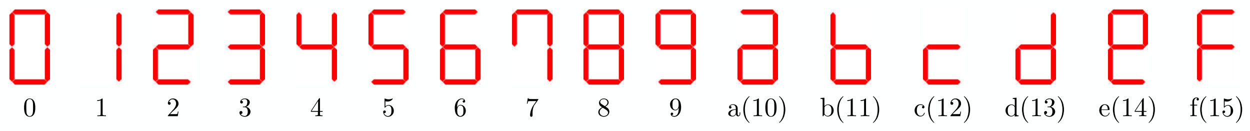
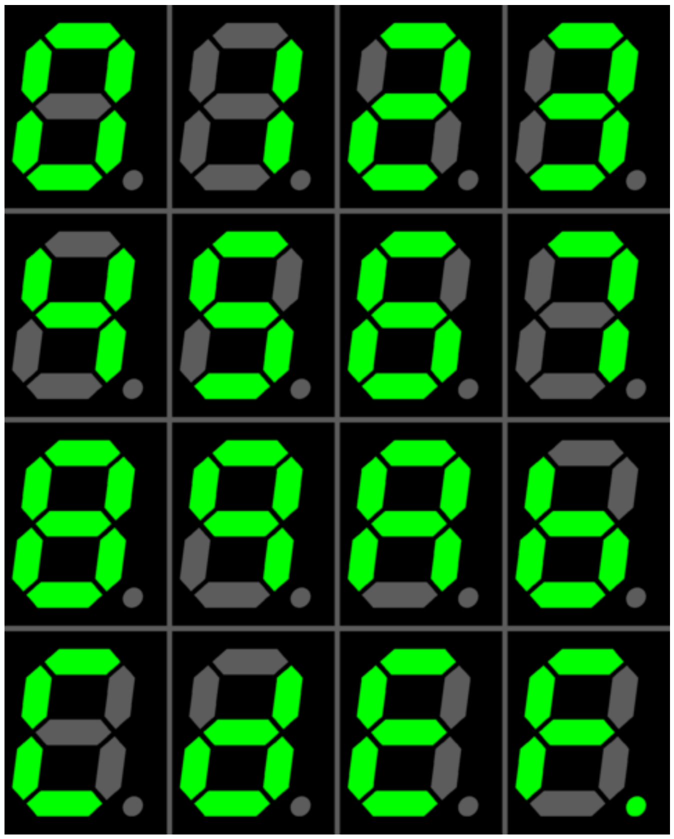
Project #1

April 1, 2021

- Due: 11:59pm 11 Apr.
- Goal: To build the 7-segment display circuit.
- Build the circuit using the `HardwareSimulator`. You can download the necessary files (`hdlempty.zip`) from MS Teams.
  - `Encoder16.tst`: Test script. DO NOT modify this file.
  - `Encoder16.cmp`: Correct result file. DO NOT modify this file.
  - `Encoder16.hdl`: Complete the circuit.
  - `Decoder7Seg.tst`: Test script. DO NOT modify this file.
  - `Decoder7Seg.cmp`: Modify the output by setting all `X`'s to correct values (`1` or `0`) which is the same as the truth table of (Table 2).
    (NOTE: You should modify only the `X` letters not other characters including white spaces. Otherwise you may get an error.)
  - `Decoder7Seg.hdl`: Complete the circuit.
  - `Seg7.tst`: DO NOT modify this file.
  - `Seg7.cmp`: Modify the output of the file which should be the same as the output of `Decoder7Seg.cmp`.
    (NOTE: You should modify only the `X` letters not other characters including white spaces. Otherwise you may get an error.)
  - `Seg7.hdl`: DO NOT modify this file.
- `Seg7.hdl`
  - input: 16 buttons (`in[0]`–`in[15]`) associated with numbers 0–15
  - output: a 7-segment display components (`out[0]`–`out[6]`) displaying the corresponding input (hexadecimal) numbers below. (Number 1 is aligned on the right.)



  - Note that our display is different from the typical hexadecimal display, e.g., (image courtesy of Wikipedia.)



- The circuit MUST be composed of two parts:
  - 16-to-4 encoder (`Encoder16.hdl`)
  - 7-segment decoder (`Decoder7Seg.hdl`)

- `Encoder16.hdl`
  - The truth table for the 16-to-4 encoder:

							in										out		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

  - For this circuit, you do not need to take care of invalid inputs not listed in the truth table.

* Analysis

in																out			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

⇒ 0이 많은 sparse의 형태.

in[2] in[4] ··· in[14] → XNOR! 굳이? OR랑 차이가 없음.

out[0] = in[1] + in[3] + in[5] + ··· + in[15]

out[1] = in[2] + in[3] + in[6] + in[7] + in[10] + in[11] + in[14] + in[15]

out[2] = in[4] + ··· + in[7] + in[12] + ··· + in[15]

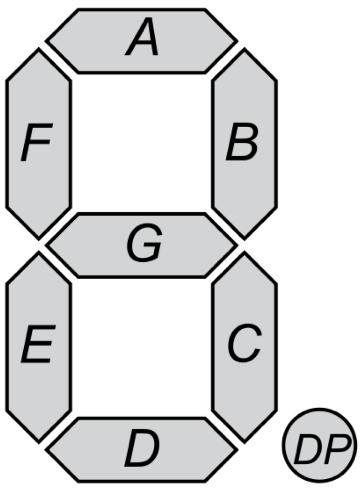
out[3] = in[8] + ··· + in[15]
in[1]

in[1] ~ in[15]는 서로 배반사건 : for i,j ∈ [0,15], i≠j

→ ∀ XNOR시 개수가 짝수여야 1.

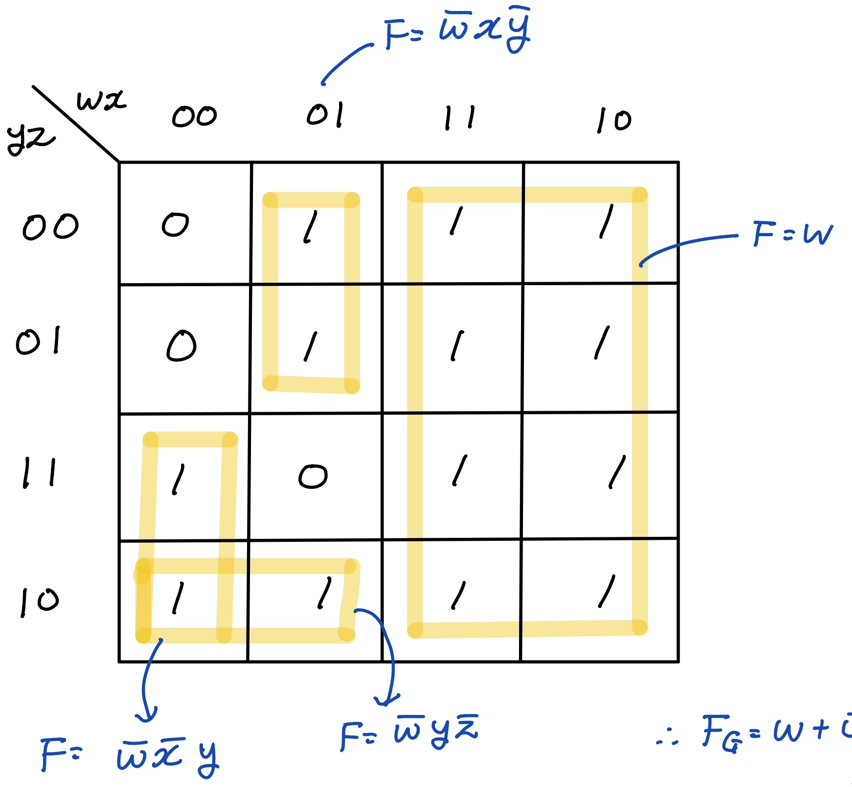
1 ③ 5 ⑦ 9 ⑪ 13 ⑮
2 ③ 6 ⑦ 10 ⑪ 14 ⑮
4 5 6 7 12 13 14 15
8 9 10 11 12 13 14 15

in				out						
3	2	1	0	6	5	4	3	2	1	0
w	x	y	z	A	B	C	D	E	F	G
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	1	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	1	1	1	1	1	0	1
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0	0	0	0	1	1	0	1
1	1	0	1	0	1	1	1	1	0	1
1	1	1	0	1	1	0	1	1	1	1
1	1	1	1	1	0	0	0	1	1	1



Ⓖ

$$\begin{pmatrix} 0000 \\ 0001 \\ 0111 \end{pmatrix}^c$$



yz \ wx	00	01	11	10
00	0	1	1	1
01	0	1	1	1
11	1	0	1	1
10	1	1	1	1

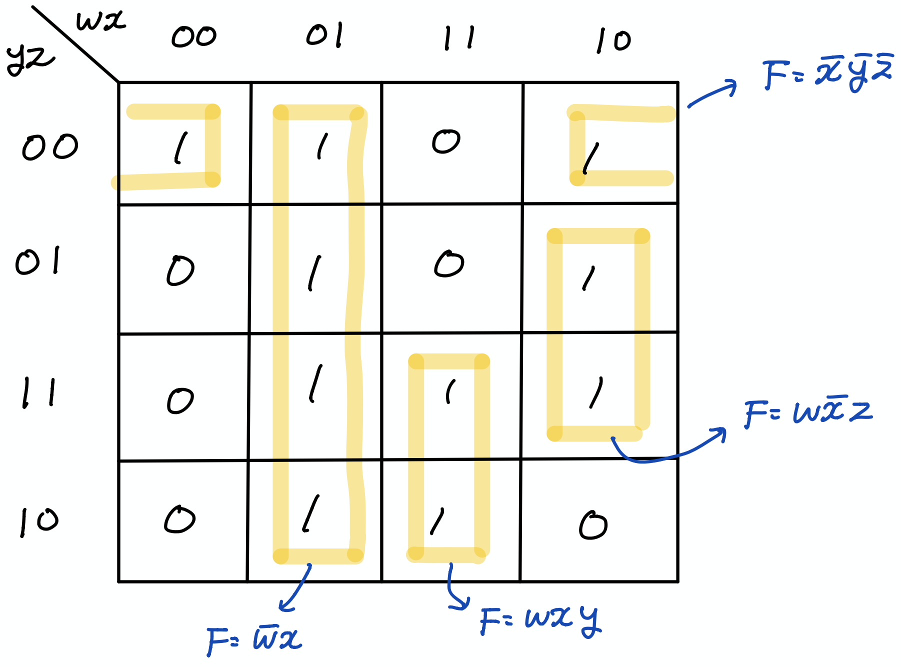
$$\therefore F_G = w + \bar{w}x\bar{y} + \bar{w}\bar{x}y + \bar{w}y\bar{z}$$

$$w + \bar{w}\cdot(x \otimes y) + \bar{w}y\bar{z}$$

① ② ③ ④ ⑤ ⑥

Ⓕ

$$\begin{pmatrix} 0001 \\ 0010 \\ 0011 \\ 1010 \\ 1100 \\ 1101 \end{pmatrix}^c$$



yz \ wx	00	01	11	10
00	1	1	0	1
01	0	1	0	1
11	0	1	1	1
10	0	1	1	0

$$\therefore F_F = \bar{w}x + wxy + w\bar{x}z + \bar{x}\bar{y}\bar{z}$$

$$= x(\bar{w} + wy) + w\bar{x}z + \bar{x}\bar{y}\bar{z}$$

① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨

Ⓔ

$$\begin{pmatrix} 0001 \\ 0011 \\ 0100 \\ 0101 \\ 0111 \\ 1001 \end{pmatrix}^{c}$$

yz \ wx	00	01	11	10
00	1	0	1	1
01	0	0	1	0
11	0	0	1	1
10	1	1	1	1

$F = \bar{x}\bar{y}\bar{z}$

$F = \bar{w}y\bar{z}$

$F = wx$

$F = w\bar{x}y$

$\therefore F_E = wx + w\bar{x}y + \bar{w}y\bar{z} + \bar{x}\bar{y}\bar{z}$

$= \underbrace{\underbrace{\underbrace{wx}_{1} + \underbrace{y(\underbrace{\underbrace{w\bar{x}}_{2}+\underbrace{\bar{w}\bar{z}}_{3}}_{5})}_{7}}_{8} + \underbrace{\underbrace{\bar{x}\bar{y}}_{4}\bar{z}}_{6}}_{9}$

Ⓓ

$$\begin{pmatrix} 0001 \\ 0100 \\ 0111 \\ 1111 \end{pmatrix}^{c}$$

yz \ wx	00	01	11	10
00	1	0	1	1
01	0	1	1	1
11	1	0	0	1
10	1	1	1	1

$F = w\bar{y}$

$F = \bar{x}\bar{y}\bar{z}$

$F = x\bar{y}z$

$F = \bar{x}y$

$F = xy\bar{z}$

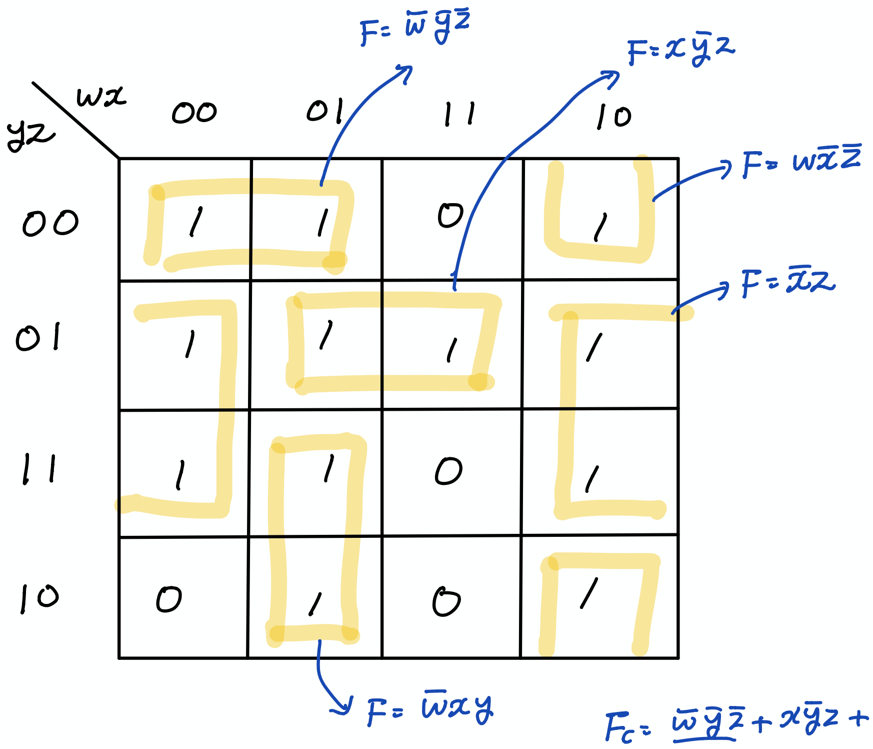
$F_D = w\bar{y} + \bar{x}\bar{y}\bar{z} + \underline{x\bar{y}z} + \bar{x}y + \underline{xy\bar{z}}$

$= w\bar{y} + \bar{x}\bar{y}\bar{z} + \bar{x}y + x(y \otimes z)$

$= \underbrace{\underbrace{\bar{y}\,(\underbrace{w + \underbrace{\bar{x}\bar{z}}_{1}}_{4})}_{6} + \underbrace{\underbrace{\bar{x}y}_{2} + \underbrace{x(\underbrace{y \otimes z}_{3})}_{5}}_{7}}_{8}$

Ⓒ

$$\begin{pmatrix} 0010 \\ 1100 \\ 1110 \\ 1111 \end{pmatrix}^{c}$$



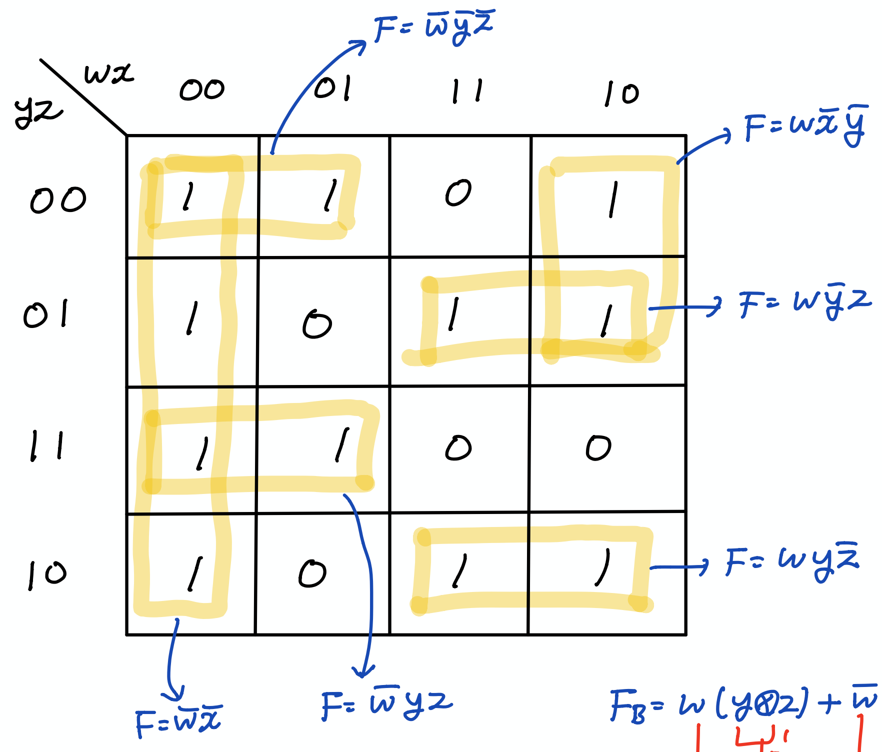
$$\begin{array}{c|cccc} yz \backslash wx & 00 & 01 & 11 & 10 \\ \hline 00 & 1 & 1 & 0 & 1 \\ 01 & 1 & 1 & 1 & 1 \\ 11 & 1 & 1 & 0 & 1 \\ 10 & 0 & 1 & 0 & 1 \end{array}$$

$$F_c = \underline{\bar w\bar y\bar z} + x\bar y z + \overline{w\bar x\bar z} + \overline{\bar x z} + \underline{\bar w x y}$$

$$= \underbrace{\underbrace{\underbrace{\bar w\,\underbrace{(\underbrace{xy}_{1}+\underbrace{\bar y\bar z}_{2})}_{5}}_{8} + \underbrace{\bar x\,\underbrace{(\underbrace{w\bar z}_{3}+z)}_{6}}_{9}}_{10} + \underbrace{\underbrace{x\bar y}_{4} z}_{7}}_{11}$$

Ⓑ

$$\begin{pmatrix} 0101 \\ 0110 \\ 1011 \\ 1100 \\ 1111 \end{pmatrix}^{c}$$



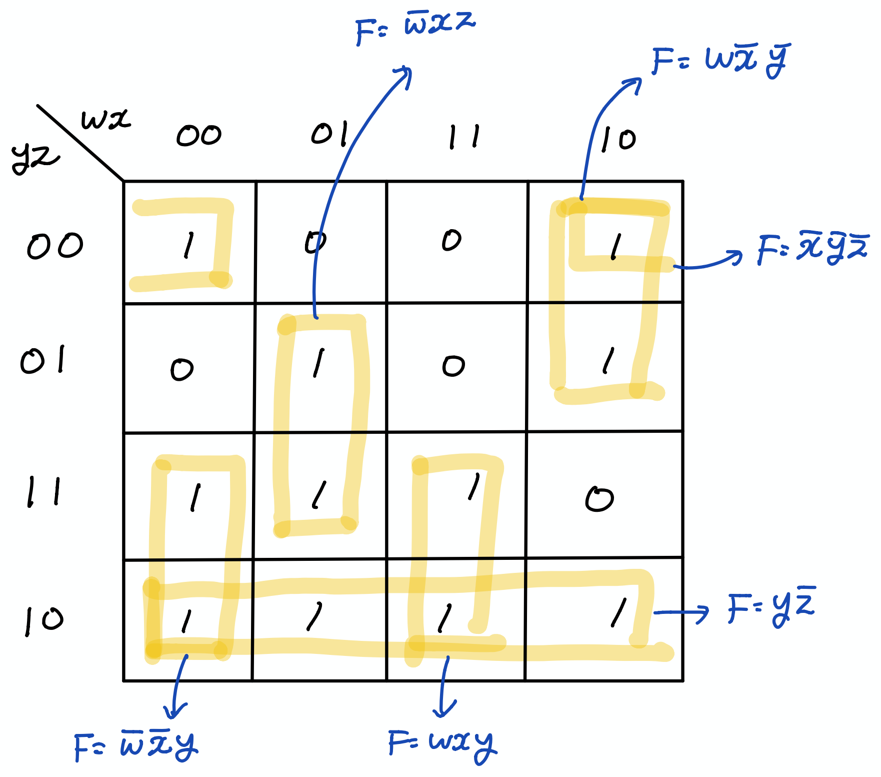
$$\begin{array}{c|cccc} yz \backslash wx & 00 & 01 & 11 & 10 \\ \hline 00 & 1 & 1 & 0 & 1 \\ 01 & 1 & 0 & 1 & 1 \\ 11 & 1 & 1 & 0 & 0 \\ 10 & 1 & 0 & 1 & 1 \end{array}$$

$$F_B = \underbrace{\underbrace{\underbrace{w\,\underbrace{(y \otimes z)}_{1}}_{5} + \underbrace{\bar w\,\underbrace{(\underbrace{yz}_{2}+\underbrace{\bar y\bar z}_{3})}_{6}}_{8}}_{10} + \bar w\bar x + w\bar x\bar y}$$

$$+ \underbrace{\underbrace{\bar x\,\underbrace{(\bar w + \underbrace{w\bar y}_{4})}_{7}}_{9}}_{11}$$

Ⓐ

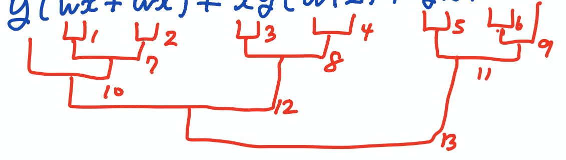
$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix} C$$



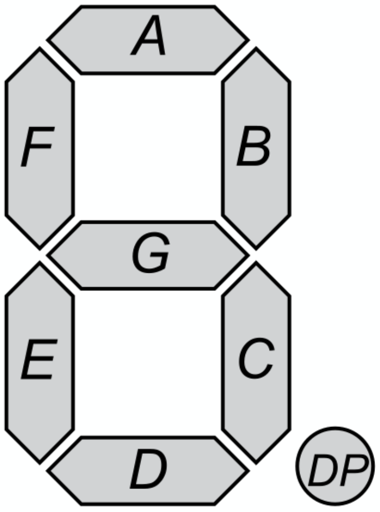
$\bar{x}\bar{y}(w+\bar{z})$

$$F_A = y(\bar{w}\bar{x} + wx) + y\bar{z} + \bar{x}\bar{y}\bar{z} + w\bar{x}\bar{y} + \bar{w}xz$$

$$= y(\bar{w}\bar{x} + wx) + \bar{x}\bar{y}(w+\bar{z}) + y\bar{z} + \bar{w}xz$$



- `Decoder7Seg.hdl`
  - The 7-segment display is composed of the following 8 segments, but we only use the 7 segments A–G below. (Do not confuse those pin names A–G with the hexadecimal numbers A–F.)



  - The truth table for the 7-segment decoder (You need to complete the table yourself in `Decoder7Seg.cmp`.) Note that `out[0]`–`out[6]` correspond with A–G above in **reverse order**. See the table below.

`in`				`out`						
`3`	`2`	`1`	`0`	`6` A	`5` B	`4` C	`3` D	`2` E	`1` F	`0` G
0	0	0	0	X	X	X	X	X	X	X
0	0	0	1	X	X	X	X	X	X	X
0	0	1	0	X	X	X	X	X	X	X
0	0	1	1	X	X	X	X	X	X	X
0	1	0	0	X	X	X	X	X	X	X
0	1	0	1	X	X	X	X	X	X	X
0	1	1	0	X	X	X	X	X	X	X
0	1	1	1	X	X	X	X	X	X	X
1	0	0	0	X	X	X	X	X	X	X
1	0	0	1	X	X	X	X	X	X	X
1	0	1	0	X	X	X	X	X	X	X
1	0	1	1	X	X	X	X	X	X	X
1	1	0	0	X	X	X	X	X	X	X
1	1	0	1	X	X	X	X	X	X	X
1	1	1	0	X	X	X	X	X	X	X
1	1	1	1	X	X	X	X	X	X	X

- Note
  - AND, OR, NOT, Xor
  - You can use any gate provided by `nand2tetris`.
  - You don't have to minimize the number of gates used. As long as it works, you're ok.