VLSI 設計自動化導論

Final Project 書面報告

1. input 檔案

```
CIRCUIT avqlarge
INSTANCE S    PAD //Source
    N1(3,o) N2(2,o)
INSTANCE V1    ai2s
    N1(3,i) N3(1,o) N4(1,o)
INSTANCE V2    i1s
    N2(2,i) N5(1,o) N6(2,o)
INSTANCE V3    ai2s
    N3(1,i) N8(3,o) N9(2,o)
INSTANCE V4    ai2s
    N4(1,i) N5(1,i) N7(2,o)
INSTANCE V5    dsr2s
    N9(2,i) N10(2,o)
INSTANCE V6    ai2s
    N6(2,i) N13(4,o)
INSTANCE V7    i1s
    N7(2,i) N8(3,i) N11(5,o) N12(1,o)
INSTANCE V8    i1s
    N13(4,i) N14(3,o) N17(5,o)
INSTANCE V9    i1s
    N10(2,i) N11(5,i) N16(3,o)
INSTANCE V10    i1s
    N12(1,i) N14(3,i) N15(3,o)
INSTANCE D     PAD //Destination
    N15(3,i) N16(3,i) N17(5,i)
ENDCIRCUIT
```

2. 程式作法說明

資料結構 >>

```
struct Node { // 放各個節點
        string name;
        int weight;
        bool visited;
};

struct Path { // 放各個路徑
        string name;
        int weight;
        string start;
        string end;
        bool visited;
};

struct Queue { // 假Queue,其實只是陣列
        string name;
        int weight;
};

struct TmpPath { // 放需要處理的路徑
        string tmp1; // instance
        string tmp2; // node
        string name;
        int weight;
        string dir;
};
```

做法說明 >>

```
void Merge(int n1, Node * node, Queue * queue)
void Sort(int index, int n1, Queue * queue)
void Compute_Dijkstra(int n1, int n2, Node * node, Path * path, Queue * queue)
void PrintResult(int n1, Node * node)
bool isNode(string tmp, int n1, Node * node)
void getData(string tmp, string & name, string & dir, int & weight)
void DFS(int n1, int n2, Node * node, Path * path,int index)
void Traversal(int n1, int n2, Node * node, Path * path)
```

以上是我的所有 Function

請使用者輸入 input 檔名，接著開始讀檔
讀第一次的時候，先計算有幾個 instance，因為之後要宣告動態的陣列，需要知道大小
之後再讀第二次檔，把一些資料經過處理後放進我所宣告的物件裡(node,queue,path)之類的
最後呼叫 Compute_Dijkstra 和 Traversal 去做最後的運算，把結果做出來
然後再用 PrintResulf 把答案印出來

3. output 結果

```
DFS:
S -> V1 -> V3 -> V5 -> V9 -> D -> V7 -> V10 -> V4 -> V2 -> V6 -> V8

Dijkstra:
From S to V1 = 3
From S to V2 = 2
From S to V3 = 4
From S to V4 = 3
From S to V5 = 6
From S to V6 = 4
From S to V7 = 5
From S to V8 = 8
From S to V9 = 8
From S to V10 = 6
From S to D = 9
```

4. 執行方式

執行程式時，要先輸入 input 檔的名稱，之後就會印出 DFS 和 Dijkstra。

DFS:
依照各個走訪的節點輸出(節點名稱越小越先走訪)。

Dijkstra:
從 S 出發到各個節點的長度。