Lab 5: Finite State Machines

Group 3 : 109060013張芯瑜 109062328 吳邦寧

# Design Explanation

## Content-addressable memory

1. 設計說明：

利用register創造一個可存放16個8 bit數字的空間CAM。

當ren = 1時，我們由存放位置序號較大的，也就是CAM[4’b1111] 開始找起，整個運算利用if else判斷。若有存放的數字與din相同，則將下一個clock的dout設為該數字存放位置，且hit設為1。若有兩位置皆存放與din相同的值，會因為先後順序而選擇位置序號較大的作為輸出。當跑完整個CAM仍然沒找到與din相同的值時，則輸出dout = 0 和 hit = 0。

當ren = 0且wen = 1時，將din值在下一個clock時存入CAM的第addr個位置。

當ren = 0且 wen = 0，則輸出dout = 0 和 hit = 0。

1. 驗證：

測試資料 #1

測試以下情況：

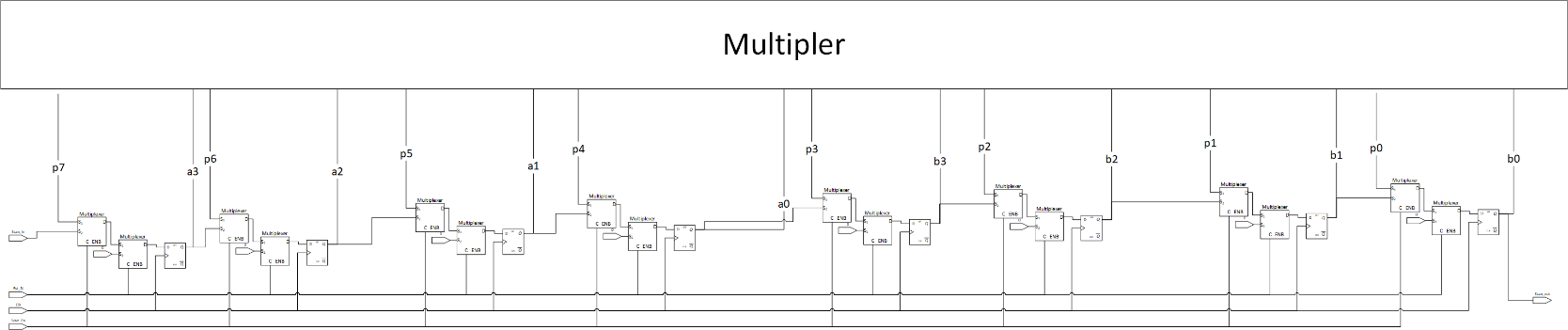
* ren = 1且wen = 1，不會寫資料，只會進行讀取。
* ren = 1且wen = 0，讀取資料，且讀不到時將輸出歸零。
* ren = 0且wen = 1，將資料寫入正確位置。
* ren = 0且wen = 0，不進行任何動作，並將輸出歸零。



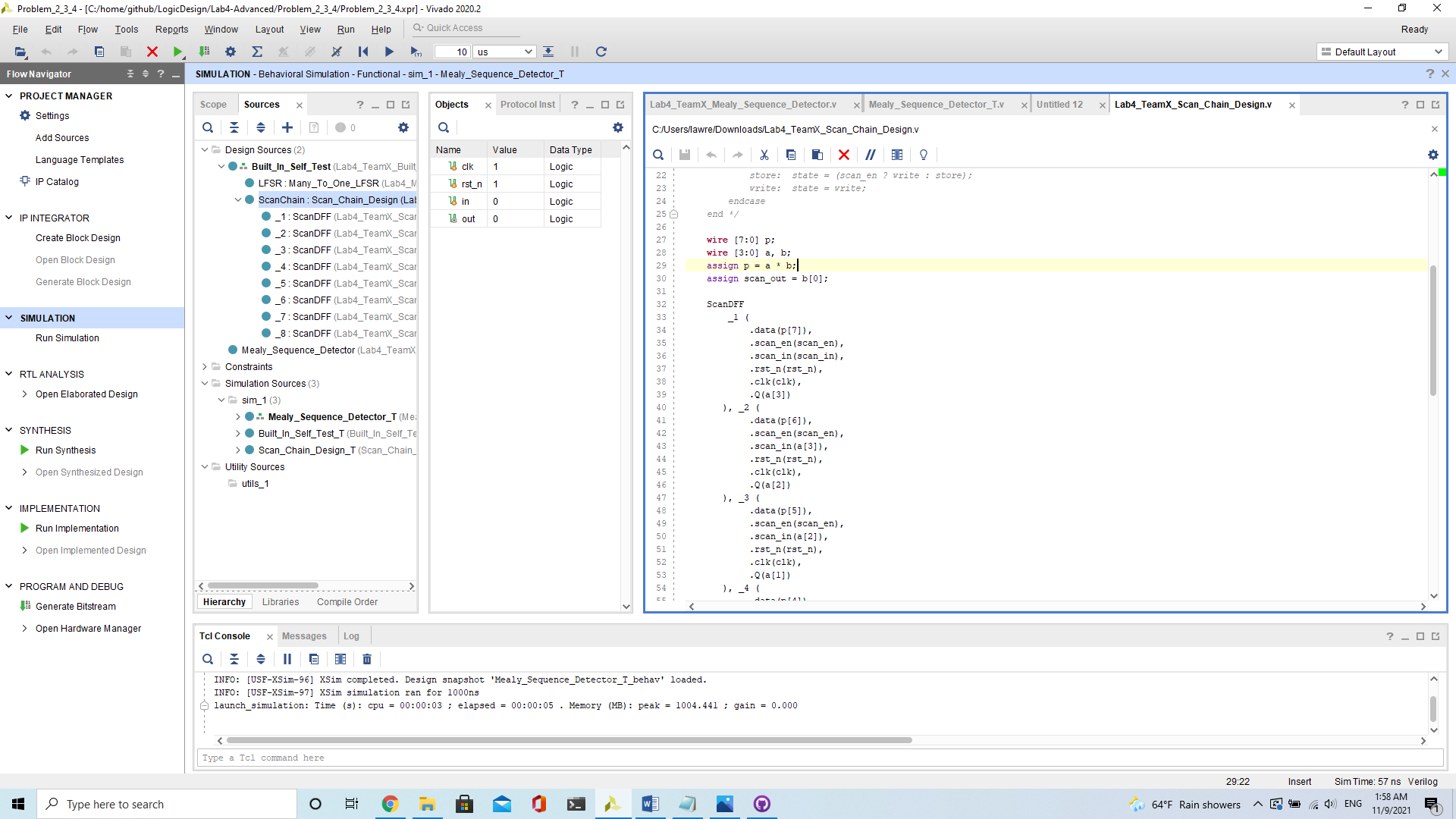
## Scan Chain Design

1. 設計說明：

首先，跟著規格設計Scan DFF。



接下來，利用內建的乘法運算子實作乘法器，再將資料打回Scan DFF。

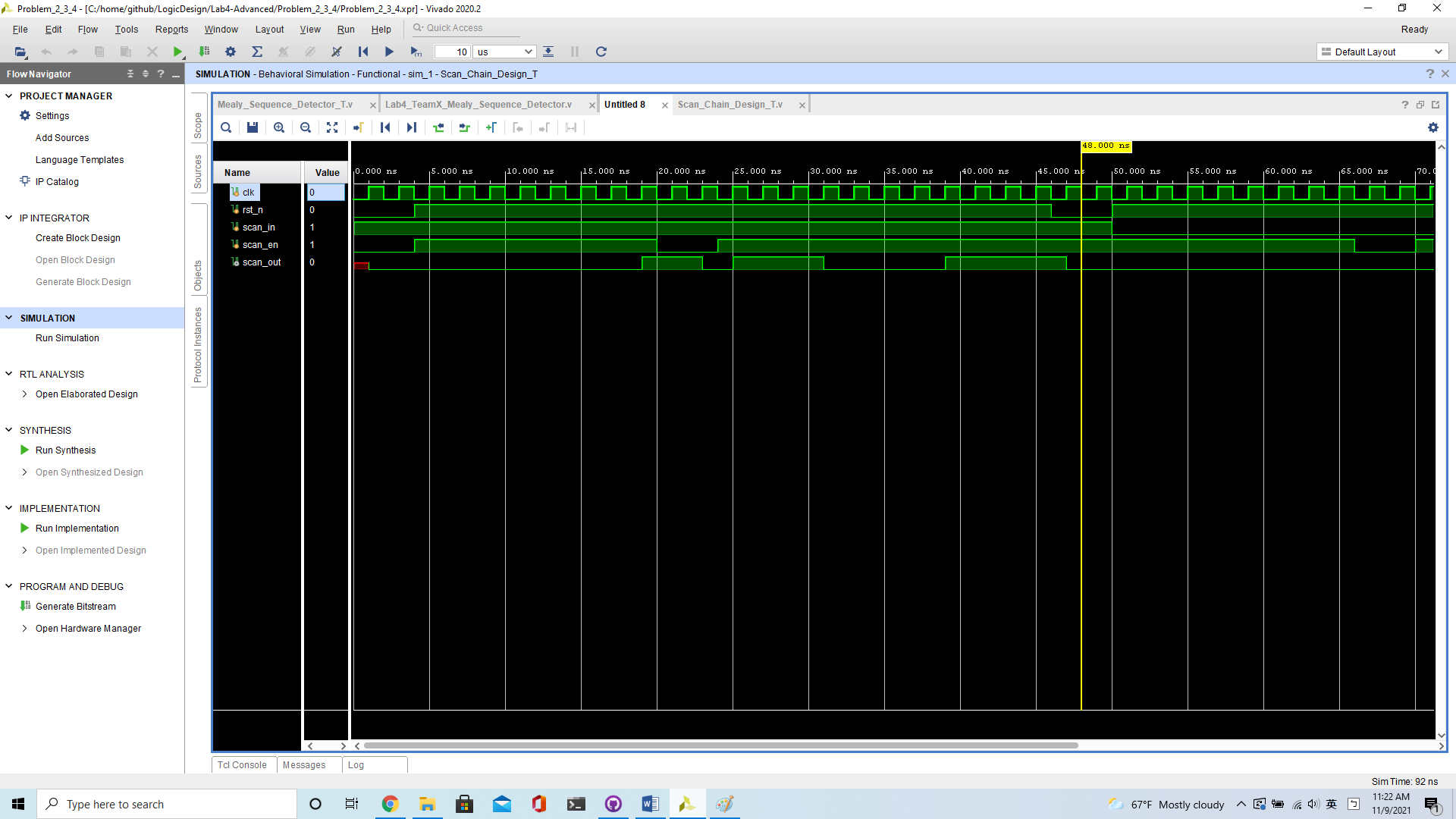


利用上述兩個設計，即可完成題目所需之要求。

1. 驗證：

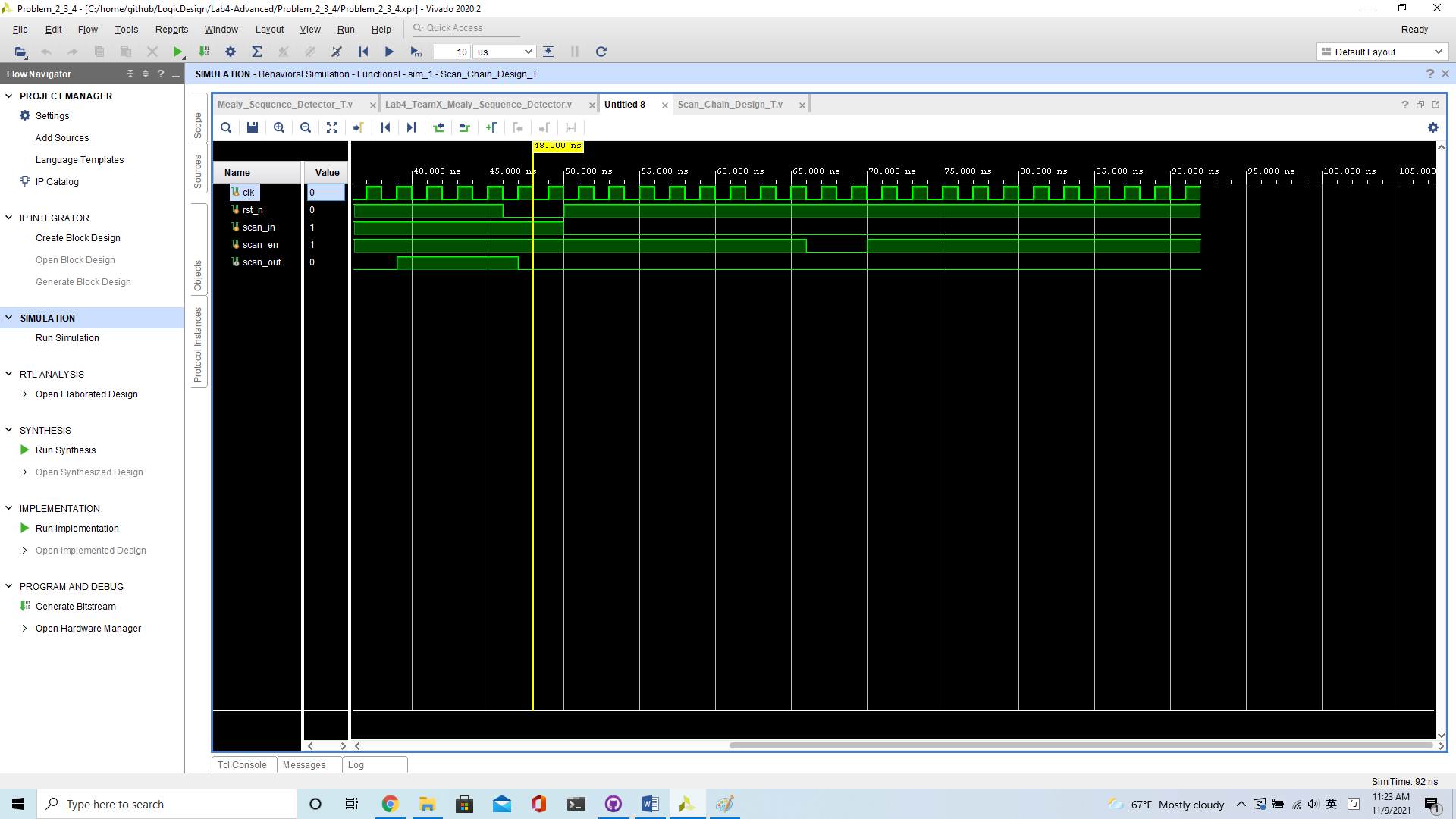
測試資料 #1

讓所有輸入都是高電位。



測試資料 #2

讓所有輸入都是低電位。



## Greatest Common Divisor

1. 設計說明：

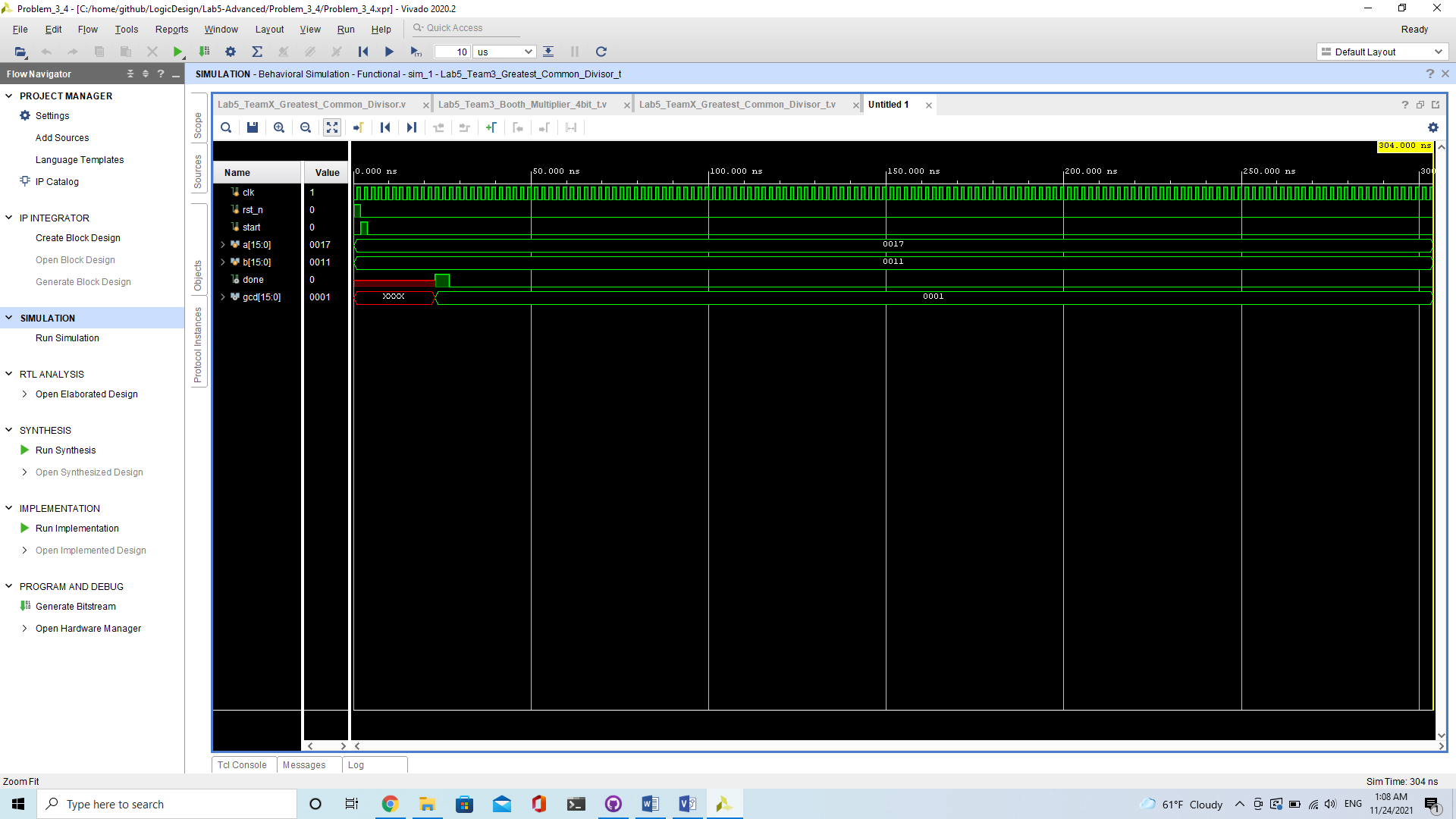
依照助教給定的虛擬碼進行實作，在重設時改變狀態到WAIT，在啟動時改變狀態到CAL，完成運算時（也就是A,B其中之一歸零時）改變狀態到FINISH。

為了維持有兩個時脈的Done，在完成運算時，我們將Done設為高電位，並且設立一個Dummy State，在FINISH狀態等候一個時脈後，轉移到Dummy State，並且將Done設為低電位。藉由設立一個新的Dummy State，就能輕鬆延長Done的維持時間。

1. 驗證：

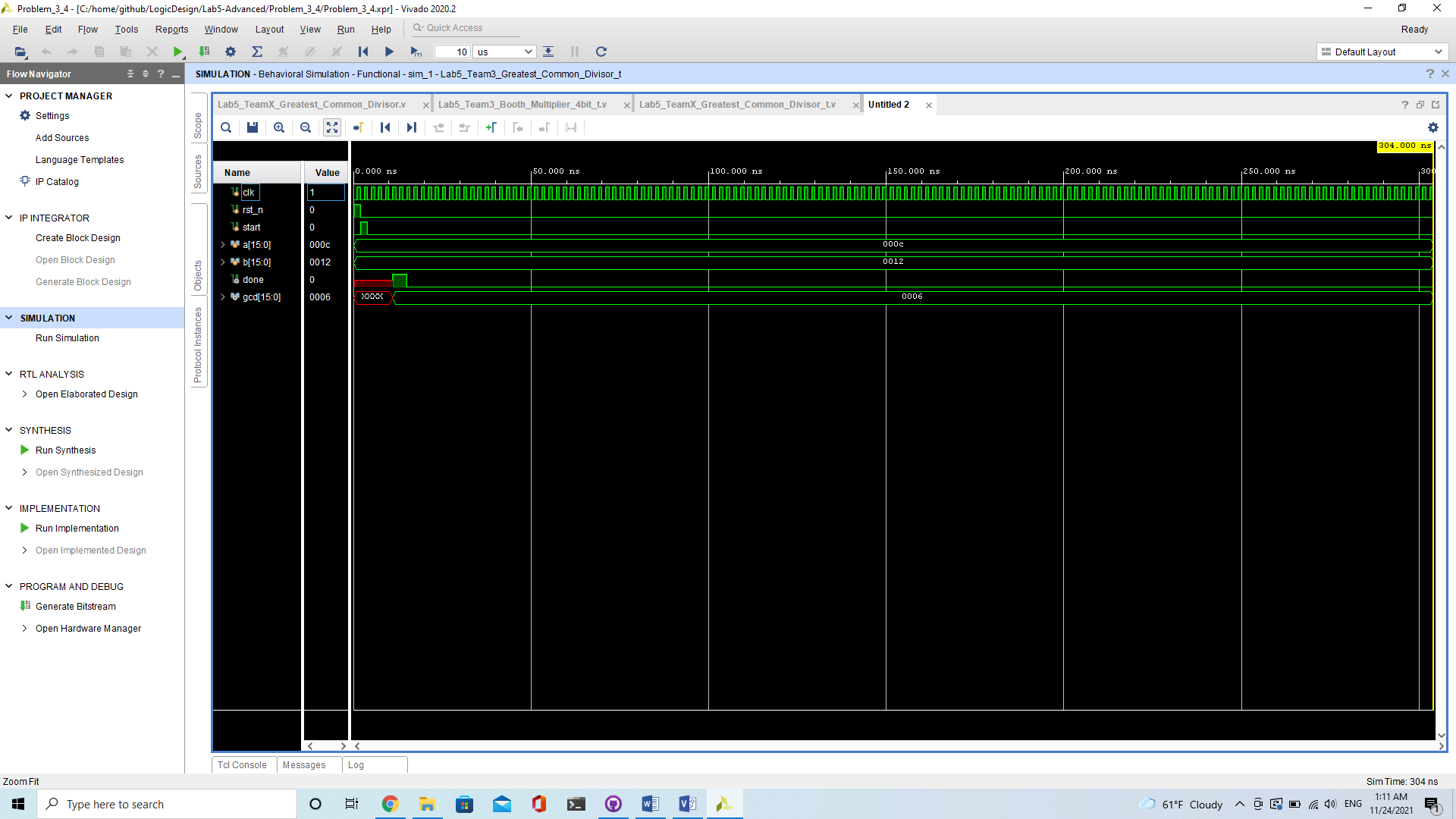
測試資料 #1

A=23, B=17



測試資料 #2

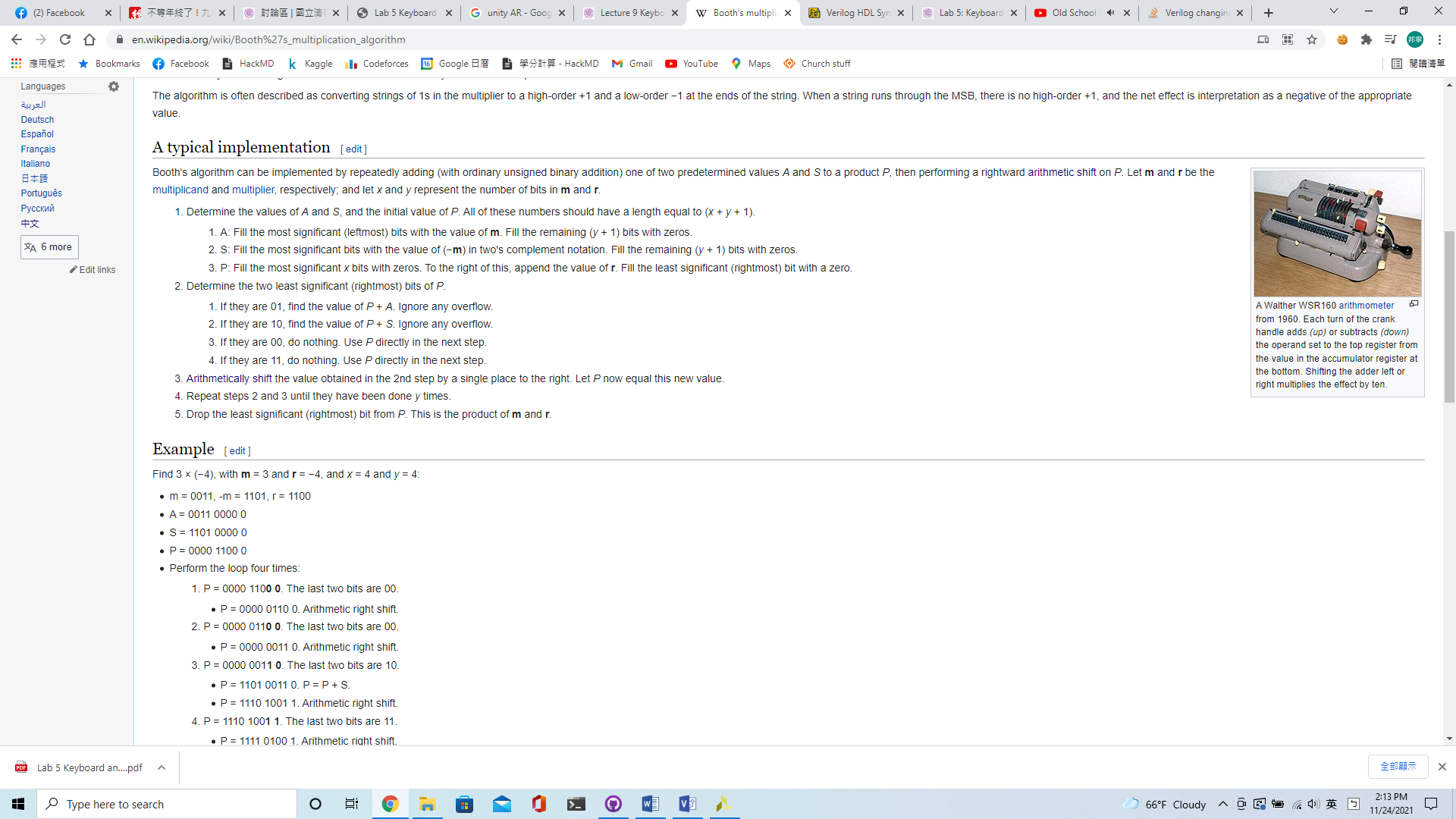
A=12 B=18



## Booth Multipler

1. 設計說明：

參考自維基百科



該算法的核心精神為「利用二進位的性質，將一堆加法組合成少量的加法與減法」，如果要計算一個位數布林乘法，該演算法需要個時脈，並且需要個電晶體。與暴力法相較，暴力法需要個時脈，但是需要個電晶體，由此可知該算法有「以時間換取空間」的特性。

假定為最負整數，也就是在位數二補數系統下的，的值沒辦法在位數二補數系統下表示，因為。為了避免溢位，我們將轉為位數二補數系統，並在前面append一個空格，藉此放入額外的位元。

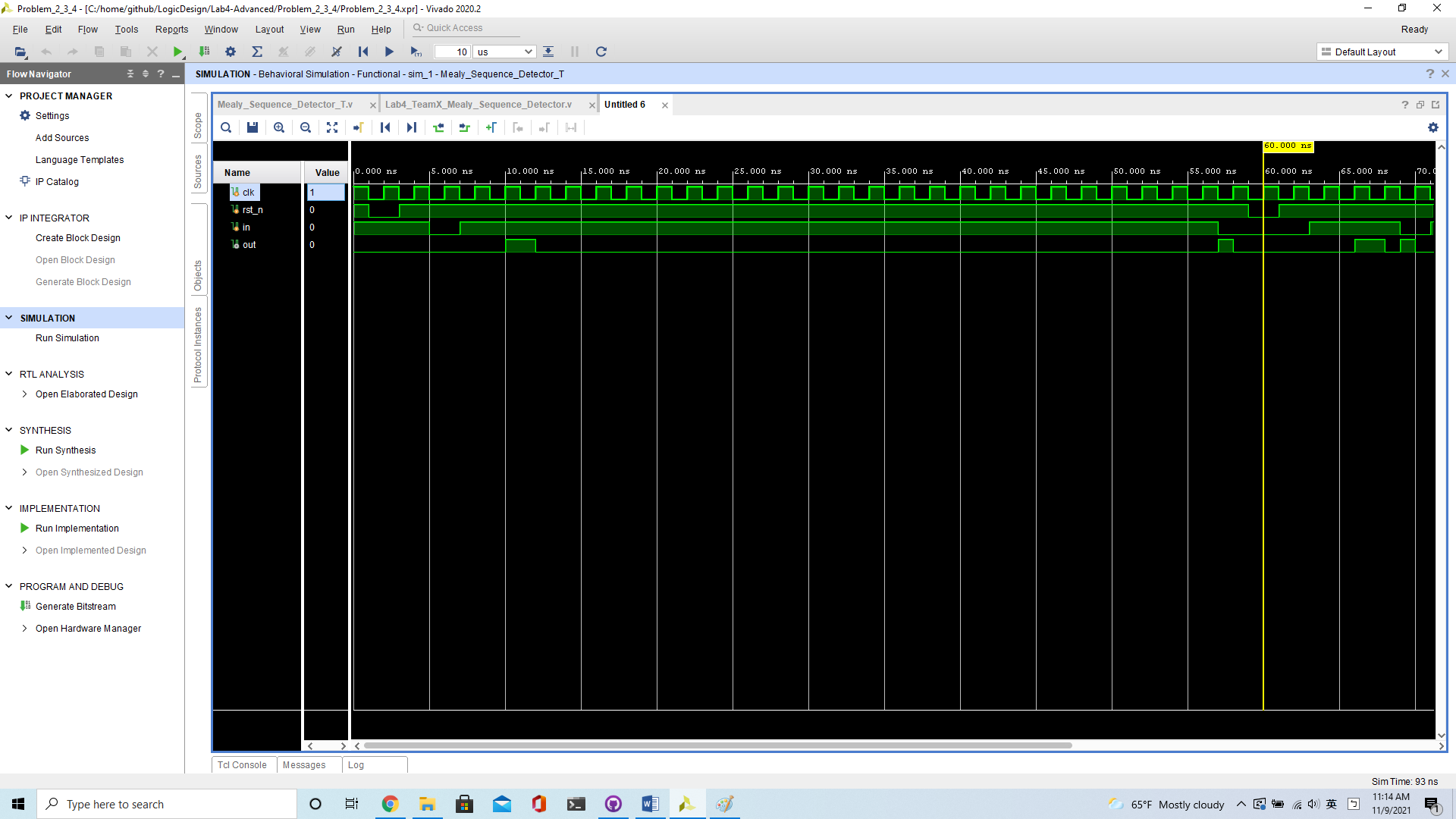
從位數二補數系統轉換到位數二補數系統的流程如下，我們分成兩個情況來討論，第一種情況是待轉換的數字。

第二種情況是待轉換的數字，

1. 驗證：

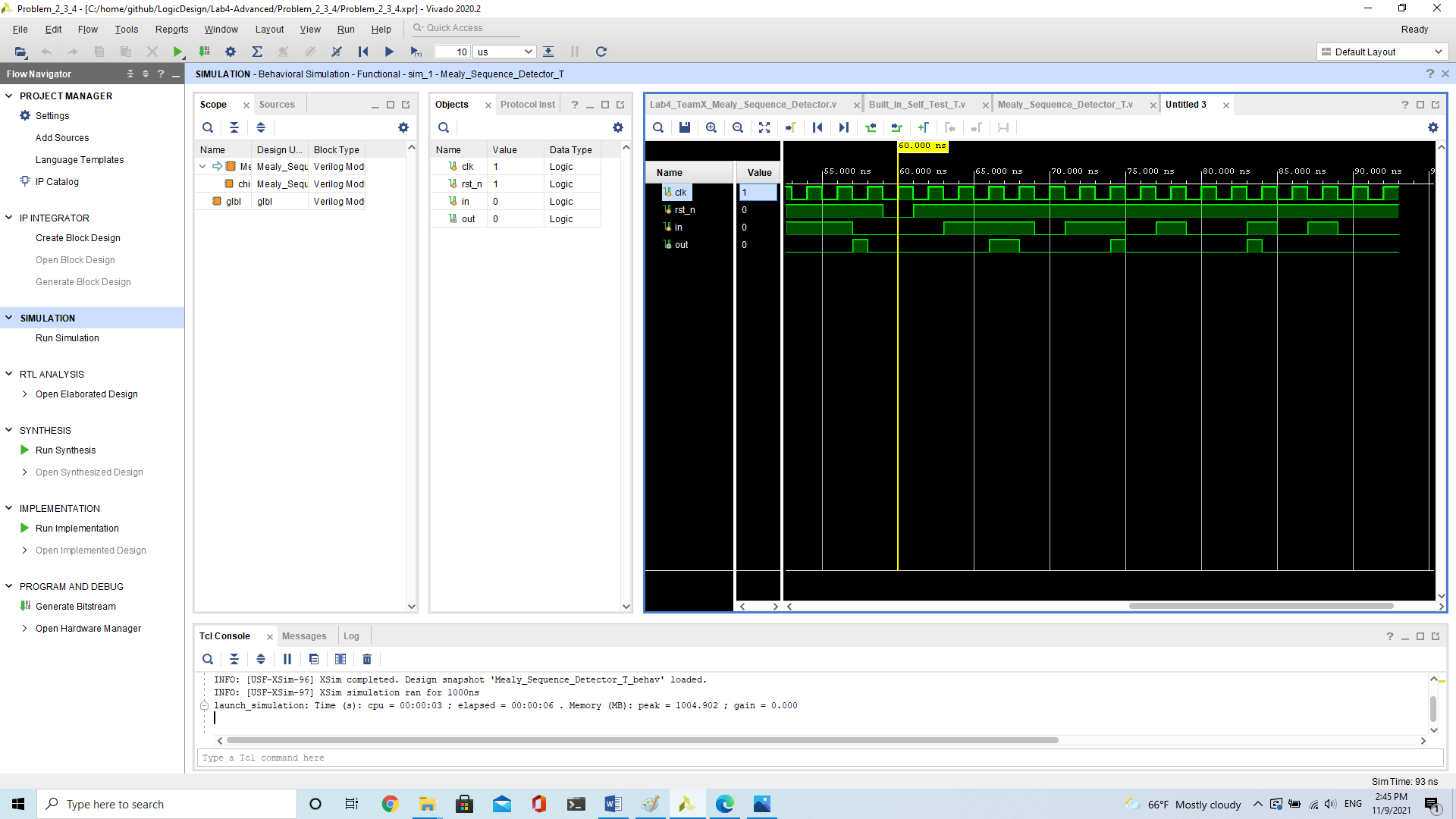
測試資料 #1

先讓資料都是高電位，時不時把電位拉下來。



測試資料 #2

同助教測試資料。



# Contribution

## Lawrence Wu

第二、三以及五題之實作以及報告撰寫。

## Ariel Chang

第一題實作及報告撰寫。

# What have we learned?

1. 多去看討論區！
2. 瀑布開發的效率很高，非常適合趕工用。
3. 不要寫髒Code，未來的自己會討厭你。
4. 盡量維持模組的可讀性，未來的自己會很感謝你。