基本情報技術者対策
勉強会
コンピュータの回路を知る編

担当

長谷川　魁(はせがわ　いさむ)

目次

１．論理演算とベン図

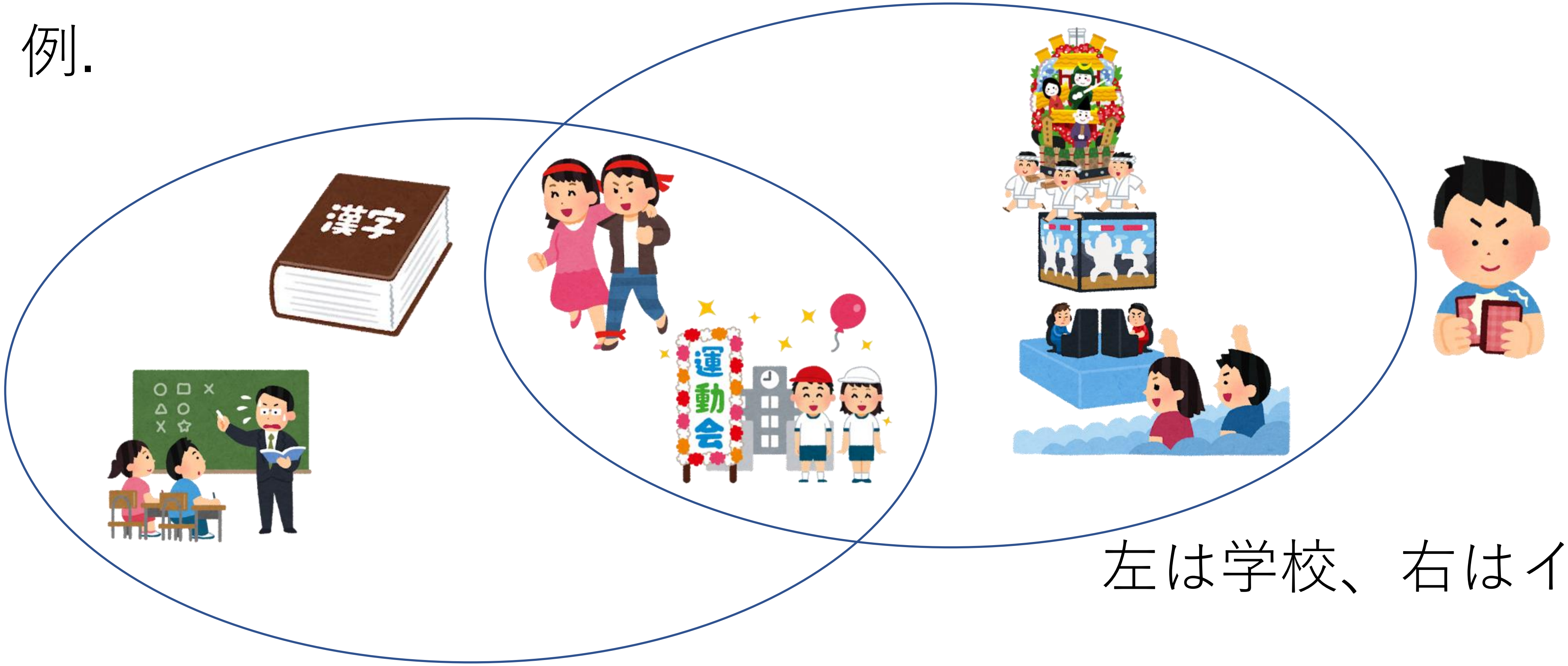
そもそも論理演算って？

AND(AかつB)、OR(AまたはB)、NOT(AはBではない)の結果を
真(条件を満たしている:TRUE)と
偽(条件を満たしていない:FALSE)
の２つの値で行う演算

ベン図って？

グループ同士の関係を図に表したもの

例.

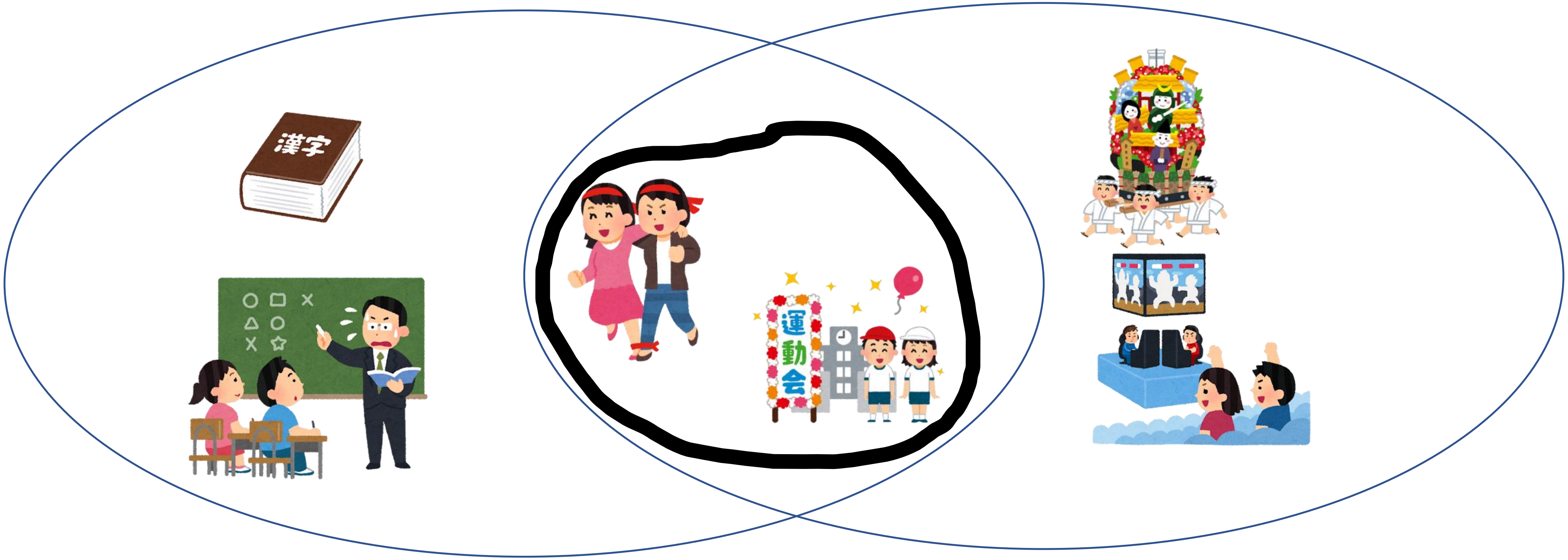


左は学校、右はイベント

論理積(AND)

２つの条件の双方に一致する場合のみ真とする

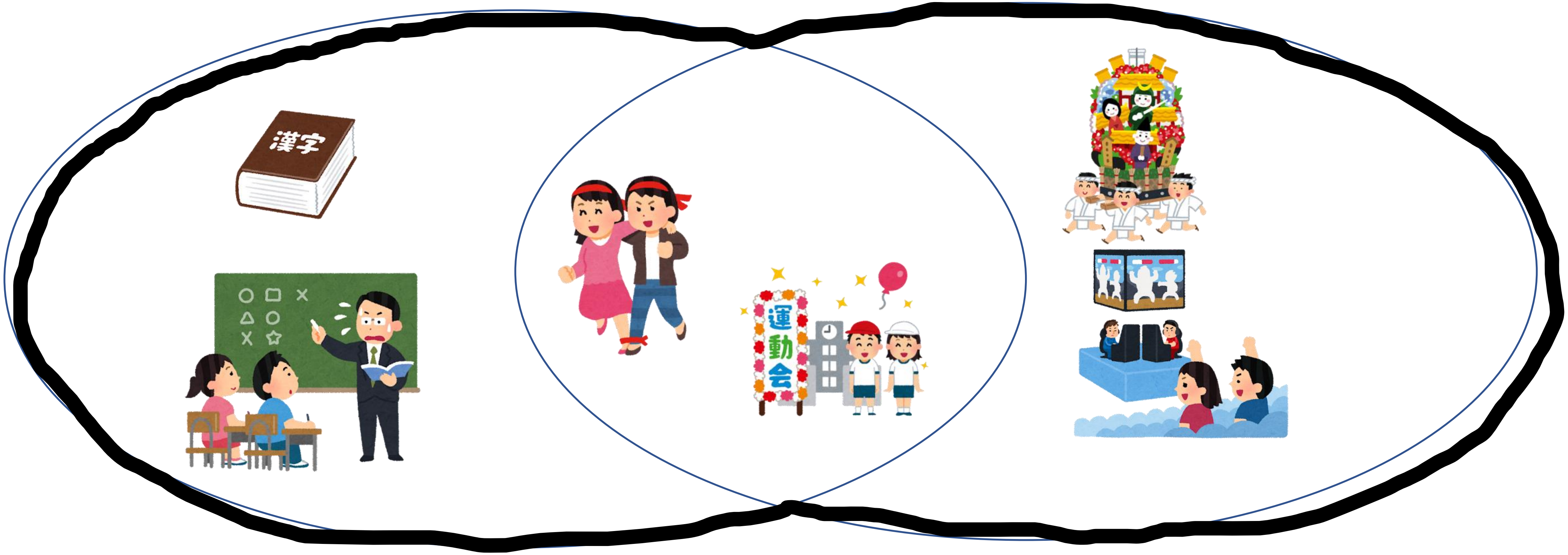
例.　下図の場合、学校かつイベントは運動会(体育祭)



論理和(OR)

２つの条件のどちらかに一致する場合、真とする

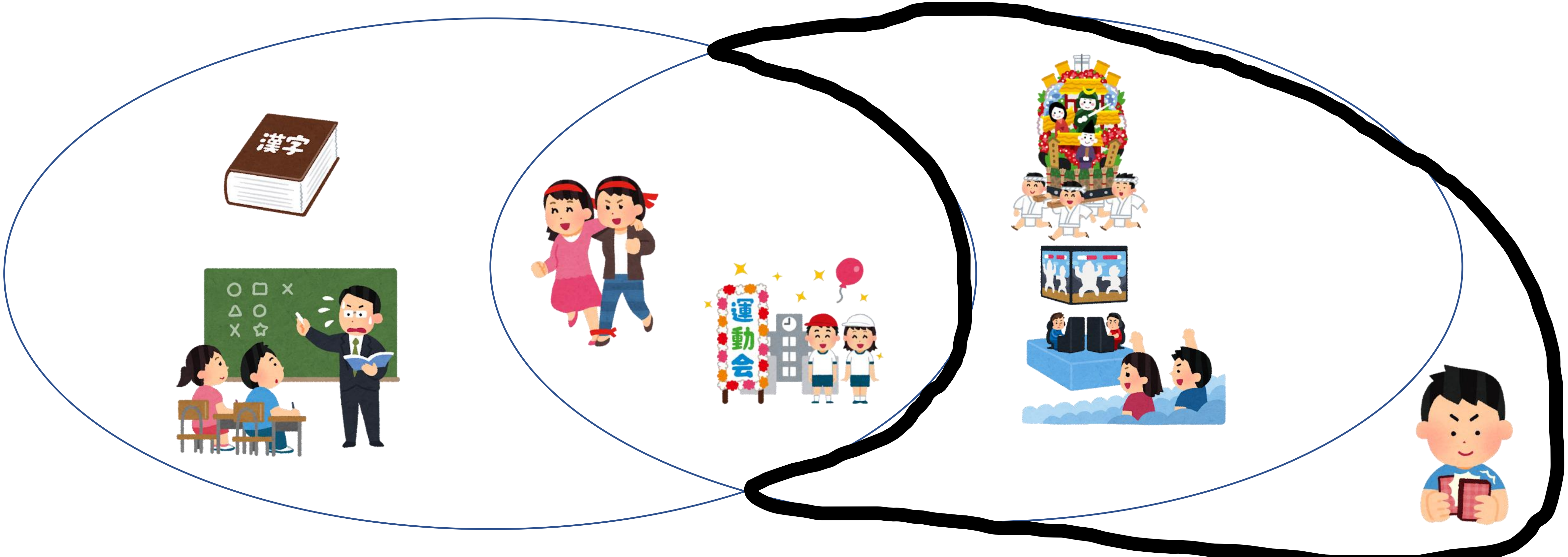
例.　下図の場合、学校かイベントのどちらか



否定(NOT)

ある条件を満たしていないものを真とする

例.　下図の場合、学校とは関係性のない(例外は考えない)もの



表にすると…

aの値	bの値	a AND bの演算結果
1(真)	1(真)	1(真)
0(偽)	1(真)	0(偽)
1(真)	0(偽)	0(偽)
0(偽)	0(偽)	0(偽)

aの値	bの値	a OR bの演算結果
1(真)	1(真)	1(真)
0(偽)	1(真)	1(真)
1(真)	0(偽)	1(真)
0(偽)	0(偽)	0(偽)

aの値	bの値	a XOR bの演算結果
1(真)	1(真)	0(偽)
0(偽)	1(真)	1(真)
1(真)	0(偽)	1(真)
0(偽)	0(偽)	0(偽)

aの値	NOT aの演算結果
1(真)	0(偽)
0(偽)	1(真)

排他的論理和(XOR)は後ほど説明

どんな風に問われるか？

問.　論理式$\overline{A}\cdot\overline{B}\cdot C+A\cdot\overline{B}\cdot C+\overline{A}\cdot B\cdot C+A\cdot B\cdot C$

と等しい物はどれか？

（・は論理積、＋は論理和、$\overline{A}$は否定を表す）

ア.$\underline{A\cdot B\cdot C}$

イ.$\underline{A\cdot B\cdot C+\overline{A}\cdot\overline{B}\cdot C}$

ウ.$\underline{A\cdot B+B\cdot C}$

エ.$\underline{C}$

解き方

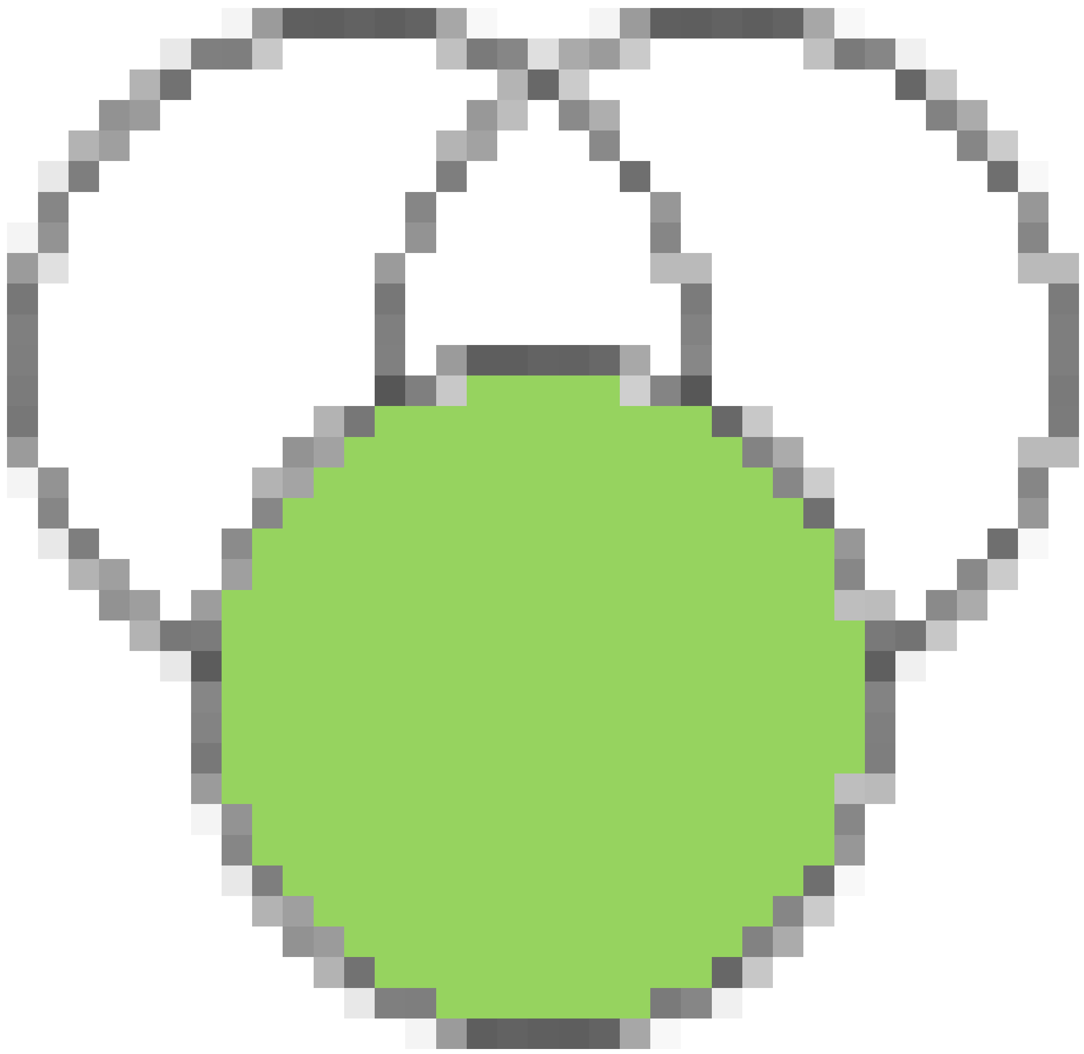
1．ベン図を用いる※左上－＞A、右上－＞B、下－＞C

2．ベン図に一つずつ埋める

~~A~~・~~B~~・C　＋　A・~~B~~・C　＋　~~A~~・B・C　＋　A・B・C

答え

エ.<u>C</u>



２．論理回路と基本回路

論理回路って？

論理演算を行う、コンピュータなどの電気回路。
電流が流れれば真、流れなければ偽と対応させる。

※Goo辞書参照

基本回路には何があるの？

- 論理積回路(AND回路)
- 論理和回路(OR回路)
- 否定回路(NOT回路)

の三つ

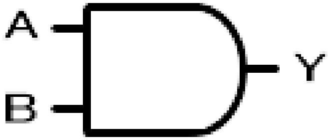
論理積回路

- 入力がどちらも真(1)なら、１を出力

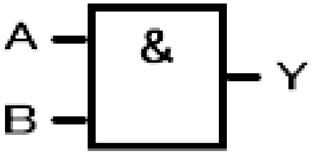
AND

論理式：Y = A・B

回路記号(MIL)



回路記号(JIS)



真理値表

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

論理和回路

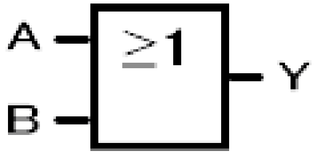
• 入力がどちらかが真(1)なら、１を出力

OR

論理式：Y = A + B

回路記号(MIL)

回路記号(JIS)



真理値表

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

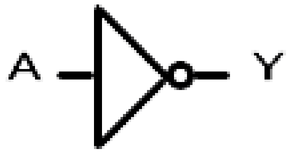
否定回路

- 入力が真(1)なら 0 を、偽(0)なら 1 を出力

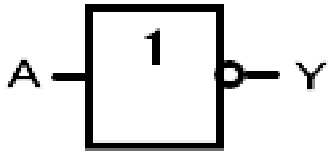
NOT

論理式：$Y = \overline{A}$

回路記号(MIL)



回路記号(JIS)



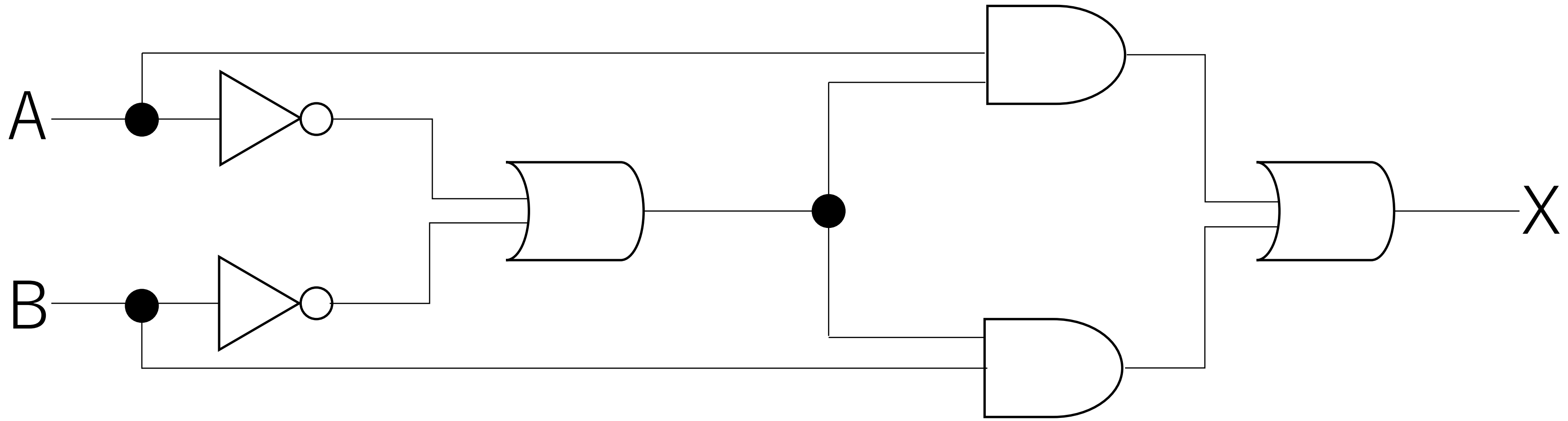
真理値表

A	Y
0	1
1	0

どんな風に問われるか？

問．図に示すディジタル回路と等価な論理式はどれか。

ここでの論理式中の“・”は論理積、“＋”は論理和、$\overline{X}$は否定とする。

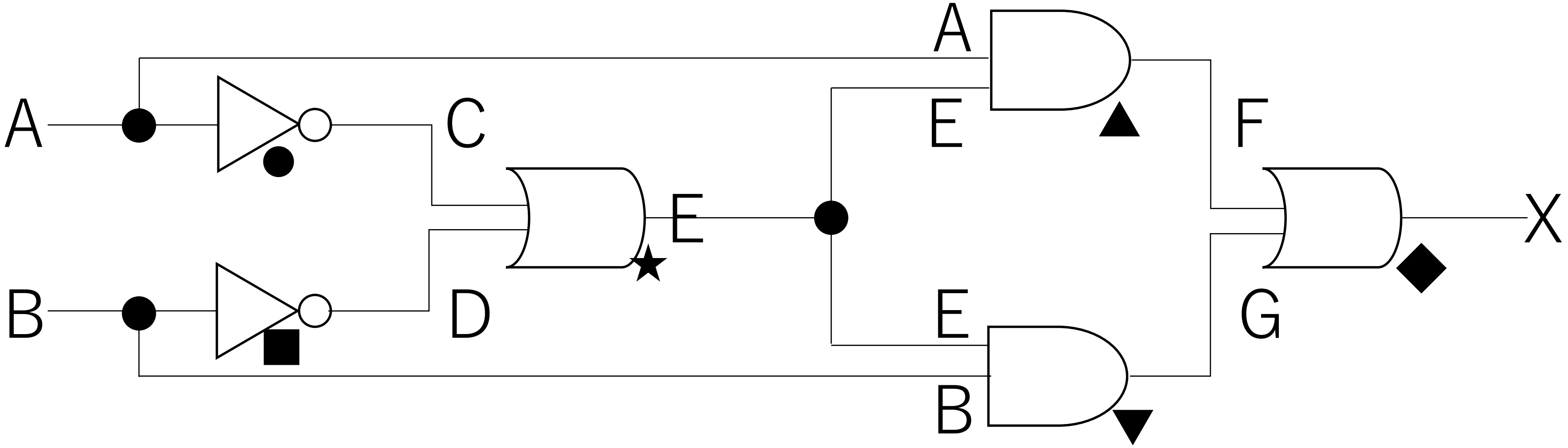


ｱ.$X = A \cdot B + \overline{A \cdot B}$　ｲ.$X = A \cdot B + \overline{A} + \overline{B}$

ｳ.$X = A \cdot \overline{B} + \overline{A} \cdot B$　ｴ.$X = (\overline{A} + B) \cdot (A + \overline{B})$

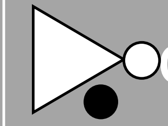
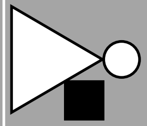
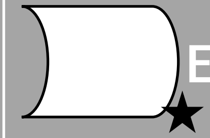
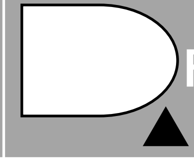
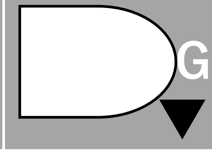
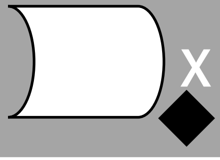
解き方

１．問題図に記号などで各ブロックに分ける



解き方

２．各ブロックを真理値表に表す

A	B	Aの否定  C	Bの否定  D	CとDの  E	AとEの  F	BとEの  G	FとGの  X
0	0	1	1	1	0	0	0
0	1	1	0	1	0	1	1
1	0	0	1	1	1	0	1
1	1	0	0	0	0	0	0

解き方

３．各選択肢を真理値表に置き換える

ア				
A	B	A・B＝H	$\overline{A \cdot B}$＝I	H+I=X
0	0	0	1	1
0	1	0	1	1
1	0	0	1	1
1	1	1	0	1

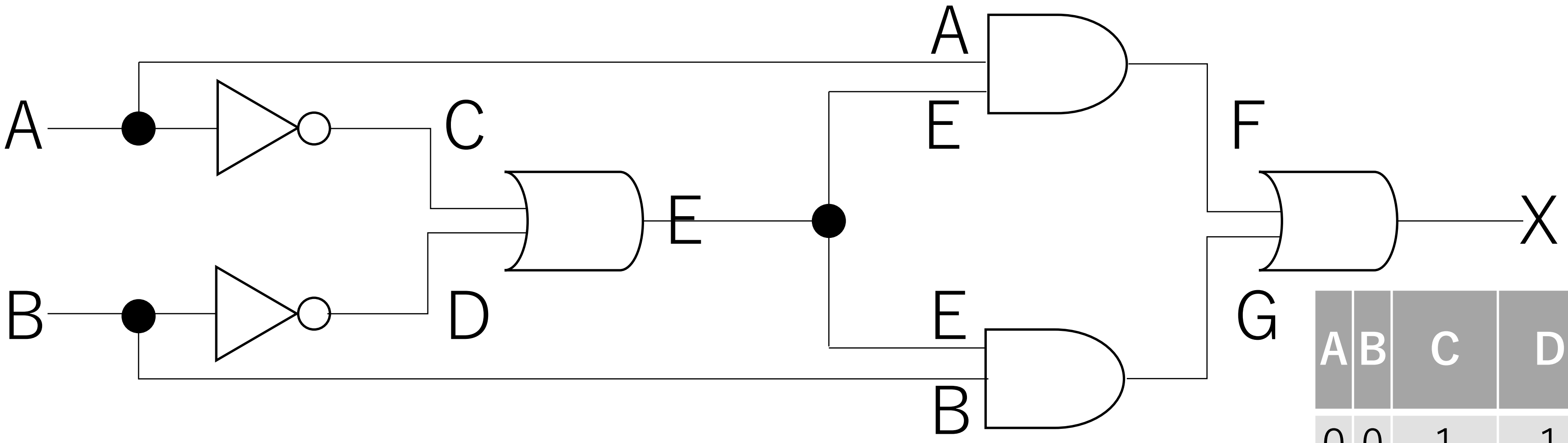
イ				
A	B	A・B＝J	$\overline{A}$・$\overline{B}$＝K	J+K=X
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

ウ				
A	B	A・$\overline{B}$＝L	$\overline{A}$・B＝M	L+M=X
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	0	0	0

エ				
A	B	$\overline{A}$+B＝N	A+$\overline{B}$＝O	N+O=X
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	1	1	1

答え

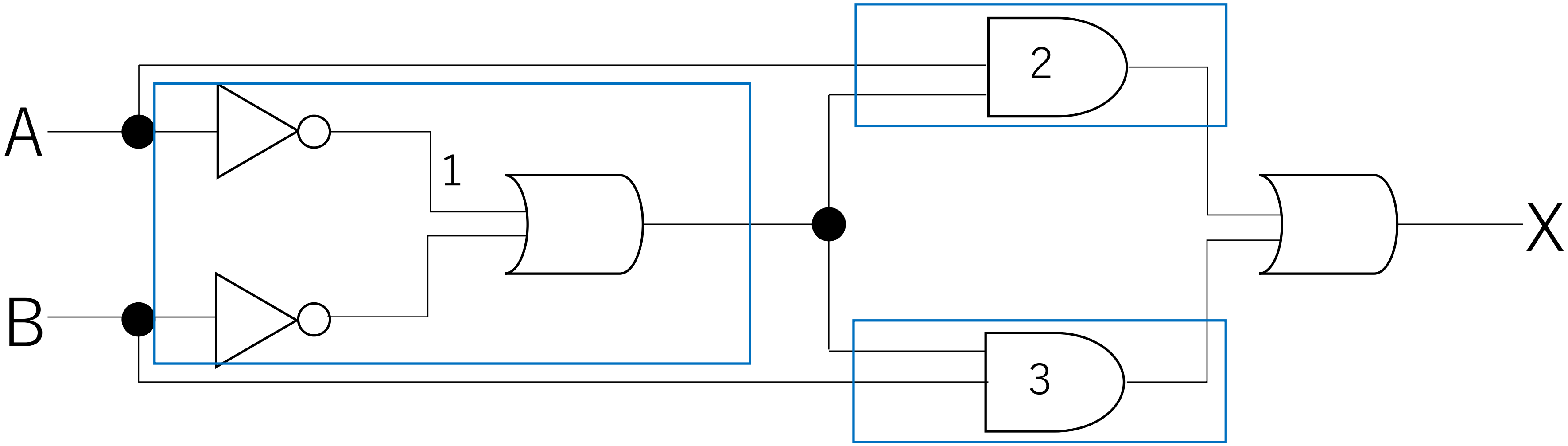
ウ．$X=A\cdot\overline{B}+\overline{A}\cdot B$



A	B	C	D	E	F	G	X
0	0	1	1	1	0	0	0
0	1	1	0	1	0	1	1
1	0	0	1	1	1	0	1
1	1	0	0	0	0	0	0

別解

３つに分解して考える。ド・モルガン則により、
１はAとBの否定論理積(後述)、
２はAと１の論理積、３はBと１の論理積。

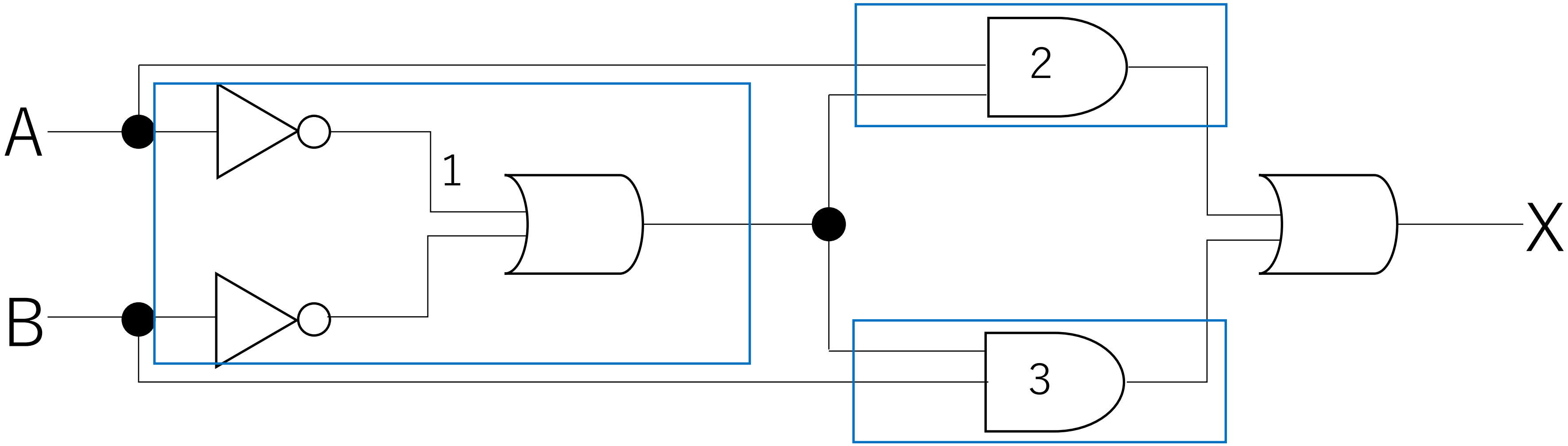


別解

１－＞~~A~~＋~~B~~、２－＞A・~~A~~＋~~B~~、　３－＞B・~~A~~＋~~B~~。

全体は２と３の論理和でX=A・~~A~~＋A・~~B~~＋B・~~A~~＋B・~~B~~。

上記の結果A・~~A~~＝０、B・~~B~~＝０なのでX＝A・~~B~~＋B・~~A~~となる

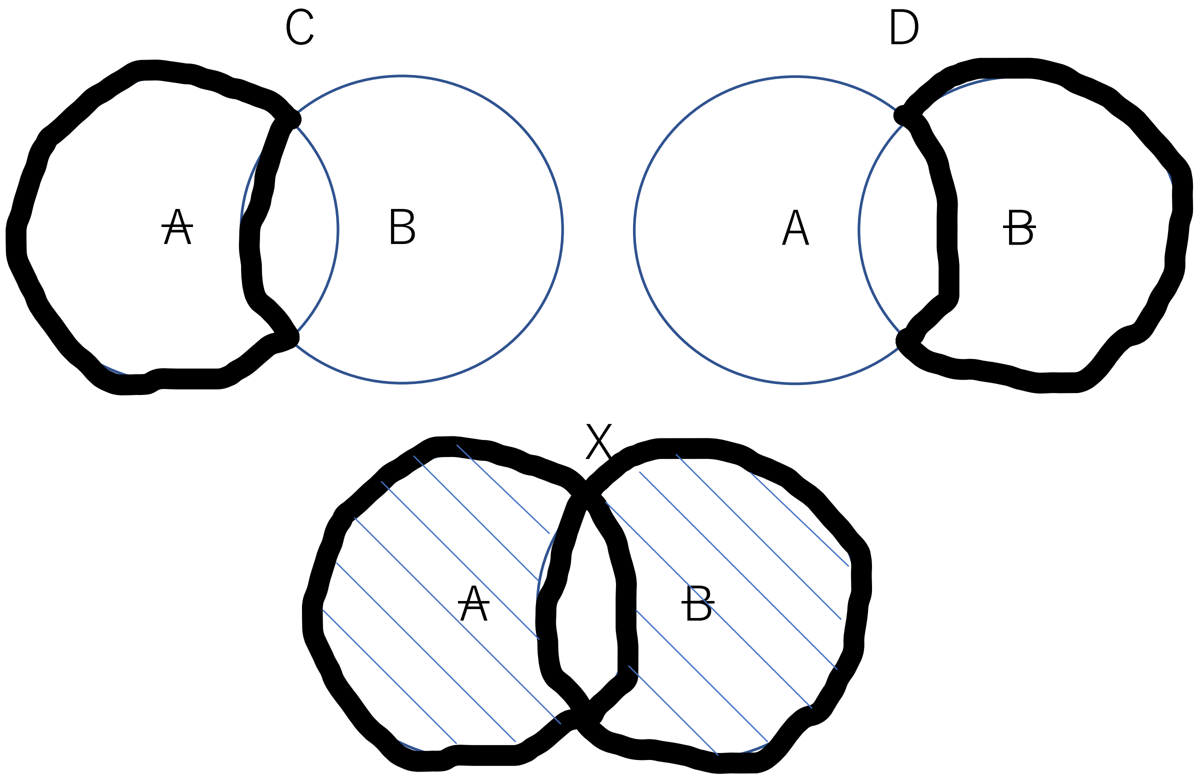


別解

ベン図に表すと、排他的論理和(後述)になることに気づける

A	B	Aの否定C	Bの否定D
0	0	1	1
0	1	1	0
1	0	0	1
1	1	0	0

A	B	CとDの論理和E	AとEの論理積F	BとEの論理積G	FとGの論理和X
0	0	1	0	0	0
0	1	1	0	1	1
1	0	1	1	0	1
1	1	0	0	0	0



3.
基本回路を組み合わせた
論理回路

基本回路の組み合わせ？

基本回路は前述した、AND、OR、NOTの三つだが、
この三つを組み合わせた論理回路が存在する。

代表的な組み合わせた論理演算

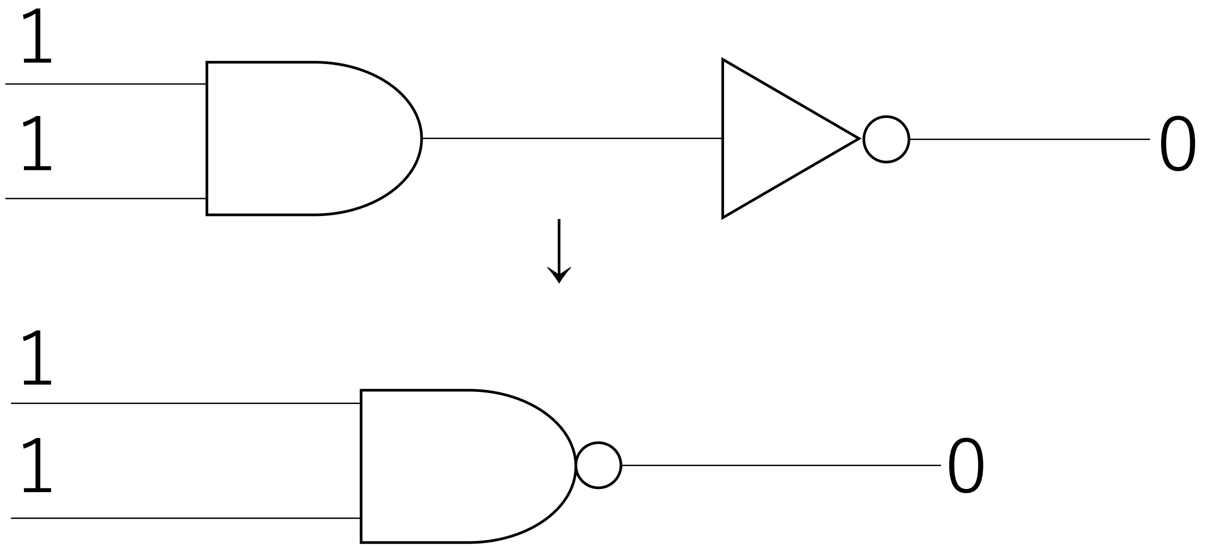
１．否定論理積(NAND)

２．否定論理和(NOR)

３．排他的論理和(XOR)

否定論理積回路(NAND)

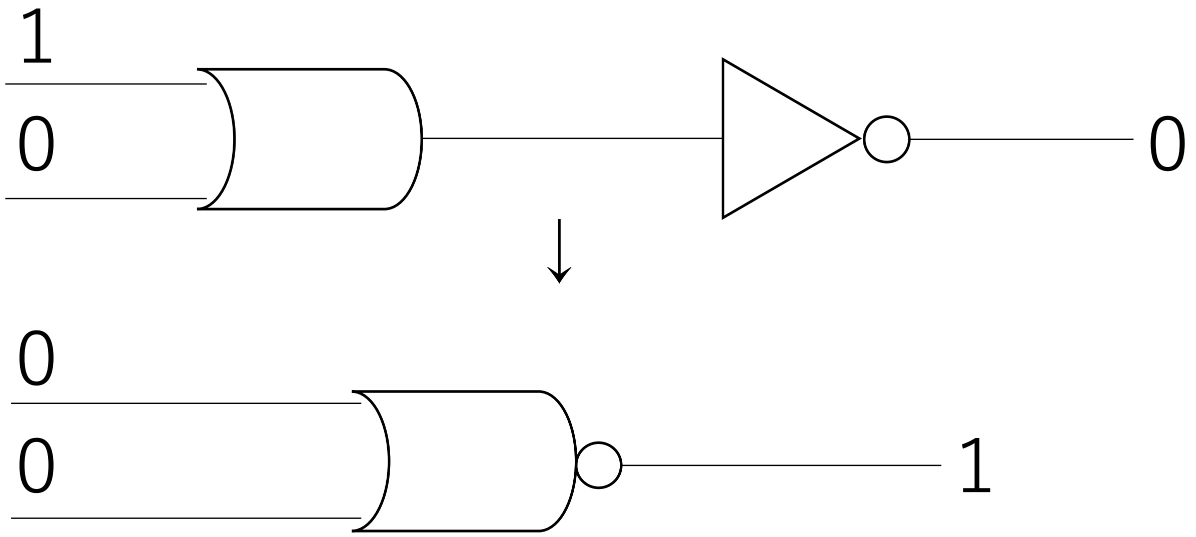
論理積(AND)と否定(NOT)を組み合わせた論理回路(ANDの逆！)



論理式－＞~~A・B~~＝Y(一つ前の問題のX=A・B＋~~A・B~~が該当)

否定論理和回路(NOR)

論理和(OR)と否定(NOT)を組み合わせた論理回路(ORの逆！)

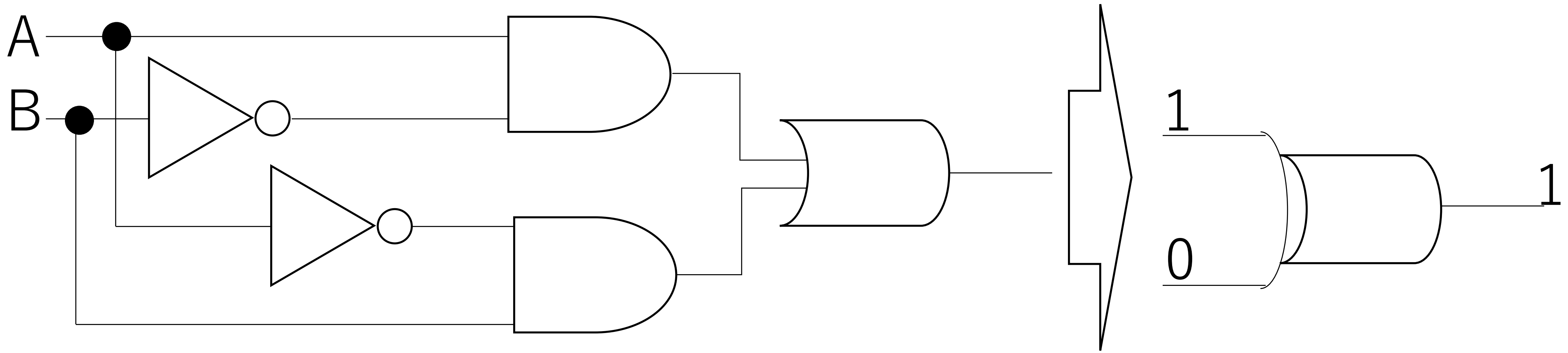


論理式－＞~~A＋B~~＝Y

排他的論理和回路(EOR又はXOR)

論理和(OR)から論理積(AND)を 差し引いた論理回路
(たとえ方は諸説あり)

中身(予備知識として)　　　MIL記号　　　簡略



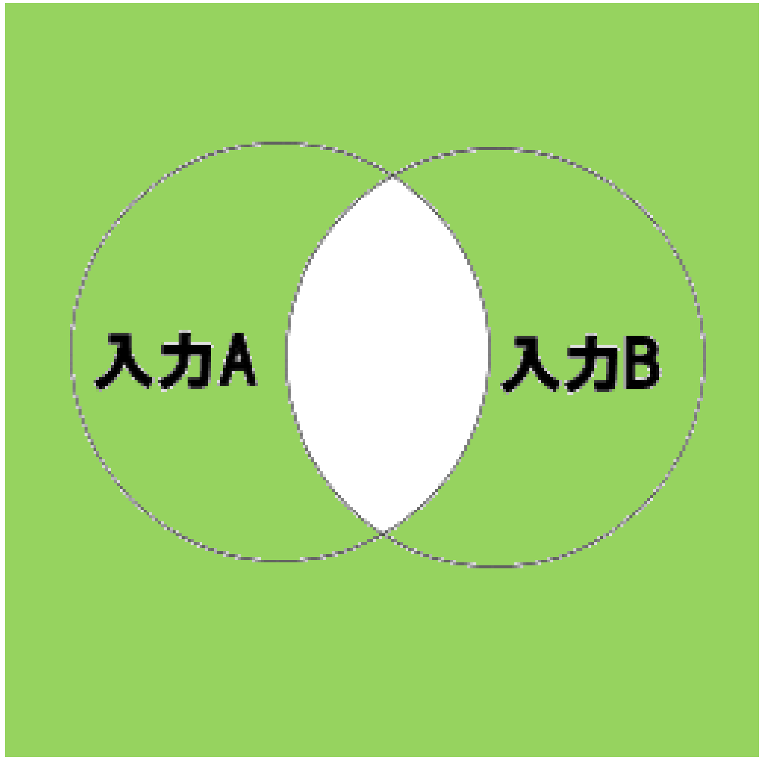
論理式 (一つ前の問題のX=A・~~B~~+~~A~~・Bが近い)

真理値表とベン図(否定論理積回路)

真理値表

入力A	入力B	出力Y
0	0	1
0	1	1
1	0	1
1	1	0

ベン図

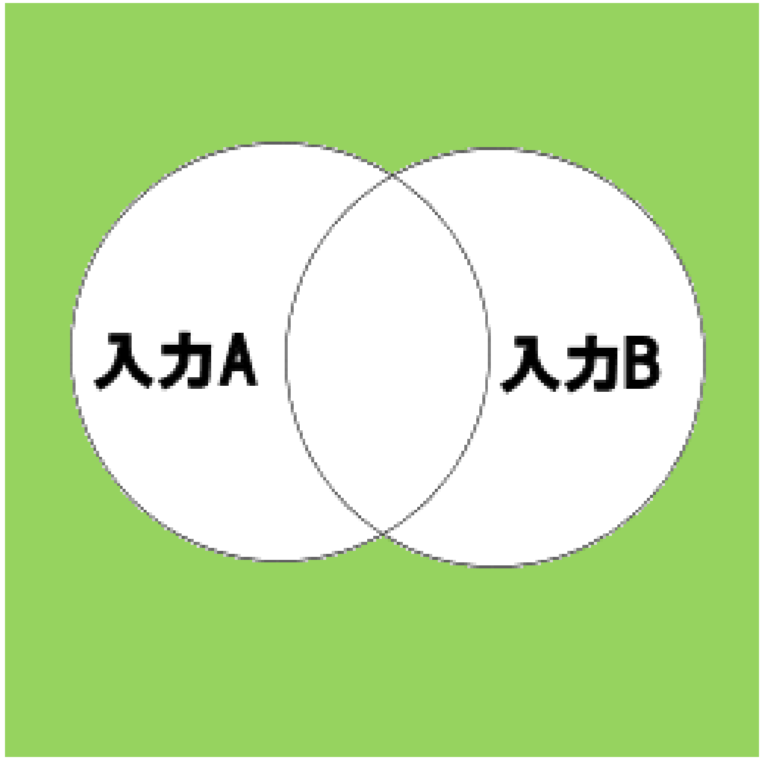


真理値表とベン図(否定論理和回路)

真理値表

入力A	入力B	出力Y
0	0	1
0	1	0
1	0	0
1	1	0

ベン図

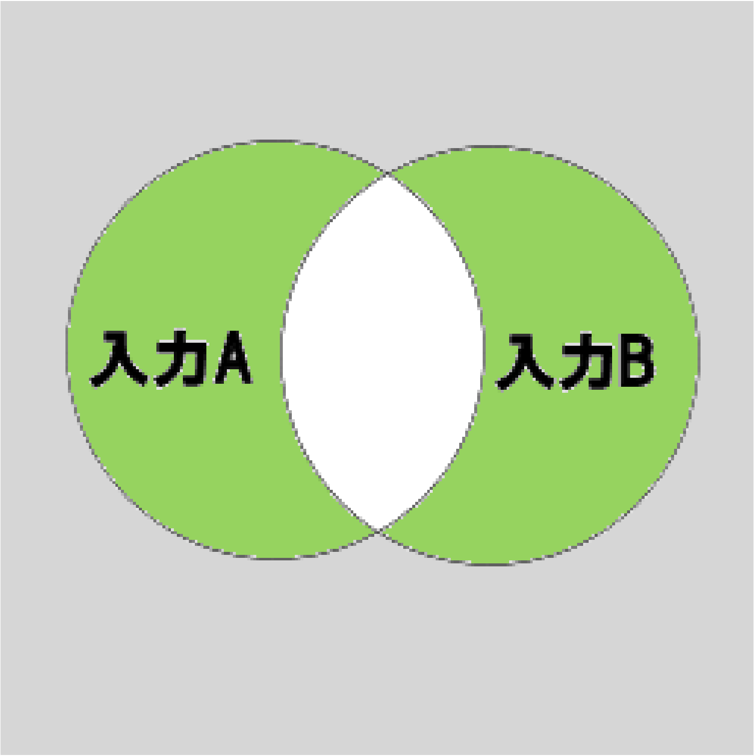


真理値表とベン図(排他的論理和回路)

真理値表

入力A	入力B	出力Y
0	0	0
0	1	1
1	0	1
1	1	0

ベン図



どんな風に問われるか？

問．XとYの否定論理積X NAND Yは、
NOT(X AND Y)として定義される。
X OR YをNANDだけを使って表した論理積はどれか。

ア、((X NAND Y)NAND X)NAND Y
イ、(X NAND X)NAND(Y NAND Y)
ウ、(X NAND Y)NAND(X NAND Y)
エ、X NAND(Y NAND(X NAND Y))

HINT

X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

解き方

1-1.真理値表にしてみる

ア、

(X NAND Y)NAND(X NAND Y)

X	Y	X NAMD Y = A^	A^ NAND X = B^	B^ NAND Y
0	0	1	1	1
0	1	1	1	0
1	0	1	0	1
1	1	0	1	0

イ、(X NAND X)NAND(Y NAND Y)

X	Y	X NAMD X = A^	Y NAND Y = B^	A^ NAND B^
0	0	1	1	0
0	1	1	0	1
1	0	0	1	1
1	1	0	0	1

解き方

1-2.真理値表にしてみる

ウ、

((X NAND Y)NAND X)NAND Y

X	Y	X NAMD Y = A^	X NAND Y = B^	A^ NAND B^
0	0	1	1	0
0	1	1	1	0
1	0	1	1	0
1	1	0	0	1

エ、 X NAND(Y NAND(X NAND Y))

X	Y	X NAMD X = A^	Y NAND A^= B^	X NAND B^
0	0	1	1	1
0	1	1	0	1
1	0	1	1	0
1	1	0	1	0

答え

ア、

X	Y	X NAMD Y = A^	A^ NAND X = B^	B^ NAND Y
0	0	1	1	1
0	1	1	1	0
1	0	1	0	1
1	1	0	1	0

イ、

X	Y	X NAMD X = A^	Y NAND Y = B^	A^ NAND B^
0	0	1	1	0
0	1	1	0	1
1	0	0	1	1
1	1	0	0	1

ウ、

X	Y	X NAMD Y = A^	X NAND Y = B^	A^ NAND B^
0	0	1	1	0
0	1	1	1	0
1	0	1	1	0
1	1	0	0	1

エ、

X	Y	X NAMD X = A^	Y NAND A^= B^	X NAND B^
0	0	1	1	1
0	1	1	0	1
1	0	1	1	0
1	1	0	1	0

X OR Y

と一致

するのは

イ

X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

４．　半加算器と全加算器

加算器(Adder)って？

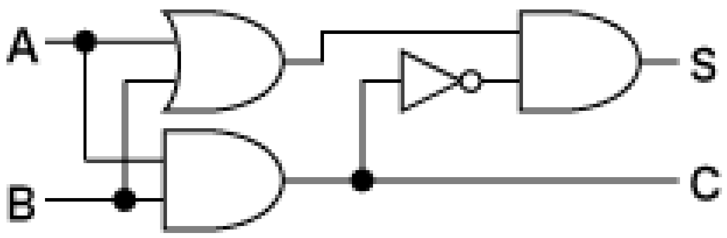
加算を行う演算装置のこと (Adder)

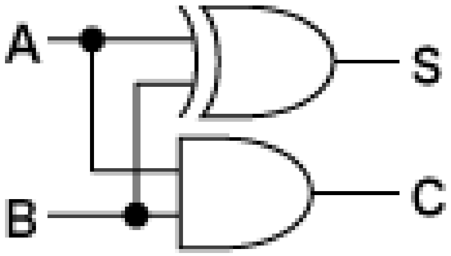
(~~加算機で調べると電卓が出てきます~~)

※一部ウィキペディア参照

半加算器(Half Adder)とは

2進数の同じ桁同士の演算をし、
桁上りは桁上げ出力(Carry out)によって出力(Sum)。
ただし、下位からの桁上がりを考慮しないため、一桁目しか処理できない
左の図はAND、OR、NOTを組み合わせた回路
右の図はXORを使った回路





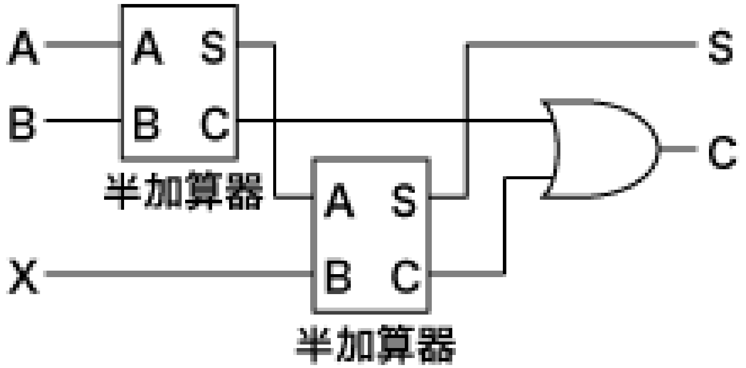
真理値表

A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

※ウィキペディア参照

全加算器(Full Adder)とは

2進数の最下位以外の桁同士の演算をし、
下位からの桁上入力を含めて出力。
半加算器ではできない多くの桁に対応できる。
図はAND、OR、NOTを組み合わせた回路
右の表は全加算器の真理値表(Xは桁上げ出力)



A	B	X	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

※ウィキペディア参照

どんな風に問われるか？

半加算器もしくは全加算器の回路図について問われる。
しかし平成21年秋以降、半加算器と全加算器は
問われていない。回路図は一応覚えておこう。

５．ビット操作と
マスクパターン

ビットの反転

ビットを反転するには、排他的論理和を使う

ビットの反転(全ビット)

反転させたいビット列　　0 0 0 0 1 1 0

反転させる位置に１を入れたビット列　　1 1 1 1 1 1 1

$$
\begin{array}{r}
0\ 0\ 0\ 0\ 1\ 1\ 0 \\
+\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \\
\hline
1\ 1\ 1\ 1\ 0\ 0\ 1
\end{array}
$$

ビットの反転(一部のビット)

反転させたいビット列	0 0 0 0 1 1 0
反転させる位置に１を入れたビット列	0 0 0 0 1 1 1

$$
\begin{array}{r}
0\ 0\ 0\ 0\ 1\ 1\ 0 \\
+\ 0\ 0\ 0\ 0\ 1\ 1\ 1 \\
\hline
0\ 0\ 0\ 0\ 0\ 0\ 1
\end{array}
$$

ビットの取り出し

ビットを取り出すには論理積を使う

対象のビット列	0 1 1 0 1 1 0
取り出す位置に１を入れたビット列	0 0 0 0 1 1 0

$$
\begin{array}{r}
0\ 1\ 1\ 0\ 1\ 1\ 0 \\
\cdot\ 0\ 0\ 0\ 0\ 1\ 1\ 0 \\
\hline
0\ 0\ 0\ 0\ 1\ 1\ 0
\end{array}
$$

マスクパターンとは

ビットを反転させたい(取り出したい)位置に
１を入れたビット列のこと

どんな風に問われるか？
(今回は平成31年春期の過去問から)

最上位をパリティビット(データの転送などで生じる誤りを検知できるように算出・付加される符号)とする８ビット符号において、パリティビット以外の下位７ビットを得るためのビット演算はどれか。

ア．16進数0F(00001111)とのANDをとる
イ．16進数0F(00001111)とのORをとる
ウ．16進数7F(01111111)とのANDをとる
エ．16進数FF(11111111)とのXORをとる

解き方

８ビットのデータを置き(今回は１００１１０１０とする)、
下位７ビットをビット演算で得る

ア.

```
  1 0 0 1 1 0 1 0
  0 0 0 0 1 1 1 1  (論理積
  ---------------
  0 0 0 0 1 0 1 0
```

イ.

```
  1 0 0 1 1 0 1 0
  0 0 0 0 1 1 1 1  (論理和
  ---------------
  1 0 0 1 1 1 1 1
```

ウ.

```
  1 0 0 1 1 0 1 0
  0 1 1 1 1 1 1 1  (論理積
  ---------------
  0 0 0 1 1 0 1 0
```

エ. (~~論理和~~

```
  1 0 0 1 1 0 1 0
  1 1 1 1 1 1 1 1
  ---------------
  0 1 1 0 0 1 0 1
```

解き方

下位７ビットを得るのはデータと論理演算の結果の
下位７ビットが同じであればいいので、答え．ウ

ア．

```
  1 0 0 1 1 0 1 0
  0 0 0 0 1 1 1 1  (論理積
  ---------------
  0 0 0 0 1 0 1 0
```

イ．

```
  1 0 0 1 1 0 1 0
  0 0 0 0 1 1 1 1  (論理和
  ---------------
  1 0 0 1 1 1 1 1
```

ウ．

```
  1 0 0 1 1 0 1 0
  0 1 1 1 1 1 1 1  (論理積
  ---------------
  0 0 0 1 1 0 1 0
```

エ．

```
  1 0 0 1 1 0 1 0
  1 1 1 1 1 1 1 1  (~~論理和~~
  ---------------
  0 1 1 0 0 1 0 1
```