논리 회로 설계 CD, RD, PM 구현 설명

총 6 개의 함수 구현

예시를 들면서 설명하겠다. (def petric 에서는 다른 예시로 설명하겠다.)

4bit, 11 개의 minterm(0,2,5,6,7,8,10,12,13,14,15) 이 주어졌다고 가정한다.

1. def solution

기존 1, 2 단계에서는 minterm 이 주어지면 combine 된 2 진 변화가 된 PI list 를 return 해주었지만, 여기서는 combine 될 때 minterm 자체를 결합 해줌으로써 mimterm 조합인 list 를 return 해주는 함수

(ex. 기존 return 값이 101_ 이면 [10,11] 로 return 해주는 함수)

```
['11--', '1--0', '-0-0', '-11-', '-1-1', '--10'
```

-->

```
PI number  0  :  [0, 2, 8, 10]
PI number  1  :  [2, 6, 10, 14]
PI number  2  :  [5, 7, 13, 15]
PI number  3  :  [6, 7, 14, 15]
PI number  4  :  [8, 10, 12, 14]
PI number  5  :  [12, 13, 14, 15]
```

이런식으로 list 를 가져온다.

2. def my_b

2 진 변환시켜주는 함수, solution 함수에서 combine 될 때 사용됨.

```
def my_b(n, m):
    li = []
    for i in range(m):
        q = int(n / 2)
        r = int(n % 2)
        if n != 0:
            if (r == 0):
                li.insert(0, '0')
            else:
                li.insert(0, '1')
            n = n / 2
        else:
            li.insert(0, '0')
    return li
```

3. def colum_dominance

```
def column_dominance(PI_list2):
    pp=[]
    for i in range(0, len(minterm)):
        cc = 0
        cc2 = 0
        for j in range(len(PI_list)):
            if (minterm[i] in PI_list[j] and check[minterm[i]]==1):
                cc = cc + 1
                cc2 = j
        if(cc==1):
            print("column_dominance : ", minterm[i],"in PI_"+str(cc2),":",PI_list[cc2])
            for j in range(len(PI_list[cc2])):
                check[PI_list[cc2][j]]=0

            for n in range(len(PI_list[cc2])):
                for m in range(len(PI_list)):
                    if(m==cc2): continue
                    if(PI_list[cc2][n] in PI_list2[m]): PI_list2[m].remove(PI_list[cc2][n])
            pp.append(cc2)
    pp2=[]
    for i in range(len(PI_list2)):
        if(i not in pp):
            pp2.append(PI_list2[i])
        else: pp2.append([])

    print(pp2)
    return pp2
```

초기 설정 : 2 중 for 문 i, j 를 돌리는데, I 는 minterm 의 수만큼, j 는 pi 의 수만큼 돌린다.

방법 : minterm 에 대해서 이걸 가지고 있는 pi 가 하나임을 따진다.

맞으면 그 pi 가 가지고 있는 minterm 들은 check 를 시켜 주고, 없애준다.

그리고 CD 가 된 pi 의 mimterm 들을 다른 pi 에서 볼 필요 없기 때문에 나머지 pi 에서도 CD 가 된 minterm 들의 요소를 제거해준다.

마지막 처리 : CD 가 되면 check 가 됐고, minterm 이 지워지기 때문에 빈 list([])가 된다. 그걸 살려주고 결과값을 return 해준다. (처리를 해주지 않으면 pi 갯 수를 유지 할수 없다.)

```
    0   2   5   6   7   8   10  12  13  14  15
pi0 V   V               V   V
pi1     V       V           V               V
pi2         V       V               V       V
pi3             V   V                   V   V
pi4                     V   V   V       V
pi5                         V   V   V   V

 1 번째 실행
column_dominance :  0 in PI_0 : [0, 2, 8, 10]
column_dominance :  5 in PI_2 : [5, 7, 13, 15]
[[], [6, 14], [], [6, 14], [12, 14], [12, 14]]
    0   2   5   6   7   8   10  12  13  14  15
pi0
pi1             V                       V
pi2
pi3             V                       V
pi4                         V           V
pi5                         V           V
```

← before

← after

4. def row_dominance

```
def row_dominance():
    pp=[]
    for i in range(len(PI_list)):
        for j in range(i+1,len(PI_list)):
            union = list(set(PI_list[i]) & set(PI_list[j]))

            if(PI_list[j]==union and len(ab_PI_List[i])>=len(ab_PI_List[j]) and PI_list[j]!=[]):
                print("Row_dominance : PI",j,"< PI",i)
                PI_list[j]=[]
                pp.append(j)

            elif(PI_list[i]==union and len(ab_PI_List[i])<=len(ab_PI_List[j]) and PI_list[i]!=[]):
                print("Row_dominance : PI", i, "< PI", j)
                PI_list[i]=[]
                pp.append(i)

    pp2 = []
    for i in range(len(PI_list)):
        if (i not in pp and PI_list[i]!=[]):
            pp2.append(PI_list[i])

        else: pp2.append([])

    # print("After R_D : ",pp2)

    return pp2
```

초기 설정 : pi list 를 2 중 for 문을 이용해 포함이 되는지 여부를 따진다.

방법 : 교집합 method 를 이용하여 PI:A 가 PI:B 에 포함이 된다면 PI:A 지워 준다.

반대로 PI:B 가 PI:A 에 포함이 된다면 PI:B 를 지워준다.

추가 조건 : 이때 interchangeble 한 경우가 생겼을 때, cost 관점에서 minterm 을 더 많이 가지고 있는쪽을 선택하는게 낫다.

마지막 처리 : RD 가 된 pi 는 제거가 됐다. 이때는 빈 list([])로 처리해준다. (처리 해주지 않으면 pi 갯 수가 줄어든다.)

```
    0   2   5   6   7   8   10  12  13  14  15
pi0
pi1             V                       V
pi2
pi3             V                       V
pi4                             V       V
pi5                             V       V
Row_dominance : PI 3 < PI 1
Row_dominance : PI 5 < PI 4
checking : 0: o  2: o  5: o  6: x  7: o  8: o  10: o  12: x  13: o  14: x  15: o
    0   2   5   6   7   8   10  12  13  14  15
pi0
pi1             V                       V
pi2
pi3
pi4                             V       V
pi5
```

← before

← after

위 예제의 CD, RD 의 최종 결과이다.

```
SETUP
PI number  0  :  [0, 2, 8, 10]
PI number  1  :  [2, 6, 10, 14]
PI number  2  :  [5, 7, 13, 15]
PI number  3  :  [6, 7, 14, 15]
PI number  4  :  [8, 10, 12, 14]
PI number  5  :  [12, 13, 14, 15]
checking : 0: x  2: x  5: x  6: x  7: x  8: x  10: x  12: x  13: x  14: x  15: x
    0   2   5   6   7   8   10  12  13  14  15
pi0 V   V               V   V
pi1     V       V           V           V
pi2         V       V               V       V
pi3             V   V                   V   V
pi4                     V   V   V       V
pi5                             V   V   V   V

 1 번째 실행
column_dominance :  0 in PI_0 : [0, 2, 8, 10]
column_dominance :  5 in PI_2 : [5, 7, 13, 15]
[[], [6, 14], [], [6, 14], [12, 14], [12, 14]]
    0   2   5   6   7   8   10  12  13  14  15
pi0
pi1             V                       V
pi2
pi3             V                       V
pi4                             V       V
pi5                             V       V
Row_dominance : PI 3 < PI 1
Row_dominance : PI 5 < PI 4
checking : 0: o  2: o  5: o  6: x  7: o  8: o  10: o  12: x  13: o  14: x  15: o
    0   2   5   6   7   8   10  12  13  14  15
pi0
pi1             V                       V
pi2
pi3
pi4                             V       V
pi5

 2 번째 실행
column_dominance :  6 in PI_1 : [6, 14]
column_dominance :  12 in PI_4 : [12, 14]
[[], [], [], [], [], []]
    0   2   5   6   7   8   10  12  13  14  15
pi0
pi1
pi2
pi3
pi4
pi5
checking : 0: o  2: o  5: o  6: o  7: o  8: o  10: o  12: o  13: o  14: o  15: o
    0   2   5   6   7   8   10  12  13  14  15
pi0
pi1
pi2
pi3
pi4
pi5

Process finished with exit code 0
```

5. def petrick

```
def petrick(sstr):
    for i in minterm:
        if(check[i]==1):
            sstr+='('
            for j in range(len(PI_list)):
                if(i in PI_list[j]):
                    sstr+="PI_"
                    sstr+=str(j)
                    sstr+=' + '
            sstr=sstr[:len(sstr)-3]
            sstr+=')'
    return sstr
```

시작 조건 : CD, RD 를 진행 했는데 바뀐게 없고, 아직 모든 minterm 이 check 가 되지 않았을때 시작한다.

방법 : check 가 되지 않는 minterm 마다 이 minterm 을 가지고 있는 PI 들을 더해주고 묶어준다. Ex(PI3 + PI5) 그리고 이렇게 나온 묶음들을 다 곱해준다. EX.(PI3+PI5)(PI6+PI8)

ex. 3bit, 6 개의 minterm(0,1,2,5,6,7)이 주어졌을때의 결과이다.

```
SETUP
PI number  0  :  [0, 1]
PI number  1  :  [0, 2]
PI number  2  :  [1, 5]
PI number  3  :  [2, 6]
PI number  4  :  [5, 7]
PI number  5  :  [6, 7]
checking : 0: x  1: x  2: x  5: x  6: x  7: x
    0   1   2   5   6   7
pi0 V   V
pi1 V       V
pi2     V       V
pi3         V       V
pi4             V       V
pi5                 V   V

 1 번째 실행
[[0, 1], [0, 2], [1, 5], [2, 6], [5, 7], [6, 7]]
    0   1   2   5   6   7
pi0 V   V
pi1 V       V
pi2     V       V
pi3         V       V
pi4             V       V
pi5                 V   V
checking : 0: x  1: x  2: x  5: x  6: x  7: x
    0   1   2   5   6   7
pi0 V   V
pi1 V       V
pi2     V       V
pi3         V       V
pi4             V       V
pi5                 V   V

Petric
PI number  0  :  [0, 1]
PI number  1  :  [0, 2]
PI number  2  :  [1, 5]
PI number  3  :  [2, 6]
PI number  4  :  [5, 7]
PI number  5  :  [6, 7]
Petrick method :
(PI_0 + PI_1)(PI_0 + PI_2)(PI_1 + PI_3)(PI_2 + PI_4)(PI_3 + PI_5)(PI_4 + PI_5)

Process finished with exit code 0
```

←CD 를 했는데 바뀐게 없음

←RD 를 했는데 바뀐게 없음(둘다 바뀐게 없으니 탈출)

while 문이 종료됐는데, 아직 minterm 이 다 checking 되지 않았으니 def petrick 로 진입

6. def printt

```
def printt():
    table = [[0 for i in range(max1+1)] for j in range(len(PI_list))]
    for i in range(len(PI_list)):
        for j in PI_list[i]:
            table[i][j]=1
    print("\t",end='')
    for i in minterm: print(i,end='\t')
    print()
    for i in range(len(table)):
        print("pi"+str(i),end='\t')
        for j in minterm:
            if(table[i][j]==0): print(" ",end='\t')
            else: print("V",end='\t')

        print()
```

```
     0   1   2   5   6   7
pi0  V   V
pi1  V       V
pi2      V       V
pi3          V       V
pi4              V       V
pi5                  V   V
```

현재 PI 상태가 어떤상태인지 보여주는 함수 이다. 왼쪽 그림처럼 깔끔하게 보여준다.

출력 결과물 1 : m = [4,8,2,6,8,9,10,11,14,15]

```
SETUP
PI number  0  :  [2, 6, 10, 14]
PI number  1  :  [8, 9, 10, 11]
PI number  2  :  [10, 11, 14, 15]
checking : 2: x  6: x  8: x  9: x  10: x  11: x  14: x  15: x
    2   6   8   9   10  11  14  15
pi0 V   V           V       V
pi1         V   V   V   V
pi2                 V   V   V   V

 1 번째 실행
column_dominance :  2 in PI_0 : [2, 6, 10, 14]
column_dominance :  8 in PI_1 : [8, 9, 10, 11]
column_dominance :  15 in PI_2 : [10, 11, 14, 15]
[[], [], []]
    2   6   8   9   10  11  14  15
pi0
pi1
pi2
checking : 2: o  6: o  8: o  9: o  10: o  11: o  14: o  15: o
    2   6   8   9   10  11  14  15
pi0
pi1
pi2

Process finished with exit code 0
```

출력 결과물 2-1 : m=[4,13,0,2,3,4,5,6,7,8,9,10,11,12,13]

```
SETUP
PI number  0  :  [0, 2, 4, 6]
PI number  1  :  [0, 2, 4, 6, 8, 10]
PI number  2  :  [0, 2, 4, 6, 8, 12]
PI number  3  :  [2, 3, 6, 7]
PI number  4  :  [2, 3, 6, 7, 10, 11]
PI number  5  :  [4, 5, 6, 7]
PI number  6  :  [4, 5, 6, 7, 12, 13]
PI number  7  :  [8, 9, 10, 11]
PI number  8  :  [8, 9, 10, 11, 12, 13]
checking : 0: x  2: x  3: x  4: x  5: x  6: x  7: x  8: x  9: x  10: x  11: x  12: x  13: x
    0   2   3   4   5   6   7   8   9   10  11  12  13
pi0 V   V       V       V
pi1 V   V       V       V       V       V
pi2 V   V       V       V       V               V
pi3     V   V           V   V
pi4     V   V           V   V           V   V
pi5             V   V   V   V
pi6             V   V   V   V                   V   V
pi7                             V   V   V   V
pi8                             V   V   V   V   V   V

 1 번째 실행
    0   2   3   4   5   6   7   8   9   10  11  12  13
pi0 V   V       V       V
pi1 V   V       V       V       V       V
pi2 V   V       V       V       V               V
pi3     V   V           V   V
pi4     V   V           V   V           V   V
pi5             V   V   V   V
pi6             V   V   V   V                   V   V
pi7                             V   V   V   V
pi8                             V   V   V   V   V   V
Row_dominance : PI 0 < PI 1
Row_dominance : PI 3 < PI 4
Row_dominance : PI 5 < PI 6
Row_dominance : PI 7 < PI 8
checking : 0: x  2: x  3: x  4: x  5: x  6: x  7: x  8: x  9: x  10: x  11: x  12: x  13: x
    0   2   3   4   5   6   7   8   9   10  11  12  13
pi0
pi1 V   V       V       V       V       V
pi2 V   V       V       V       V               V
pi3
pi4     V   V           V   V           V   V
pi5
pi6             V   V   V   V                   V   V
pi7
pi8                             V   V   V   V   V   V
```

출력 결과물 2-2 : m=[4,13,0,2,3,4,5,6,7,8,9,10,11,12,13]

```
 2 번째 실행
column_dominance :  3 in PI_4 : [2, 3, 6, 7, 10, 11]
column_dominance :  5 in PI_6 : [4, 5, 6, 7, 12, 13]
column_dominance :  9 in PI_8 : [8, 9, 10, 11, 12, 13]
    0   2   3   4   5   6   7   8   9   10  11  12  13
pi0
pi1 V
pi2 V
pi3
pi4
pi5
pi6
pi7
pi8
Row_dominance : PI 2 < PI 1
checking : 0: x  2: o  3: o  4: o  5: o  6: o  7: o  8: o  9: o  10: o  11: o  12: o  13: o
    0   2   3   4   5   6   7   8   9   10  11  12  13
pi0
pi1 V
pi2
pi3
pi4
pi5
pi6
pi7
pi8

 3 번째 실행
column_dominance :  0 in PI_1 : [0]
    0   2   3   4   5   6   7   8   9   10  11  12  13
pi0
pi1
pi2
pi3
pi4
pi5
pi6
pi7
pi8
checking : 0: o  2: o  3: o  4: o  5: o  6: o  7: o  8: o  9: o  10: o  11: o  12: o  13: o
    0   2   3   4   5   6   7   8   9   10  11  12  13
pi0
pi1
pi2
pi3
pi4
pi5
pi6
pi7
pi8

Process finished with exit code 0
```

출력 결과물 3 : m=[4,9,1,2,3,7,9,10,11,13,15]

```
SETUP
PI number  0  :  [1, 3, 9, 11]
PI number  1  :  [2, 3, 10, 11]
PI number  2  :  [3, 7, 11, 15]
PI number  3  :  [9, 11, 13, 15]
checking : 1: x  2: x  3: x  7: x  9: x  10: x  11: x  13: x  15: x
    1   2   3   7   9   10  11  13  15
pi0 V       V       V       V
pi1     V   V           V   V
pi2         V   V           V       V
pi3                 V       V   V   V

 1 번째 실행
column_dominance :  1 in PI_0 : [1, 3, 9, 11]
column_dominance :  2 in PI_1 : [2, 3, 10, 11]
column_dominance :  7 in PI_2 : [3, 7, 11, 15]
column_dominance :  13 in PI_3 : [9, 11, 13, 15]
    1   2   3   7   9   10  11  13  15
pi0
pi1
pi2
pi3
checking : 1: o  2: o  3: o  7: o  9: o  10: o  11: o  13: o  15: o
    1   2   3   7   9   10  11  13  15
pi0
pi1
pi2
pi3

Process finished with exit code 0
```

출력 결과물 4-1 : 교수님 스프레드 시트 예제

PI_list=[[3,4,5,11],[4,5,6,7],[4,6,9,10],[4,5,8,9],
[1,4,6,11],[1,2,6,10],[0,3,5,8],[2],[0]]

m=[4,12,0,1,2,3,4,5,6,7,8,9,10,11]

```
SETUP
PI number  0  :  [3, 4, 5, 11]
PI number  1  :  [4, 5, 6, 7]
PI number  2  :  [4, 6, 9, 10]
PI number  3  :  [4, 5, 8, 9]
PI number  4  :  [1, 4, 6, 11]
PI number  5  :  [1, 2, 6, 10]
PI number  6  :  [0, 3, 5, 8]
PI number  7  :  [2]
PI number  8  :  [0]
checking : 0: x  1: x  2: x  3: x  4: x  5: x  6: x  7: x  8: x  9: x  10: x  11: x
    0   1   2   3   4   5   6   7   8   9   10  11
pi0             V   V   V                       V
pi1                 V   V   V   V
pi2                 V       V           V   V
pi3                 V   V           V   V
pi4     V           V       V                   V
pi5     V   V               V               V
pi6 V           V       V           V
pi7         V
pi8 V

 1 번째 실행
column_dominance :  7 in PI_1 : [4, 5, 6, 7]
    0   1   2   3   4   5   6   7   8   9   10  11
pi0             V                               V
pi1
pi2                                     V   V
pi3                                 V   V
pi4     V                                       V
pi5     V   V                               V
pi6 V           V                   V
pi7         V
pi8 V
Row_dominance : PI 7 < PI 5
Row_dominance : PI 8 < PI 6
checking : 0: x  1: x  2: x  3: x  4: o  5: o  6: o  7: o  8: x  9: x  10: x  11: x
    0   1   2   3   4   5   6   7   8   9   10  11
pi0             V                               V
pi1
pi2                                     V   V
pi3                                 V   V
pi4     V                                       V
pi5     V   V                               V
pi6 V           V                   V
pi7
pi8
```

출력 결과물 4-2 : 교수님 스프레드 시트 예제

```
 2 번째 실행
column_dominance :  0 in PI_6 : [0, 3, 8]
column_dominance :  2 in PI_5 : [1, 2, 10]
    0   1   2   3   4   5   6   7   8   9   10  11
pi0                                         V
pi1
pi2                                 V
pi3                                 V
pi4                                         V
pi5
pi6
pi7
pi8
Row_dominance : PI 4 < PI 0
Row_dominance : PI 3 < PI 2
checking : 0: o  1: o  2: o  3: o  4: o  5: o  6: o  7: o  8: o  9: x  10: o  11: x
    0   1   2   3   4   5   6   7   8   9   10  11
pi0                                         V
pi1
pi2                                 V
pi3
pi4
pi5
pi6
pi7
pi8

 3 번째 실행
column_dominance :  9 in PI_2 : [9]
column_dominance :  11 in PI_0 : [11]
    0   1   2   3   4   5   6   7   8   9   10  11
pi0
pi1
pi2
pi3
pi4
pi5
pi6
pi7
pi8
checking : 0: o  1: o  2: o  3: o  4: o  5: o  6: o  7: o  8: o  9: o  10: o  11: o
    0   1   2   3   4   5   6   7   8   9   10  11
pi0
pi1
pi2
pi3
pi4
pi5
pi6
pi7
pi8

Process finished with exit code 0
```