


논리 회로 설계 도전 과제

프로젝트 명	Row Dominance, Column Dominance, Petrick Method 구현
팀 명	
문서 제목	결과 보고서

Version	1.0
Date	2022-05-26

팀원	

	국민대학교 소프트웨어학부 논리회로설계	결과보고서		
		프로젝트 명	Row Dominance, Column Dominance , Petrick Method 구현	
		팀 명		
		Confidential Restricted	Version 1.0	2022-MAR-26

CONFIDENTIALITY/SECURITY WARNING

이 문서에 포함되어 있는 정보는국 국민대학교 소프트웨어융합대학 소프트웨어학부 및 소프트웨어학부 개설 교과목 논리회로설계 수강 학생 중 도전과제를 수행하는 팀 “이혁규”의 팀원들의 자산입니다. 국민대학교 소프트웨어학부 및 팀 “이혁”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다.

문서 정보 / 수정 내역

Filename	논리회로설계 _결과보고서.doc
원안작성자	
수정작업자	

수정날짜	대표수정자	Revision	추가/수정 항목	내 용
2022-05-26		1.0	최초 작성	

	국민대학교 소프트웨어부 논리로설계	결과보고서	
		프로젝트 명	Row Dominance, Column Dominance , Petrick Method 구현
		팀 명	
		Confidential Restricted	Version 1.0 2022-MAR-26

1.1 개발내용 및 결과물

```
# column dominance를 시키기 전에 epi가 가지고 있는 minterm들이 nepi가 가지고 있는 minterm에 속해있다면 제거해준다.
def remove_EPI_Minterm(removed_epi, epi):
    removed_epi_minterm = removed_epi
    for epi_key in epi:
        term = epi[epi_key]
        for nepi_key in removed_epi:
            removed_epi_minterm[nepi_key] = [x for x in removed_epi_minterm[nepi_key] if x not in term]

    # 제거 후 NEPI가 가지고 있는 값이 빈 리스트([])인 경우 EPI들에 지배받는 경우(Row Dominance)이므로 먼저 한 번 제거해준다.
    dominated = []
    for key in removed_epi_minterm:
        if (removed_epi_minterm[key] == []):
            dominated.append(key)
    for i in dominated:
        del removed_epi_minterm[i]

    return removed_epi_minterm
```

1.1.1 개발내용

– remove_EPI_Minterm 함수

1. Column Dominance 를 하기 전 , EPI 가 가지고 있는 minterm 들이 NEPI 가 가지고 있는 Minterm 에 속해있다면 제거해준다.
2. 제거 후 NEPI 가 가지고 있는 값이 빈 리스트인 경우는 EPI 들에게 지배받는 경우(Row Dominance) 이므로 먼저 제거해준다.
3. EPI 의 Minterm 이 제거 된 pi 테이블을 리턴한다.

	국민대학교 소프트웨어부 논리회로설계	결과보고서	
		프로젝트 명	Row Dominance, Column Dominance , Petrick Method 구현
		팀 명	
		Confidential Restricted	Version 1.0 2022-MAR-26

```

for key1 in mintermTable:
    dominate = mintermTable[key1]
    for key2 in mintermTable:
        dominated = mintermTable[key2]
        if (key1 == key2):
            continue
        # 지배하는 minterm에서 지배 당하는 minterm을 제거해줌, 지배 당하는 minterm에서 지배하는 minterm을 제거해준다
        remove_dominated = [x for x in dominate if x not in dominated]
        dominated = [x for x in dominated if x not in dominate]
        # 지배하는 Minterm의 길이가 0 보다 크고 지배 당하는 minterm이 비어있으면 지배하는 minterm을 리스트에 추가해준다
        if (len(remove_dominated)> 0 and dominated == []):
            Cdominate_Minterm.append(key1)
        # 지배하는 minterm, 지배당하는 minterm 둘다 비어있을 경우 interchagable하기에 다른 리스트에 추가해준다.
        elif (remove_dominated == [] and dominated == []):
            # 중복을 사전에 방지하지 위해 집합 형식으로 추가해준다.
            Interchage[key1] = key2

# 중복 제거
value = list(Interchage.items())
plus_minterm = []
remove_minterm = []
for i in range(len(value)):
    if (value[i][0] not in remove_minterm):
        plus_minterm.append(value[i][0])
        remove_minterm.append(value[i][1])

Cdominate_Minterm = list(set(Cdominate_Minterm))
Cdominate_Minterm += plus_minterm

# pi의 minterm 중 지배하는 minterm들을 제거해준다
for i in Cdominate_Minterm:
    for key in afterCD:
        term = afterCD[key]
        if (i in term):
            del term[term.index(i)]
            afterCD[key] = term

# 지배하는 minterm을 제거해준 뒤 minterm을 가지지 않는 pi를 제거해준다.
Cdominate_pi = []
for key in afterCD:
    if (afterCD[key] == []):
        Cdominate_pi.append(key)
for i in Cdominate_pi:
    del afterCD[i]

return afterCD, Cdominate_Minterm, Cdominate_pi

```

- Column Dominance 함수

1. minterm 이 row 가 되고 pi 가 column 이 되는 table 을 생성한다.
2. table 을 활용해서 지배하는 minterm 을 찾아낸다.
3. 서로 지배하는 관계(Interchagable)도 추가로 찾아낸 후, 첫번째 나오는 값을 지배하는 minterm 으로 설정한다.
4. 기존 pi 테이블에서 지배하는 minterm 을 제거한다.
5. 지배하는 minterm 을 제거해준 뒤 minterm 을 가지지 않은 pi 를 테이블에서 삭제해준다.
6. pi table, 지배하는 minterm, 지배하는 pi 를 리턴한다.

	국민대학교 소프트웨어부 논리회로설계	결과보고서	
		프로젝트 명	Row Dominance, Column Dominance , Petrick Method 구현
		팀 명	
		Confidential Restricted	Version 1.0 2022-MAR-26

```
def rowDominance(afterCD):
    afterRD = afterCD
    Rdominated_pi = []
    Interchage = {}

    for key1 in afterRD:
        dominate = afterRD[key1]
        for key2 in afterRD:
            dominated = afterRD[key2]
            if (key1 == key2):
                continue
            remove_dominated = [ x for x in dominate if x not in dominated ]
            dominated = [ x for x in dominated if x not in dominate ]
            if (len(remove_dominated) > 0 and dominated == []):
                Rdominated_pi.append(key2)
            elif (remove_dominated == [] and dominated == []):
                Interchage[key1] = key2

    # Interchage 중복을 제거하기 위한 과정. plus_pi를 Rdominated_pi에 추가할 것이다.
    value = list(Interchage.items())
    plus_pi = []
    rm_pi = []
    for i in range(len(value)):
        if (value[i][0] not in rm_pi):
            plus_pi.append(value[i][0])
            rm_pi.append(value[i][1])


    Rdominated_pi = list(set(Rdominated_pi))
    Rdominated_pi += plus_pi


    for i in Rdominated_pi:
        del afterRD[i]

    return afterRD, Rdominated_pi
```

- Row Dominance 함수

1. table 을 활용해서 지배당하는 PI 을 찾아낸다.
2. 서로 지배하는 관계(Interchagable)도 추가로 찾아낸 후, 첫번째 나오는 값을 지배당하는 PI 로 설정한다.
3. 지배 당하는 PI 를 PI 테이블에서 제거한다.
4. PI 테이블과 지배당하는 PI 를 리턴한다.

	국민대학교 소프트웨어학부 논리로설계	결과보고서	
		프로젝트 명	Row Dominance, Column Dominance , Petrick Method 구현
		팀 명	
		Confidential Restricted	Version 1.0 2022-MAR-26

	국민대학교 소프트웨어부 논리회로설계	결과보고서	
		프로젝트 명	Row Dominance, Column Dominance , Petrick Method 구현
		팀 명	
		Confidential Restricted	Version 1.0 2022-MAR-26

```
def petrick(answer):
    # pi에 해당하는 minterm이 하나인 경우, 이는 epi이므로 제외시켜준다.
    epi = []
    for key in answer:
        if (len(answer[key]) == 1):
            epi.append(key)
    print("EPI : ", epi, "\n")
    for i in epi:
        del answer[i]

    minterm = []
    for key in answer:
        minterm += answer[key]
    minterm = list(set(minterm))

    # minterm(row)- pi(column) 형식의 테이블 생성
    mintermTable = {}
    for i in minterm:
        term = []
        for pi in answer:
            if (i in answer[pi]):
                term.append(pi)
        mintermTable[i] = term

    #
    minterm = set(minterm)
    sop = []

    for key in mintermTable:
        # 만약 이전 반복에 의해 minterm이 포함 되었다면 continue
        if (key not in minterm):
            continue
        cnt = 0
        flag = False
        choice = mintermTable[key][cnt]
        choice_minterm = set(answer[choice])
        # 이전 반복에 이미 선택된 pi이면 해당 minterm을 포함하는 다른 pi를 순차적으로 탐색한다.
        if (choice in sop):
            choice_length = len(mintermTable)
            while(True):
                cnt += 1
                choice = mintermTable[key][cnt]
                # 모든 pi가 선택되었으면 flag 변수 값을 참으로 만든 후 break and continue
                if (cnt == choice_length-1):
                    flag = True
                    break
                # 해당하는 pi가 선택된 pi가 아니라면 이 pi를 선택한다.
                if (choice not in sop):
                    break
            if (flag):
                continue
        # 선택 된 pi를 sop 리스트에 추가한다. 그리고 선택된 pi가 포함하는 minterm을 제거한다.
        sop.append(choice)
        minterm -= choice_minterm

    return sop
```

	국민대학교 소프트웨어부 논리회로설계	결과보고서	
		프로젝트 명	Row Dominance, Column Dominance , Petrick Method 구현
		팀 명	
		Confidential Restricted	Version 1.0 2022-MAR-26

- Petrick 함수

1. π 에 해당하는 minterm 이 하나인 경우, 다음 findEPI 에서 검출되어야 할 EPI 이므로 Petrick Method 에서 제외시켜준다.
2. minterm(row) - PI(column) 형식의 테이블을 생성한다.
 3. minterm 의 수만큼 for 문을 돌며, 이전 반복에 의해 minterm 이 포함되지 않고 π 가 선택되지 않은 경우, minterm 에 해당하는 π 를 sop 에 추가한다
4. 최종 결과값 sop 를 리턴한다.

	국민대학교 소프트웨어부 논리회로설계	결과보고서	
		프로젝트 명	Row Dominance, Column Dominance , Petrick Method 구현
		팀 명	
		Confidential Restricted	Version 1.0 2022-MAR-26

1.1.2 시스템 구조 및 설계도


```
def test(minterm):
    answer = getPI(minterm)
    print("PI : ", answer, "\n")

    cnt = 1
    answer, epi = findEPI(answer, minterm)
    answer = remove_EPI_Minterm(answer, epi)
    print("EPI - ", cnt, " : ", list(epi.keys()))
    print("After Removing EPI : ", answer, "\n")
    while(True):
        answer, Cdominate_Minterm, Cdominate_pi = columnDominance(answer)
        print("After CD> PI Table : ", answer)
        print("Column Dominance Minterm : ", Cdominate_Minterm)
        print("Column Dominance PI : ", Cdominate_pi, "\n")

        answer, Rdominated_PI = rowDominance(answer)
        print("After RD> PI Table : ", answer)
        print("Row Dominance PI : ", Rdominated_PI, "\n")
        )
        if (epi == {} and Cdominate_pi == []):
            sop = petrick(answer)
            print("After Petrick Method : ", sop)
            break

        cnt += 1;
        answer, epi = findEPI(answer, minterm)
        answer = remove_EPI_Minterm(answer, epi)
        print("EPI - ", cnt, " : ", list(epi.keys()))
        if (answer == {}):
            break;
```

- (1) PI 테이블에서 모든 PI 를 찾는다.
 - (2) 테이블에서 EPI 를 찾고 제거한다.
- => 더이상 NEPI 가 없으면 QUIT

	국민대학교 소프트웨어부 논리회로설계	결과보고서	
		프로젝트 명	Row Dominance, Column Dominance , Petrick Method 구현
		팀 명	
		Confidential Restricted	Version 1.0 2022-MAR-26

(3) Column Dominance 를 적용한다.

(4) Row Dominance 를 적용한다.

(5) (2)와 (3)을 통해 어떠한 최적화가 되었다면 (2)로, 되지 않았다면 Petrick method 를 적용한다.

1.1.3 결과물 목록

1 번 테스트

```
test([4, 11, 0, 2, 5, 6, 7, 8, 10, 12, 13, 14, 15])
```

케이스



```
> python -u "/home/ehyeok9/sophomore/intelligence_for_vehicle/QM_method.py"
PI : ['11--', '1--0', '-0-0', '-11-', '-1-1', '--10']

EPI - 1 : ['-0-0', '-1-1']
After Removing EPI : {'11--': [12, 14], '1--0': [12, 14], '-11-': [6, 14], '--10': [6, 14]}

After CD> PI Table : {'11--': [12], '1--0': [12], '-11-': [6], '--10': [6]}
Column Dominance Minterm : [14]
Column Dominance PI : []

After RD> PI Table : {'1--0': [12], '--10': [6]}
Row Dominance PI : ['11--', '-11-']

EPI - 2 : ['1--0', '--10']
```

2 번 테스트 케이스

```
test([4, 13, 0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13])
```

	국민대학교 소프트웨어부 논리회로설계	결과보고서	
		프로젝트 명	Row Dominance, Column Dominance , Petrick Method 구현
		팀 명	
		Confidential Restricted	Version 1.0 2022-MAR-26

```

-----
PI : ['01--', '0-1-', '0--0', '10--', '1-0-', '-01-', '-0-0', '-10-', '--00']
EPI - 1 : []
After Removing EPI : {'01--': [4, 5, 6, 7], '0-1-': [2, 3, 6, 7], '0--0': [0, 2, 4, 6], '10--': [8, 9, 10, 11],
0, 4, 8, 12]}

After CD> PI Table : {'01--': [5, 7], '0-1-': [3, 7], '0--0': [0], '10--': [9, 11], '1-0-': [9, 13], '-01-': [3, 11], '-10-': [10, 12]}
Column Dominance Minterm : [2, 4, 6, 8, 10, 12]
Column Dominance PI : []

After RD> PI Table : {'01--': [5, 7], '0-1-': [3, 7], '10--': [9, 11], '1-0-': [9, 13], '-01-': [3, 11], '-10-': [10, 12]}
Row Dominance PI : ['0--0', '-0-0']

EPI : ['--00']

After Petrick Method : ['0-1-', '01--', '10--', '1-0-']
~ /sophomore/intelligence_for_vehicle

```

- 강의 자료의 petrick 파트에 있는 Minterm 을 예시로 활용하였다.

◦ 1 번 테스트 케이스

	CD' (2,6,10,14)	BC (6,7,14,15)	AD' (8,10,12,14)	AB (12,13,14,15)
6	X	X		
12			X	X
14	X	X	X	X

Row 14 *dominates* both row 6 and row 12. That is, row 14 has an “X” in every column where row 6 has an “X” (and, in fact, row 14 has “X”’s in other columns as well). Similarly, row 14 has in “X” in every column where row 12 has an “X”. Rows 6 and 12 are said to be *dominated by* row 14.


1) Column dominance 후 minterm 14 가 minterm 6,12 를 지배하므로 minterm 14 를 제거한다.

```

After CD> PI Table : {'11--': [12], '1--0': [12], '-11-': [6], '--10': [6]}
Column Dominance Minterm : [14]
Column Dominance PI : []

```

	CD' (2,6,10,14)	BC (6,7,14,15)	AD' (8,10,12,14)	AB (12,13,14,15)
6	X	X		
12			X	X

	국민대학교 소프트웨어학부 논리로설계	결과보고서	
		프로젝트 명	Row Dominance, Column Dominance , Petrick Method 구현
		팀 명	
		Confidential Restricted	Version 1.0 2022-MAR-26

2) '11--'과 '1-0', '-11-'과 '--10'이 서로를 지배하는 관계(Interchagable)이므로 임의로 제거한다

```
After RD> PI Table : {'1--0': [12], '--10': [6]}
Row Dominance PI : ['11--', '-11-']
```

	국민대학교 소프트웨어부 논리회로설계	결과보고서		
		프로젝트 명	Row Dominance, Column Dominance, Petrick Method 구현	
		팀 명		
		Confidential Restricted	Version 1.0	2022-MAR-26

° 1 번 테스트 케이스

	$A'D'$	$B'D'$	$C'D'$	$A'C$	$B'C$	$A'B$	BC'	AB'	AC'
0	X	X	X						
2	X	X		X	X				
3				X	X				
4	X		X			X	X		
5						X	X		
6	X			X		X			
7				X		X			
8		X	X					X	X
9								X	X
10		X			X			X	
11					X			X	
12			X				X		X
13							X		X

1) Column

Dominance : 2 →

3, 4 → 5, 6 → 7, 8 → 9, 10 → 11, 12 → 13 이렇게 지배하는 - 지배당하는 구조가 형성된다.

지배하는 [2, 4, 6, 8, 10, 12] minterm 을 제거한다.

After CD> PI Table : { '01--': [5, 7], '0-1-': [3, 7], '0--0': [0], '10--': [9, 11], '1-0-': [9, 13], '-01-': [3, 11], '-10-': [5, 13], '--00': [0] }
Column Dominance Minterm : [2, 4, 6, 8, 10, 12]
Column Dominance PI : []

	$A'D'$	$B'D'$	$C'D'$	$A'C$	$B'C$	$A'B$	BC'	AB'	AC'
0	X	X	X						
3				X	X				
5						X	X		
7				X		X			
9								X	X
11					X			X	
13							X		X

2) Row Dominance : '-

-00', '0-0', '-0-0'이 서로를 지배하는 관계(Interchagable)이다. 임의로 하나의 pi 만 남긴다.

After RD> PI Table : { '01--': [5, 7], '0-1-': [3, 7], '10--': [9, 11], '1-0-': [9, 13], '-01-': [3, 11], '-10-': [5, 13], '--00': [0] }
Row Dominance PI : ['0--0', '-0-0']

	$A'D'(**)$	$A'C$	$B'C$	$A'B$	BC'	AB'	AC'
(○)0	X						
3		X	X				
5				X	X		
7		X		X			
9						X	X
11			X			X	
13					X		X

결과보고서

of the copyright owner.

	국민대학교 소프트웨어부 논리로설계	결과보고서		
		프로젝트 명	Row Dominance, Column Dominance , Petrick Method 구현	
		팀 명		
		Confidential Restricted	Version 1.0	2022-MAR-26

3) Petrick Method : minterm 0 에 해당하는 pi 는 다음 단계에서 찾아야 할 epi 이므로 제외시켜준다. 나머지 minterm 들을 통해 최소 논리곱의 합(SOP)를 얻는다.

EPI : ['--00']

After Petrick Method : ['0-1-', '01--', '10--', '1-0-']

2 참고 문헌

번호	종류	제목	출처	발행년도	저자	기타
1	파일	https://cseweb.ucsd.edu/classes/su14/cse140-a/handouts/Quine.pdf				