Python程式設計-期末專案

成績座號：2

學號：407415560

姓名：劉孫和

Part 1 系統描述

（1）系統架構圖

出於python的解釋執行，動態語言的特點，導致python不可避免的再運行速度上遠遠落後于C，C++，JAVA等語言（後文會詳細提及）。

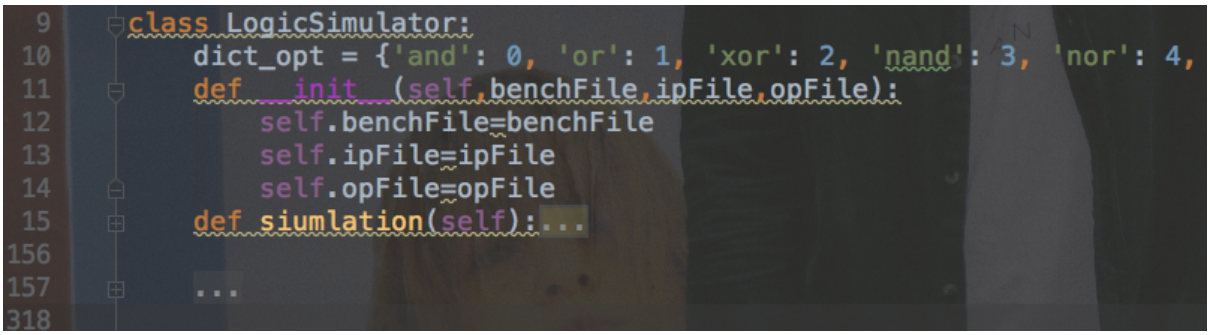
為了盡可能的提高運行速度，本工程減少函数的调用层次。以犧牲了代碼的可讀性換取了執行速度。每一层函数调用都会带来不小的开销，特别对于调用频率高，但单次消耗较小的calltree，對於本工程而言，就是AND,OR,XOR,NAND,NOR,XNOR,BUF,NOT,這一組邏輯運算函數，以及fillInput，gatheroutput函數，多层的函数调用开销就很大，这个时候可以考虑将其展开。

結構：

主類: LogicSimulator

類的主要操作函數： siumulation

將 parseBenchFile，fillInput, gatherOutput, doSim函數都直接在其中展開，而不是單獨寫出成為函數，減少函數的呼叫。在測試過程中發現大致可以提高20%的運行速度。



LogicSiumlation 的兩個主要函數

siumlation 函數的主要操作部分，由注釋來區分開來

（2）資料結構設計

原始的java的數據結構具有較好的可讀性，但是使用string來比對，必然會導致運行時間緩慢。本工程修稿了數據類型，具體如下：

gateValue = []

使用數組來保存每一個**gate**的值，**index**通過**gateDict**（**dictionary**類型）來找到所對應的**gate**名字

gateDict = {}

字典類型，（**gateName , indexOFgateValue**）**dict**採用**haspmap**來實現，有很好的查找效率，在**parseBenchFile**函數中將所有的**string**類型的**gateName**全部替換為**int**型的**index**，提高尋找速度

gatelist = []

保存邏輯門之間的運算式，內容形式為（**IndexOFOutPutGate，optType，IndexOFInputGate1[,IndexOFInputGate2,IndexOFInputGate3,...]**）

outPutStartIndex = 0 middleGateStartIndex = 0

由於所有的**gateValue**都保存在同一個數組，於是需要添加兩個變量，來分別出**OutPutGate**和邏輯中間的運算門

（3）演算法設計

3.1 doSim函數

將邏輯門的操作類型改為int類型，由下所定義

```
dict_opt = {'and': 0, 'or': 1, 'xor': 2, 'nand': 3, 'nor': 4, 'xnor': 5, 'buf': 6, 'not': 7}
```

算法結構為：

```
if (optType == 4):
    # NOR
elif (optType < 4):
    if (optType > 1):
        if (optType == 2):
            # XOR
        else:  # 3
    elif (optType == 1):
        # OR
    else:
        # AND
elif (optType > 4):
    if (optType > 6):
   elif (optType == 6):
    else:
else:
    raise Exception("Unknown Gate OPT")
```

在沒有更好的選擇調用函數下，通過觀察bench文件發現，不同的操作的調用頻率相差較大，我們採用類似于平衡二叉樹 （balanced binary Tree）的結構來編號，改變if-elif-else的結構，
來減少平均判斷時間

3.2 邏輯門運算函數（以AND為例）

```
# AND
    tot = gateValue[gateInfo[2]]
    for item in gateInfo[3:]:
        tot &= gateValue[item]
    v = tot
```

採用位元運算 & | ^ 的速度會比 加法運算 或者是 == 運算快速更多

（4）差異

4.1 數據結構組織

BasicSim.java 主要採用hashmap 用string類型的gateName 來查找 value ，但是每一次的運算查找在python的解釋編譯下會造成驚人的開銷，所以採用線性的list類型的gateValue，（折中插入數值和隨機讀數故選擇 list），使得 doSim 函數在 index 查找 value的方式來提高效率。

4.2 基本邏輯門運算

採用了 & ^ | 為基礎的運算，以提高速度。

在測試中發現 如果使用+ 為基礎的操作，在運行速度上沒有和 & ^ | 相差多少，原因在於 操作類型是 int 而不是真正的bit 而高級編程語言沒有涉及到bit的數據類型，所以似乎在這裡也沒有太好的辦法。

Part 2 程式碼

```
import time
# import gc


class MismatcheError(RuntimeError):
    def __init__(self, arg):
        self.args = arg
class LogicSimulator:
    dict_opt = {'and': 0, 'or': 1, 'xor': 2, 'nand': 3, 'nor': 4, 'xnor': 5, 'buf': 6, 'not':
7}
    def __init__(self,benchFile,ipFile,opFile):
        self.benchFile=benchFile
        self.ipFile=ipFile
        self.opFile=opFile
    def siumlation(self):
        gateValue = []
        gateDict = {}
        outPutStartIndex = 0
        middleGateStartIndex = 0
        gatelist = []
        # try:
        #將bench逻辑文件转化
        with open(self.benchFile, 'r') as f:
            count = 0
            meetMiddleFlag = False
            meetOutputFlag = False

            for aline in f:
                aline = aline.replace('\n', '')

                if (aline.startswith("#") or aline == None or len(aline) == 0):
                    continue
                elif aline.startswith("INPUT"):
                    tt = aline.split('(')
                    gName = str(tt[1].replace(')', ''))
                    gateDict[gName] = count
                    gateValue.append(None)
                    count += 1


                elif aline.startswith("OUTPUT"):
                    if meetOutputFlag == False:
                        meetOutputFlag = True
```

```
                    outPutStartIndex = count

                tt = aline.split('(')
                gName = str(tt[1].replace(')', ''))

                gateDict[gName] = count
                gateValue.append(None)
                count += 1
            else:
                if meetMiddleFlag == False:
                    meetMiddleFlag = True
                    middleGateStartIndex = count
                aline = aline.replace(' ', '')
                aline = aline.replace('=', ',')
                aline = aline.replace('(', ',')
                aline = aline.replace(')', '')
                tt = aline.split(',')
                if (str(tt[0]) not in gateDict):
                    gateDict[str(tt[0])] = count
                    gateValue.append(None)
                    tt[0] = count
                    count += 1
                else:
                    tt[0] = gateDict[str(tt[0])]
                tt[1] = LogicSimulator.dict_opt[tt[1]]
                for i in range(2, len(tt)):
                    tt[i] = int(gateDict[str(tt[i])])
                gatelist.append(tt)
        # except IOError:
        #     print ("Error: 没有找到文件或读取文件失败")
        # else:
        # try:
        with open (self.ipFile,'r') as ip:
            with open(self.opFile, 'w') as fw:
                for ipvs in ip:
                    ipvs=ipvs.replace('\n','')
                    #fillInput
                    count = 0
                    for i in ipvs:
                        gateValue[count] = int(i)
                        count += 1
                    #doSim
                    for gateInfo in gatelist:
                        outGateIndex = gateInfo[0]
                        optType = gateInfo[1]
                        # optType=optType.lower()
                        if (optType == 4):
```

```
                        # NOR
                        tot = gateValue[gateInfo[2]]
                        for item in gateInfo[3:]:
                            tot |= gateValue[item]
                        v = ~tot
                    elif (optType < 4):
                        if (optType > 1):
                            if (optType == 2):
                                # XOR
                                v = gateValue[gateInfo[2]] ^ gateValue[gateInfo[3]]
                            else:  # 3
                                # NAND
                                tot = gateValue[gateInfo[2]]
                                for item in gateInfo[3:]:
                                    tot &= gateValue[item]
                                v = ~tot
                        elif (optType == 1):
                            # OR
                            tot = gateValue[gateInfo[2]]
                            for item in gateInfo[3:]:
                                tot |= gateValue[item]
                            v = tot
                        else:
                            # AND
                            tot = gateValue[gateInfo[2]]
                            for item in gateInfo[3:]:
                                tot &= gateValue[item]
                            v = tot
                    elif (optType > 4):
                        if (optType > 6):
                            v = ~ gateValue[gateInfo[2]]
                        elif (optType == 6):
                            v = gateValue[gateInfo[2]]
                        else:
                            tot = gateValue[gateInfo[2]]
                            for item in gateInfo[3:]:
                                tot |= gateValue[item]
                            v = ~tot
                    else:
                        raise Exception("Unknown Gate OPT")
                    gateValue[outGateIndex] = v
                #gatherOutput
                opvs = ''
                for item in gateValue[outPutStartIndex:middleGateStartIndex]:
                    opvs+=str(item)
                fw.write(ipvs + ' ' + opvs + '\n')
        # except MismatcheError:
```

```
        #     print(MismatcheError,"Input Size Mismatch")
        # except Exception:
        #     print(Exception,"Unknown Gate OPT")
        # except IOError:
        #     print ("Error:InputFile 没有找到文件或读取文件失败")
```

```
# gc.disable()
start = time.process_time()
benchFile = "/Users/liusunhe/testfile/bench/c2670.bench.txt"
ipFile = "/Users/liusunhe/testfile/input/c2670_10k_ip.txt"
opFile='/Users/liusunhe/testfile/outputPython/c2670_10k_op.txt'
test=LogicSimulator(benchFile,ipFile,opFile)
test.siumlation()
end = time.process_time()
print('Running time: %s Seconds' % (end - start))
# gc.enable()
```

Part 3 系統測試表

1.[基本測試]: 以自行撰寫版本進行基本測試

-使用c432.bench+c432_1k_ip.txt，結果不正確者，不進行後續測試。

正確　　不正確

2.電腦效能基本測試

1. 電腦配置

CPU	RAM	**OS**	**HDD**
1.6 GHz Intel Core i5	8 GB 1600 MHz DDR3	macOS Mojave 10.14	128GB SSD

2. BasicSim（java）執行時間

	C432	**C7552**
1M	18.762 sec(s)	457.833 sec(s)
5M	90.612 sec(s)	
10M	170.381 sec(s)	

3. 測試各電路執行時間時(打V處)，需先確定結果正確，否則沒有意義。

	C432	**C2670**	**C7552**
1K/Seconds	0.234892		4.09279
10K/Seconds	2.30749800	24.454648	40.230065
1M/Seconds	229.633872		4027.020112
5M/Seconds	1560.341894		
10M/Seconds	2244.288002		

4.通过混乱处理的模拟化电路　　未能通过