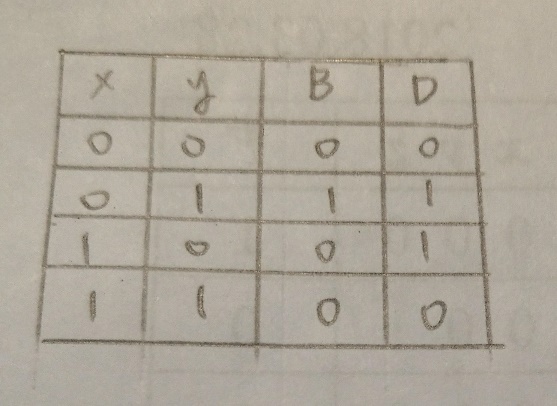
(1)

先寫出半減器的真值表:

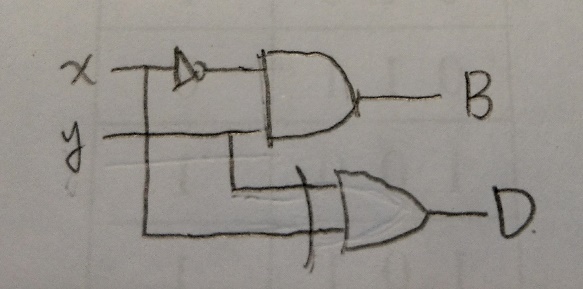


由真值表可以寫出布林代數式:

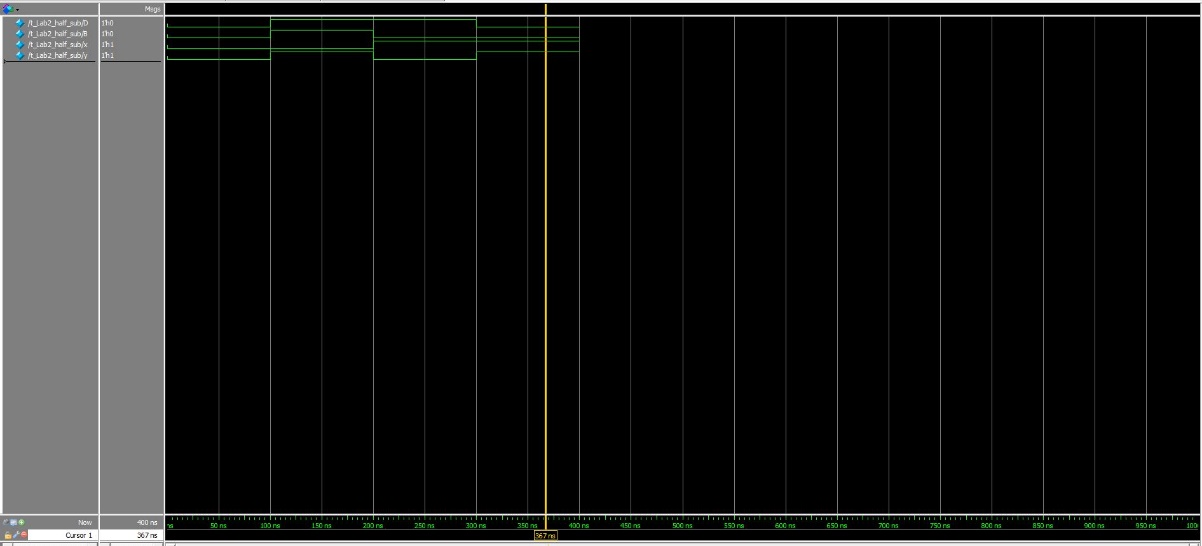
B = x’y

D = x’y + xy’

再從布林代數式畫出邏輯電路圖:



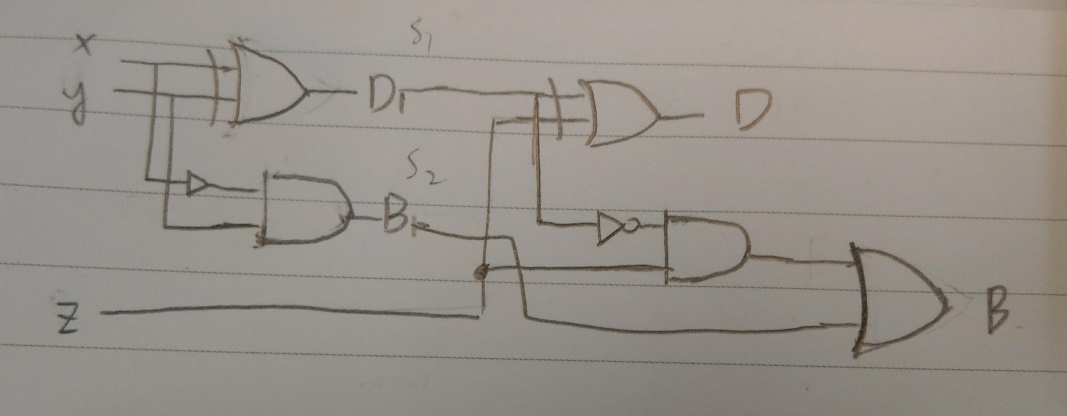
由於gatelevel 是以電路圖形式寫成、dataflow則以布林代數式為主、behavior則以類似布林代數式寫成，此三種形式雖然不同，但都描述同一個電路圖，所以模擬出來的波形圖相同，從波形圖來看也與真值表相同，故模擬正確。

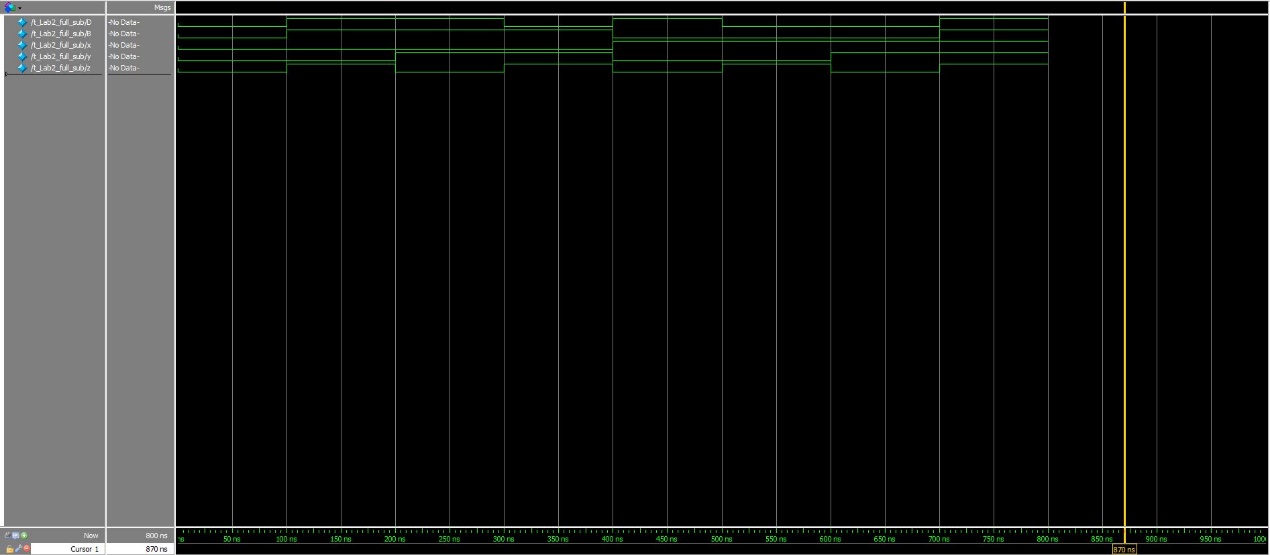


(2)

一個全減器可用兩個半減器建構而成，經過第一個半減器之後，得到D1和B1，接著再做第二次減法，為第二個半減器，D為D1和z XOR，B則由!D1 and z 在和 z or 一起(由布林代數式化簡成半減器的形式)，便形成一個全減器。

電路方塊圖則如下:

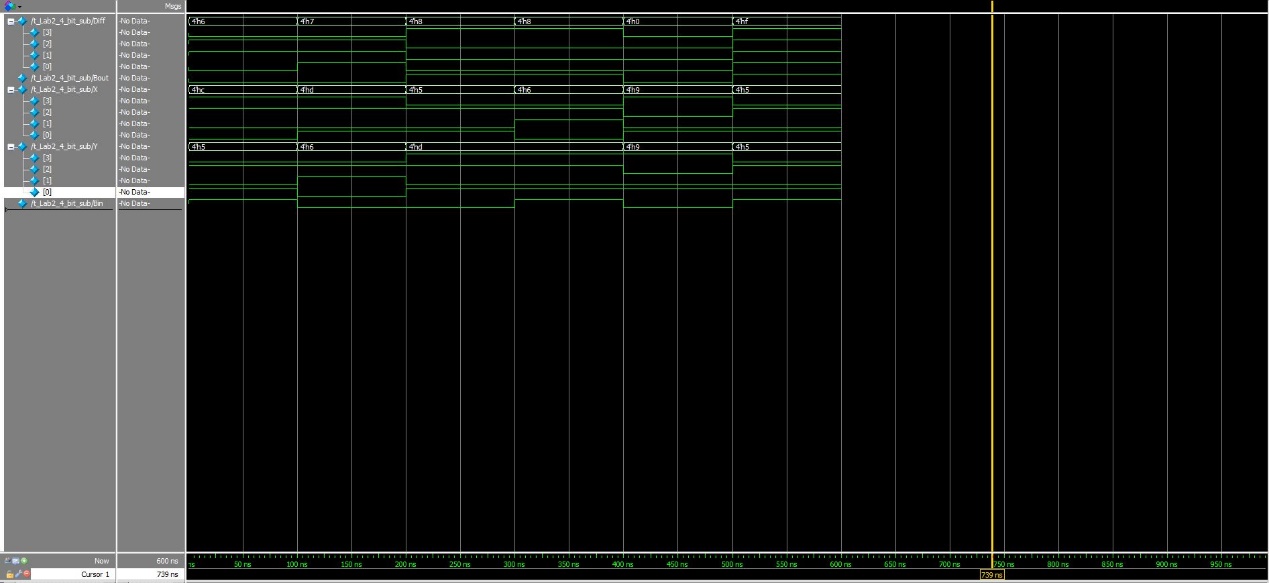




觀察模擬的波形圖，與結果相符合。

(3)

將輸入的四位元被減數和減數分成4個單獨的位元進行個別相減，第一個全減器做X[0]、Y[0]和Bin的減法，得到D[0]和借位B1，接著第二個全減器做X[1]、Y[1]和B1的減法，得到D[1]和B2，第三個全減器做X[3]、Y[3]和B2的減法，得到D[3]和B3，接著第四個全減器做X[4]、Y[4]和B3的減法，得到D[4]和Bout。



觀察模擬的波形圖，與結果相符。

(4)

由於(3)的作法Delay比較長，因此使用borrow lookahead，Delay相對少。

令P1 = X[0] ^ Y[0]

G1 = !X[0]&&Y[0]

則Diff[0] = Bin^P1

B1 = G1 ||!P1 && Bin

令P2 = X[1]^Y[1]

G2 = !X[1] &&Y[1]

則Diff[1] = B1^P2

B2 = G2|| !P2 && B1

令P3 = X[2]^Y[2]

G3 = !X[2] &&Y[2]

則Diff[2] = B2^P3

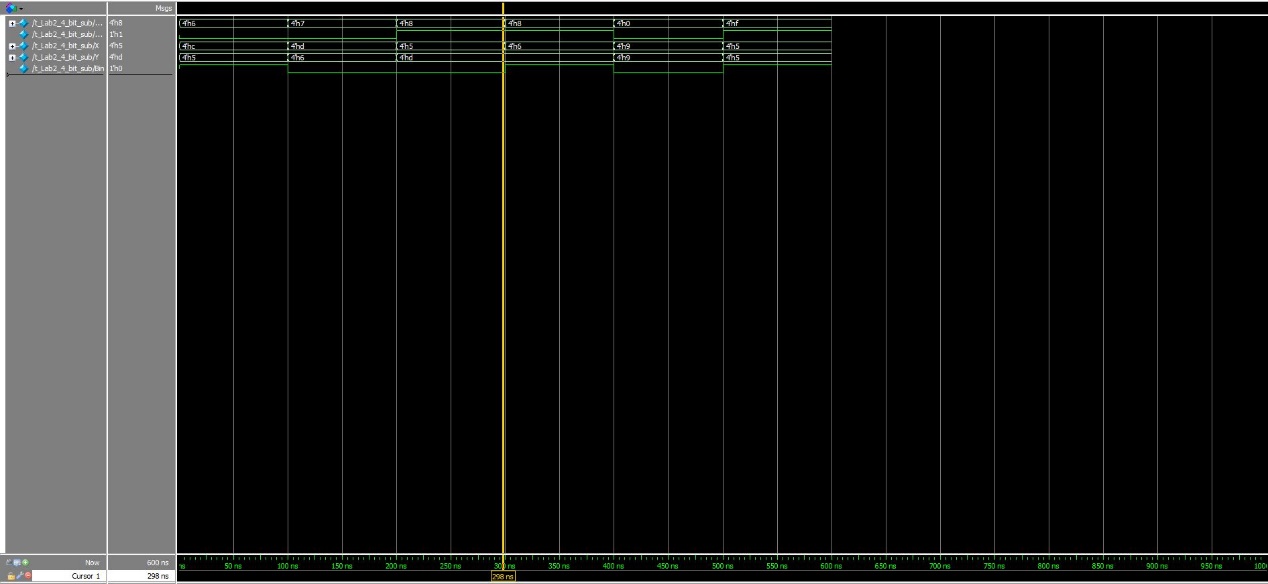
B3 = G3|| !P3 && B2

令P4 = X[3]^Y[3]

G4 = !X[3] &&Y[3]

則Diff[3] = B3^P4

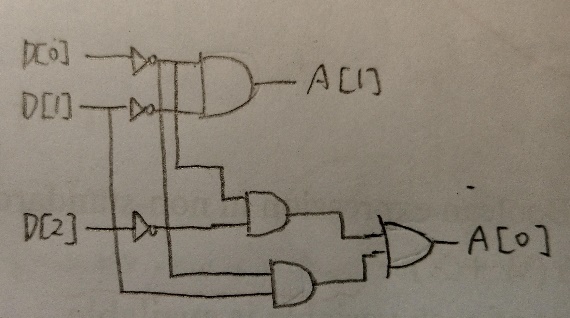
Bout = G4|| !P4 && B3

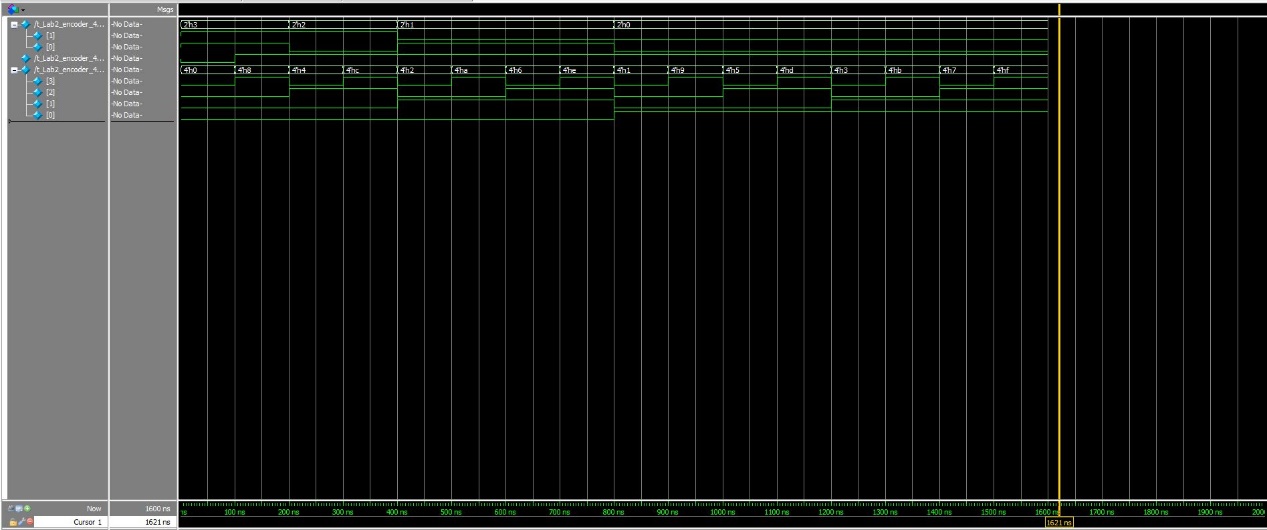


觀察模擬的波形圖，與(3)的結果相同。

(5)

觀察四輸入優先編碼器的真值表，由於D[0]的優先序最高，所以D[0]為1時，其他為don’t care，且輸出A為00，V為1，以此類推到D[3]，最後畫出K map，可以得到A[1] 的布林代數式為!D[1] && !D[0]，A[0]則為 !D[0] && !D[2] || !D[0] && D[1] ，V為D[0] || D[1] || D[2] || D[3] ，畫成電路方塊圖:





觀察模擬的波形圖，與結果相符合。

(6)

這次的作業雖然比較多也比較難，但是也透過這次的LAB，讓我更熟悉講義的內容，也順便做一次完整的複習。