實驗主題(Lab03)

B083022053

黃啟桓

Exercise 1:

Verify x'y'z + x'yz + xy' = xy'+x'z

(using Schematic)

Exercise 2:

Verify Postulate 4(b) x + yz = (x + y) (x + z)

(using Dataflow modeling)

Exercise 3:

Verify xy + xy'z + x'yz = xy + xz+yz

(using Structural level modeling)

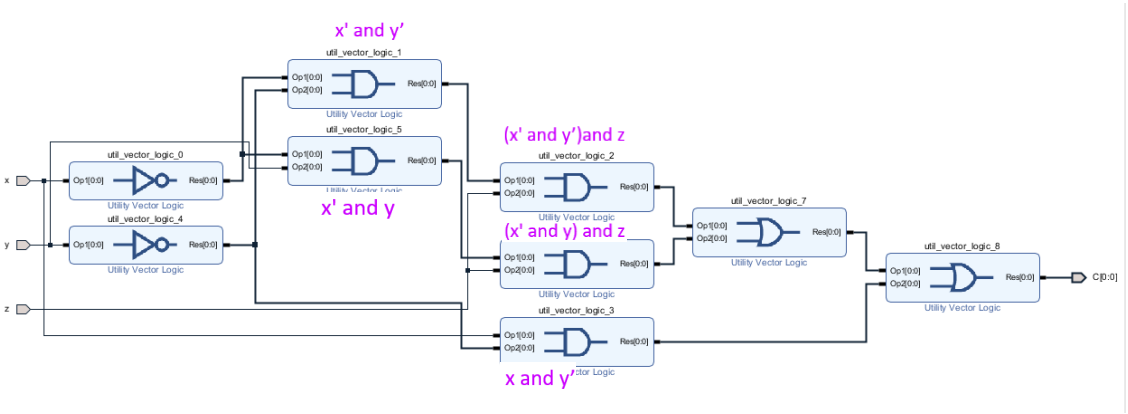
實驗內容

一、Exercise 1:

Verify　　x'y'z + x'yz + xy' = xy'+x'z

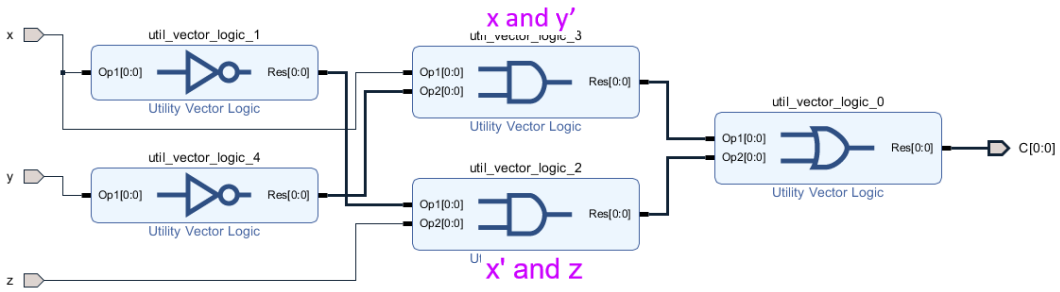
(using Schematic)

(一)　等式左方

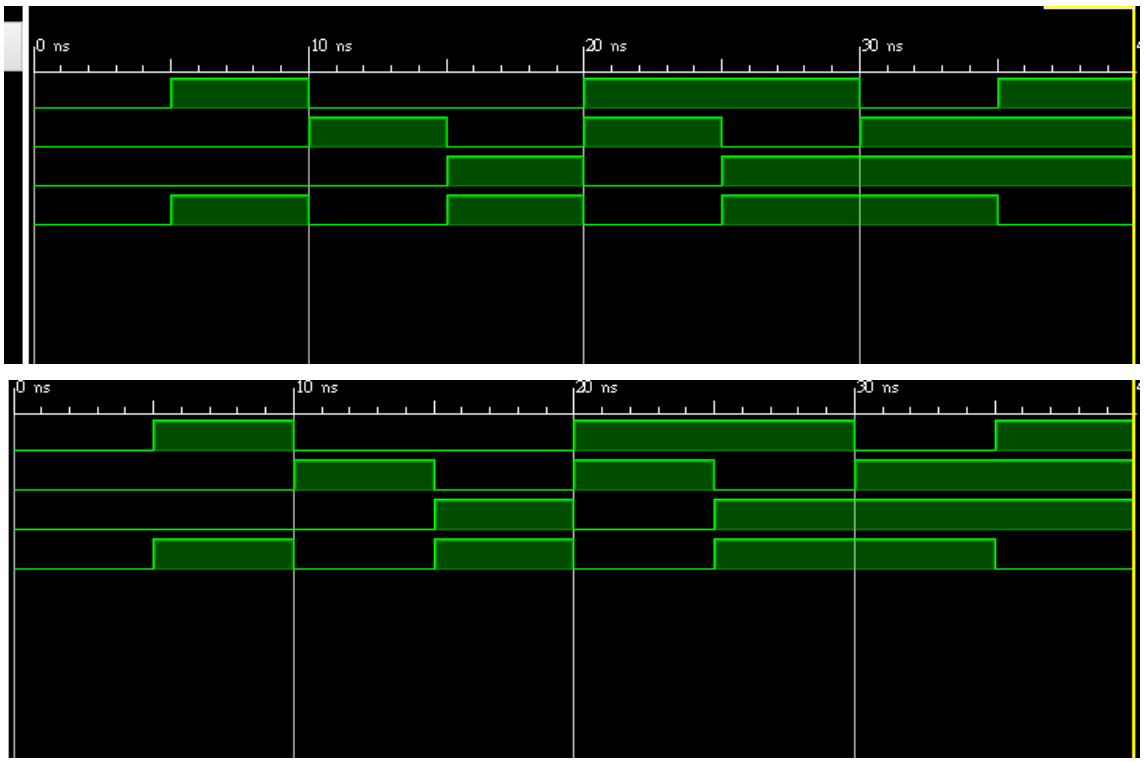


因為 vivado 沒有 3 輸入的 and 與 or 邏輯閘，故較為複雜，因為 and 與 or 具有分配性可以使用 2 個 2 輸入的邏輯閘完成 3 輸入運算。

(二)　等式左方



方程式左方比上式更簡化，且僅需 2 輸入的 and、or 邏輯閘，電晶體需求少，成本更低，省電，體積更小，延遲更少



由電壓圖可知 x'y'z + x'yz + xy' = xy'+x'z 是成立的

二、Exercise 2:

Verify Postulate 4(b) x + yz = (x + y) (x + z)
(using Dataflow modeling)

(一) 等式左方 程式碼表示如右：

因 and (&) 優先權高於 or (|) 所以不須加括號，由程式碼可知運算時需要一個 and 邏輯閘和一個 or 邏輯閘，且 and 邏輯閘在第一層，而 or 邏輯閘在第二層

```
`timescale 1ns / 1ps



module exercise_2(C,x,y,z);
    output C;
    input x,y,z;
    assign C = x | y & z;
endmodule
```

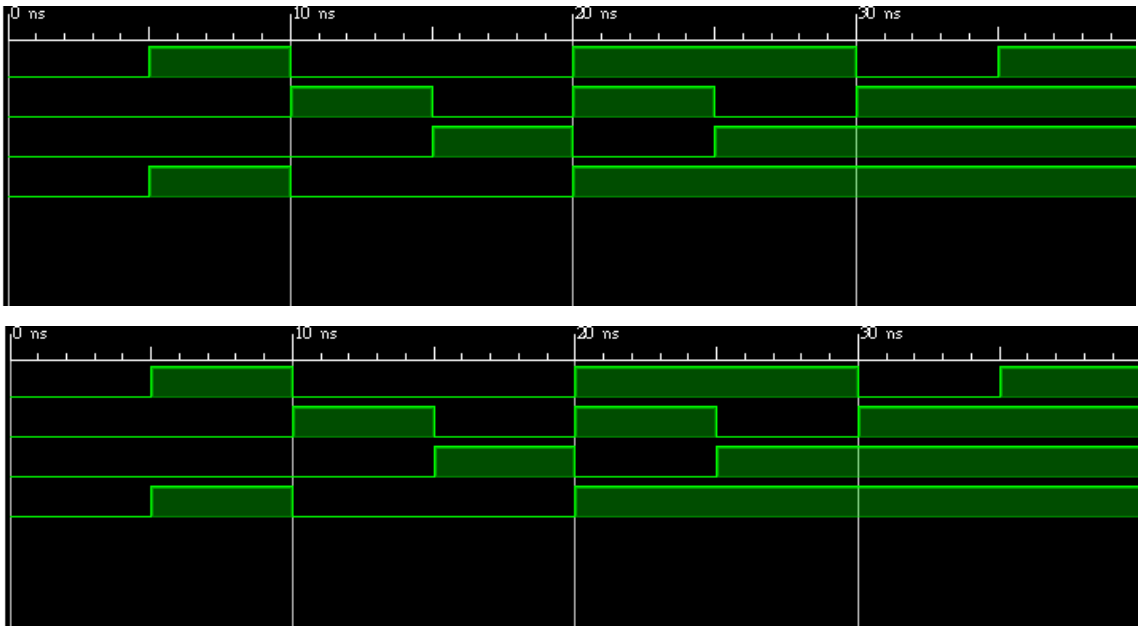
(二) 等式右方 程式碼表示如右：

因 and (&) 優先權高於 or (|) 所以須加括號，由程式碼可知運算時需要一個 and 邏輯閘和兩個 or 邏輯閘，且 or 邏輯閘在第一層，而 and 邏輯閘在第二層。製作成本比上方的高

```
`timescale 1ns / 1ps



module exercise_2(C,x,y,z);
    output C;
    input x,y,z;
    assign C =( x | y )&(x | z);
endmodule
```



比較 2 張電壓圖，可以證實等式成立

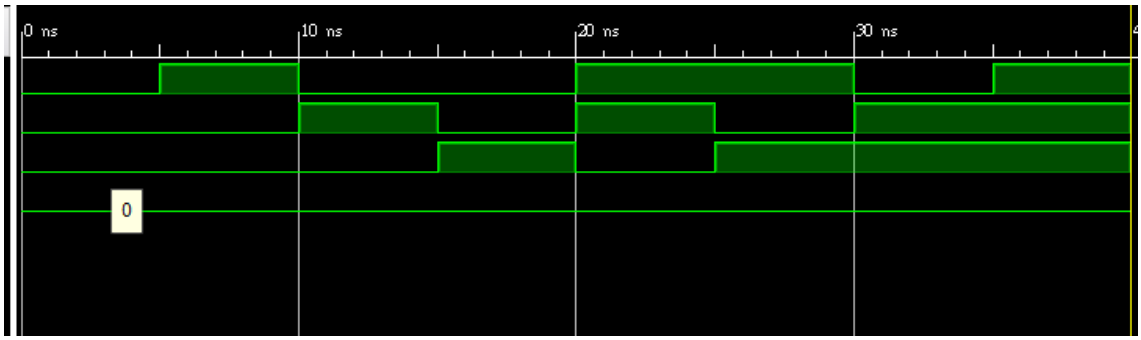
(三)　使用 XOR (^)判斷等式是否成立

已知 x ^ y = x'y + xy'
在 2 輸入的情況下，可以視為「輸入相同則輸出 0，輸入不同則輸出 1」 由此推斷等式是否成立

```
`timescale 1ns / 1ps



module exercise_2(C,x,y,z);
    output C;
    input x,y,z;
    assign C =( x | y & z) ^ (( x | y )&( x | z ));
endmodule
```



電壓圖輸出為 0，證實等式成立，但有些缺憾的是此圖未顯示 x + yz 電壓變化

三、Exercise 3:

Verify xy + xy'z + x'yz = xy + xz + yz
(using Structural level modeling)

(一) 等式左方 程式碼表示如右：

```
module exercise_3(C,x,y,z);
    output C;
    input x,y,z;
    wire n3,n4,n5;

    and(n3,x,y);
    and(n4,x,~y,z);
    and(n5,~x,y,z);
    or (C,n3,n4,n5);


endmodule
```

wire ：為 Structural level modeling 非系統輸入也非系統輸出的線路宣告。

and(n,a,b,c); ：為 and 邏輯閘的函數，n (第一個變數)為輸出，僅能存在一個。而 a、b、c 為輸入 and 邏輯閘的變數，幾個變數代表幾個輸入，且至少 2 輸入。

or(n,a,b,c); ：為 or 邏輯閘的函數，n (第一個變數)為輸出，僅能存在一個。而 a、b、c 為輸入 or 邏輯閘的變數，幾個變數代表幾個輸入，且至少 2 輸入。

~x :為 x 經過 not 運算的值。為 not 邏輯閘的簡化
EXP: not(n1,x); (其中 n1 即等同於 ~x

因此此程式碼需要 2 個 not 閘，3 個 and 閘，1 個 or 閘

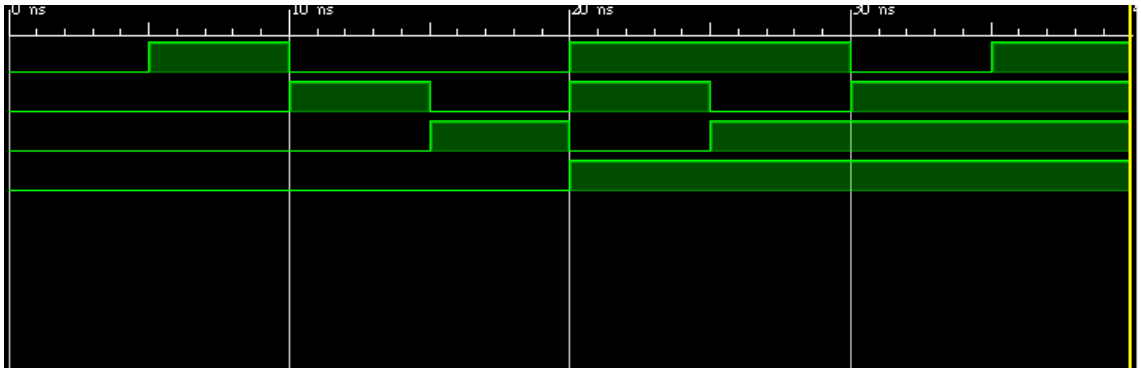
(一) 等式右方 程式碼表示如右：

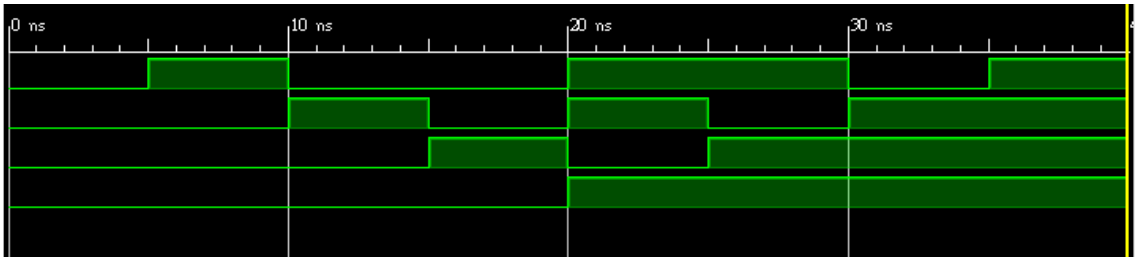
相較於 等式左方 此程式碼需要 0 個 not 閘，3 個 and 閘，1 個 or 閘。故效率更高，亦更節省成本，體積更小。

```
module exercise_3(C,x,y,z);
    output C;
    input x,y,z;
    wire n3,n4,n5;

    and(n3,x,y);
    and(n4,x,z);
    and(n5,y,z);
    or (C,n3,n4,n5);


endmodule
```





比較 2 張電壓圖，可以證實等式成立

實驗心得

1. Structural level modeling 中邏輯閘使用函數表示，而 Dataflow modeling 中邏輯閘使用算式表示
2. Structural level modeling 可以很直觀的轉換成 Schematic
3. Structural level modeling 需要宣告邏輯閘之間的線路
4. Dataflow modeling 要特別注意優先權，尤其是程式越複雜時越容易出錯
5. 所有的 modeling 都需要注意該邏輯閘是否有交換律或結合律(例如: NAND、NOR 沒有 distributivity)
6. 驗證等式是否成立時應將等式 2 邊同時輸出，比較容易對照電壓圖