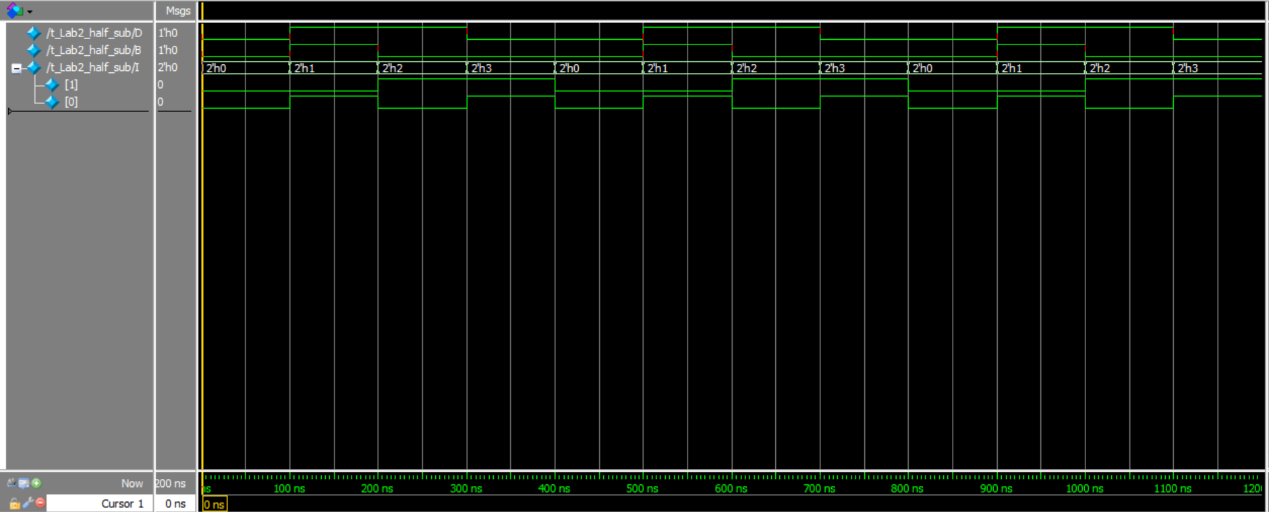
1. **2A iv (半減器) 模擬結果波形圖**



＊測試模組中I[1]代表x、I[0]代表y

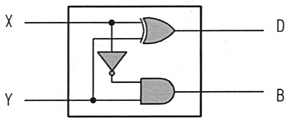
x - y 的差值為D、借位為B真值表如下

|  |  |  |  |
| --- | --- | --- | --- |
| x | y | B | D |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 |

則可推出

B = ~x & y  
D = x ^ y

**Gate-level modeling**



xor G1(D, x, y)

and G2(B, ~x, y)

波形圖為0至400ns

**Dataflow modeling**

assign D = x ^ y

assign B = ~x & y

波形圖為400至800ns

**Behavior modeling**

當每次輸入改變的時候

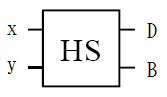
D = x ^ y

B = ~x & y

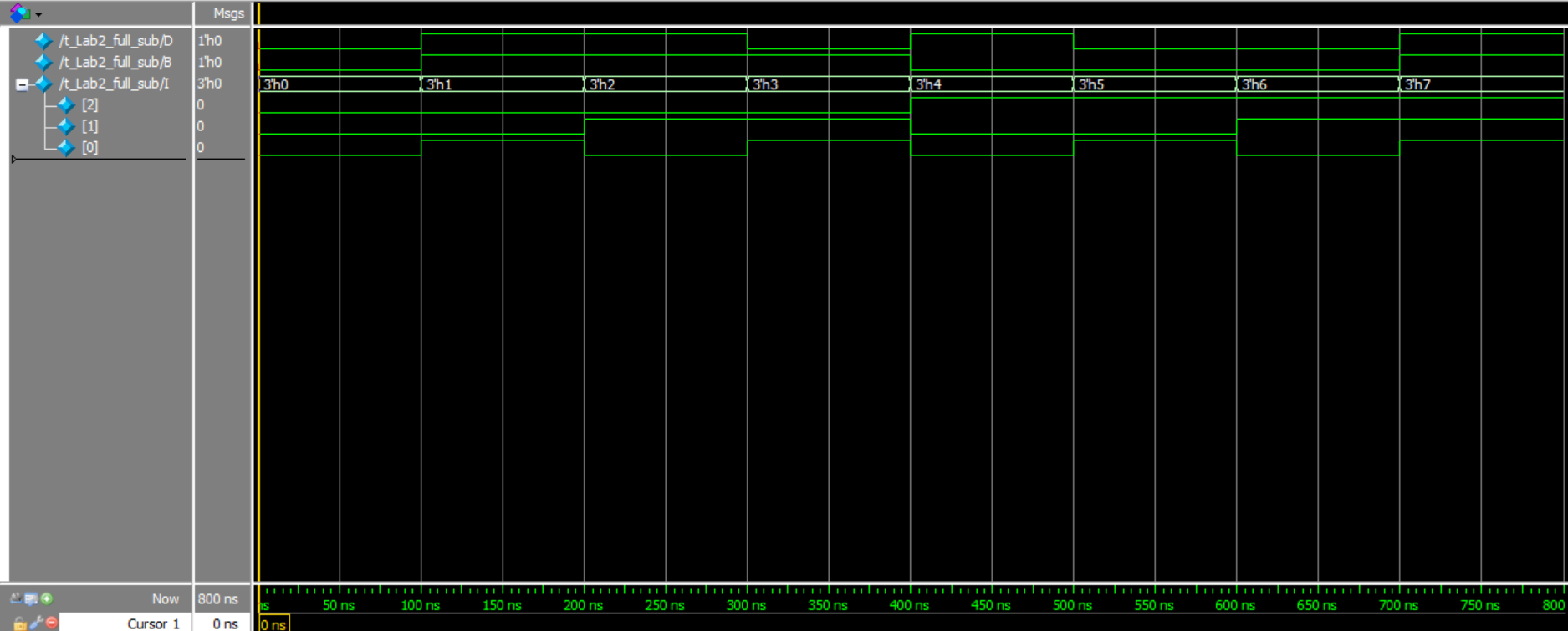
波形圖為800至1200ns

用以上三種不同的方式撰寫半減器，模擬後所得的波形圖並無不同，且波形圖與所推導的真值表結果相符，故三種方式撰寫的半減器皆正確。

以下利用方塊圖來代表半減器



1. **2B ii (全減器) 模擬結果波形圖**

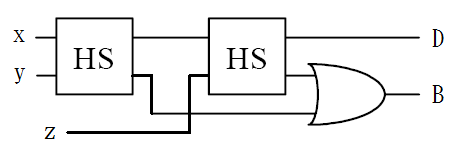


＊測試模組中I[2]代表x、I[1]代表y、I[0]代表z

x - y - z 的差值為D、借位為B真值表如下

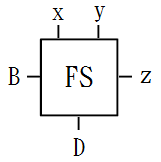
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x | y | z | B | D |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

可利用兩個半減器兜出全減器

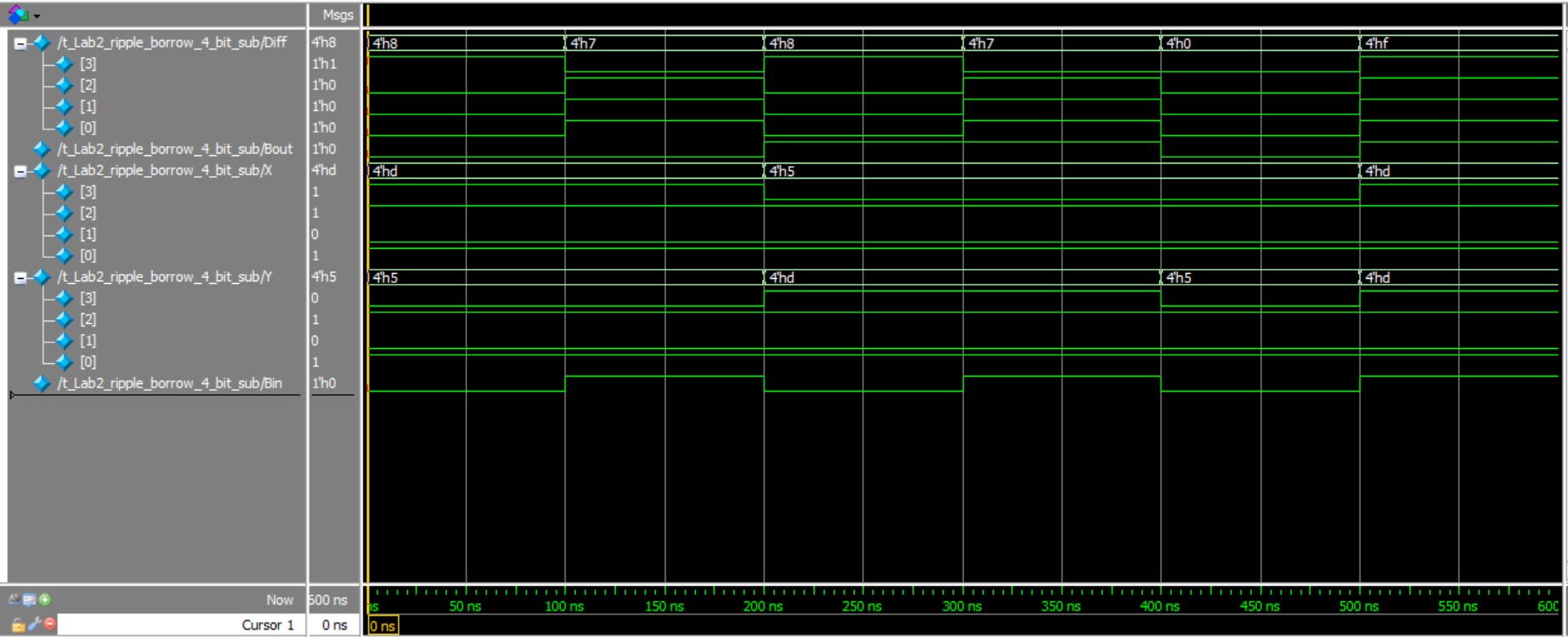


模擬後所得的波形圖與真值表相符故正確。

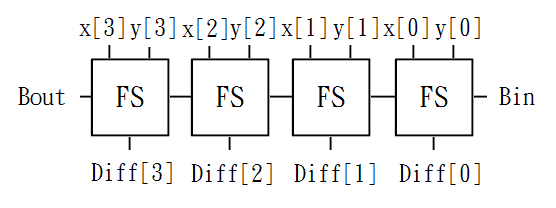
以下利用方塊圖來代表全減器



1. **2C ii (4-bit RBS) 模擬結果波形圖**



可利用四個全減器兜出4-bit RBS

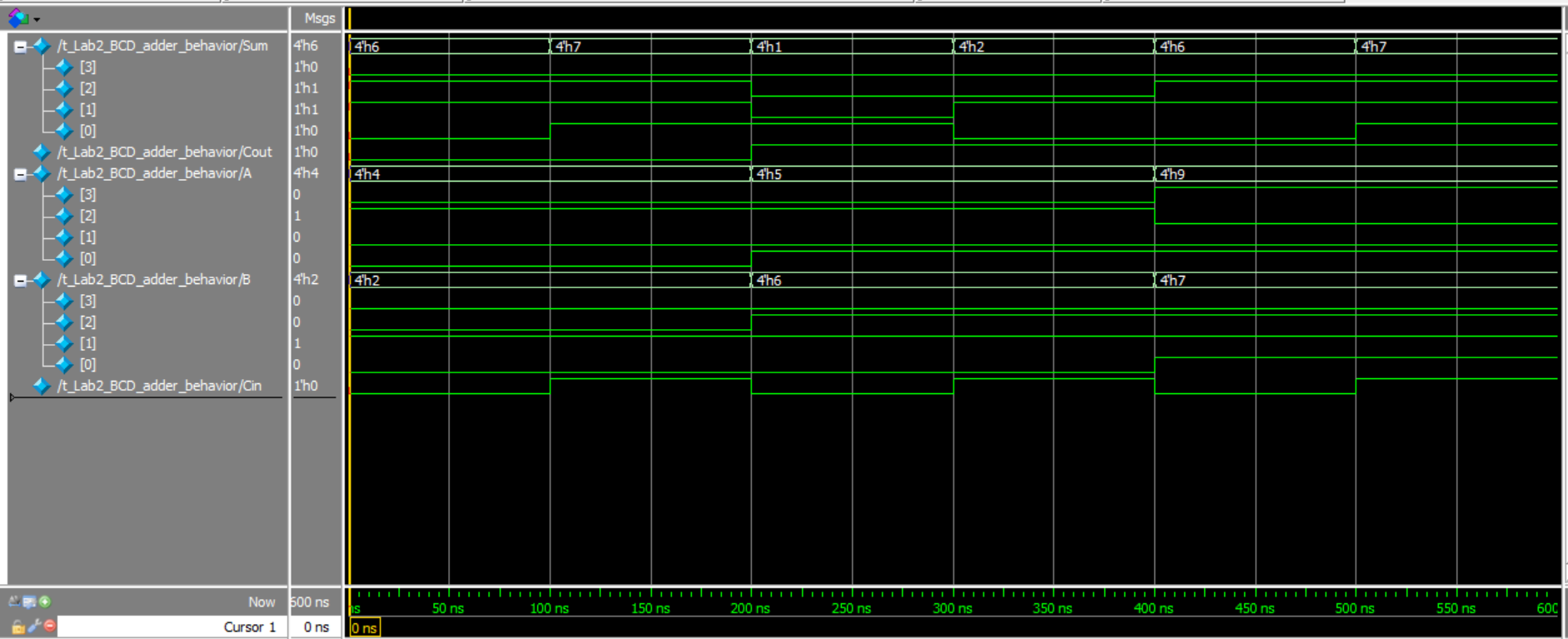


以下測試資料在波形圖上顯示結果皆正確

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x | y | Bin | Diff | Bout |
| 1101 | 0101 | 0 | 1000 | 0 |
| 1101 | 0101 | 1 | 0111 | 0 |
| 0101 | 1101 | 0 | 1000 | 1 |
| 0101 | 1101 | 1 | 0111 | 1 |
| 0101 | 0101 | 0 | 0000 | 0 |
| 1101 | 1101 | 1 | 1111 | 1 |

利用程式測試所有輸入組合所得的答案皆正確

1. **2D ii (BCD加法器) 模擬結果波形圖**



利用一個5-bit的reg存A + B + Cin的值(最大值為10011)，若超過01001則要再加00110(跳過6個沒用到的編碼)，且把Cout的值設為1。Sum值為5-bit的reg後4-bit的值。

以下測試資料在波形圖上顯示結果皆正確

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | Cin | Sum | Cout |
| 0100 | 0010 | 0 | 0110 | 0 |
| 0100 | 0010 | 1 | 0111 | 0 |
| 0101 | 0110 | 0 | 0001 | 1 |
| 0101 | 0110 | 1 | 0010 | 1 |
| 1001 | 0111 | 0 | 0110 | 1 |
| 1001 | 0111 | 1 | 0111 | 1 |

利用程式測試所有輸入組合所得的答案皆正確

1. 在這次作業中，我更加熟悉了如何利用gate-level、dataflow的技巧設計電路，並且學會了behavior的設計方法。體會到用一個個已經設計好的小電路兜出大電路的感覺，多累積一些基本的電路，之後就可以設計出更多更複雜的電路。在撰寫測試模組時，我學到了可以利用repeat指令生成n – bit所有的排列組合，非常的方便易讀。希望之後可以學到同步順序電路的設計方法。