積體電路電腦輔助設計概論

CAD　LAB1

Stuck-At Fault

實驗日期： 2019/03/27

學號： B063040061

姓名： 陳少洋

一、實驗內容過程及結果

1.實驗內容：

此次實驗是使用錯誤模型（Fault Model）中的一種測試方式 Stuck-At Fault 進行。錯誤模型是一種結構化的測試，他利用內部的元件像是邏輯閘去產生最適合這個函數的測試。

Stuck-At Fault 主要分成兩種狀態：stuck-at-0（SA0）以及 stuck-at-1（SA1）。一個輸入或輸出有可能為 SA0 或是 SA1，為了找出所有錯誤，我們除了可以用所有的 1 和 0 的組合來進行測試，也就是所謂的暴力法外，我們還可以用一組經過簡化的測試組合來進行測試，這也是本次實驗所要做的。

2.程式解釋：

(1) 變數宣告(Pic1)

```
bool same, end ;
int n[9], t[9], temp, y0, y1, ans[17][17], ansnum[17] = {0}, max, maxset ;
int j, sum, minset[10] = {0}, minnum = 0, alreadyChosen[17] = {0}, count[17] = {0} ;
```

▲Pic1

(2) 0 跟 1 總共有 2^4=16 種可能組合，將這 16 種組合由十進位轉成二進位。(Pic2)

```
// a0, a1, a2, a3, n1, n2, n3, y
for ( int i = 0 ; i < 16 ; i++ ) {
    // i from 0000 to 1111
    temp = i ;
    // transfer int to int array as bit using
    for ( j = 0 ; j < 4 ; j++ ) {
        n[j] = temp%2 ;
        temp /= 2 ;
    }
```

▲Pic2

(3) 呼叫 GetOutput 函式得到初始的 output y 值。(Pic 3)

Getoutput 函式傳入 wire n1,n2,n3,n4 以及要改的線，並判斷要改的是哪一條線。(Pic 4)

```
for ( int x = 0 ; x < 8 ; x++ ) { //sa0 a0 a1 a2 a3 n1 n2 n3 y
    y0 = GetOutput( n, 0 ) ;// y0=initialize normal output
```

▲Pic3

```
int GetOutput( int *n, int a ) { //n1 n2 n3 n4 + 要改哪一條線
    // SA1 or SA0 check
    if ( a != 4 )
        n[4] = !( n[2]&n[3] ) ;
    if ( a != 5 )
        n[5] = !n[1] ;
    if ( a != 6 )
        n[6] = !( n[5]|n[0] ) ;
    if ( a != 7 )
        n[7] = !( n[4]&n[6] ) ;
    return n[7] ;
}
```

▲Pic4

(4) 再建立一個測試用陣列 t,將資料複製到 t 中。將 t 陣列中要測試的線設為 1,代表測試是否為 SA1,並呼叫 GetOutput 函式,得到測試後的 ouput y 值。(Pic 5)

```
for ( j = 0 ; j < 9 ; j++ )
    t[j] = n[j] ; //copy to test
t[x] = 1 ; //測試 是不是sa1

y1 = GetOutput( t, x ) ; //改變后的y
```

▲Pic5

(5) 判斷 y 值是否一樣,不一樣則代表 SA1,將 SA1 的值轉成 10 進位存入 answer 陣列。(Pic 6)

```
if ( y0 != y1 ) {//如果不一樣代表sa1
    sum = n[3]*8+n[2]*4+n[1]*2+n[0] ;//2進位轉10進位

    // store nonrepeat int to answer buffer
        ans[x][ansnum[x]++] = sum ;
}
```

▲Pic6

(6) 重複上述動作，但這次是將 t 陣列中要測試的線設為 0,代表測試是否為 SA0,並呼叫 GetOutput 函式,得到測試後的 ouput y 值。判斷 y 值是否一樣,不一樣則代表 SA0,將 SA0 的值轉成 10 進位存入

answer 陣列。(Pic 7)

```
for ( int x = 8 ; x < 16 ; x++ ) { //sa1
    y0 = GetOutput( n, 0 ) ; //得到y值
    for ( j = 0 ; j < 9 ; j++ )
        t[j] = n[j] ;
    t[x-8] = 0 ;//測試 是不是sa0

    y1 = GetOutput( t, x-8 ) ;//改變后的y

    if ( y0 != y1 ) {//如果不一樣代表sa0
        sum = n[3]*8+n[2]*4+n[1]*2+n[0] ;//2進位轉10進位

    // store nonrepeat int to answer buffer
            ans[x][ansnum[x]++] = sum ;
    }
}
```

▲Pic7

(7) 此時,會得到所有 SA0 (Pic 8) 和 SA1 (Pic 9)的值,但是有許多重複的,所以要再進行加工。

```
SA0
8
0011 0111 1011
9
0010 0110 1010
10
1110
11
1110
12
0010 0110 1010
13
0000 0100 1000
14
0010 0110 1010
15
0000 0001 0011 0100 0101 0111 1000 1001 1011 1100 1101 1110 1111
```

```
SA1
0
0010 0110 1010
1
0000 0100 1000
2
1010
3
0110
4
1110
5
0010 0110 1010
6
0000 0001 0011 0100 0101 0111 1000 1001 1011
7
0010 0110 1010
```

▲Pic8 Pic9

(8) 如果其中一條線所測試出來的結果只有一種可能 SA0 或 SA1,則這條線一定要被選擇才能找出錯誤。因此將只有一種可能發生 SA0 或 SA1 的值存入 minimun set 中。(Pic 10)

```
// ( only a answer ) choose first
for ( int x = 0 ; x < 16 ; x++ ) {
    if ( ansnum[x] == 1 ) {
        same = false ;
        for ( int i = 0 ; i < minnum ; i++ )
            if ( minset[i] == ans[x][0] )
                same = true ;
        if ( !same )
            minset[minnum++] = ans[x][0] ;
    }
        // store into minimum set
}
```

▲Pic10

(9) 將 minimun set 的值呼叫函式 Update 做更新。(Pic 11)
Update 函式中將已經選擇過的線做記號,避免重複選擇,如果每條線皆已選擇完畢則回傳 true,否則回傳 false。(Pic 12)

```
end = Update( alreadyChosen, minset, minnum, ans, ansnum ) ;//確認沒有重複  用already chosen做記號
```

▲Pic11

```
bool Update( int *alreadyChosen, int *minset, int num, int ans[][17], int *ansnum ) {
    // 更新已選擇過的線
    bool end = true ;
    for ( int x = 0 ; x < 16 ; x++ ) {
        if ( alreadyChosen[x] == 0 ) {
            // 避免重複做記號
            for ( int i = 0 ; i < ansnum[x] ; i++ ) {
                for ( int j = 0 ; j < num ; j++ )
                    if ( minset[j] == ans[x][i] )
                        alreadyChosen[x] = 1 ;
            }
        }
    }
    for ( int x = 0 ; x < 16 ; x ++ )
    // 是否已全部線皆確認過
        if ( alreadyChosen[x] == 0 )
            end = false ;
    return end ;
```

▲Pic12

(10)　當還有沒選擇過的線時,找出重複性最高的 SA0 或 SA1 值,存入 minimun set 中。並且再次呼叫 Update 函式做記號。重複此動作直到 Update 函式回傳 true 代表所有線皆已做記號。(Pic 13)

```
while ( !end ) {
    // 每次都推一個尚未確認的線且統計數據重複最多的set進入minimun set
    // 再重新確認是否已達到目標並做記號(update)
    memset( count, 0, 16 ) ;//count歸0
    max = 0 ;
    for ( int x = 0 ; x < 16 ; x++ ) { //算哪一個組合最多 存到max
        if ( alreadyChosen[x] == 0 ) {
            for ( int i = 0 ; i < ansnum[x] ; i++ ) {
                count[ans[x][i]]++ ;
                if ( count[ans[x][i]] >= max ) {
                    max = count[ans[x][i]] ;
                    maxset = ans[x][i] ;
                }
            }
        }
    }
    minset[minnum++] = maxset ; //max推入minset
    end = Update( alreadyChosen, minset, minnum, ans, ansnum ) ;//確認沒有重複  用already chosen做記號
}
```

▲Pic13

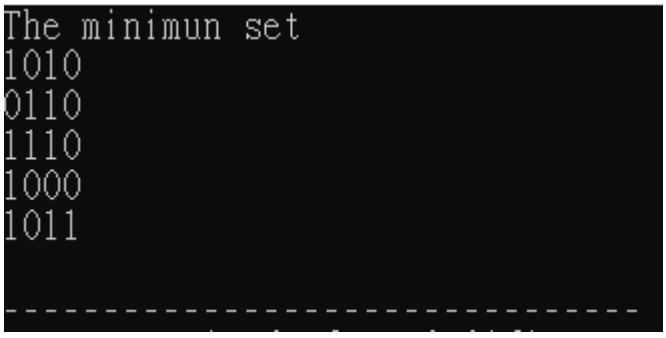
（11）將 minimum set 中的值轉為二進位,並且印出來。(Pic 14)

```
cout << "The minimun set" << endl ;
for ( int x = 0 ; x < minnum ; x++ ) { //印出minimum set
    // output
    temp = minset[x] ;
    for ( j = 0 ; j < 4 ; j++ ) { //轉2進位
        n[j] = temp%2 ;
        temp /= 2 ;
    }
    cout << n[3] << n[2] << n[1] << n[0] << endl ;
}
```

▲Pic14

3.輸出結果：

透過 Stuck-at fault 的方式,我們最少能用 5 種值找出所有的錯誤。(Pic 15)



▲Pic15

附註：全部組合(5)

1.0000

2.0110

3.1110

4.0000 or 0100 or 1000

5.0011 or 0111 or 1011

二、實驗心得

這是第一次實做積體電路課程的實驗，透過老師上課講解的 Stuck-at Fault 的概念，親自實作出如何透過程式運算來達成最簡單的驗證電路結果之正確性與否，以最少的組數來驗證全部可能出錯的地方。雖然這次的驗證電路比較小，但對於第一次接觸的我來說就足以讓我花時間思考該如何達成這樣看似簡單的目標，那更大的電路一定會更複雜，也讓我更能體會老師會說大部分這類型的問題都是 NP complete 的問題。希望透過這堂課程與實作，讓我更能摸索硬體相關的領域，提升自己的基本能力。

三、附圖：手動模擬