**논리회로설계**

**도전과제 보고서**

|  |  |
| --- | --- |
| 프로젝트 명 | *CD + RD + PM* |
| 팀 명 |  |
| 문서 제목 | 결과보고서 |

|  |  |
| --- | --- |
| **Version** | 1.0 |
| **Date** | 2022-05-20 |

|  |  |
| --- | --- |
| **팀원** |  |
|  |
|  |
|  |
|  |

**문서 정보 / 수정 내역**

|  |  |
| --- | --- |
| **Filename** | 최종보고서-CDRDPM.doc |
| **원안작성자** |  |
| **수정작업자** |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
| 2022-05-20 |  |  | 초안 | 초기설정 및 구현방법 |

**목 차**

[1 개요 4](#_Toc43103653)

[2 개발 내용 및 결과물 5](#_Toc43103654)

[2.1 목표 5](#_Toc43103655)

[2.2 개발 내용 및 결과물 6](#_Toc43103656)

[2.2.1 개발 내용 6](#_Toc43103657)

[2.2.2 시스템 구조 및 설계도 6](#_Toc43103658)

[2.2.3 활용/개발된 기술 6](#_Toc43103659)

[2.2.4 현실적 제한 요소 및 그 해결 방안 6](#_Toc43103660)

[2.2.5 결과물 목록 7](#_Toc43103661)

[3 자기평가 8](#_Toc43103662)

[4 참고 문헌 8](#_Toc43103663)

[5 부록 8](#_Toc43103664)

[5.1 사용자 매뉴얼 8](#_Toc43103665)

[5.2 설치 방법 8](#_Toc43103666)

# 개요

Visual studio를 이용하여 c++언어로 Row dominance, Column dominance, Petrick’s method를 구현합니다. 출력을 보다 편하게 보기위해 format라이브러리를 사용하였고 이를위해 컴파일러를 c++20으로 설정해주었습니다. 기본적인 pi, epi, nepi를 찾는 과정은 1단계 과제와 동일합니다. 쓰인 STL 및 라이브러리는 iostream, vector, string, set, map, unordered\_map, algorithm, format이 있습니다.

# 개발 내용 및 결과물

## 목표

|  |  |  |
| --- | --- | --- |
| 적용단계 | 내용 | 적용 여부 |
| 1단계 | pi찾기 | 적용 |
| 2단계 | Epi 및 nEpi찾기 | 적용 |
| 3단계 | Column dominance 구현 | 적용 |
| 4단계 | Row dominance 구현 | 적용 |
| 5단계 | Petrick’s method 구현 | 적용 |

## 개발 내용 및 결과물

### 개발 내용

* **1단계. pi찾기**
* 기본적으로 주어지는 minterm을 1단계 과제와 동일한 input으로 들어온다고 가정하고 해당 minterm들을 이진수로 변환하여 pi를 찾아 이를 pi와 pi가 커버하는 minterm의배열쌍으로 저장하고 (pair<string, vector<int>>) 이런 pi쌍을 배열에 모두 저장합니다.
* **2단계. Epi 및 nEpi찾기**
* map자료형을 사용한 변수를 선언하여 key값을 minterm, value를 key에해당하는 minterm을 커버하고있는 minterm의 수에 해당하는 값을 넣습니다. 따라서 이 변수(m이라고 하겠습니다.)의 value가 1인 key를 cover하는 pi를 epi, 나머지를 nepi로 나눠 pi와 같은 자료구조로 저장합니다.
* **Column dominance관계 찾기**
* 현재 pi의 자료구조가 pi를 중심으로 되어 있어 minterm을 중심으로 해당하는 minterm을 커버하는 pi들을 저장하는 변수를 map자료형을 사용하여 재배열합니다. 이후 2중 for문을 이용하여 dominate관계를 알아냅니다.
* **Row dominance관계 찾기**
* row dominance의 경우 이미 pi를 중심으로 한 자료구조를 사용할 수 있습니다. 2중for문을 돌며 만약 지배받는 pi가 커버하는 minterm들이 지배하는 pi가 커버하는 minterm들에 존재하지 않으면 지배관계가 아님을 이용하여 dominate관계를 알아냅니다.
* **Petrick’s Method 적용**
* CD때와 마찬가지로 minterm을 기준으로 한 자료형이 필요해서 같은방식으로 변수를 선언합니다. 모든 minterm을 커버할때까지 while문을 돌며 각 minterm을 커버하는 pi를 골라내는 작업을 합니다. 작업중 pi를 선택하는 방식은 커버해야할 minterm을 가장 많이 보유한 pi를 골라내는 방식으로 합니다.

### 활용/개발된 기술

Iostream : 진행상황을 확인 할 수 있게 출력하는데 사용합니다

Vector : pi의 배열을 저장하는데 사용 push\_back등으로 동적으로 크기를 변화시키고 erase등으로 특정위치의 값을 삭제하는데 필요합니다.

String : pi를 표현 하는 데에 있어서 char[]보다 유용합니다.

Map, unordered\_map : 해당 key값에 대한 값을 저장 하는 데에 유용합니다. vector로는 인덱스 / value로만 저장 할 수 있는 점을 보완해줍니다. Unordered\_map은 정렬이 필요 없는 경우 약간이나마 시간을 줄이기 위해 사용해보았습니다.

Algorithm : search 및 sort를 직접 구현 할 필요 없이 미리 잘 짜여진 알고리즘을 사용합니다.

Format : 보다 편하게 출력을 보기 위해 사용합니다.

### 현실적 제한 요소 및 그 해결 방안

Petrick’s method를 구현하는 데에 있어서 사실 모든 경우의 수를 고려해서 최적인 답안을 고르고 싶었는데, 일단 모든 경우의 수가 너무 많고 해당하는 minterm들의 조합이 가변적이기때문에 for문을 중첩하는 방법이 떠오르지 않아, 차선책으로 POS까지 가지 않고 SOP의 단계에서 가장 많이 등장하는 pi를 순차적으로 골라 모든 minterm을 커버합니다.

이에 대한 문제점이 단일한 해답이 나온다는 것입니다. Optimized된 pi의 조합이 단일하지 않을 수 있는데 이에 대한 해답을 하나밖에 제시하지 못합니다.

### 결과물 목록

논회설EPI.sln :프로젝트들의 실행에 필요한 visual studio 솔루션파일입니다.

CD/ColumnDominance.cpp, RD/rowdominance.cpp, PM/PetricksMethod.cpp : 각각 cd, rd, pm을 구현한 cpp파일입니다. 별도의 h파일은 존재하지않습니다.

# 부록

## 사용자 매뉴얼

**sln파일을 열어 visual studio를 실행한 후 원하는 프로젝트를 시작프로젝트로 설정 한 후 실행하면됩니다.**