Chapter 3 : Applications of Boolean Algebra

Design Applications

1 . **You receive a 5-bit signed binary number ABCDE corresponding to the temperature of a freezer in degrees Celsius. If the temperature rises above −14 ◦C, the compressor should turn on. If the temperature rises above −5 ◦C, a warning light should turn on. Define both output variables and express them as minimum SOP functions.**

Compressor Bang-Bang control: 0 means that the temperature is below -14C and the compressor is off, 1 means temperature is above -14 C and the compressor is on.

Temperature Warning sensor: 0 means that the temperature is below -5C, 1 means temperature is above -5 C

In [3]:
```python
from IPython.display import Markdown
import ttg
import pandas as pd
def tt(args_list,exprs = None):
    df =  (ttg.Truths(args_list,exprs).as_pandas())[::-1]
    df = df.reset_index(drop = True)
    return df

fn_columns = pd.DataFrame({
    'temperature' : [x for x in (range(0,16))] +  [x for x in range(-16,0)],
    })

df  = tt(['A', 'B', 'C', 'D', 'E'],
   ['not A or (B and C) or (B and D and E)',
   'not (A and not B and not C and not D)',
   ])

pd.set_option('display.max_rows', 500)
df = pd.concat([df,fn_columns ],axis = 1)
df.index = df.temperature
df = df.drop('temperature',axis = 1)
df =df.rename_axis('temperature')
display ( Markdown('**warning light function**: A\' + BC + BDE'))
display ( Markdown("**compressor control function**: (AB\'C\'D\')'"))
df
```

warning light function: A' + BC + BDE

compressor control function: (AB'C'D')'

Out[3]:

temperature	A	B	C	D	E	not A or (B and C) or (B and D and E)	not (A and not B and not C and not D)
0	0	0	0	0	0	1	1
1	0	0	0	0	1	1	1
2	0	0	0	1	0	1	1
3	0	0	0	1	1	1	1
4	0	0	1	0	0	1	1
5	0	0	1	0	1	1	1
6	0	0	1	1	0	1	1
7	0	0	1	1	1	1	1
8	0	1	0	0	0	1	1
9	0	1	0	0	1	1	1
10	0	1	0	1	0	1	1
11	0	1	0	1	1	1	1
12	0	1	1	0	0	1	1
13	0	1	1	0	1	1	1
14	0	1	1	1	0	1	1
15	0	1	1	1	1	1	1
-16	1	0	0	0	0	0	0
-15	1	0	0	0	1	0	0
-14	1	0	0	1	0	0	1
-13	1	0	0	1	1	0	1
-12	1	0	1	0	0	0	1
-11	1	0	1	0	1	0	1
-10	1	0	1	1	0	0	1
-9	1	0	1	1	1	0	1
-8	1	1	0	0	0	0	1
-7	1	1	0	0	1	0	1
-6	1	1	0	1	0	0	1
-5	1	1	0	1	1	1	1
-4	1	1	1	0	0	1	1
-3	1	1	1	0	1	1	1
-2	1	1	1	1	0	1	1
-1	1	1	1	1	1	1	1

2. **You receive two 2-bit numbers designated as AB and CD.\

If AB > CD, a red LED should turn on with all other LEDs off.
If AB < CD, a green LED should turn on with all other LEDs off.\ If AB = CD, a blue LED should turn on with all other LEDs off.
Derive minimum SOP expressions to control each LED. Assume that a value of zero corresponds to an LED that is off, and a value of one corresponds to an LED that is on**

In [76]:
```python
from IPython.display import Markdown

df  = tt(['A', 'B', 'C', 'D'])

pd.set_option('display.max_rows', 500)
                    # 0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5
df["RED_LED : AC\' + BC\'D\' + ABD\'"]   = [0,0,0,0,1,0,0,0,1,1,0,0,1,1,1,0]
df["GREEN_LED :  A'C + A'B'D + B'CD "] = [0,1,1,1,0,0,1,1,0,0,0,1,0,0,0,0]
df["BLUE_LED :  A'B'C'D' + A'BC'D + AB'CD' + ABCD"]  = [1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1]
df
```

Out[76]:

	A	B	C	D	RED_LED : AC' + BC'D' + ABD'	GREEN_LED : A'C + A'B'D + B'CD	BLUE_LED : A'B'C'D' + A'BC'D + AB'CD' + ABCD
0	0	0	0	0	0	0	1
1	0	0	0	1	0	1	0
2	0	0	1	0	0	1	0
3	0	0	1	1	0	1	0
4	0	1	0	0	1	0	0
5	0	1	0	1	0	0	1
6	0	1	1	0	0	1	0
7	0	1	1	1	0	1	0
8	1	0	0	0	1	0	0
9	1	0	0	1	1	0	0
10	1	0	1	0	0	0	1
11	1	0	1	1	0	1	0
12	1	1	0	0	1	0	0
13	1	1	0	1	1	0	0
14	1	1	1	0	1	0	0
15	1	1	1	1	0	0	1

3.

Create a programmable logic gate with inputs A and B, control bits X and Y , and output F.

If X = Y = 0, then F = AB.

If X = Y = 1, then F = A + B.

If X = 1 and Y = 0, then F = 0.

Anyother combination of X and Y will never occur. Derive a minimum SOP expression for F.

Minterm Expressions

1. **Find a minimum Boolean expression (either SOP or POS) for $F(A, B, C) = \Sigma m(0, 1, 5, 6)$**

In [77]:
```python
pd.concat([tt(["A","B","C"]),pd.DataFrame({"F" : [1,1,0,0,0,1,1,0]})],axis = 1)
```

Out[77]:

	A	B	C	F
0	0	0	0	1
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

AB/C kmap

11
00
10
01

$F = A'B' + B'C' + ABC'$

2. **Find a minimum Boolean expression (either SOP or POS) for:**

- $F(A, B, C, D) = \Sigma m(0, 8, 9, 13, 15) + \Sigma d(0, 1, 5, 6)$

In [78]:
```python
                                                    #0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5
pd.concat([tt(["A","B","C","D"]),pd.DataFrame({"F" : [0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1]})],axis = 1)
```

Out[78]:

	A	B	C	D	F
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	1

AB/CD Kmap

```
x       x       0       0
0       x       0       x
0       1       1       0
1       1       0       0
```

$F = B'C'D' + AB'C' + ABD$(NOT CONSIDERING DON'T CARES)
$F = B'C' + ABD$ (CONSIDERING DON'T CARES)

3. **Find a minimum SOP expression for:**
   - $F(A, B, C) = \Sigma m(0, 1, 3, 4, 5)$

In [79]:
```python
pd.concat([tt(["A","B","C"],[" (not A and C) or (not B)"]),pd.DataFrame({"F" : [1,1,0,1,1,1,0,0]})],axis = 1)
```

Out[79]:

	A	B	C	(not A and C) or (not B)	F
0	0	0	0	1	1
1	0	0	1	1	1
2	0	1	0	0	0
3	0	1	1	1	1
4	1	0	0	1	1
5	1	0	1	1	1
6	1	1	0	0	0
7	1	1	1	0	0

AB/C KMAP

```
1       1
0       1
0       0
1       1
```

$F = AB' + A'C + A'B'$

$F = A'C + (B'(A' + A))$

$F = A'C + B'$

Maxterm Expressions

1. Find a minimum SOP expression for $F(A, B, C) = \Pi M(2, 3, 5, 7)$

In [80]:
```python
pd.concat([tt(["A","B","C"]),pd.DataFrame({"F" : [0,0,1,1,0,1,0,1] })],axis = 1)
```

Out[80]:

	A	B	C	F
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

Maxterms : $(A + B' + C)(A + B' + C')(A' + B + C')(A' + B' + C')$

SOP : $AC' + A'B'$

2. Find a minimum SOP expression for $F(A, B, C) = \Pi M(2, 3, 5)\Pi D(0, 7)$

Indicate which (if any) of the don't care terms you use to minimize the expression

In [81]:
```python
pd.concat([tt(["A","B","C"],["(A and not C) or (not A and not B)"]),pd.DataFrame({"F\'" : [0,0,1,1,0,1,0,0] })]
```

Out[81]:

	A	B	C	(A and not C) or (not A and not B)	F'
0	0	0	0	1	0
1	0	0	1	1	0
2	0	1	0	0	1
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	1	0
7	1	1	1	0	0

Maxterms : $(A + B' + C)(A + B' + C')(A' + B + C')$

$\Pi M(2, 3, 5)\Pi D(0, 7)$

$\Sigma m(1, 4, 6) + \Sigma d(0, 7)$

AB/C kmap

```
x       1
0       0
1       x
1       0
```

F = AC' + A'B'

I used the 0 don't care

3. Find a minimum SOP expression for $\Pi M(1, 2, 14)\Pi D(0, 3, 8, 15)$

Indicate which (if any) of the don't care terms you use to minimize the expression

In [82]:
```python
pd.concat([tt(["A","B","C","D"]),pd.DataFrame({})],axis = 1)
```

Out[82]:

	A	B	C	D
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

AB/CD KMAP

```
x       1       x       1
0       0       0       0
0       0       x       1
x       0       0       0
```

$F = A'B' + ABC$

I used all of the don't care terms except 8.

Minterm and Maxterm Expansions

1. Express $F = (A + C)D + ABCD$ as a minterm expansion of four variables.

In [12]:
```python
tt(["A","B","C","D"],["((A or C) and D) or (A and B and C and D)"])
```

Out[12]:

	A	B	C	D	((A or C) and D) or (A and B and C and D)
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	1

AB/CD KMAP

```
0       0       1       0
0       0       1       0
0       1       1       0
0       1       1       0
```

minterms = A'B'CD + A'BCD + AB'C'D + AB'CD+ ABC'D + ABCD

2. Express $F = AB + BD + ABC$ as a minterm expansion of four variables.

In [14]:
```python
tt(["A","B","C","D"],["(A and B) or (B and D) or (A and B and C)"])
```

Out[14]:

	A	B	C	D	(A and B) or (B and D) or (A and B and C)
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	1

AB/CD KMAP

```
0       0       0       0
0       1       1       0
1       1       1       1
0       0       0       0
```

minterms = A'BC'D + A'BCD + ABC'D' + ABC'D + ABCD' +ABCD

3. Express $F = A(B + C)(C + D)$ as a maxterm expansion of four variables.

In [17]:
```python
tt(["A","B","C","D"],["A and (B or C) and (C or D)"])
```

Out[17]:

	A	B	C	D	A and (B or C) and (C or D)
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	1

$F = \Pi M(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 12) = (A' + B' + C' + D')(A' + B' + C' + D)(A' + B' + C + D')(A' + B' + C + D)(A + B' + C + D)(A' + B + C' + D$

In [ ]:

In [ ]: