CS 5160-00 FPGA Architecture & CAD Final Project

108062537 魏聖修

1. How to run my code

--How to Compile

In this directory , enter the following command:

$ make

It will generate the executable file "Map" in the same directory.

If you want to remove it please enter the following command:

$ make clean

--How to Run

Usage: ./<exe> <agg> <k> <output_name>

e.g.: ./map ../testcases/alu4.aag 3 ../output/alu4.out

```
[[zscaxd5651@ic56 src]$ make
g++ -g -std=c++11 -O3 main.cpp -o  map
[zscaxd5651@ic56 src]$ time ./map ../testcases/alu4.aag 3 ../output/alu4.out
```

2.

K = 4

	alu4	bigkey	c1908	c5315
depth	7	3	10	28
toggling	2838.91	2195.4	312.45	1723.12
runtime(sec)	5.59	0.56	8.57	10.17

前面 3 個 case 都有按照 flowmap 做出來，

但 c5315 的部分會出錯 size illegal，但是把 K 壓到 2 就會對，

我覺得是 label 的部分沒有寫好。

但沒有足夠的時間做修改。

所以只能暫時用 k = 2 來勉強得出個答案。

3. Algorithm

這次我是使用講義上的 Flowmap 演算法來實作，為了就是確保能夠達成 depth-optimal mapping。基本上程式碼分成四大部分，首先

Parse

```
struct Node
{
  int id;
  int  type;            //0 for PI,  1 for PO,  2 for Internal Nodes
  double toggling_rate;
  vector<int> fan_in;
  //PI no fan_in, PO no toggling_rate
};
```

儲存 agg 資訊的主要資料結構就是這個 node，這部分主要就是把各個 node 包含的資訊記錄下來，包含他的 node id, type 是 PI, PO, or internal node, toggling rate 以及 fan-in。並將全部 node 都存放到一個 vector 內來使用。

此外實作上有點不方便的地方是，node id 和 node 在 vector 內的 index 並不會相等，為了方便起見，我做了一個 id_to_index 的 map，access 就方便許多。

TopologicalSort

基本上就是取得一個全部 node 的 topological sort，算是為了做 labeling phase 的前置作業。因為如果要得到某一個 node 的 label 必須先知道他所有 predecessor 的 label，因此必須使用 topological order 來實作。

Labeling

算是這個演算法內最重要且困難的部分。這部分我是依照”FlowMap: an optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs” 這篇 paper 實作。

首先我們先將 PI 都 label 成 0。之後開始對每個 node 依照 topological order 去做 labeling。對於每個 node 來說，我們必須先找到此 node 的所有 predecessor 之 max-label p，然後開始決定這個 node 的 label = p or p+1。只會有 p or p+1 這兩種可能是因為這一個 node 最糟的情況就是再多用 1 個 LUT 就好了，不會再更多了。

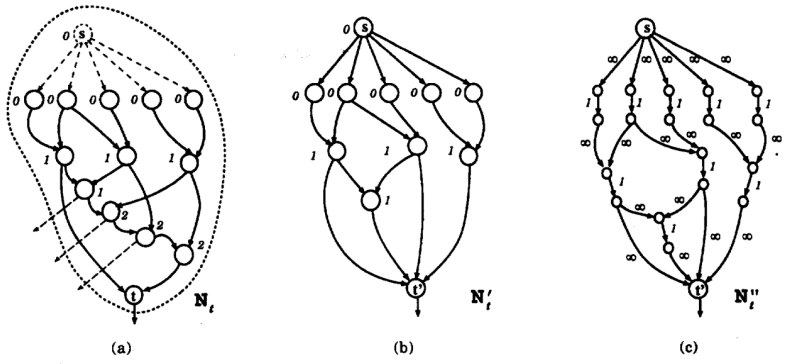

Fig. 5. Network transformations in computing a minimum height K-feasible cut in N_t ($K = 3$).

再來如何決定是 p or p+1，我們必須看看在，如上圖的 Nt 是否存在一個 K-feasible cut 在 height p – 1。如果存在 label 就是 p，否則就是 p+1。為此，我們會將 label 為 p 的 node 集合成一個新的 node t’，並獲得新的圖 Nt’。然後看看 Nt’中是否有 K-feasible cut，若存在就代表在 Nt 存在一個 K-feasible cut 在 height p – 1。

那怎麼決定 Nt’中是否有 K-feasible cut，我們先將 Nt’轉成 Nt’’，每個 node 都複製一次，然後 node 和 node 間的 capacity 是 1，其餘全部是無限大。然後依照 paper 中的 Lemma 所說

Lemma 4: *N_t' has a K-feasible cut if and only if N_t'' has a cut whose edge cut-size is no more than K.* □

因此我們必須實作 Ford-Fulkerson 演算法來找出 max-flow&min-cut。若 max-flow <= K，就代表 Nt’中有 K-feasible cut，也就代表在 Nt 存在一個 K-feasible cut 在 height p – 1。所以 label(t) = p，否則的話 label(t) = p+1。

Mapping

有了 label 這部分就比較簡單了，但是我這部分並不是按照 paper 而是依照自己的做法。先將所有的 PO 都放到一個 candidate queue 內，然後一個個 candidate 開始往 fan-in 拆解。若拆解過程中遇到的 node 之 label 和 candidate 相同的就繼續拆解，若 label 不同就把這個 node 放到 candidate queue 內(如果這個 node 是 PI 就不用放了)，並且記錄這些 node 來當作此 LUT 的 input，candidate 當作此 LUT 的 output。這樣就能得到所有的 LUT。

下圖是 sample.agg 的 mapping