資料結構與程式設計

Final Project

Fraig

姓名:張瑞騰

學號:B06901111

系級:電機二

email:b06901111@ntu.edu.tw

FB:https://www.facebook.com/ruiteng.zhang

一.前言

因為自己的HW6有幾個的bug，在處理fanin及fanout會有問題，讀檔所花費的時間也極長，因此這次project中HW6的部份是採用同學的，學號是b06602037，工海二 徐子程。

在這次final project中我只完成了strash和optimize及strash，因此主要會針對資料結構還有myhashmap以及上面提及三項指令進行說明。

二.資料結構

class cirMgr

|  |  |  |
| --- | --- | --- |
| 變數類別 | 名稱 | 意義 |
| unsigned | \_miloa[5] | 儲存目前gate的總數，PI/PO/latch/AIG的數量 |
| Idlist(typedef vector<unsigned> ) | \_input | 儲存PI的ID |
| Idlist(typedef vector<unsigned> ) | \_latch | 儲存latch的ID |
| Idlist(typedef vector<unsigned> ) | \_output | 儲存PO的ID |
| Idlist(typedef vector<unsigned> ) | \_aig | 儲存AIG的ID |
| Gatelist(typedef vector<Cirgate\*>) | \_gateList | 儲存每筆gate的資料 |
| Gatelist(typedef vector<Cirgate\*>) | \_dfsList | 儲存經過DFS處理後gate的資料 |
| vector<string> | \_comments | 儲存comment for the aag file |
| ofstream \* | \_simLog |  |

class cirGate

|  |  |  |
| --- | --- | --- |
| 變數類別 | 名稱 | 意義 |
| unsigned | \_lineNo | 儲存在aag file中被讀到時是在第幾行 |
| mutable unsigned | \_ref | 用來進行DFS |
| static unsigned | \_globalRef |
| unsigned | \_id | gate的ID |
| string | \_typeStr | gate的type |
| string \* | \_symbol | gate的symbol |
| uint8\_t | \_typeID | gate的type ID |
| CirGate \* | \_fin1 | gate的fanin1 |
| CirGate \* | \_fin2 | gate的fanin2 |
| GateList(typedef vector<Cirgate\*>) | \_outList | gate的fanout |
| bool | \_inv1 | 判斷gate的fanin1是否invert |
| bool | \_inv2 | 判斷gate的fanin2是否invert |

此外還有把每種gate繼承成class (UNDEF\_gate / PI\_gate / PO\_gate / AIG\_gate / CONST\_gate )

三.演算法

在進行以下三個指令時，由於要merge的關係，會用到兩個這次在cirGate中新增的函式

void removeFin()

判斷這個gate有無fanin，如果有則把fanin中含有這個gate的\_outlist(=fanout)刪除。

void removeFou()

判斷這個gate有無fanout，如果有則把fanout含有這個gate的fanin刪除。

接下來是三種指令的演算法說明

A. CIRSWeep

原理是刪除\_gateList中\_dfsList沒出現的AIG，因為這些gate最後不會接到PO。

首先建立兩個vector，remove\_ID和remaingate，remaingate和原本的\_gateList大小是相同 的，用來紀錄出現在\_dfsList中的AIGgate的位置，如果有則會被mark，可以和\_gateList同步 比對。

第二步是將沒被mark到的AIG的ID存到remove\_ID，同時在\_aig中刪除這些沒被mark到的 ID，這主要是用來更新CIRPrint。

最後將\_gateList與remove\_ID作比對，出現在remove\_ID中的gate會從\_gateList中刪除，同 時呼叫removeFin()與removeFou()更新gate之間的關係。

B. CIROPTimize

原理是若是有fanin == 0或是兩個fanin互相反向，把所有的fanout 接到const0;按DFS順序， 若是有一個fanin == 1或是兩個fanin一樣,把所有的fanout接到另一個fanin。

在這邊會用到建立在cirGate的兩個函式：

void mergeConstant(int tomerge, bool inv)

用來處理fanin有0或是1的情況，tomerge是0或1出現在fanin的位置(\_fin1 或 \_fin2)，inv 則是紀錄是否反向(true是0有反向，false是0沒反向)。

首先會把呼叫這個函式的gate的fanin給移除掉( 呼叫removeFin() )。

第二步是判斷這個gate是屬於fanout們中的fanin1或fanin2。

第三步是決定fanout的新fanin是否反向，這邊運用XOR的運算子來判斷。

第四步是根據tomerge 和inv判斷原本的fanin要成為fanout的fanin1或fanin2。

最後再根據tomerge和inv去決定fanin的新fanout(中間的gate已不在)。

void mergeIdentical(GateList & gate, int mode)

用來處理fanin1跟fanin2是相同的時候，gate用來傳送\_gateList以呼叫gate[0]和gate[1]，

mode則是判斷fanin是同向還反向(mode 1是fanin同時都反向 / mode 2是fanin同時都正向

/ mode 3 是fanin為一正一反)。

首先會把呼叫這個函式的gate的fanin給移除掉( 呼叫removeFin() )。

第二步是判斷mode，mode 1和 mode 2會歸在一類，mode 3則獨立出來。

第三步是判斷這個gate是屬於fanout們中的fanin1或fanin2。

第四步是決定fanout的新fanin是否反向，這邊運用XOR的運算子來判斷，

fanout的新fanin一律是原本的fanin 1，同時更新原本的fanin 1的fanout。

接著是運作optimize的過程說明

首先建立一個vector<unsigned> remove\_ID，用來儲存要移除的gate的ID。

第二步呼叫setGlobalRef()初始化DFS，因為optimize會重建\_dfsList。

第三步是判斷fanin的關係為何種狀況，進而呼叫 mergeIdentical()或mergeConstant()。

第四步是如果確定符合任一一種可以optimize的狀況，則讓isDel = true，會根據這個來判斷是 否要push\_back()\_dfsList中的gate的ID到remove\_ID。

第五步是依remove\_ID中紀錄的ID去更新\_aig，並刪除這個ID的gate。

第六步是初始化\_dfsList並重建。

C. CIRSTRash

原理是按 DFS 順序,以兩 fanin 和為 key insert hash,失敗表示可 merge。

這裡我在cirGate 新增一個函式void mergeStrash(CirGate \* gate)以及設計自己的HashMap來 create hash table < fanins, this gate >

void mergeStrash(CirGate \* gate)

這裡的gate是指\_dfsList裡的gate，在strash()中宣告一個CirGate\* gate呼叫此函式來重建 fanin和fanout。

以下是運作步驟說明

首先宣告HashMap<HashKey, CirGate \*> strashMap(getHashSize(\_dfsList.size()))來存放 fanins，以及宣告remove\_ID來紀錄需要被移除的gate的ID。

第二步是宣告HashKey key = gatekey(\_dfsList[i])， 以兩 fanin 和為 key，還有CirGate\* gate。

第三步是strashMap.check(key, gate)，把上一步得到的key去做check，如果成功了代表key 已經在map裡面，這時要呼叫 mergeStrash()還有用\_dfsList去呼叫removeFin() & removeFou()，並紀錄要移除 的gate的ID到remove\_ID。。

第四步是更新\_aig及根據remove\_ID移除掉在\_gateList中的gate。

第五步是重置DFS並重建。

D. myHashMap.h

class HashKey

這部份是用來建立key用的，會搭配在cirFraig.cpp中宣告的HashKey key = gatekey(\_dfsList[i])來產生key。

class HashMap

這部份是要建出hash table的形式，基本架構沿用HW7的myHashSet.h

1. 資料結構

typedef pair<HashKey, HashData> HashNode

size\_t \_numBuckets

vector<HashNode>\* \_buckets

2.實做

iterator begin() -> Point to the first valid data

iterator end() -> Pass the end

bool empty() 判斷begin()是否等於end()

size\_t size() 設一個變數s去加每一個\_buckets[i]的size，即可得到全部的數量

bool check(const HashKey& k, HashData& d) const

確認是否d有在hash table裡面，如果有則回傳true並賦予d 那格的值。

bool query(const HashKey& k, HashData& d) const

原理同check

bool update(const HashKey& k, HashData& d)

確認是否d有在hash table裡面，如果有則回傳true並賦予那格d的值，沒有則push\_back進 去\_buckets[pos]。

bool insert(const HashKey& k, const HashData& d)

會先檢查k是否有在hash table裡，有的話回傳false，沒有則push\_back(HashNode(k, d))到

\_buckets[pos]裡並回傳true。

bool remove(const HashKey& d)

會呼叫check(d)來檢查是否存在d，有的話則移除並回傳true，沒找到則回傳false。

class Iterator:

1.資料結構：

vector<HashNode>\* \_bucket

HashNode\* \_node

size\_t \_id, \_pos, \_numbkt

2.實做

在做 ++ / -- 時基本概念相同，會先檢查\_pos在\_bucket[\_id][ pos ]的位置，如果在界線內就是往 \_bucket[\_id ][ pos + 1] / \_bucket[\_id ][ pos - 1]的方向移動，如果已經在界線上則\_id ++/\_id - -

\_pos則在0或是尾端的位置。

四.實驗比較

sim09.aag

|  |  |  |
| --- | --- | --- |
| 我的 | optimize | strash |
| 時間 (s) | 0.17 | 0.07 |
| 記憶體 (M) | 2.148 | 2.477 |
| Ref |  |  |
| 時間 (s) | 0.01 | 0 |
| 記憶體 (M) | 1.242 | 1.82 |

sim10.aag

|  |  |  |
| --- | --- | --- |
| 我的 | optimize | strash |
| 時間 (s) | 0.02 | 0.01 |
| 記憶體 (M) | 1.582 | 1.582 |
| Ref |  |  |
| 時間 (s) | 0 | 0 |
| 記憶體 (M) | 1.82 | 1.82 |

sim13.aag

|  |  |  |  |
| --- | --- | --- | --- |
| 我的 | sweep | optimize | strash |
| 時間 (s) | 107.2 | 0.02 | 0.05 |
| 記憶體 (M) | 18.8 | 18.8 | 22.1 |
| Ref |  |  |  |
| 時間 (s) | 0.08 | 0.02 | 0.04 |
| 記憶體 (M) | 14.58 | 15.55 | 18.39 |

五.修課心得

其實原本這學期沒有打算修這門課的，考慮到loading的關係想要大三再修，不過因為有學長推薦可以大二修，修完程式會變很強，一個推坑下就來修這門課了。

經過這學期，我覺得來修這門課是正確的選擇，雖然得要面對每兩個禮拜就出一次的作業，這學期也因此有好幾天為了趕作業沒睡覺，儘管早已有心裡準備，但還是比想像中的負擔還要大，卻也因為這種高強度的訓練下，自己真的在程式方面有所進步，正如教授剛開始說的，修這門課得有自己上網尋找方法的能力，而這也造就我學會了許多之前沒學過的用法，很多時候僅限於投影片是沒辦法完成作業的，並不是說上課沒教很多，相反的這門課是我這學期修的課之中收穫最多的，教授教的真的很棒，只能說這是一堂cp值很高的課。

修完資結後不管是在讀code方面還是各種STL的運用，又或是Dlist / ADT / BST / Hash&Heap ，尤其是BST更讓我聯想到平常所使用的資料夾的路徑分類也是使用Tree，修完這堂課讓我對這些平常所使用的工具背後的運作原理有更深一步的了解。

很高興這學期有修這堂課，我覺得非得說要走CS的人才要修，這門課不僅是教資料結構與程式設計，同時也是在訓練自己的邏輯思考能力，雖然短期之類應該不會想再看到code，不過至少自己不會害怕寫code了，總之真的很推這門課，最後也要感謝教授這學期的指教！