Test Pattern Generation

Apply the smallest sequence of test vectors to prove each node is not stuck

實驗日期：2024.03.18

學號姓名：B092040016 陳昱逢

實驗內容及過程

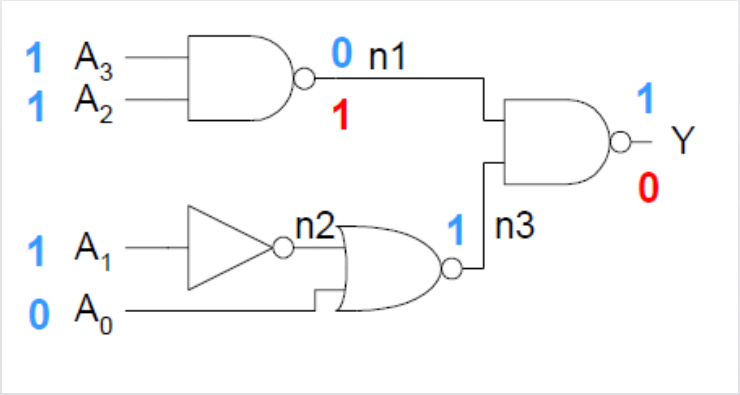
- **主要內容：**根據實驗給定的電路圖，去找出最少的測試向量集合能夠測試出所有節點是否有 Stuck-At Fault 的情況發生

- **主要過程：**Loop 每個電路中的所有節點，去測試有哪些輸入向量會發生 Stuck-At 1 (SA1) 以及 Stuck-At 0 (SA0)，針對每個 Stuck-At Fault 情況，找出所有可以測試出此錯誤情況的輸入向量，最後再找出最小交集，也就是找出最少的測試向量集合。

實驗過程

首先實現實驗給定的電路圖，我會有另外一個函式專門來實現 Stuck-At 情況的電路，該函式會額外接收兩個參數：分別為**哪個節點發生 Stuck-At 的情況**以及是 **Stuck-At 1 還是 Stuck-At 0 的情況發生**

電路圖：



電路圖的程式碼實現：

```
// Function to simulate the logic gate behavior
int simulate_gate(const string& gate, int a, int b = 0) {
    if (gate == "NAND") return !(a && b);
    if (gate == "NOR") return !(a || b);
    if (gate == "NOT") return !a;
}

// Function to simulate the entire circuit with a given input vector
int simulate_circuit(const vector<int>& input) { //vector<int>: 0110 (A3,A2,A1,A0)
    // Simulate each gate's behavior based on the input
    int n1 = simulate_gate("NAND", input[0], input[1]);
    int n2 = simulate_gate("NOT", input[2]);
    int n3 = simulate_gate("NOR", n2, input[3]);
    int y = simulate_gate("NAND", n1, n3);

    return y;
}
```

實驗過程

接下來我會去 Loop 每個電路中的節點，針對每個節點去記錄所有的測試向量能夠測出該節點發生 SA1 以及 SA0 的情況，記錄在一個二維陣列中，也就是完成填表。左圖是要填入的表格，右圖是我填入表格的結果，以二維陣列存放，並後續利用此二維陣列找出最少的測試向量集。

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S1	A3 SA1							1									
S2	A3 SA0															1	
S3	A2 SA1											1					
S4	A2 SA0															1	
S5	A1 SA1					1											
S6	A1 SA0							1									
S7	A0 SA1							1									
S8	A0 SA0								1								
S9	n1 SA1															1	
S10	n1 SA0							1									
S11	n2 SA1							1									
S12	n2 SA0					1											
S13	n3 SA1						1										
S14	n3 SA0							1									
S15	Y SA1							1									
S16	Y SA0															1	

```
0 0 0 0 0 0 1 0 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 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0
0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0
0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0
0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0
0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0
1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0
1 1 0 1 1 1 0 1 1 1 0 1 0 0 0 0
0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0
0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0
1 1 0 1 1 1 0 1 1 1 0 1 1 1 1 1
```

實驗過程

最後利用前一步產生的二維陣列去找出最少的測試向量集，我的主要想法利用遞回的方式去找，對該陣列的每一行去做加總後，會得到尚未解決 Stuck-At Fault 情況使用該測試向量的次數，每次選擇最大的，選擇該向量後，對該向量能夠解決的 Stuck-At Fault 情況中，那情況的橫排就必須全部變為 0，接著重複上述步驟，直到此陣列的值全部為 0，如此一來，我們就能得到最少的測試向量集合。程式碼如圖：

```
void FindMinimumSet(int **arr, vector<int>& miniset){
    //sum column
    int sum_column[16] = {0};
    for (int j = 0;j<16;j++){
        for(int i = 0;i<16;i++){
            sum_column[j] += arr[i][j];
        }
    }

    //get biggest element -> index save to the miniset
    int max_index = 0;
    for (int i =0;i<16;i++){
        if(sum_column[i] > sum_column[max_index])
            max_index = i;
    }
    miniset.push_back(max_index);

    //get row num vector to eliminate 1-row
    vector<int> eliminate_one_to_zero;
    for(int i=0;i<16;i++){
        if(arr[i][max_index]==1)
            eliminate_one_to_zero.push_back(i);
    }



    //start to eliminate 1-row
    for(int i=0;i<eliminate_one_to_zero.size();i++){
        int row = eliminate_one_to_zero[i];
        for(int j=0;j<16;j++){
            arr[row][j] = 0;
        }
    }

    //check if all zero
    int check_all_zero = 0;
    for(int i = 0;i<16;i++){
        for(int j =0;j<16;j++){
            if(arr[i][j]!=0)
                check_all_zero+=1;
        }
    }

    if(check_all_zero == 0){
        //cout<<"exit properly";
        return;
    }

    FindMinimumSet(arr,miniset);

}
```

實驗結果及分析

我對每個情況印出可以測試出該 Stuck-At Fault 的所有測試向量，並使用前一頁敘述的方法找出最少的測試向量集，根據結果顯示，最少的測試向量集有五個分別為：

0000 (0)
0011 (3)
0110 (6)
1010 (10)
1110 (14)

最少的測試向量集合：

```
Minimum Set: {  0000    0011    0110    1010    1110    }
```

可以測出該 Stuck-At 情況的所有向量：

```
A3 SA1:
       0110
A3 SA0:
       1110
A2 SA1:
       1010
A2 SA0:
       1110
A1 SA1:
       0000    0100    1000
A1 SA0:
       0010    0110    1010
A0 SA1:
       0010    0110    1010
A0 SA0:
       0011    0111    1011
n1 SA1:
       1110
n1 SA0:
       0010    0110    1010
n2 SA1:
       0010    0110    1010
n2 SA0:
       0000    0100    1000
n3 SA1:
       0000    0001    0011    0100    0101    0111    1000    1001    1011
n3 SA0:
       0010    0110    1010
Y SA1:
       0010    0110    1010
Y SA0:
       0000    0001    0011    0100    0101    0111    1000    1001    1011    1100    1101    1110    1111
```

實驗心得

此次實驗讓我理解了什麼是 Stuck-At Fault 問題，以及該如何解決，在從得到的二維陣列中，也花了一些時間思考該利用怎麼樣的演算法才能找出確定是最少的測試向量集，在成功實現的當下，也小有成就感。透過此次實驗不僅更熟悉了電路圖，也加深了我對 Stuck-At Fault 問題的印象。