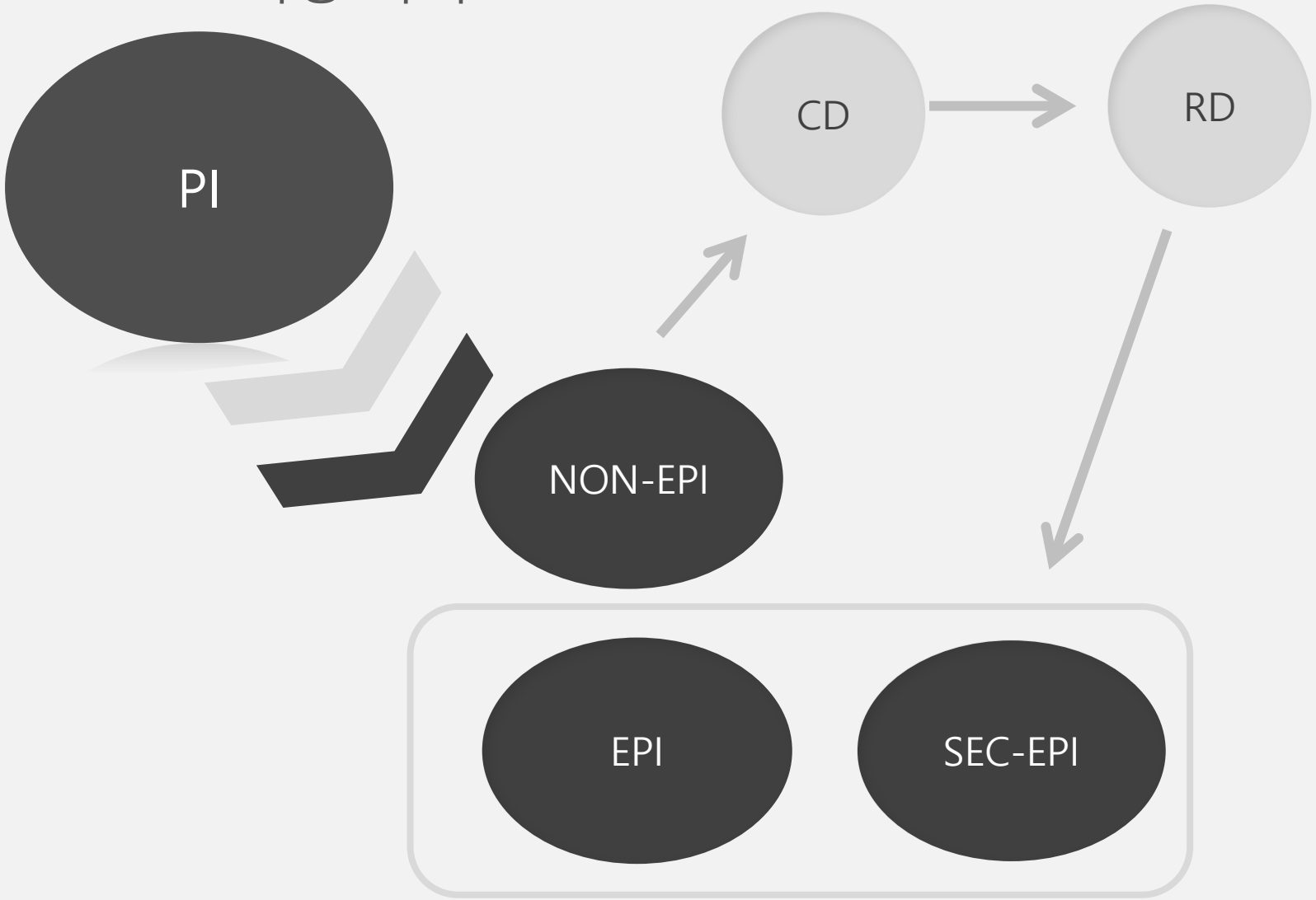
논리회로설계 추가과제

Anonymous

CONTENTS

Ⅰ. RD 와 CD 사용 목적

Ⅰ. RD CD 사용 목적



Ⅰ. RD CD 사용 목적

- NON EPI -> SEC EPI
- Use CD and RD
- First use CD and then use RD

Ⅱ. Column Dominance

Ⅱ. Column Dominance

```
def col_dominance_print(last, col_minus_epi):
    print("<<CM>>")
    answer = []
    if (len(last) == 0 ):
        print("CM doesn't exist")
        return answer
    for i in range(0,len(last)):
        if last[i] == -1 :
            col_is_dominated = last[i-2]
            col_dominates = last[i-1]
            print("minterm "+str(col_minus_epi[col_dominates]) +
                  " dominates minterm " + str(col_minus_epi[col_is_dominated]))
            answer.append(col_minus_epi[col_is_dominated])
        if last[i] == -2 :
            col_is_dominated = last[i-2]
            col_dominates = last[i-1]
            print("minterm "+str(col_minus_epi[col_dominates]) +
                  " interchangeable minterm " + str(col_minus_epi[col_is_dominated]))
    return answer
```

- Column Dominance : EPI를 정한 후 EPI가 가지고 있는 Minterm은 제외한 상태에서 나머지를 Cover 하기 위해 SEC-EPI를 찾는 과정 에서 사용.

- Column Dominance : Minterm들을 비교하는 것 해당 Minterm이 어느 PI에 담겨있는 지를 비교. 즉, Minterm의 기준에서 해당하는 PI를 list 형태로 담아 다른 Minterm과 list 원소를 각각 비교.

- List : Minterm 기준 자신을 Cover하는 PI를 원소로 한다.
- Dominate : 길이가 짧은 List의 원소들이 길이가 긴 List 안에 모두 포함된 경우. 이 때 Dominate 당한 쪽을 선택.
- Interchangeable : 길이가 같고 포함된 원소들이 같은 경우.

Ⅲ. Row Dominance

Ⅲ. Row Dominance

```
def row_dominance_print(row_answer):
    print("<<RM>>")
    if len(row_answer) == 0 :
        print("RM doesn't exist")
        return 0
    for i in range(0,len(row_answer)):
        if row_answer[i] == -1 :
            if(row_answer[i-2].count('2') != 0 ):
                row_answer[i-2] = row_answer[i-2].replace('2','-')
            if(row_answer[i-1].count('2') != 0 ):
                row_answer[i-1] = row_answer[i-1].replace('2','-')
            print("PI ("+ row_answer[i-2] + ") dominantes PI (" + row_answer[i-1] + ")")
        if row_answer[i] == 2 :
            if(row_answer[i-2].count('2') != 0 ):
                row_answer[i-2] = row_answer[i-2].replace('2','-')
            if(row_answer[i-1].count('2') != 0 ):
                row_answer[i-1] = row_answer[i-1].replace('2','-')
            print("PI ("+ row_answer[i-2] + ") interchangeable PI (" + row_answer[i-1] + ")")
```

- Row Dominance : EPI가 Cover하는 Minterm을 제외한 후 Column Dominance로 나머지 Minterm의 일부를 제외한 상태에서 적용

- Row Dominance : PI를 비교하는 것
  해당 PI 가 어느 Minterm을 Cover 하는 지 비교.
  즉, PI 의 기준에서 Cover 하는 Minterm 를 list 형
  태로 담아 다른 PI list 원소를 각각 비교.

- List : PI 기준 Cover하는 Minterm 을 원소로 한다.
- Dominate : 길이가 짧은 List의 원소들이 길이가 긴 List 안에 모두 포함된 경우. 이 때 Dominate 한 쪽을 선택.
- Interchangeable : 길이가 같고 포함된 원소들이 같은 경우.

+ Draw

Draw

Table

EPI

```
def draw_maze(draw_pi,draw_epi,numdata, all_minterm,maze_pi,maze_epi,col_answer,row_answer,
              col_interchange,row_interchange):
    maze = [[ "-" for j in range(numdata+1)] for i in range(len(draw_pi)+1 +len(draw_epi))]
    # row : r
    print('')
    print("--
    print("<<
    maze[0][(
    for i in
        maze[
    for i in
        maze[
    for i in
        maze[

    for i in
        if ma
            f

    for i in
        for
            i

        for
            i
```

```
    how_max = 0
    how_max_index = 0
    if(col_interchange != 0 ) and(len(col_interchange) != 0) :
        for i in range(1,len(col_interchange)):
            how = 0
            inter_index = maze[0].index(col_interchange[i])
            for k in range(1,len(draw_pi)+1+len(draw_epi)):
                if maze[inter_index][k] == "V":
                    how += 1
            if how > how_max :
                how_max = how
                how_max_index = i
    if(how_max_index != 0 ):
        maze[0][how_max_index] = "IC"


    if(col_answer != 0) and (len(col_answer) >1 ):
        for m in range(1, numdata+1) :
            if (maze[0][m] != "X") and(maze[0][m] not in col_answer) :
                maze[0][m] = "CM"
        print('')
        print("----------<<  MAP >>----------")
        print("<<Check CM>> ")
        for i in range(0,len(maze)):
            for k in range(0,len(maze[i])):
                print(maze[i][k] , end= ' ')
            print('')
    else:
        print('')
        print("<<Check CM>> ")
        print("CM : " + "maze doesn't modify ")
            maze[i][m] = "V"
```

```
if (row_interchange != 0 ) and (len(row_interchange) != 0 ):
    for i in range(0,len(row_interchange)):
        for k in range(1,len(draw_pi)+1+len(draw_epi)):
            if ( maze[k][0] == row_interchange) and(i % 2 == 0):
                for m in range(1,numdata+1):
                    if maze[k][m] == "V":
                        row_how_0 += 1
                row_index_0 = k
            else :
                for m in range(1,numdata+1):
                    if maze[k][m] == "V":
                        row_how_1 += 1
                row_index_1 = k

        if (row_index_0 != 0 ) and( row_index_1 != 0 ):
            if (row_how_0 > row_how_1):
                maze[row_index_0][0] = "IR"
            else :
                maze[row_index_1][0] = "IR"
            row_index_0 = 0
            row_index_1 = 0
            row_how_0 = 0
            row_how_1 = 0

if (row_answer != 0 ) and (len(row_answer) >1 ) :
    for i in range(1,len(draw_pi)+1+len(draw_epi)):
        for k in range(0,len(row_answer)):
            if (maze[i][0] == "'"+str(row_answer[k])+"'" ) and (k % 3 == 1 ):
                maze[i][0] = "  RM  "
    print('')
    print("----------<<  MAP >>----------")
    print("<<Check RM>> ")
    for i in range(0,len(maze)):
        for k in range(0,len(maze[i])):
            print(maze[i][k] , end= ' ')
```

Result

출력

```
<<CD>>
minterm 14 dominates minterm 6
minterm 14 dominates minterm 12
<<RD>>
RD doesn't exist

----------<<  MAP >>----------
<<Check Minterm that Pi has>>
PI/MIN 0 2 5 6 7 8 10 12 13 14 15
'1--0' - - - - - - V  V  V  -  V
'-11-' - - - V V - -  -  -  V  V
'--10' - V - V - - V  -  -  V  -
'-0-0' V V - - - V V  -  -  -  -
'-1-1' - - V - V - -  -  V  -  V

----------<<  MAP >>----------
<<Check EPI>>
PI/MIN X X X 6 X X X 12 X 14 X
'11--' - - - - - - - -  V  V  V  V
'1--0' - - - - - - V V  V  -  V  -
'-11-' - - - V V - -  -  -  V  V
'--10' - V - V - - V  -  -  V  -
  EPI  V V - - - V V  -  -  -  -
  EPI  - - V - V - -  -  V  -  V

----------<<  MAP >>----------
<<Check CD>>
PI/MIN X X X 6 X X X 12 X CD X
'11--' - - - - - - - -  V  V  V  V
'1--0' - - - - - - V V  V  -  V  -
'-11-' - - - V V - -  -  -  V  V
'--10' - V - V - - V  -  -  V  -
  EPI  V V - - - V V  -  -  -  -
  EPI  - - V - V - -  -  V  -  V

<<Check RD>>
RD : maze doesn't modify

ALL DONE
```

Case2 : EPI에 해당하는 PI는 삭제

Row Dominance로 Dominate 당한 PI는 "RD"로 대체

출력

```
<<CD>>
CD doesn't exist
<<RD>>
PI (101-) dominantes PI (10-0)
PI (11-1) dominantes PI (110-)

----------<<  MAP >>----------
<<Check Minterm that Pi has>>
PI/MIN 0 4 8 10 11 12 13 15
'101-' - - - V  V  -  -  -
'10-0' - - V V  -  -  -  -
'110-' - - - -  -  V  V  -
'11-1' - - - -  -  -  V  V
'1-11' - - - -  V  -  -  V
'--00' V V V -  -  V  -  -

----------<<  MAP >>----------
<<Check EPI>>
PI/MIN X X X 10 11 X 13 15
'101-' - - - V  V  -  -  -
'10-0' - - V V  -  -  -  -
'110-' - - - -  -  V  V  -
'11-1' - - - -  -  -  V  V
'1-11' - - - -  V  -  -  V
  EPI  V V V -  -  V  -  -

<<Check CD>>
CD : maze doesn't modify

----------<<  MAP >>----------
<<Check RD>>
PI/MIN X X X 10 11 X 13 15
'101-' - - - V  V  -  -  -
  RD   - - V V  -  -  -  -
  RD   - - - -  -  V  V  -
'11-1' - - - -  -  -  V  V
'1-11' - - - -  V  -  -  V
  EPI  V V V -  -  V  -  -

ALL DONE
```

Case1 : EPI에 해당하는 PI는 삭제

Column Dominance로 Dominate 한 Minterm 은 "CM"로 대체