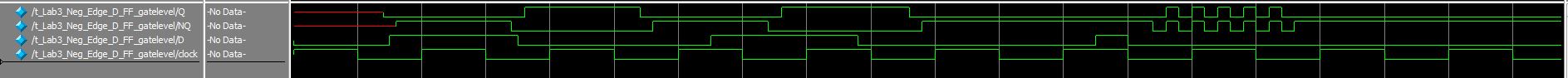


一開始NQ和Q各有2ns和4ns的延遲，因為S和R各為1和0，為SET狀態，Q變成1，接下來S和R皆為0，為no change，Q維持1，當S和R為0和1時，為RESET，Q變成0，再來S和R為0時，為no change，Q維持0，再來S和R為1時，進入forbidden狀態，Q無法確定，因此在1和0間不斷震盪，

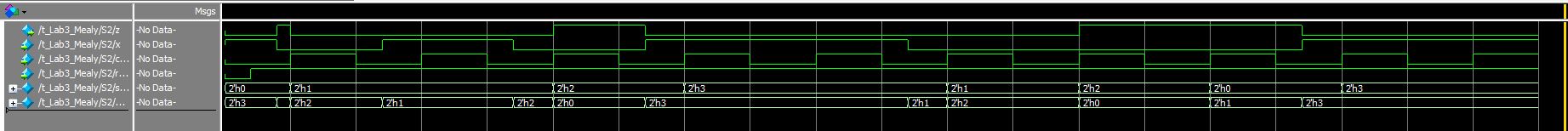
波形圖正確。

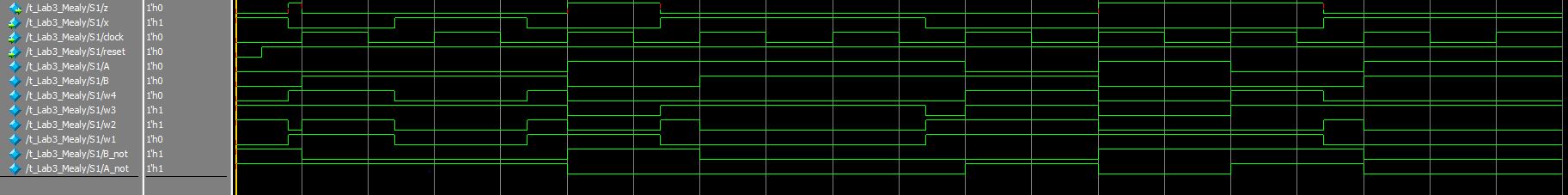
2.



一開始的clock為0，會使Q維持原本狀態，但是一開始的Q沒有定值，所以維持不確定的狀態，接著10ns時，clock由1變0，D為0，R變成1，S變成0，進入reset狀態，Q為0，30ns時，clock由1變0，D為1，進入set狀態，Q為1，50ns時，clock由1變0，D為0，進入reset狀態，Q為0，70ns時，clock由1變0，D為1，進入set狀態，90ns時，clock由1變0，D為0，進入reset狀態，110ns時，clock由1變0，D為0，進入no change狀態，130ns時，clock由1變0，D為0，進入forbidden狀態，Q在0和1之間震盪。波形圖正確。

3.





分析狀態圖後發現，在S1狀態(01)和S3狀態(11)時，不管輸入x多少，輸出都是0，其他狀態則是和x輸入相反。

Testbench主要是先藉由reset強制狀態為S0，在藉由輸入的控制，跑過每一個狀態，流程為S0--->S1--->S1--->S2--->S3--->S3--->S1--->S2--->S0--->S3。

一開始將reset設為0，x設為1，此時會進入S0狀態(00)，此時的輸出和x不同，5ns時clock來，而x為1，跑到S1狀態(01)，此時的輸出不管輸入都是0，15ns時clock來，x為1，回到S1狀態(01)，輸出一樣都是0，25ns時clock來，x為0，跑到S2狀態(10)，輸出和x相反，35ns時，clock來，x為1，跑到S3狀態(11)，此時不管輸入，輸出都是0，45ns時，clock來，x為1，回到S3狀態(11)，輸出維持0，再55ns時，clock來，x為0，跑到S1狀態(01)，輸出維持0，再65ns時，clock來，x為0，跑到S2(10)狀態，此時輸出和x相反，再75ns，clock來，x為0，跑到S0狀態(00)，輸出和x相反，再85ns，clock來，x為1，跑到S3狀態，輸出不管x的值都是0。

比對兩張模擬的波形圖皆正確。

4.

這次的作業code雖然不難寫，但是在驗證時十分耗腦力且不容易檢查，但是也因此讓我對於latch 和 同步順序電路更為了解，也才發現我存在著一些錯誤的觀念，實作和理論果然大不相同。