-디지털 논리회로-

컴퓨터정보공학부 2018202074 김상우

1. Problem statement

위 문제는 비트의 길이와 몇 개의 don’t care value, midterm을 txt파일을 통해 입력 받은 후, 퀸-맥클러스키(Quine Mccluskey)를 이용해 필수주항과 그것들을 논리식으로 나타냈을 때에 드는 최소 트랜지스터 개수를 구하고 그것들을 기록한 txt파일을 생성하는 코드를 c혹은 c++로 만드는 문제이다. 위 과정수행을 위해 Quine Mccluskey에서 사용할 주항차트 등의 자료를 생성하는 코드가 필요할 것이다. 이를 구하기 위해 아래와 같은 과정이 필요할 것이다.

1. **txt파일로부터 다음 값들을 입력받음. (we will using Example of Homework)**

**Number of bit(=4), number of value(=9), input\_midterm.txt’s texts**

1. **Column1과 Column2의 선언(Column1은 text파일을, 나머지와 Column2는 ?로 초기화)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **d/m** | **blank** | **Value[1]** | **Value[2]** | **…** | **Value[number of BIT]** |
| **1** | **D** |  | **0** | **0** |  | **0** |
| **:** |  |  |  |  |  |  |
| **Number of value** | **D** |  | **1** | **1** |  | **1** |
| **:** |  |  |  |  |  |  |
| **{(Number of Value)^2/2}+1** | **?** | **?** | **?** | **?** |  | **?** |

1. **Care(=Midterm)의 선언 후 Column1의 d/m을 이용해 Midterm을 배열 내로 가져온다.**

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **0** | **0** | **0** |
| **0** | **1** | **1** | **1** |
| **1** | **1** | **0** | **1** |
| **1** | **1** | **1** | **1** |

1. **d/m이 적힌 자리를 (num of value)번째 행까지 0으로 초기화하고, 공백이 있던 열에 value 내 1의 개수를 입력하고 그 값이 1 차이나는 value끼리 비교한다.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **Number of 1** | **Value[1]** | **Value[2]** | **Value[3]** | **Value[4]** |
| **1** | **0** | **0** | **0** | **0** | **0** | **0** |
| **2** | **0** | **1** | **0** | **1** | **0** | **0** |
| **3** | **0** | **2** | **0** | **1** | **0** | **1** |
| **4** | **0** | **2** | **0** | **1** | **1** | **0** |
| **:** |  |  |  |  |  |  |
| **Number of value** | **0** | **4** | **1** | **1** | **1** | **1** |
| **Number of value+1** | **0** | **?** | **?** | **?** | **?** | **?** |
| **:** |  |  |  |  |  |  |
| **{(Number of Value)^2/2}+1** | **0** | **?** | **?** | **?** | **?** | **?** |

**5. 상응하는 value의 비교결과 다른 value와 ‘-’ 1개를 사용해 표현이 가능하다면(Method 1이라 지칭하자.), d/m이 적혀있던 자리에 1을 추가한다.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **Number of 1** | **Value[1]** | **Value[2]** | **Value[3]** | **Value[4]** |
| **1** | **1** | **0** | **0** | **0** | **0** | **0** |
| **2** | **3** | **1** | **0** | **1** | **0** | **0** |
| **3** | **3** | **2** | **0** | **1** | **0** | **1** |
| **4** | **2** | **2** | **0** | **1** | **1** | **0** |
| **:** |  |  |  |  |  |  |
| **Number of value** | **d** | **4** | **1** | **1** | **1** | **1** |
| **Number of value+1** | **?** | **?** | **?** | **?** | **?** | **?** |
| **:** |  |  |  |  |  |  |
| **{(Number of Value)^2/2}+1** | **?** | **?** | **?** | **?** | **?** | **?** |

**Ex) 0010 , 0110 -> 0-10 // 0000, 1000->-000**

1. **위 계산 결과 d/m이 있던 row에 0이 기록되어있는 value를 Final(=Pls)로 옮긴다. 만약 1이상의 값이 있다면 value들을 Column2로 옮기고 d/m이 있던 row의 값들을 0으로 해준다.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **Number of 1** | **Value[1]** | **Value[2]** | **Value[3]** | **Value[4]** |
| **1** | **0** | **0** | **0** | **0** | **0** | **0** |
| **2** | **0** | **1** | **0** | **1** | **0** | **0** |
| **:** |  |  |  |  |  |  |
| **{(Number of Value)^2/2}+1** | **?** | **?** | **?** | **?** | **?** | **?** |

**-Final 배열**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Value[1]** | **Value[2]** | **Value[3]** | **Value[4]** |
| **1** | **1** | **0** | **1** | **0** |
| **2** | **?** | **?** | **?** | **?** |
| **:** |  |  |  |  |
| **{(Number of Value)^2/2}+1** | **?** | **?** | **?** | **?** |

1. **Column2를 Column1으로 옮기고 Column2 전체를 ?로 초기화한 후에 5-7를 반복한다.**

**Column2로 이동할 때 바뀐 것이 없다면 8번으로 이동한다.**

1. Final값을 y축으로 care값을 x축으로 하는 배열(=ese\_Pls)을 선언하고 Final과 care의 value들을 비교한다. Method1에 대입시 성립하면 ‘X’아닐시, ‘O’를 쓰는 주항차트를 생성한다. 또한 열에 존재하는 X의 값을 배열 가장 아래에 추가대입한다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **0100** | **0101** | **0110** | **1001** | **1010** |
| **1010** | **O** | **O** | **O** | **O** | **X** |
| **0-00** | **X** | **O** | **O** | **O** | **O** |
| **10-1** | **O** | **O** | **O** | **X** | **O** |
| **01--** | **X** | **X** | **X** | **O** | **O** |
| **-1-1** | **O** | **X** | **O** | **O** | **O** |
| **Number of ‘X’** | **2** | **2** | **1** | **1** | **1** |

X수가 1인 열중 X가 포함된 행의 문자열을 출력파일(result.txt)로 보내주고, 그 행의 모든 X의 값의 열의 (Number of ‘X’)값을 a로 한다. Method2를 이용, 트랜지스터 수를 구한다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **0100** | **0101** | **0110** | **1001** | **1010** |
| **1010** | **O** | **O** | **O** | **O** | **ⓧ** |
| **0-00** | **X** | **O** | **O** | **O** | **O** |
| **10-1** | **O** | **O** | **O** | **ⓧ** | **O** |
| **01--** | **X** | **X** | **ⓧ** | **O** | **O** |
| **-1-1** | **O** | **X** | **O** | **O** | **O** |
|  | **a** | **a** | **a** | **a** | **a** |

1. 만약 number of ‘X’의 행에 a로 되지 않은 값이 있다면 아래에 있는 value부터 이용해 모든 수치를 a로 만들어준다. 이 행도 출력파일(result.txt)로 보내준다. 또다시 Method2를 이용, 트랜지스터 수를 구한다.

(위 예시는 이를 적용할 수 없으므로 다른 경우를 사용했다.)

**Ex)파란선은 8번 빨간선은 9번 방법의 이용**

|  |  |  |  |
| --- | --- | --- | --- |
| **X** |  | **X** |  |
|  |  | **X** | **X** |
|  | **ⓧ** |  | **X** |
| **X** |  | **X** |  |
| **a** | **a** | **a** | **a** |

**Method2) Bit수와 –수가 같으면 트랜지스터 수(TR)를 0으로 하고 종료시킨다. 만약 ‘-’를제외한 수(=A)가 1이고 그 값이 0이면 TR에 2를 더해준다. A값이 2이상이면 (A+1)\*2를 TR에 더해준다. 이때 0의 수가 1보다 많으면 (1의수+1)\*2를 TR에 더하고 전체 ’수 (=NOT)도 1 더해준다. 0의 수가 더 많으면 2\*0의수를 TR에 더해준다. 만약 출력파일로 보낸 문자열의 수가 1이면 TR에서 4을 빼주며 NOT이 1보다 크면 (NOT-1)\*2만큼 TR에서 빼준다.**

**(위 방법은 A’+B’=(A\*B)’ A’\*B’=(A+B)’를 기반으로 한다. ‘의 개수가 3개이상일 경우 inverter(트랜지스터 2개)의 수가 현저히 줄며 input수가 같을 시 AND와 OR는 트랜지스터의 수가 (input수\*2+2)로 같으므로 트랜지스터 수를 줄일 수 있다. 이는 각각에도, 전체적으로 사용되는 AND에도 적용한다.)**

**10. TR의 값을 출력파일로 전달해준다..**

1. Algorithm with pseudo code and flow chart

**- pseudo code(using different name in code)**

Open file input\_minterm.txt

Check (VALUE=number of values) and (BIT=number of bits)

Restart file input\_minterm.txt

Make array ’Column1’({(VALUE)^2/2}+1\*(BIT+2)) and Array ’Column2’({(VALUE)^2/2}+1\*(BIT+2)),

Make Array ‘Minterm’ (VALUE\*BIT) and Array ‘PIs’ ({(VALUE)^2/2}\*(BIT+2))

(input\_minterm.txt)’s texts from second line -> Column1

(input\_minterm.txt)’s texts that have ‘m’ in first word -> Minterm

Repeat(during any Column1’text can’t change)

Compare between Column1’s texts{

they have only one different -> Combine them using ‘-‘

they can’t change -> Put them in PIs

their result -> Column2

Column2->Column1

Make Array ‘Chart’(Y-axis: PIs , X-axis: Minterm)

Compare between Y-axis’s texts and X-axis’s texts

If Y-axis’s text’s include X-axis, put X, else put O

number of array’s X -> Chart array’s last blank

last blank = 1 in some array -> deletes rows that include X ,that is included first X of column

and choose them.

last blank is not 1 -> choose Y-axis’s texts to change all last blank to one

X-axis’s texts that chosen->essential prime implicants

Open file result.txt

essential prime implicants->result.txt

TR(=Number of transistor in logic with (essential prime implicants) and AND GATEs)=0

NOT(=using entire not)=0, Num\_ese(=number of essential Pls)=0

Repeat{

If (Nth essential prime implicants’ number of ‘-‘s = BIT)

0->TR, TR->result.txt, finish code

Else If (Nth essential prime implicants’ number of ‘-‘s + 1 = BIT)

If 0 exist

+2->TR

Else If (BIT - Nth essential prime implicants’ number of ‘-‘s > 1)

+(BIT - Nth number of ‘-’s +1)\*2->TR

If(Nth number of ‘0’s - number of ‘-‘s > 0)

+(Nth number of ‘1’s + 1)\*2->TR, +1->NOT

Else

+(Nth number of ‘0’s)\*2->TR

IF(Num\_ese=1)

-4->TR

IF(NOT>1)

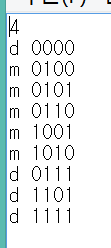
-(NOT-1)\*2->TR

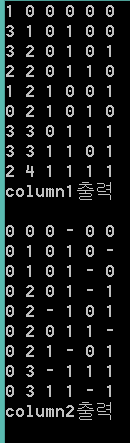
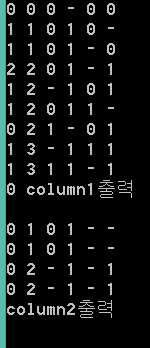
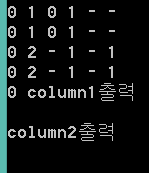
TR->result.txt

**-Flow chart**

**Verification strategy & corresponding examples with explanation**

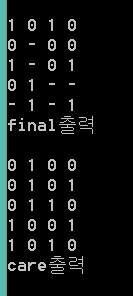
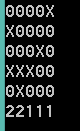
Problem statement**에서의 각각의 단계에 해당하는 코드에 출력값을 넣어 결과의 진행을 확인한다.**

 input\_minterm.txt에 입력된 값, 기본적으로 주어진 결과값이다.(1번 방법)

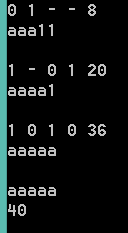
  (2-5번)

첫번째 column1이 txt의 입력값과 일치함을 알 수 있다. 1,2번째 열에도 알맞은 값이 들어가있다. (ex)변형가능 조합은(0000,0100=0-00),(0100,0101=010-),(0100,0110=01-0)로 0100의 첫째 값은 3. 2번째 값에는 1의 개수가 들어가있다.

또한 전 단계의 column2가 다음단계의 column1이 되며 최종적으로는 column2의 유효값이 모두 없어진다. 이후엔 아래와 같이 final과 care의 값이 출력됨을 알 수 있다.(6, 7번)

  (8번-1)

다음은 final과 care값, 그리고 이를 바탕으로 만들어진 주항차트이다. 값이 예상대로 나왔다는 것을 final과 care배열, 그리고 주항차트를 통해 알 수 있다.

(8번-2)

주항차트에서 3번째, 4번째, 5번째 열의 유일한 열을 가지는 4번째, 3번째, 1번째 문자열이 순서대로 대입됨을 알 수 있다. 4번째 문자열에 의해 1,2,3번째 열의 마지막 배열에 a가 대입되며, 마찬가지로 3번째 문자열에 의해 4번째 열의 마지막이, 1번째 문자열에 의해 5번째의 열의 마지막 배열의 a가 채워진다. 끝내 마지막 행이 a으로만 이루어 질 때 멈춘다.

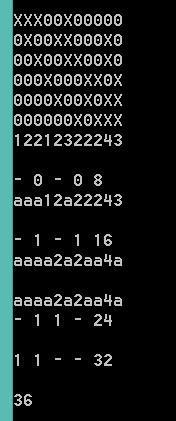
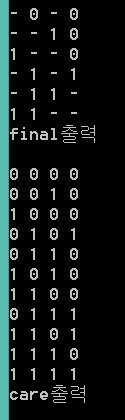
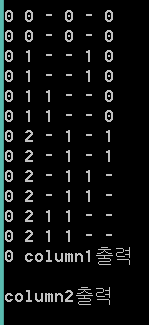
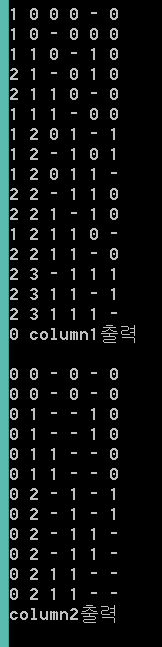
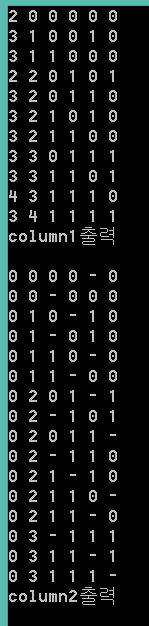
동시에 트랜지스터의 개수가 측정되고 있음을 알 수 있다.

1번줄은 0하나 1하나로 이루어져 2-input OR, 1 inverter로 (6+2)=8 이 출력된다. (8)

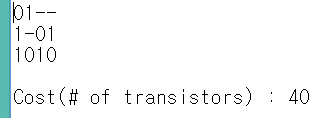
2번째 줄은 0이 1개, 1이 2개로, inverter1개, 3-input OR로 (2+8)=10이 더해진다. 이때 각 문자열을 묶어주는 AND의 input이 1개 상승하므로 2를 추가적으로 더해준다.(20)

2번째 줄은 0이 2개, 1이 2개로, inverter2개, 4-input OR로 (4+10)=14가 더해진다. 이때 각 문자열을 묶어주는 AND의 input이 1개 상승하므로 2를 추가적으로 더해준다.(36)

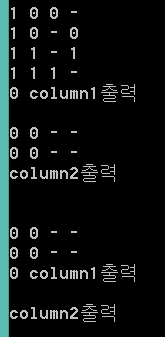
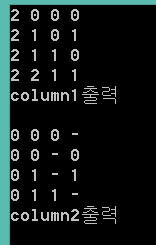
끝으로 전체를 묶어주는 3input And의 트랜지스터 수에서 2\*(3-1)을 빼준 4를 더해준다.(40)

(9번)

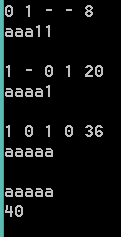
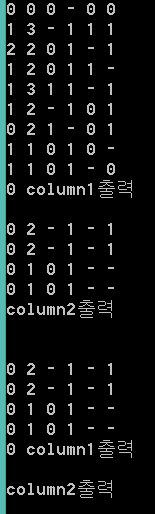
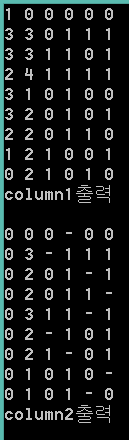
이는 열에서 유일한 X를 갖는 문자열만으로 해결할 수 없는 코드이다. (기본예시는 9번 증명 불가) 위를 보면 final의 1번째와 4번째 열이 먼저 적용되고 트랜지스터의 계산이 나온다. 이후 a로 되지 못한 열 들을 a로 만들어주기 위해 아래서부터 대입되는 것을 알 수 있다.

(10번)

끝으로 Result.txt파일을 열어 결과값을 확인할 수 있다. 위와 같은 결과값이 출력되어 저장이 잘 되었다는 것을 알 수 있다.



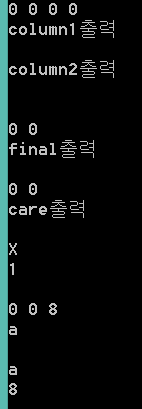
다른 예시로, 이는 2비트로 0부터 4까지 모든 값을 받은 경우이다. 결과적으로 essential PIs로 –만을 갖게 되어 트랜지스터 수로 0을 출력하게 된다.



위는 기존 예시의 순서를 바꿔 출력한 결과이다. 오류 없이 결과값이 바뀌지 않음을 알 수 있다.

**-A testbench that you think it is very hard to solve**

**Input:**  **output:** 

**Process:** 

**많은 사람들이 입력이 1개일 때의 예외처리를 놓쳤을 것이라 생각합니다. 혹은 1개일 때 트랜지스터 수를 0 두는 등의 예외처리 실수 등으로 많은 오류가 발생했을 거라 생각합니다. 추가적으로 00은 A’\*B’=(A+B)’로 바꾸어 cost를 8로 줄일 수 있으므로 트랜지스터 수의 최소화를 테스트 하는 데에도 좋을 것이라고 생각합니다.**