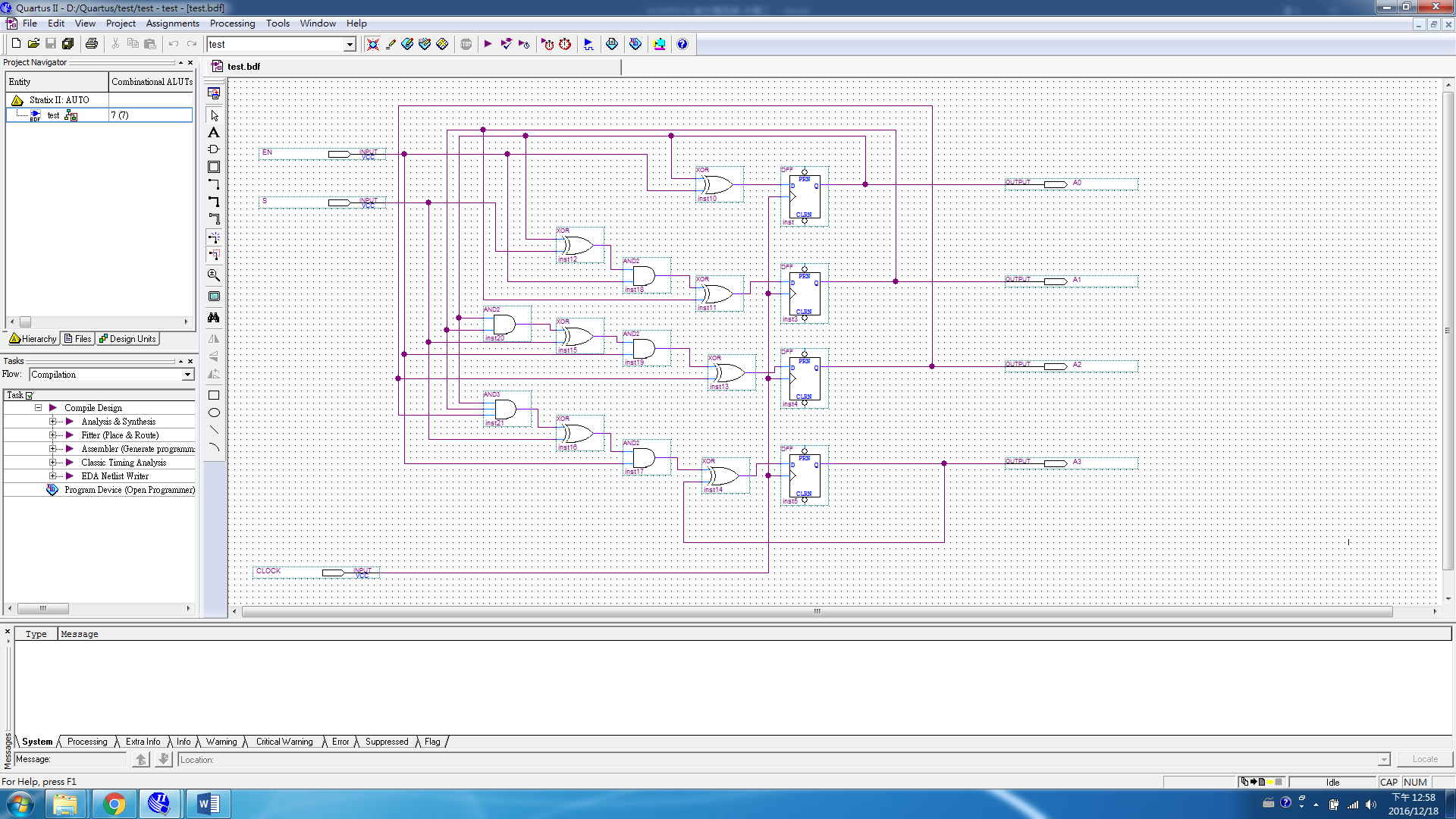
那在做這一個電路的時候中途遇到的問題就在於Function的化簡，在一開始我看到這4個Function的時候，也就是如下圖所示:



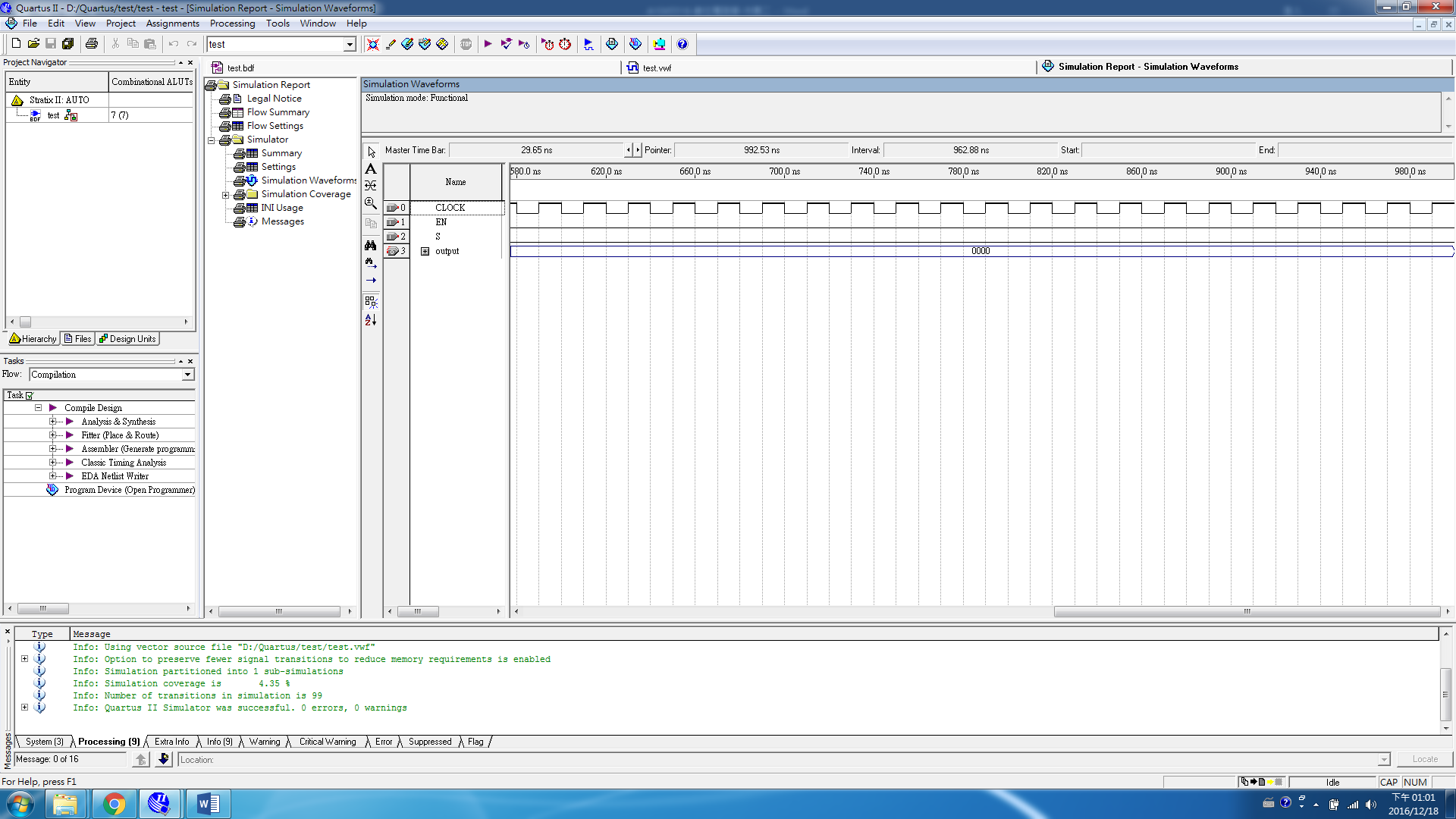
我當下的第一個反應就是認為說其、和都可以分別化簡成XOR的形式，並且使得其Function變成如下圖所示:



剛開始驗證出來的結果因為是對的所以就沒有特別注意到太多就將其做成電路的形式，就如同下圖所示:

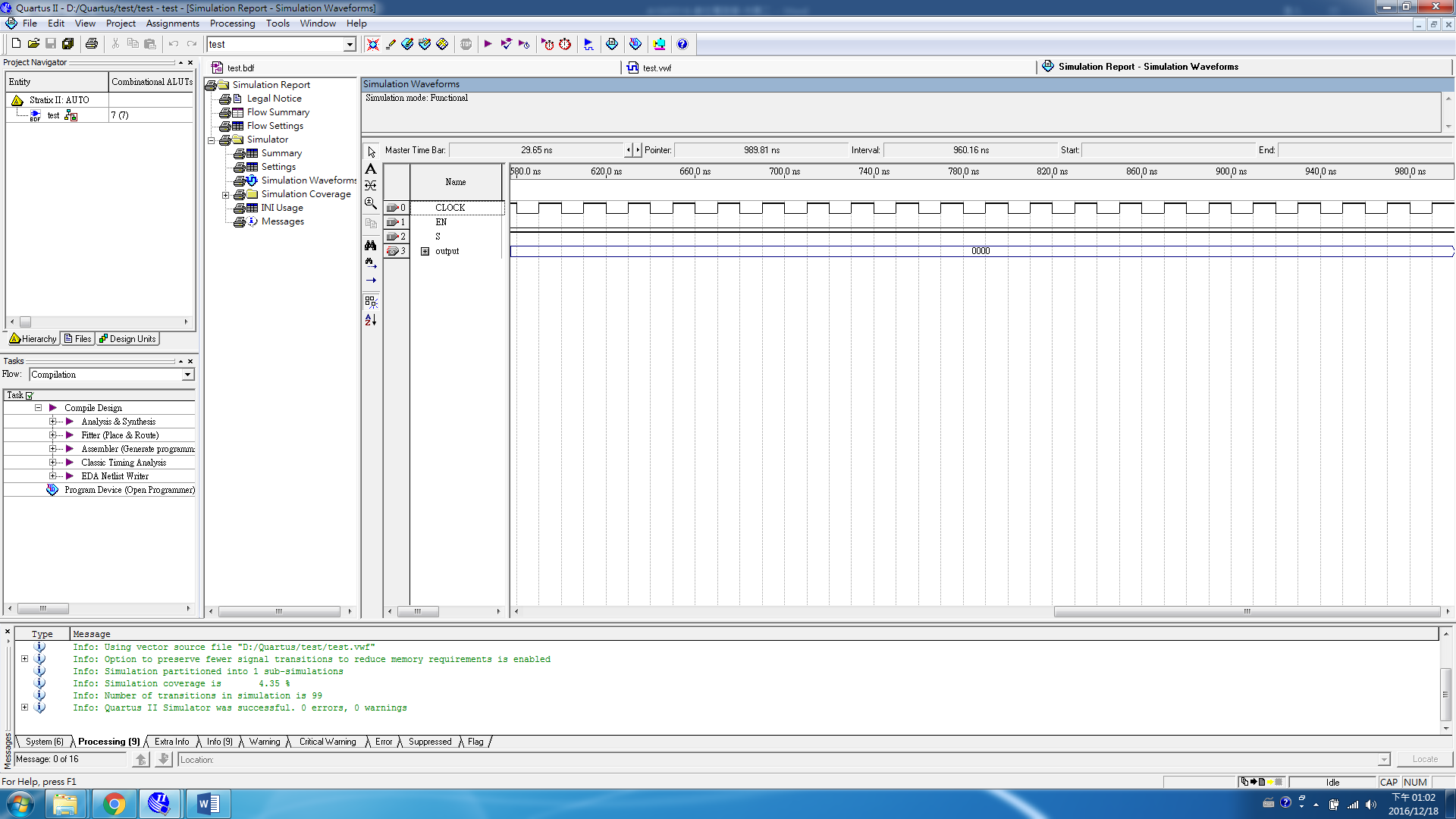


那在波型模擬的部份，當其EN = 0且S = 0的時候，其顯示結果如下圖所示:



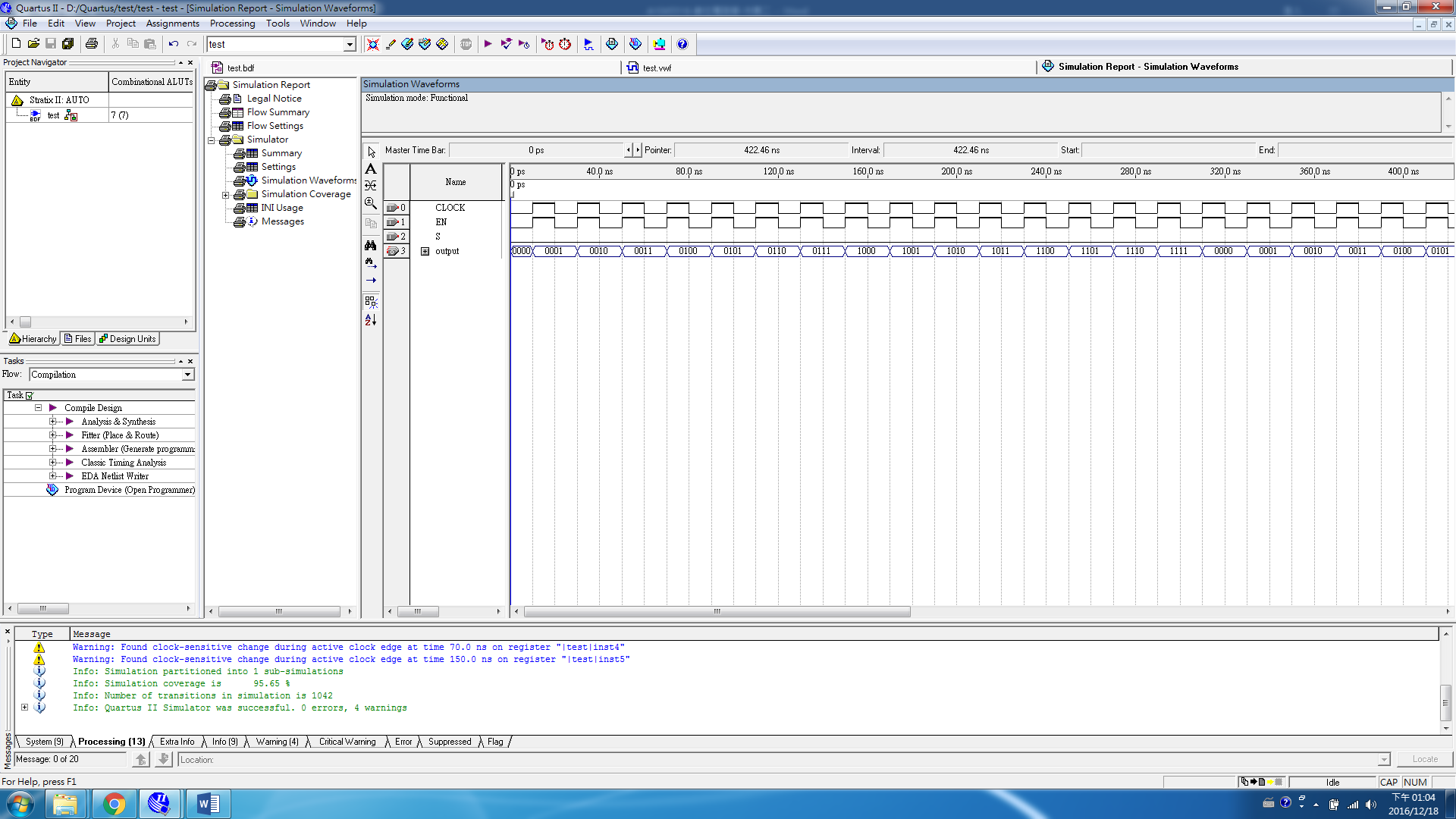
與上面正確的電路的顯示結果一模一樣。

當其EN = 0且S = 1的時候，其顯示結果如下圖所示:



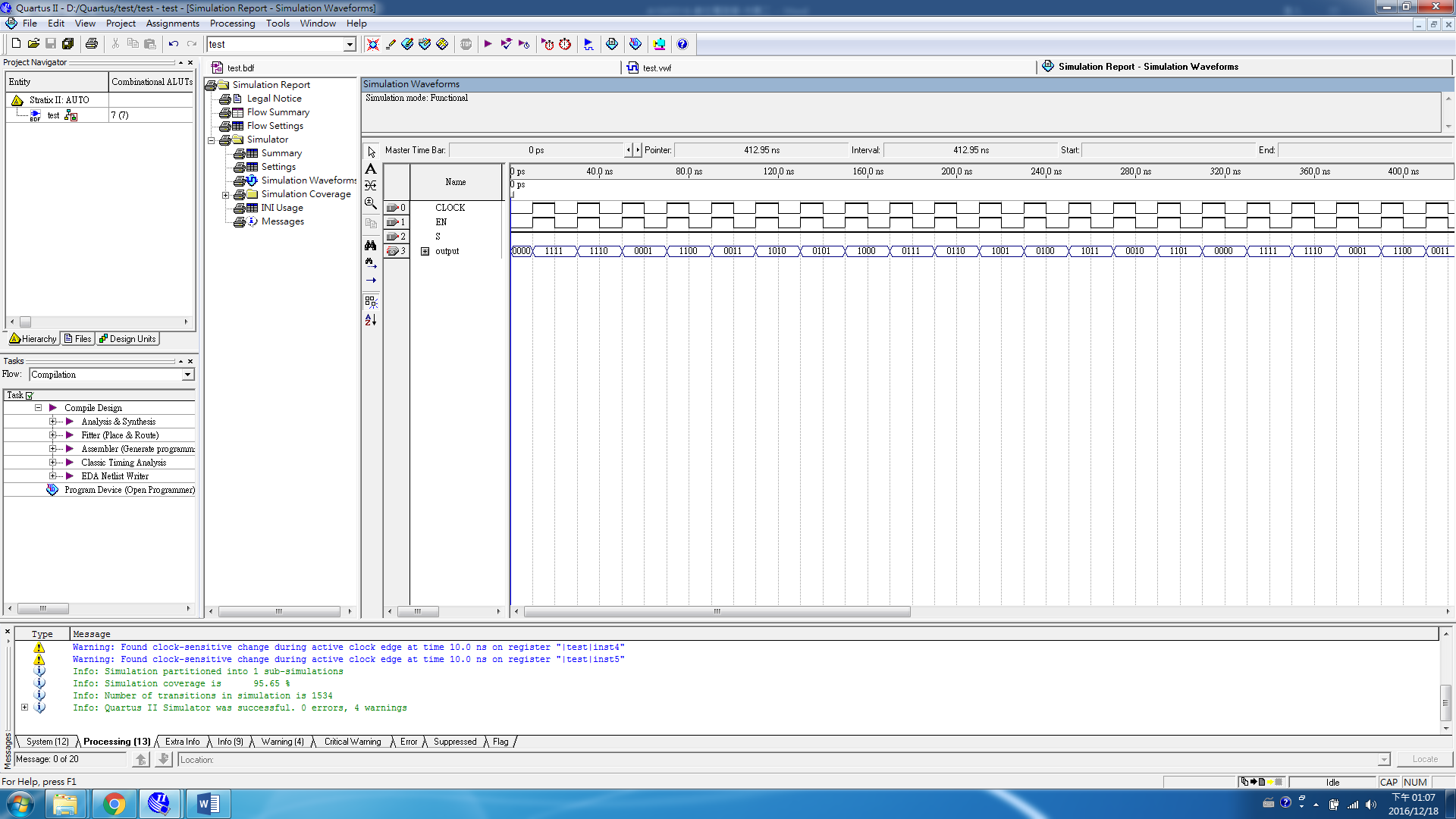
也與上面正確的電路的顯示結果一模一樣。

當其EN = 0和1的連續變化且S = 0的時候，其顯示結果如下圖所示:



可以看到說顯示的結果也與上面正確的電路的顯示結果一模一樣。

可是，當其EN = 0和1的連續變化且S = 1的時候，其顯示結果如下圖所示:



從1110之後就開始不正常

正常

可以看到說其原先在1111到1110這個過程的Downward Counting的功能是正常的，可是一到了1110之後，可以看到說其Counter的順序已經整個亂掉了，已經不是像正確電路所顯示的結果一樣是依序的Downward Counting，由此就可以得知說其所設計出來的電路是錯的，也就意味著所化簡出來的那4個Function也是錯的。

那當我再去仔細檢查我所化簡出來的那4個Function之後，我發覺到前兩個Function也就是和這兩個Function是正確的，可是其和這兩個Function是錯誤的，而會錯的原因在於其和這兩個部份是無法畫簡成XOR的形式，就拿這一個作為比方，假如我將這一個Function給它展開，如下圖所示:



從上面就可以很明顯的看出說，所以其原來化簡出來的Function是錯的，也因為如此，畫出來的電路就變成是錯誤的。

1. Discussions

從這一次的作業當中讓我理解到了在設計電路的任何過程當中，只要有任何細微的疏忽就會導致其所設計出來的電路會是錯的，並且儘管其前頭所顯示的結果是對的，也不能夠直接篤定說自己作出來的電路已經沒有任何的錯誤，並且不再去做任何後續的檢查，像這一次做錯的電路就是一個很明顯的例子，而且像這樣一開始沒有特別的注意到，等到事後完成的時候才發覺到的情況下，就意味著一切又必須從頭來過，像這樣子就會白白浪費了許多的時間，所以唯有仔細的去檢視每一個設計電路的過程，才能夠有效的去降低像這樣子的情況發生，並且也會大大的提高自己設計電路出來的速度。

1. References

(1) 105\_9\_digitalCircuit\_Chap\_6\_P2.ppt

(2) Logic and Computer Design Fundamentals, M. Morris Mano & Charles R. Kime FIFTH EDITION, 2015 Pearson Education, Inc.