CS613200 Final Project

109062630 鄭傑予

Date Structure

我總共建立了兩個 `class`, 分別為 `Parser` 及 `Subgraph`，裡面包含了 c++ stl 的一些資料結構，像是 unordered_map, vector, queue 等等，以及 LEDA 的資料結構, 像是 `GRAPH, node, edge, list, node_array` 等等。

`Parser` 負責讀檔寫檔, 主要的 data structure 維護, 以及在 FlowMap 演算法中各個流程的 function.

`Subgraph` 負責在 FlowMap 的 labeling phase 中所需的 data structure 及 function. 因為在計算子圖時所需要的 data 僅為暫存用的，因此特地建立這個 `class` 來降低演算法的空間複雜度。

`Parser:`

```
class Parser {
public:
    std::string name;
    int k;
    leda::GRAPH<std::string, int> G;
    leda::list<leda::node> inputs;
    leda::list<leda::node> outputs;
    leda::list<leda::node> gate;
    std::unordered_map<leda::node, std::string> mapGroup;
    std::unordered_map<leda::node, std::string> mapZip;
    std::unordered_map<std::string, std::vector<leda::node>> tInput;
    std::unordered_map<leda::node, std::string> mapNodeName;
    std::unordered_map<std::string, std::pair<leda::node, std::string>> mapNode;

    Parser();
    void parse(char* argv[]);
    void traverseGraph(leda::GRAPH<std::string, int> &G);
    void first_stage();
    void dfsCreateNt(Subgraph &subgraph, leda::node_array<int> &visit, leda::node tIter);
    void labeling();
    void dfsMapping(leda::node_array<bool> &visit, leda::node_array<bool> &inGroup, std::queue<leda::node> &mappingQ, std::unordered_map<s
    std::vector<std::vector<bool>> enumtruthTable(int bit);
    void calLUTfunc(leda::node_array<bool> &LUTnodeValue, std::vector<leda::node> &candGroup, std::vector<std::vector<bool>> &truthTable,
    void mapping(char* argv[]);
```

`Subgraph:`

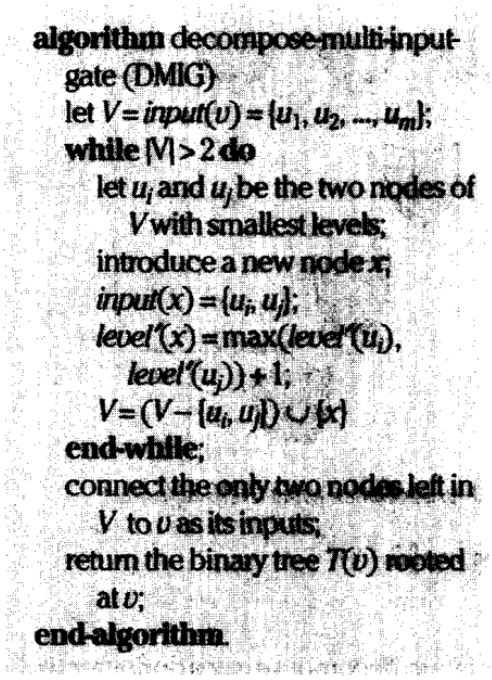
```
class Subgraph {
public:
    leda::GRAPH<std::string, int> Nt;
    std::unordered_map<std::string, leda::node> mapGtoNt;
    std::unordered_map<std::string, leda::node> mapNttoG;
    //leda::list<std::string> group;
    void dfsCollapse(leda::node_array<int> &visit, leda::node_array<int> &label, int &maxLabel, leda::node tIter);
    void nodeCollapse(std::unordered_map<leda::node, std::string> &mapZip, leda::node_array<int> &label , int &maxLabel);
    void dfsSplit(leda::node_array<int> &visit, std::unordered_map<leda::node, std::string> &mapNodeName, leda::node tIter);
    void nodeSplit();
    void dfsResidual(leda::node_array<int> &visit, std::unordered_map<leda::node, std::string> &mapGroup, leda::edge_array<int> &f
};
```

Algorithm

我參考了 FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs 這篇論文，以及跟實驗室同學互相討論，來實作這篇的演算法。

Stage1

我會根據 FlowMap 這篇論文的第三篇 reference paper, DMIG 來當作 decomposition 的演算法。



```
algorithm decompose-multi-input-
  gate (DMIG)
  let V = input(v) = {u1, u2, ..., um};
  while |V| > 2 do
    let ui and uj be the two nodes of
      V with smallest levels;
    introduce a new node x;
    input(x) = {ui, uj};
    level'(x) = max(level'(ui),
      level'(uj)) + 1;
    V = (V − {ui, uj}) ∪ {x}
  end-while;
  connect the only two nodes left in
    V to v as its inputs;
  return the binary tree T(v) rooted
    at v;
end-algorithm.
```

Figure 2. *The DMIG algorithm.*

首先從 graph 中找出 input > 2 的 gate，接著調整 edge weight 後用 shortest path 演算法找出此 gate 的 longest path，就會得到每個 input wire 的 level，將最小的兩個 wire 接到一個新建立的 gate 上後，原本的 gate 就少了一個 input，持續做到此 gate 變成 2-input gate.

Stage2

這個 stage 又分為 Labeling Phase 及 Mapping Phase.

Labeling Phase

這個階段會依照 traversal order 對每個 gate t 標示 label, label 用來決定 t 能否在 t 的子圖中跟最大的 label 的 gate 包成同一個 LUT，如果能找到 k-feasible cut 就代表可以，找不到的話 t 的 label 被標示成 maxlable+1. 因此為了判斷是否存在 k-feasible cut，會對 subgraph 進行 `dfsCollapse`, `nodeSplit` 及 `dfsResidual` 等等的 functoin，最後得到的是能跟 t 包成同一個 LUT 的 set.

`dfsCollapse` 將 t 跟子圖中 maxlabel 的 gate 合併成一個 node.

`nodeSplit` 將除了 source 跟 t 的所有 node split，建立一個分身，分身之間的 edge weight = 1，其餘的 edge weight = 999.

`dfsResidual` 會從 t 進行 dfs 走訪，找出能跟他包進同一個 LUT 的 set.

Mapping Phase

這個階段會從每個 primiry output 進行 LUT mapping，並將他的 input wire 放進 queue 中等待 mapping，接著會 enumerate 這個 LUT 的所有 function，找出 onset 之後進行寫檔，結束整個流程。

Results

k=3

	10aoi.sample01	10aoi.sample02	10aoi_alu4	10aoi_des
Level	2	2	17	10
LUTs	4	7	936	5854

k=4

	10aoi.sample01	10aoi.sample02	10aoi_alu4	10aoi_des
Level	1	2	12	7
LUTs	2	5	801	4238