**Lab1\_1**

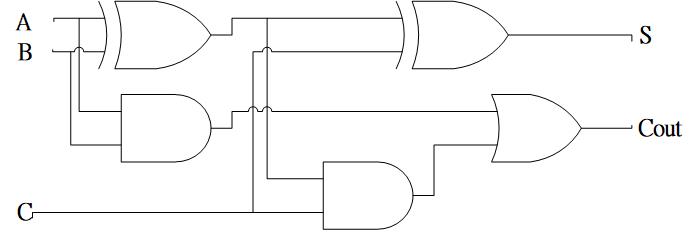
**Design Specification**

* For a full adder:

Input: a, b, cin.

Output: s, cout.

* Draw the block diagram of the design.

****

**Design Implementation**

* Truth table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a | b | cin | s | cout |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

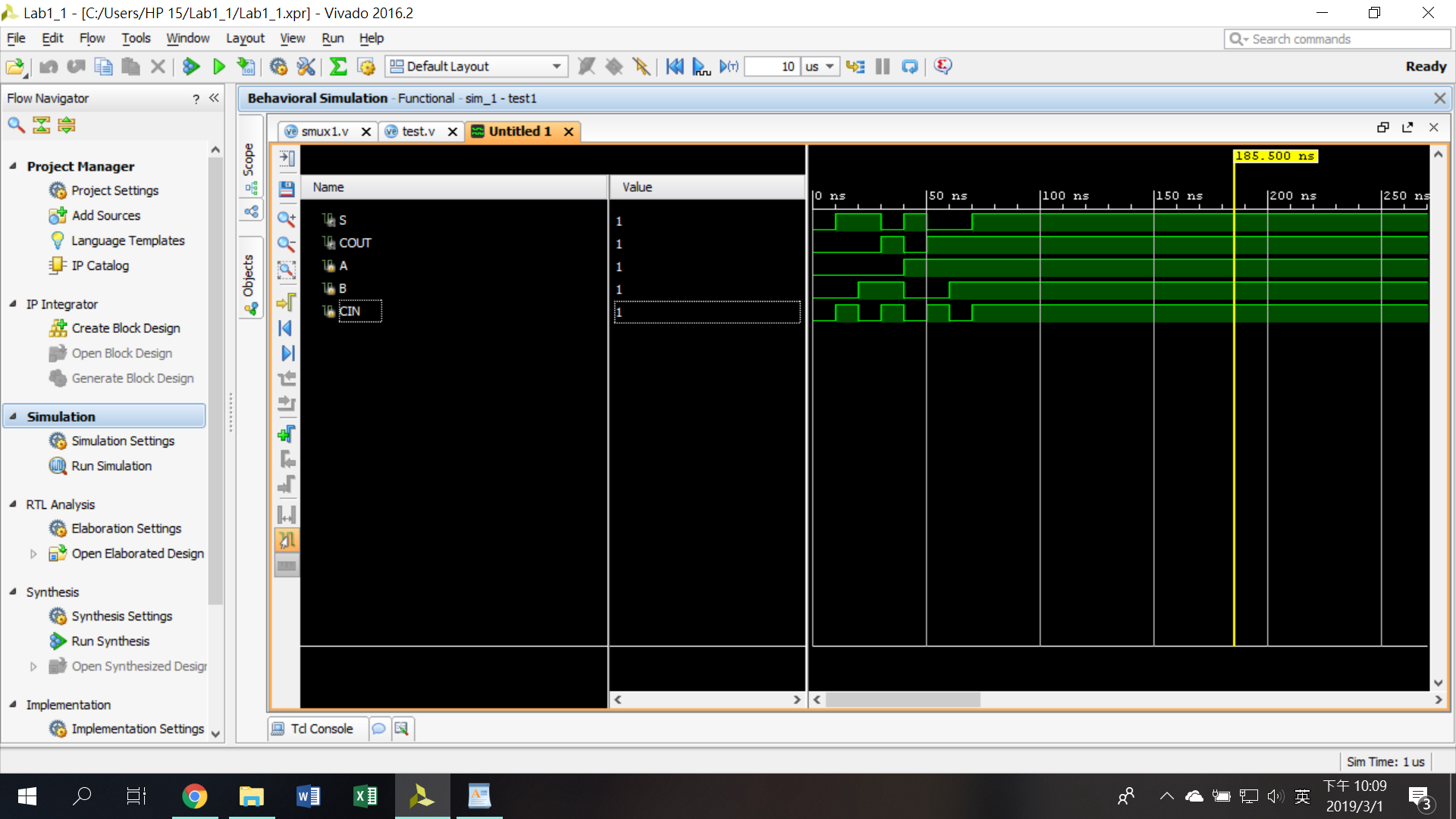
* Logic function

s = (a’．b’．cin) +(a’．b．cin’) +(a．b’．cin’) +(a．b．cin)

= a⊕b⊕cin

cout = a．b + b．cin + a．cin

* Simulation



**Lab1\_2**

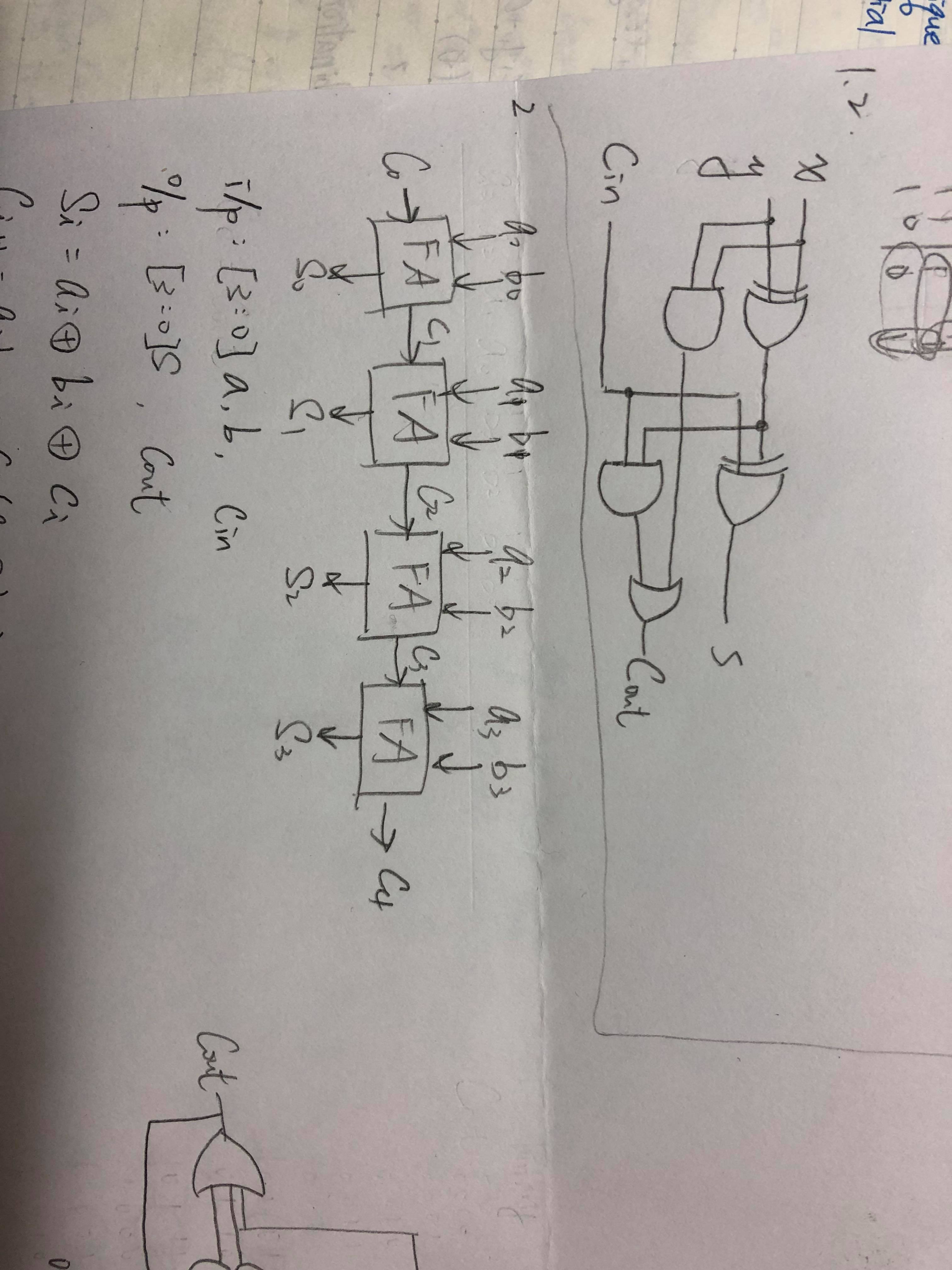
**Design Specification**

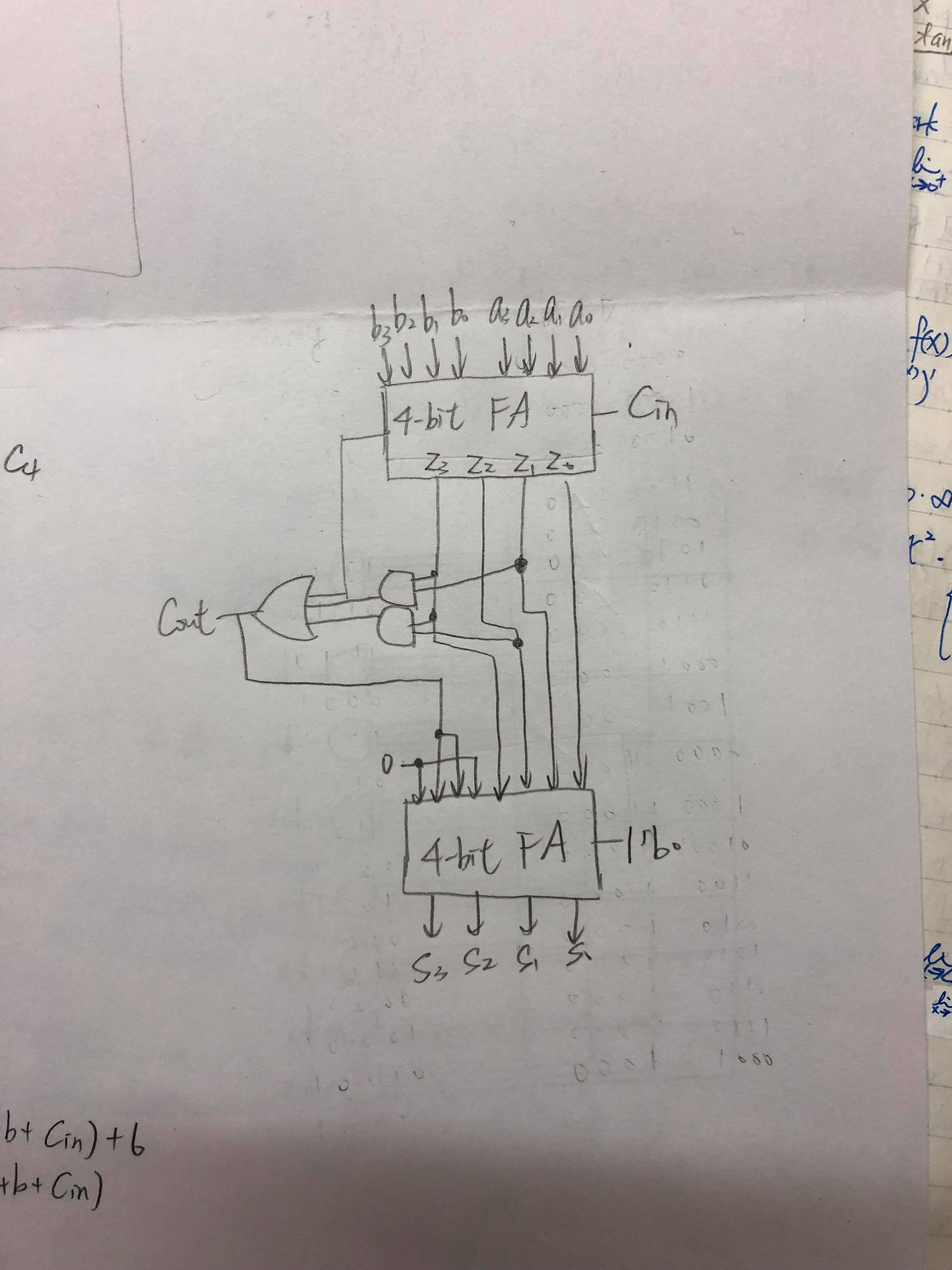
* For a 4-bit full adder:

Input: a[3:0], b[3:0], cin.

Output: s[3:0], cout.

* Draw the block diagram of the design.

****

****

**Design Implementation**

* Truth table

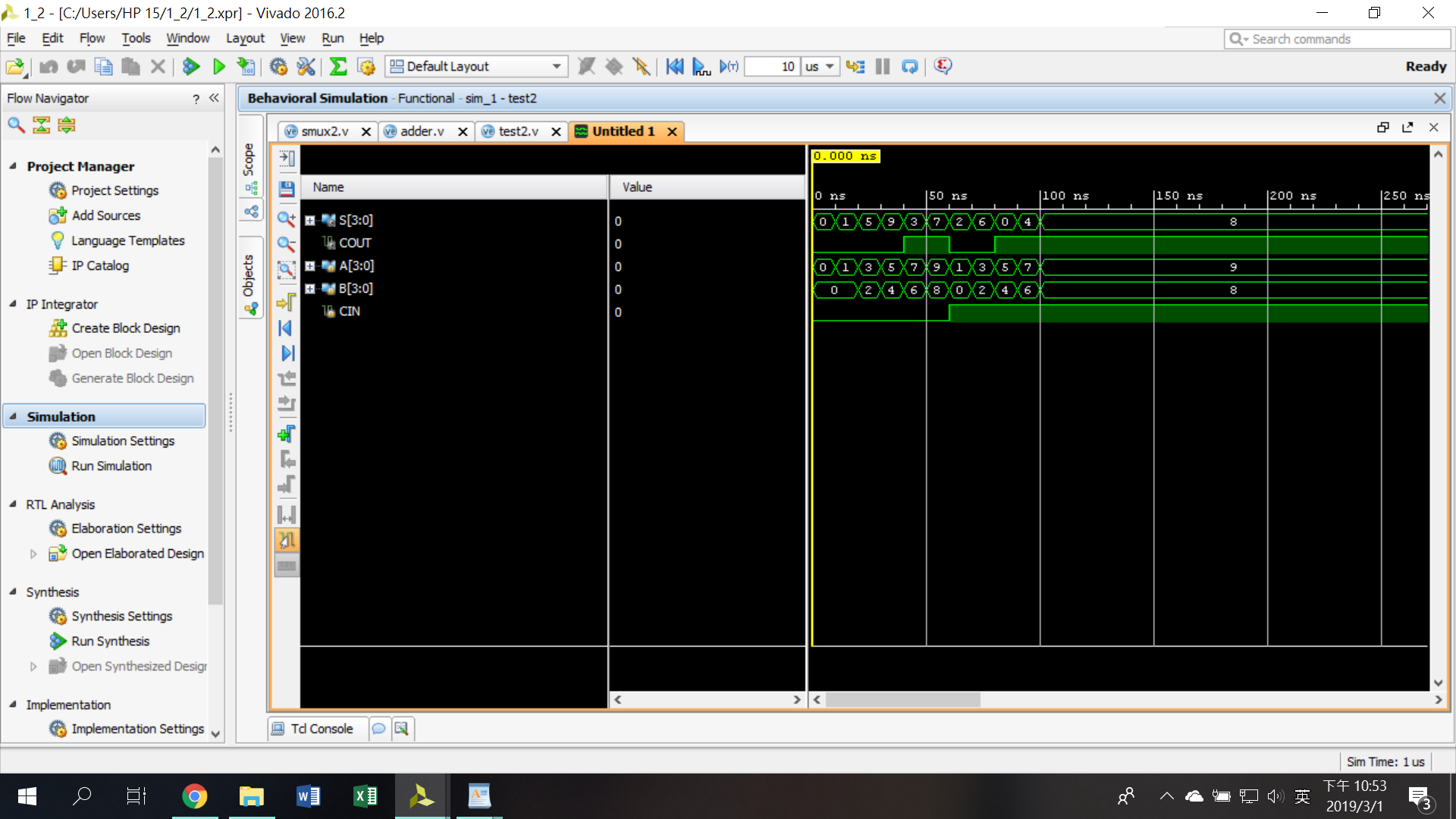
|  |  |
| --- | --- |
| Binary | BCD |
| 0000 0000 | 0000 0000 |
| 0000 0001 | 0000 0001 |
| 0000 0010 | 0000 0010 |
| 0000 0011 | 0000 0011 |
| 0000 0100 | 0000 0100 |
| 0000 0101 | 0000 0101 |
| 0000 0110 | 0000 0110 |
| 0000 0111 | 0000 0111 |
| 0000 1000 | 0000 1000 |
| 0000 1001 | 0000 1001 |
| 0000 1010 | 0001 0000 |
| 0000 1011 | 0001 0001 |
| 0000 1100 | 0001 0010 |
| 0000 1101 | 0001 0011 |
| 0000 1110 | 0001 0100 |
| 0000 1111 | 0001 0101 |
| 0001 0000 | 0001 0110 |
| 0001 0001 | 0001 0111 |
| 0001 0010 | 0001 1000 |

* Logic function

If (a+b+ci )> 9, then co=1 s=(a+b+ci)+6;

If (a+b+ci )≤ 9, then co=0 s=(a+b+ci)

* Simulation



**Lab1\_3**

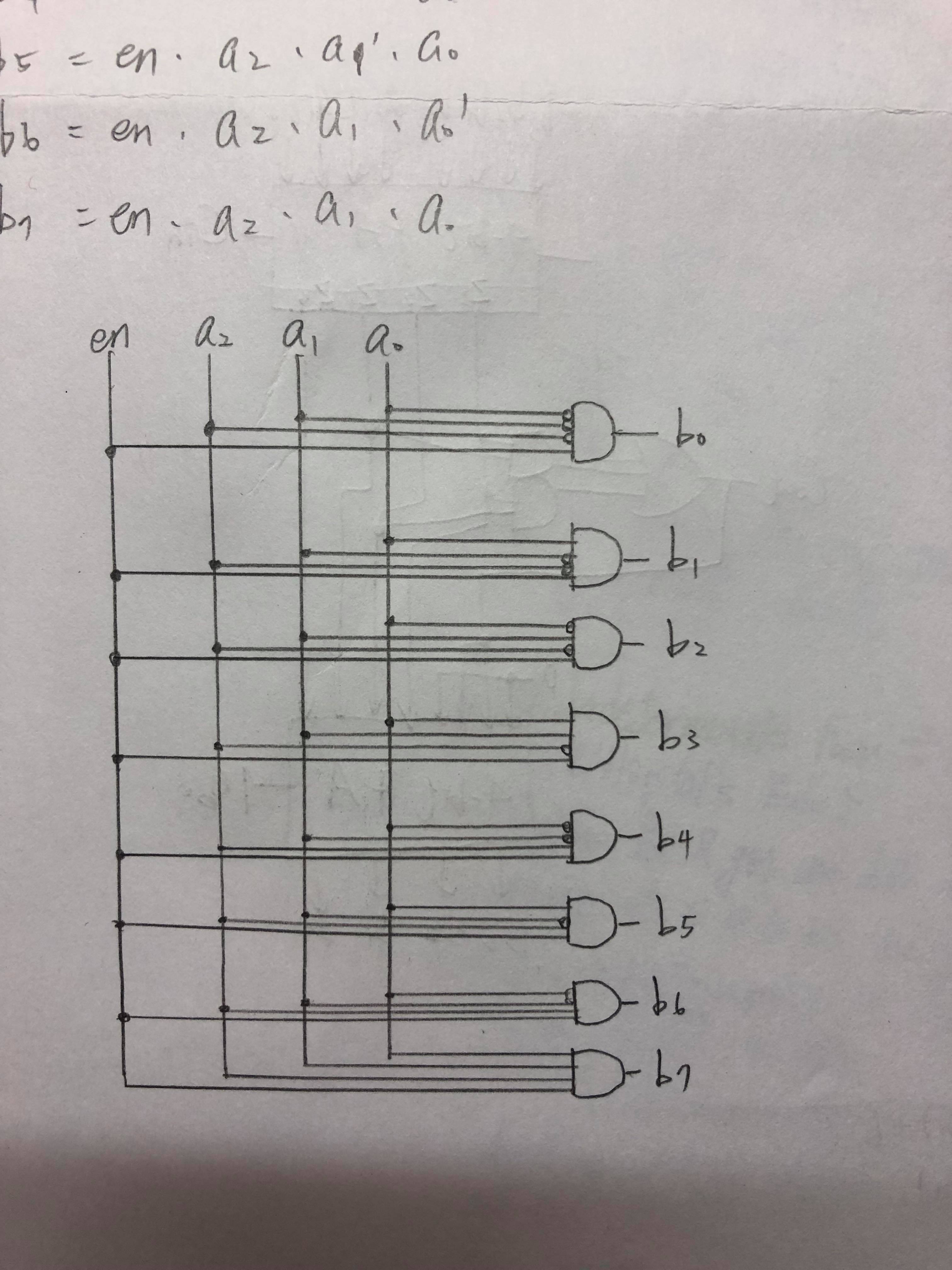
**Design Specification**

* I/P

Input: a[2:0], en.

Output: b[7:0].

* Block diagram

****

**Design Implementation**

* Truth table

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| en | a2 | a1 | a0 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| en | a2 | a1 | a0 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

* Logic function

b[0] = en．a[2]’．a[1]’．a[0]’

b[1] = en．a[2]’．a[1]’．a[0]

b[2] = en．a[2]’．a[1]．a[0]’

b[3] = en．a[2]’．a[1]．a[0]

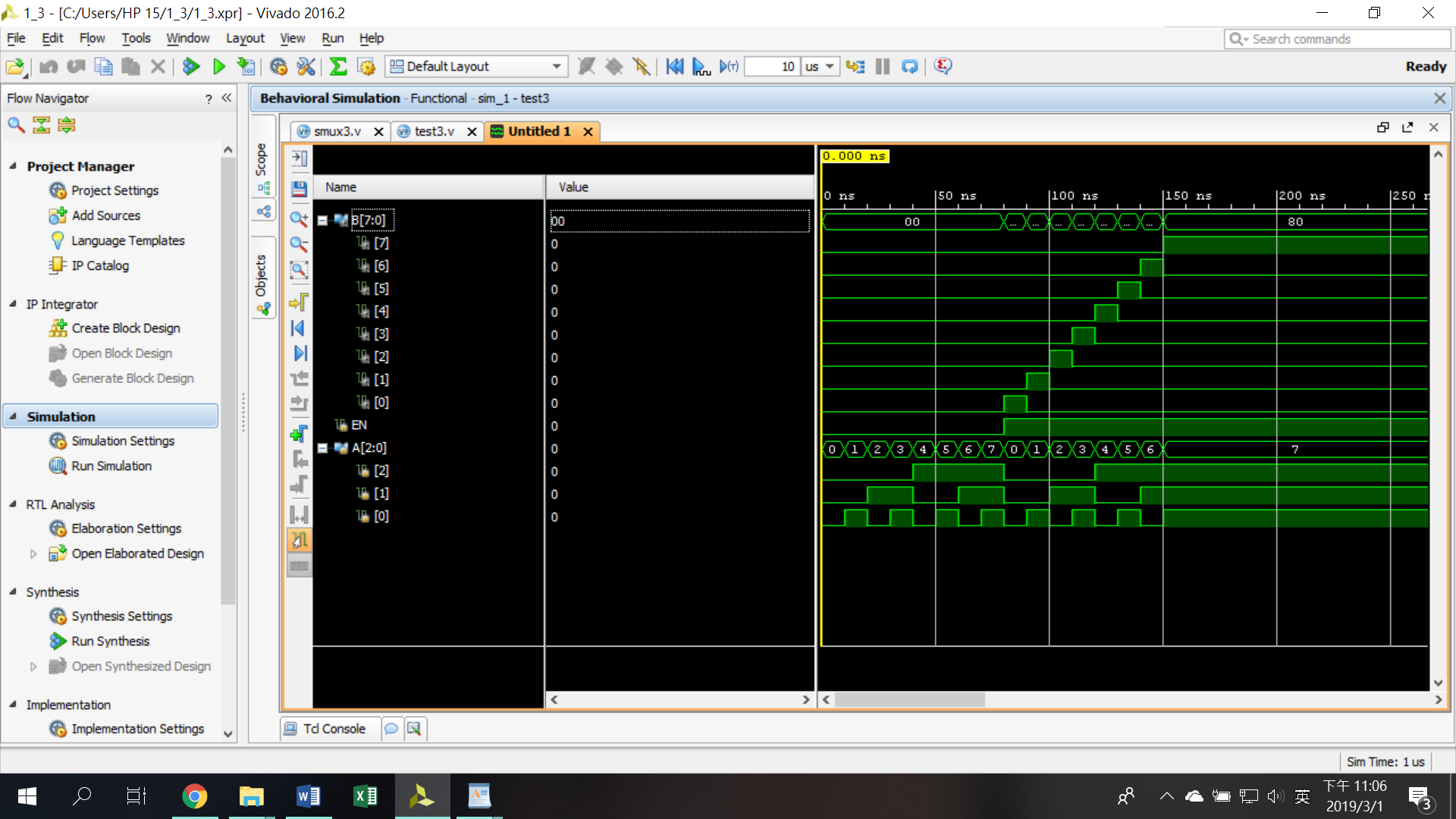
b[4] = en．a[2]．a[1]’．a[0]’

b[5] = en．a[2]．a[1]’．a[0]

b[6] = en．a[2]．a[1]．a[0]’

b[7] = en．a[2]．a[1]．a[0]

* Simulation



**Discussion**

由於第一題與第三題上學期上課內容有提過，因此在思考上比較不困難。第二題的部分原本一開始是直接用加法的方式去做，但助教表示這是軟體的寫法，所以要改成硬體邏輯的時候比較困難。因此在解第二題時，寫出truth table便可以知道在何種情況下cout為1。s[3:0]是經過兩個4-bit FA才得出的結果是因為轉成BCD時所造成的改變。如果一開始以單純adder的想法去寫很容易走偏。在設計線路時很常出現小錯誤，完全無法simulation，可以simulation時都是已經確認邏輯設計是正確後才得到波形圖，與一開始預期的結果亦相同。

**Conclusion**

這次實驗給我最大的收穫是了解到軟體的思維不能直接用在硬體邏輯的程式上，即使結果是對的，但卻沒有練習到本實驗課所要學習的內容。Lab1的題目主要是複習上學期所學，較困難的是結合不同單元的應用。這次實驗讓我了解到truth table的重要，必須自己先釐清邏輯才可以開始寫程式。