

資料結構與程式設計

Final Project Report (FRAIG)

姓名：吳岳

學號：b02901093

Email：b02901093@ntu.edu.tw

手機：0972871931

一、功能設計

● Sweep

1. Key Element & Function:

- cirGate.h

```
bool    _sw;  
void    setSw(bool s)  
bool    isSwList()
```

2. 描述

所有的gate在read_circuit時都會存一個_sw的標記，並預設成true，之後判斷該gate屬不屬於DFS List。若屬於DFS List，_sw = false，不屬於DFS List gateType !

= PI，將gate搜集入預設的gatelist。

(Sweep Gate List)

首先先將gate對應fanin跟它連線刪掉，重新整理

GateList的排序，最後再將_sw = true 的 gate刪除。

● Optimization

1. Key Element & Function :

- cirOpt.cpp

```
faninJudge(CirGate* g, GateList& opt)  
disConnect(CirGate* g, unsigned& i)  
setNewConnect(CirGate* g, CirGate* gi, size_t& phase)
```

- cirGate.h

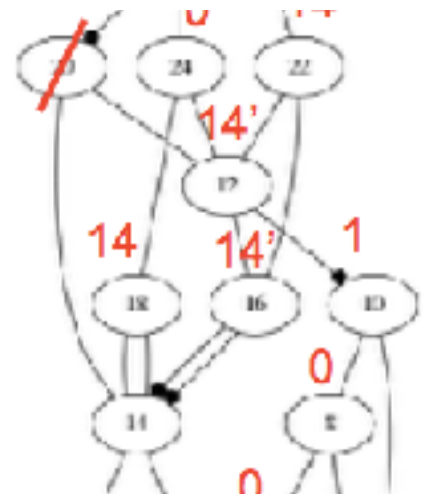
```
bool    _optDel;  
void    setOptDel(bool s)  
bool    isOptDel()
```

2. 描述：

loop整個DFS List的gate，在**faninJudge()**裡，每個gate都會判斷4種trivial optimization的條件。若符合，**disconnect()**將連線拆掉。

再者，gate跟即將取代它的gate (ex: fanin, Const0)、以及phase都會被送到**setNewConnect()**，將gate的fanout跟要取代的fanin相連。

以老師簡報裡的例子說明，gate 18會被14取代，而我先將14連到18的fanout刪掉取代成24 (14 -> 24)，再將24連到18的fanin改成14 (24 -> 14)。這樣做可以讓trivial optimization 一直沿著DFS List 做下去。



最後重新整理 GateList的排序，將_optDel = true 的 gate刪除。

● Structural Hash

1. Key Element & Function :

- myHashMap.cpp

```
template <class HashKey, class HashData>
class HashKey, HashData, HashMap
vector<HashNode>*    _buckets;
```

- cirGate.h

```
bool    _strHhDel
void    setStHDel(bool s)
bool    isStHDel()
```

2. 描述 :

Hashkey裡面存兩個對應的fanin，**HashData**則是以size_t的資料型態包含phase存對應gate，"**HashNode**"是HashKey跟HashData的pair。

loop整個DFS List的gate，每個gate的fanin都會跟存在HashMap裡的HashNode比對，若兩個fanin不一樣，建構新的HashNode將fanin 跟 gate + phase存起來。若兩個fanin都一樣，就會將gate的 _strHhDel設為true。參照optimization後段同樣的方法先**disconnect()** 再 **setNewConnection()**

● Simulation

1. Key Element & Function :

- **cirGate.h**

```
void      getValue()  
bool      getPreVal()  
void      setValue(bool v)  
void      setValToPre()  
bool      _value;  
bool      _preValue;  
pair<unsigned, unsigned> _bukSI;
```

- **cirSim.cpp**

```
piValSet(CirGate* g, char& pi)  
defValue(CirGate* g, int& it)  
setHash(CirGate* g, int& it, int& col)
```

- **cirDef.h**

```
class FECGroup
```

2. 描述 :

每次新pattern進來，都會用**piValSet()**將pi初始化，**defValue()**設定DFS List裡gate的 value。

每個get都會有**value & prevalue**，原理跟flip-flop很像，跑pattern時，會將上個state的值存到prevalue，新的值存成value。

之所以需要preValue跟Value，是因為要判斷**FEC & iFEC group**，第一次跑pattern時，所有的gate都會存在hash的第一個bucket裡面。



從第二次pattern開始，每個bucket的第一個slot是參考gate，其他的bucket都要跟它比較preValue & Value。



如果 $B.value == A.value \ \&\& \ B.Prevalue == A.Prevalue$ 合理猜測B可能跟A是FEC pair，相反的如果 $B.value != A.value \ \&\& \ B.Prevalue != A.Prevalue$ ，那他很有可能彼此是iFEC pair。



如果其中一個相等而另外一個不等，那它將不屬於A的FEC group。將拆夥產生新的bucket，同樣的，第一個slot的gate是參考點，之後進來的gate在每次有新pattern時都會跟它比較。

最後print FEC pair 得到的結果跟Reference Code一樣，
(但ID的排列並沒有按照順序, 如下圖操作sim09.aag範
例，得到的總FEC pair數目一樣、pair的成員也相同)

```

[1790] 2078 2074
[1791] 86 354
[1800] 37 124 131 1125 140 140 141 1152 163 189 185 1154 172 172 1171 1175
[1801] 705 12316 1015 1190 1209 1201 1041 1013 1312 1022 1314 11005 1051 1053 1060 1069 1082 1012 101
[1802]
[1803] 1012 2018 1930 11017 1007 1014 12015 13015 1071 1070 1940 11017
[1804] 1004 1402
[1805] 1008 1002 1003 13004 13005 1005 13012 13014 1005
[1806] 1126 1101
[1807] 1126 1101 1121 11012
[1808] 1127 1101 1101 1101
[1809] 1127 1101 1101 1101
[1810] 1127 1101 1101 1101
[1811] 1127 1101 1101 1101
[1812] 1127 1101 1101 1101
[1813] 1127 1101 1101 1101
[1814] 1127 1101 1101 1101
[1815] 1127 1101 1101 1101
[1816] 1127 1101 1101 1101
[1817] 1127 1101 1101 1101
[1818] 1127 1101 1101 1101
[1819] 1127 1101 1101 1101
[1820] 1127 1101 1101 1101
[1821] 1127 1101 1101 1101
[1822] 1127 1101 1101 1101
[1823] 1127 1101 1101 1101
[1824] 1127 1101 1101 1101
[1825] 1127 1101 1101 1101
[1826] 1127 1101 1101 1101
[1827] 1127 1101 1101 1101
[1828] 1127 1101 1101 1101
[1829] 1127 1101 1101 1101
[1830] 1127 1101 1101 1101
[1831] 1127 1101 1101 1101
[1832] 1127 1101 1101 1101
[1833] 1127 1101 1101 1101
[1834] 1127 1101 1101 1101
[1835] 1127 1101 1101 1101
[1836] 1127 1101 1101 1101
[1837] 1127 1101 1101 1101
[1838] 1127 1101 1101 1101
[1839] 1127 1101 1101 1101
[1840] 1127 1101 1101 1101
[1841] 1127 1101 1101 1101
[1842] 1127 1101 1101 1101
[1843] 1127 1101 1101 1101
[1844] 1127 1101 1101 1101
[1845] 1127 1101 1101 1101
[1846] 1127 1101 1101 1101
[1847] 1127 1101 1101 1101
[1848] 1127 1101 1101 1101
[1849] 1127 1101 1101 1101
[1850] 1127 1101 1101 1101
[1851] 1127 1101 1101 1101
[1852] 1127 1101 1101 1101
[1853] 1127 1101 1101 1101
[1854] 1127 1101 1101 1101
[1855] 1127 1101 1101 1101
[1856] 1127 1101 1101 1101
[1857] 1127 1101 1101 1101
[1858] 1127 1101 1101 1101
[1859] 1127 1101 1101 1101
[1860] 1127 1101 1101 1101
[1861] 1127 1101 1101 1101
[1862] 1127 1101 1101 1101
[1863] 1127 1101 1101 1101
[1864] 1127 1101 1101 1101
[1865] 1127 1101 1101 1101
[1866] 1127 1101 1101 1101
[1867] 1127 1101 1101 1101
[1868] 1127 1101 1101 1101
[1869] 1127 1101 1101 1101
[1870] 1127 1101 1101 1101
[1871] 1127 1101 1101 1101
[1872] 1127 1101 1101 1101
[1873] 1127 1101 1101 1101
[1874] 1127 1101 1101 1101
[1875] 1127 1101 1101 1101
[1876] 1127 1101 1101 1101
[1877] 1127 1101 1101 1101
[1878] 1127 1101 1101 1101
[1879] 1127 1101 1101 1101
[1880] 1127 1101 1101 1101
[1881] 1127 1101 1101 1101
[1882] 1127 1101 1101 1101
[1883] 1127 1101 1101 1101
[1884] 1127 1101 1101 1101
[1885] 1127 1101 1101 1101
[1886] 1127 1101 1101 1101
[1887] 1127 1101 1101 1101
[1888] 1127 1101 1101 1101
[1889] 1127 1101 1101 1101
[1890] 1127 1101 1101 1101
[1891] 1127 1101 1101 1101
[1892] 1127 1101 1101 1101
[1893] 1127 1101 1101 1101
[1894] 1127 1101 1101 1101
[1895] 1127 1101 1101 1101
[1896] 1127 1101 1101 1101
[1897] 1127 1101 1101 1101
[1898] 1127 1101 1101 1101
[1899] 1127 1101 1101 1101
[1900] 1127 1101 1101 1101
[1901] 1127 1101 1101 1101
[1902] 1127 1101 1101 1101
[1903] 1127 1101 1101 1101
[1904] 1127 1101 1101 1101
[1905] 1127 1101 1101 1101
[1906] 1127 1101 1101 1101
[1907] 1127 1101 1101 1101
[1908] 1127 1101 1101 1101
[1909] 1127 1101 1101 1101
[1910] 1127 1101 1101 1101
[1911] 1127 1101 1101 1101
[1912] 1127 1101 1101 1101
[1913] 1127 1101 1101 1101
[1914] 1127 1101 1101 1101
[1915] 1127 1101 1101 1101
[1916] 1127 1101 1101 1101
[1917] 1127 1101 1101 1101
[1918] 1127 1101 1101 1101
[1919] 1127 1101 1101 1101
[1920] 1127 1101 1101 1101
[1921] 1127 1101 1101 1101
[1922] 1127 1101 1101 1101
[1923] 1127 1101 1101 1101
[1924] 1127 1101 1101 1101
[1925] 1127 1101 1101 1101
[1926] 1127 1101 1101 1101
[1927] 1127 1101 1101 1101
[1928] 1127 1101 1101 1101
[1929] 1127 1101 1101 1101
[1930] 1127 1101 1101 1101
[1931] 1127 1101 1101 1101
[1932] 1127 1101 1101 1101
[1933] 1127 1101 1101 1101
[1934] 1127 1101 1101 1101
[1935] 1127 1101 1101 1101
[1936] 1127 1101 1101 1101
[1937] 1127 1101 1101 1101
[1938] 1127 1101 1101 1101
[1939] 1127 1101 1101 1101
[1940] 1127 1101 1101 1101
[1941] 1127 1101 1101 1101
[1942] 1127 1101 1101 1101
[1943] 1127 1101 1101 1101
[1944] 1127 1101 1101 1101
[1945] 1127 1101 1101 1101
[1946] 1127 1101 1101 1101
[1947] 1127 1101 1101 1101
[1948] 1127 1101 1101 1101
[1949] 1127 1101 1101 1101
[1950] 1127 1101 1101 1101
[1951] 1127 1101 1101 1101
[1952] 1127 1101 1101 1101
[1953] 1127 1101 1101 1101
[1954] 1127 1101 1101 1101
[1955] 1127 1101 1101 1101
[1956] 1127 1101 1101 1101
[1957] 1127 1101 1101 1101
[1958] 1127 1101 1101 1101
[1959] 1127 1101 1101 1101
[1960] 1127 1101 1101 1101
[1961] 1127 1101 1101 1101
[1962] 1127 1101 1101 1101
[1963] 1127 1101 1101 1101
[1964] 1127 1101 1101 1101
[1965] 1127 1101 1101 1101
[1966] 1127 1101 1101 1101
[1967] 1127 1101 1101 1101
[1968] 1127 1101 1101 1101
[1969] 1127 1101 1101 1101
[1970] 1127 1101 1101 1101
[1971] 1127 1101 1101 1101
[1972] 1127 1101 1101 1101
[1973] 1127 1101 1101 1101
[1974] 1127 1101 1101 1101
[1975] 1127 1101 1101 1101
[1976] 1127 1101 1101 1101
[1977] 1127 1101 1101 1101
[1978] 1127 1101 1101 1101
[1979] 1127 1101 1101 1101
[1980] 1127 1101 1101 1101
[1981] 1127 1101 1101 1101
[1982] 1127 1101 1101 1101
[1983] 1127 1101 1101 1101
[1984] 1127 1101 1101 1101
[1985] 1127 1101 1101 1101
[1986] 1127 1101 1101 1101
[1987] 1127 1101 1101 1101
[1988] 1127 1101 1101 1101
[1989] 1127 1101 1101 1101
[1990] 1127 1101 1101 1101
[1991] 1127 1101 1101 1101
[1992] 1127 1101 1101 1101
[1993] 1127 1101 1101 1101
[1994] 1127 1101 1101 1101
[1995] 1127 1101 1101 1101
[1996] 1127 1101 1101 1101
[1997] 1127 1101 1101 1101
[1998] 1127 1101 1101 1101
[1999] 1127 1101 1101 1101
[2000] 1127 1101 1101 1101

```

● Fraig

1. Key Element & Function :

- cirFraig.cpp

```

fraigLoop(CirGate* g0, GateList& fecList, SatSolver& solver)
SATSetHash(CirGate* g, bool result)
genProofModel(SatSolver& s)

```

2. 描述

我們是按照Dfs search的方式去判斷FEC group的成員跟第一個成員之間有沒有SAT，因為有範例的code，在實作上並沒有發生太大的困難。但因為CONST_Gate並不在dfs list裡面，需要額外將它加到hast。

對於那些SAT的成員，我的做法是將他們都放到新的hashSet裡面。讓成員在跟第一個成員

(`simHach[i]->getGate(0)`)判斷有沒有SAT。跟老師的code比起來，我的code會慢很多，worst case狀況下，第一個成員跟其他成員都是SAT，將會有 $n!$ 的複雜度，但可以確保不有漏網之魚。

二、修課心得

大三上時就很想修姿婕了，但害怕之下卻步了。那一年修了不少的實驗&選修，每到要寫程式總有種使不上力的感覺，後來心一橫，決定在大四跟姿婕好好大戰一場。

然而，修完之後並沒有戰勝的感覺，上課教過的很多內容老師說很多都忘了，除非作業有做到，不然它就在檔案夾裡面封塵。對於自己之後會不會走軟體相關這條路，也沒有明確的方向，只覺得程式是個我不可或缺的技能，從喜歡的技能去分類自己要做的事情太膚淺了，目前也沒有找到什麼很關注的議題，感覺都很新穎但都不是志業。

多年後，我可能忘記fraig的定義，但我一定不會忘記戰勝這門課前，老師的姿婕與人參，那不是無戰事下的嘴砲，而是大學被重課烙印後的教誨，更有歷練的鹹味。