Digital System Design Project 3 – State Minimization

姓名：張睿麟

學號：B11015030

系級：四資工二 乙

先備知識：

1. Prime Implicant、Essential Prime Implicant [1]
2. Quine–McCluskey algorithm [2]
3. Petrick's method [3]

實作步驟：

1. Build implication table for state pairs.

建立二維矩陣。

1. Remove output incompatible pairs.

快篩：對每一格檢查兩個state的input/output會不會一樣，如果不一樣，就代表此兩個state不可能相容，若input/output一樣，就代表此兩個state「有可能」可以化簡。還是需要之後進一步檢查。

1. List transition (next state) pairs.

找出current state他們的next state的關係。

1. Check compatibility of transition pair.

從每一格中，查看是否不相容，不相容的條件是next state的對應關係是否剛好為已確定為不相容的state。

1. Remove incompatible transition pairs.

每次找到不相容後，要再重投檢查一遍，直至完全沒有。

1. Merge remaining compatible states.

合併可以相容的兩個狀態，保留第一個出現的，另一個為要刪除的state。

要篩除的state，必須把在表格中有出現的地方，取代成要保留的state。

1. Update implication table

在矩陣裡，也要更新，把刪掉的state所包含的row和column都標記

kiss file format：

<input> <current-state> <next-state> <output>

程式實作說明：

儲存Terms的方式是用string資料型態

1. termsExpansion()
   1. 利用遞迴把”-”分成0、1，繼續遞迴直到讀完輸入的terms(舉例：10-1 => 1001、1011)
   2. 遇到Don’t care先看成1，但是要記錄下來，因為在取Essential Prime Implicant時，不一定需要Don’t care
2. Classification()
   1. 利用for迴圈及記錄每個Terms有多少個1
   2. 用2D的vector分類並儲存這些Terms
3. Simplify()
   1. 利用遞迴，把鄰近的Terms全部比對，若兩個Terms之間只差一個bit，則把該bit替換成”-”進行化簡。
   2. 在過程中，會去紀錄可以化簡的Terms，化簡至該迴圈的最後，會去把沒辦法化簡的Terms新增至化簡好的表格，在進行化簡。
   3. 所以程式結束後，能夠確保該表格已經不能化簡，且過程中沒辦法化簡的Terms也會儲存下來。
4. createPrimeImplicant()
   1. 把化簡後的Terms，展開，為了看哪些minterms有包含
   2. 順便去紀錄所有的minterms出現過幾次
5. createEssentialPrimeImplicant()
   1. 只有出現過1次的minterms，可作為Essential Prime Implicant
6. createPetrickTable()
   1. 整理出非Essential Prime Implicant的表，稱做othersTerms
7. 剩下的程式會把otherTerms的terms給上字母a-z標記為不同的terms，利用Petrick Algorithm排列組合出所有的可能，因為是用a-z排列組合，所以可以針對每種組合的式子各自排序過後，遇到重複的字母，可以取一次就好。
8. 在針對每個式子做統整排序，遇到相同的式子可以刪除。
9. 剩下來的式子當中，長度最短的就可以成為最佳化簡的式子

測資演示：

4 Variables

|  |  |
| --- | --- |
| Input1.pla | Output1.pla |
| .i 4  .o 1  .ilb a b c d  .ob f  .p 5  00-0 1  0-11 1  10-1 -  1-00 1  1111 1  .e | .i 4  .o 1  .lib a b c d  .ob f  .p 4  00-0 1  10-1 1  --11 1  1-00 1  .e |
| Terminal | |
| Total number of terms: 4  Total number of literals: 11 | |

5 Variables

|  |  |
| --- | --- |
| Input2.pla | Output2.pla |
| .i 5  .o 1  .ilb a b c d e  .ob f  .p 10  00-00 1  01-11 -  -01-1 -  01001 1  0-110 1  01-1- 1  101-0 1  11-00 -  11001 -  110-0 1  .e | .i 5  .o 1  .lib a b c d e  .ob f  .p 6  -1001 1  -1010 1  01-1- 1  11-00 1  -01-- 1  00-00 1  .e |
| Terminal | |
| Total number of terms: 6  Total number of literals: 21 | |

6 Variables

|  |  |
| --- | --- |
| Input3.pla | Output3.pla |
| .i 6  .o 1  .ilb a b c d e g  .ob f  .p 10  10-1-0 1  0110-1 -  1-0101 1  11-1-0 1  -00-01 -  1-0111 1  -0111- -  1-011- 1  1001-- -  -01110 -  .e | .i 6  .o 1  .lib a b c d e g  .ob f  .p 5  -00-01 1  -0111- 1  0110-1 1  1--1-0 1  1-01-- 1  .e |
| Terminal | |
| Total number of terms: 5  Total number of literals: 19 | |

參考資料：

[1]: <https://en.wikipedia.org/wiki/Implicant>

[2]: <https://en.wikipedia.org/wiki/Quine%E2%80%93McCluskey_algorithm>

[3]: <https://en.wikipedia.org/wiki/Petrick%27s_method>