論理回路入門（５）

論理回路基礎５：カルノー図と全加算器

李　亜民

2024 年 10 月 22 日 (火)

カルノー図と全加算器

ポイント

- 組み合わせ回路
- 真理値表
- 論理式
- カルノー図
- グレイコード
- 2 進数
- 半加算器
- 全加算器

組み合わせ回路

組み合わせ回路は

現在の入力のみで

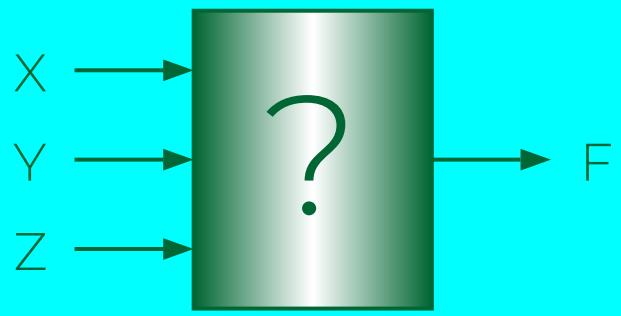
出力が決まる回路である

組み合わせ回路

- 組み合わせ回路の例：

真理値表

入力			出力
X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

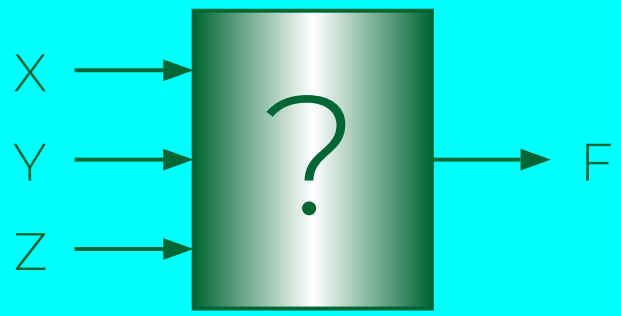


真理値表

- 組み合わせ回路の例：

真理値表

入力			出力
X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1



- どの組み合わせで出力が 1 になるか? つまり、論理式 F = ?

論理式

- 組み合わせ回路の例：

真理値表

入力			出力	
X	Y	Z	F	
0	0	0	0	
0	0	1	0	
0	1	0	1	X＝0 かつ Y＝1 かつ Z＝0 の時
0	1	1	0	
1	0	0	1	X＝1 かつ Y＝0 かつ Z＝0 の時
1	0	1	1	X＝1 かつ Y＝0 かつ Z＝1 の時
1	1	0	1	X＝1 かつ Y＝1 かつ Z＝0 の時
1	1	1	1	X＝1 かつ Y＝1 かつ Z＝1 の時

- どの組み合わせで出力が 1 になるか? つまり、論理式 F＝？

論理式

- 組み合わせ回路の例：

真理値表

入力			出力	
X	Y	Z	F	
0	0	0	0	
0	0	1	0	
0	1	0	1	$\overline{X}\,Y\,\overline{Z} = 1$ または
0	1	1	0	
1	0	0	1	$X\,\overline{Y}\,\overline{Z} = 1$ または
1	0	1	1	$X\,\overline{Y}\,Z = 1$ または
1	1	0	1	$X\,Y\,\overline{Z} = 1$ または
1	1	1	1	$X\,Y\,Z = 1$ の時, $F = 1$

- $F = \overline{X}\,Y\,\overline{Z} + X\,\overline{Y}\,\overline{Z} + X\,\overline{Y}\,Z + X\,Y\,\overline{Z} + X\,Y\,Z$ (積和標準形)
(最小項の論理和)

ブール代数の定理による論理式を簡単化

- (1) 冪等律 $A = A + A = A + A + A$　　(2) 同一律 $A \cdot 1 = A$

- $$F = \overline{X}\,Y\,\overline{Z} + X\,\overline{Y}\,\overline{Z} + X\,\overline{Y}\,Z + X\,Y\,\overline{Z} + X\,Y\,Z$$
$$= \overline{X}\,Y\,\overline{Z} + X\,Y\,\overline{Z} + X\,\overline{Y}\,\overline{Z} + X\,\overline{Y}\,Z + X\,Y\,\overline{Z} + X\,Y\,Z$$
$$= (\overline{X}\,Y\,\overline{Z} + X\,Y\,\overline{Z}) + (X\,\overline{Y}\,\overline{Z} + X\,\overline{Y}\,Z) + (X\,Y\,\overline{Z} + X\,Y\,Z)$$
$$= (\overline{X} + X)\,Y\,\overline{Z} + X\,\overline{Y}\,(\overline{Z} + Z) + X\,Y\,(\overline{Z} + Z)$$
$$= Y\,\overline{Z} + X\,\overline{Y} + X\,Y$$
$$= Y\,\overline{Z} + X\,(\overline{Y} + Y)$$
$$= Y\,\overline{Z} + X$$
$$= X + Y\,\overline{Z}$$

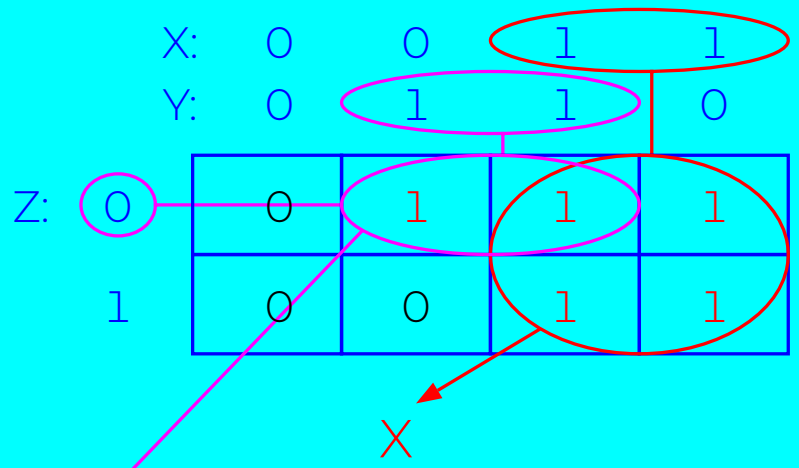
カルノー図による論理式を簡単化 3 入力

- 組み合わせ回路の例:

真理値表 (1 次元：1 列 × 8 行)

入力			出力
X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

カルノー図 (2 次元：4 列 × 2 行)

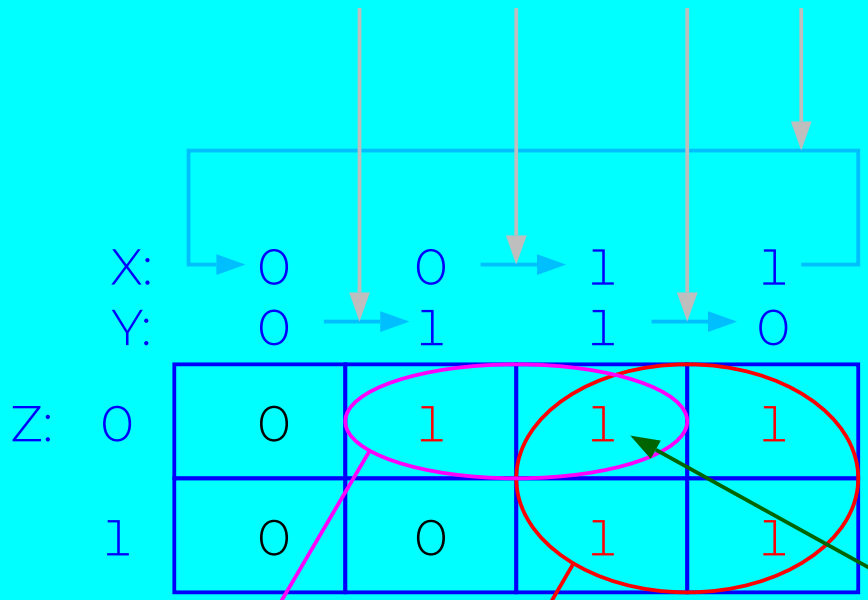


隣り合う出力が 1 のセルの数が 2^n 個 $(n = 0, 1, 2, \ldots)$ のときに長方形 (ループ) で囲む

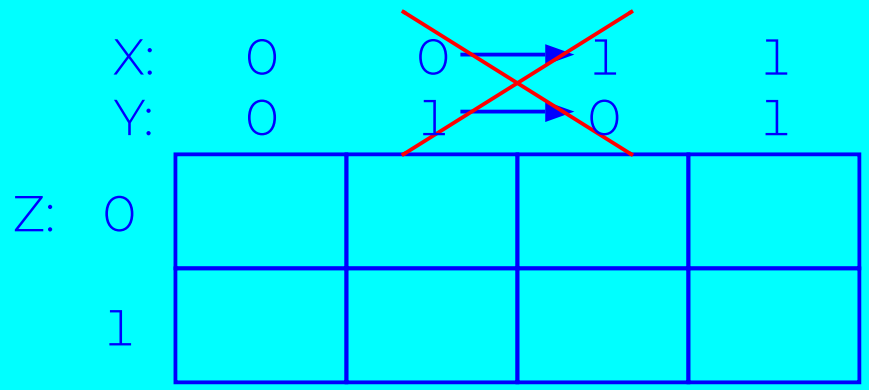
- 真理値表とカルノー図：五つの 1
- $F = X + Y\overline{Z}$ (すべての 1 が含む)
- XY: 00, 01, 11, 10: グレイコード。グレイコードは隣り合う文字のハミング距離 (互いを置換するために必要な回数) が 1 であるという特性を持つ。
- XY: 00,01,10,11 は 01 と 10 のハミング距離が 2 なのでグレイコードでない。

カルノー図による論理式を簡単化 3 入力

隣り合う文字のハミング距離 (互いを置換するために必要な回数) が 1 である



ハミング距離が 2 なので
グレイコードでない



X

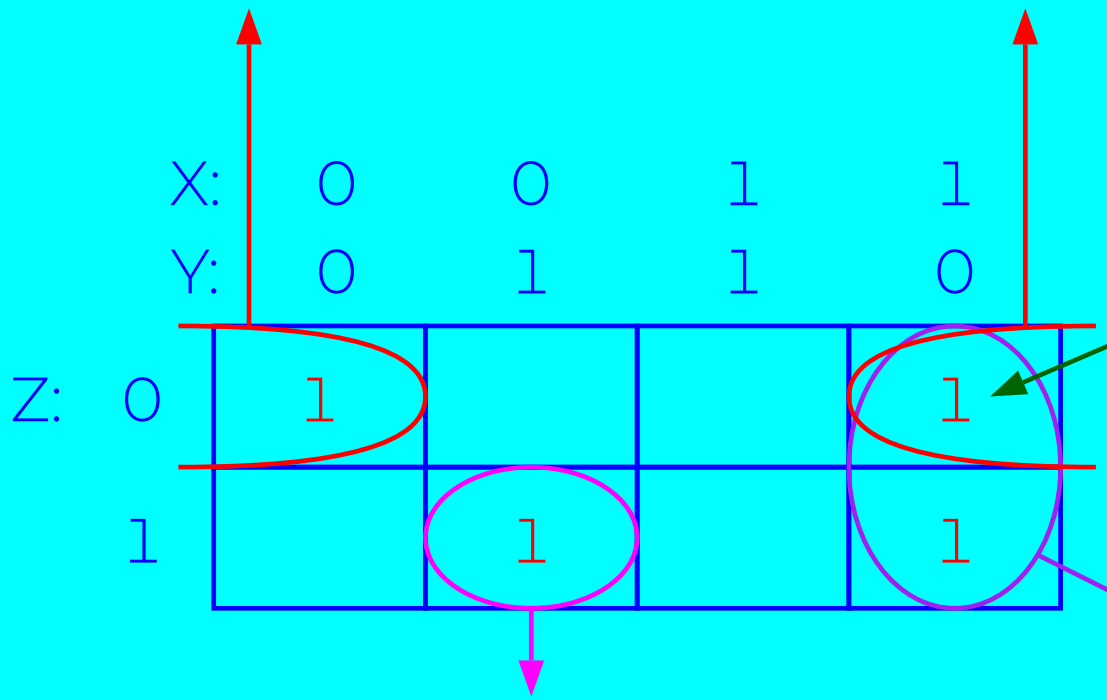
(X = 1 のとき他の入力に関わらず F = 1)

$Y\overline{Z}$ (Y = 1, Z = 0 のとき X に関わらず F = 1)

A = A + A なので、
同じセルを 2 つ以上の
ループで共有しても良い

カルノー図による論理式を簡単化 3 入力

$\overline{Y}\,\overline{Z}$ ($Y = 0, Z = 0$ のとき X に関わらず $F = 1$)



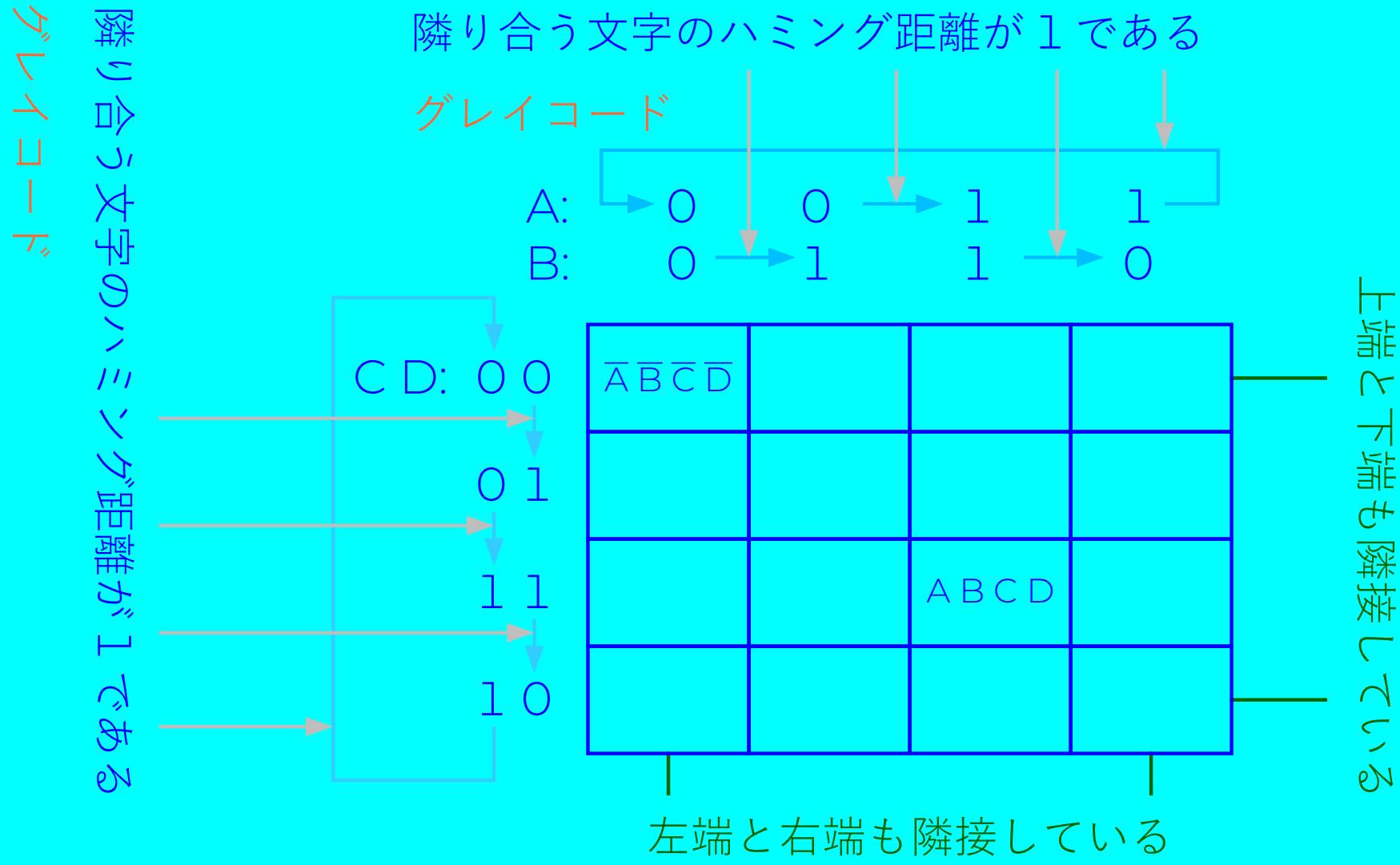
$A = A + A$ なので、
同じセルを 2 つ以上
のループで共有
しても良い

$X\,\overline{Y}$
($X = 1, Y = 0$ のとき
Z に関わらず $F = 1$)

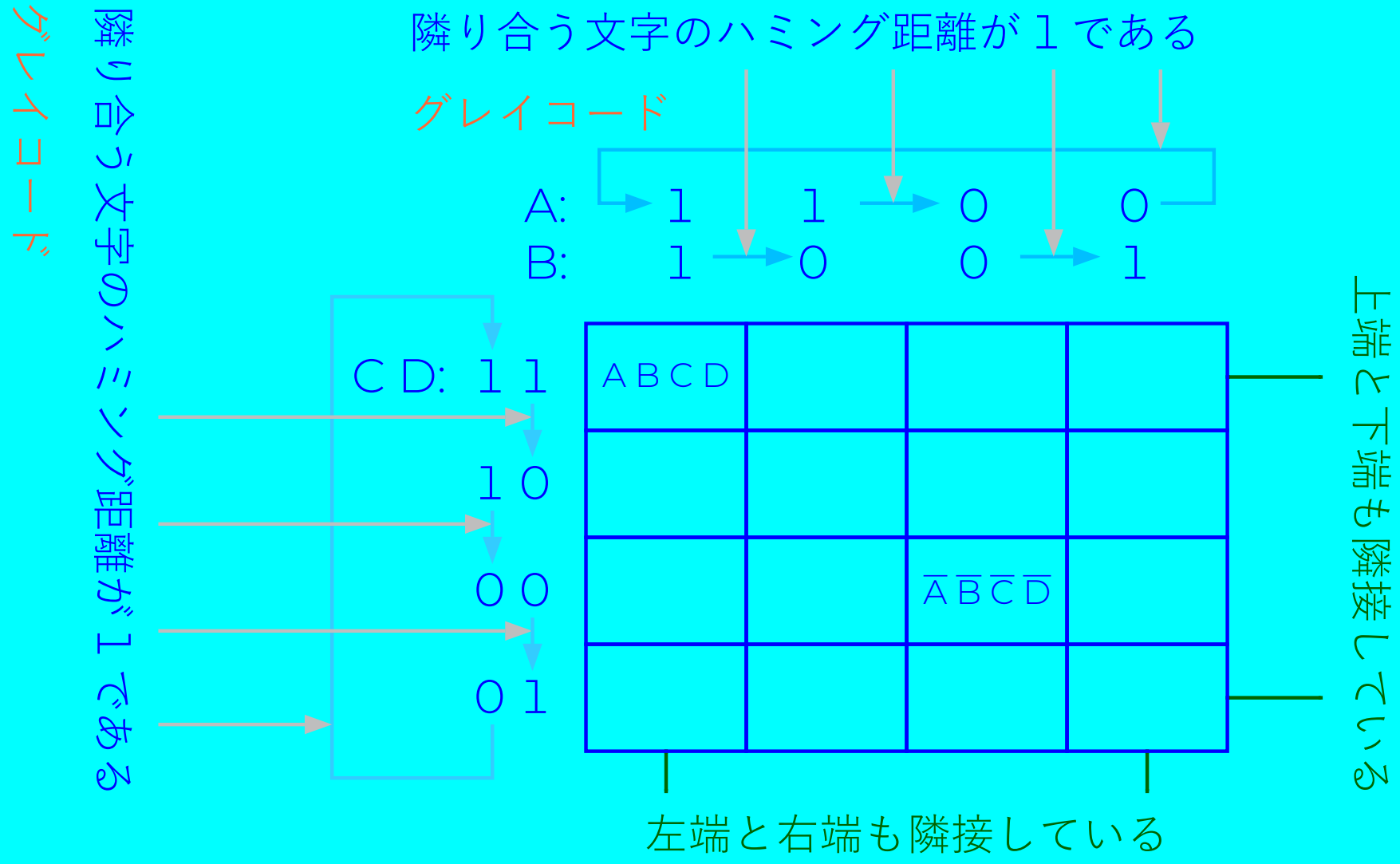
$\overline{X}\,Y\,Z$

$$F = X\,\overline{Y} + \overline{Y}\,\overline{Z} + \overline{X}\,Y\,Z$$

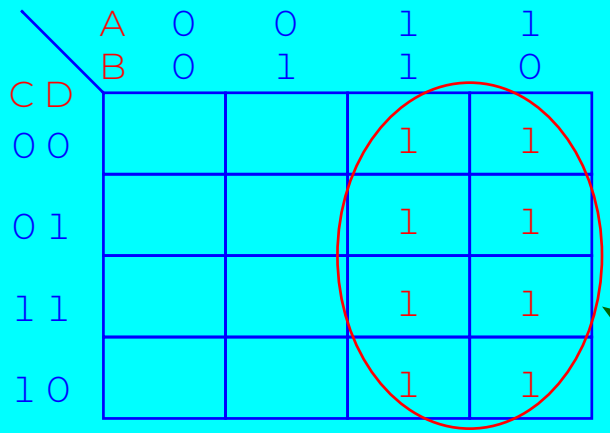
カルノー図による論理式を簡単化 4 入力



カルノー図による論理式を簡単化 4 入力

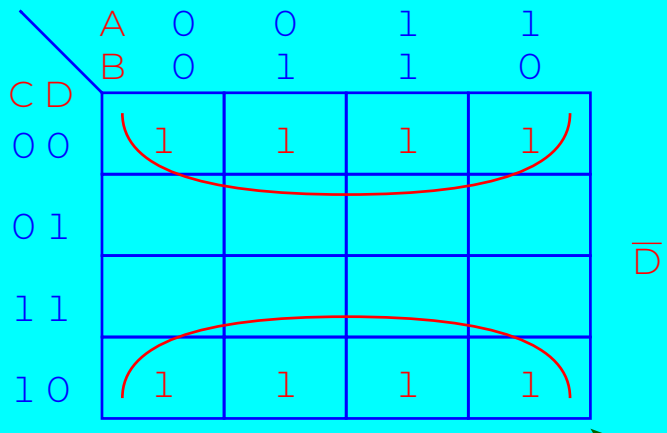


カルノー図による論理式を簡単化 4 入力



$F = A$

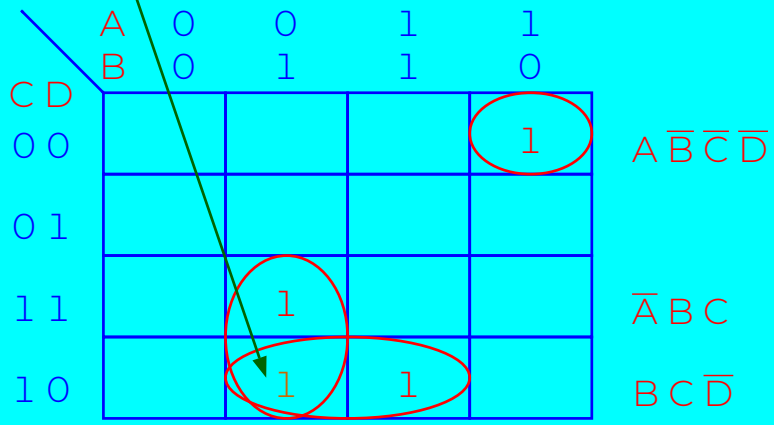
隣り合う出力が 1 のセルの数が 2^n 個 $(n = 0, 1, 2, \ldots)$ のときに長方形 (ループ) で囲む。



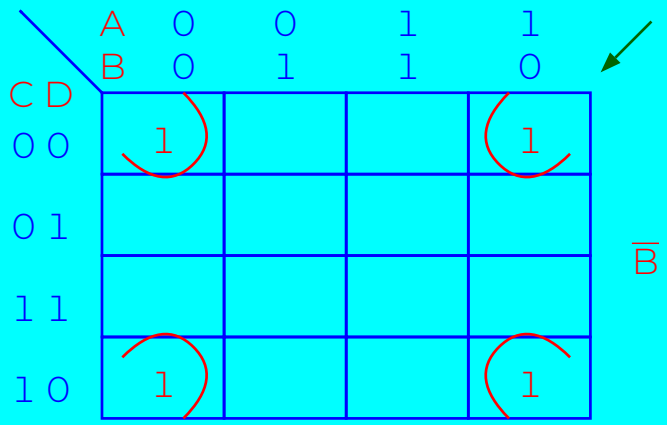
$F = \overline{D}$

カルノー図の上下の端、左右の端はそれぞれ連続していると考える。

同じセルを 2 つ以上のループで共有しても良い。

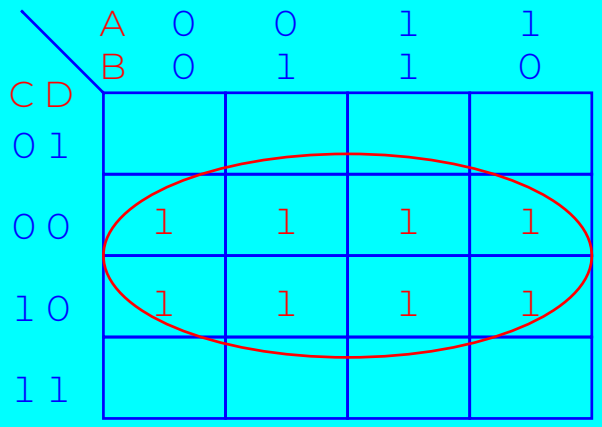


$F = A\overline{B}\overline{C}\overline{D} + \overline{A}BC + BC\overline{D}$



$F = \overline{B}\overline{D}$

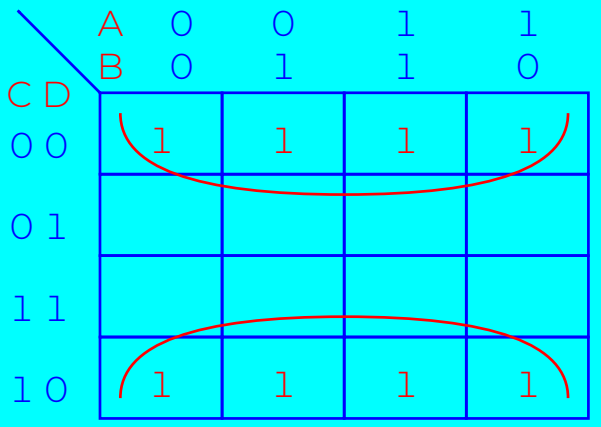
カルノー図による論理式を簡単化 4 入力



$F = \overline{D}$

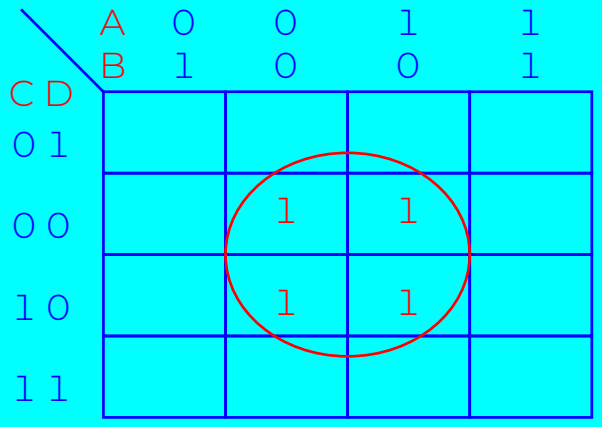
⇐

グレイコードの順序を
変更すると



$F = \overline{D}$

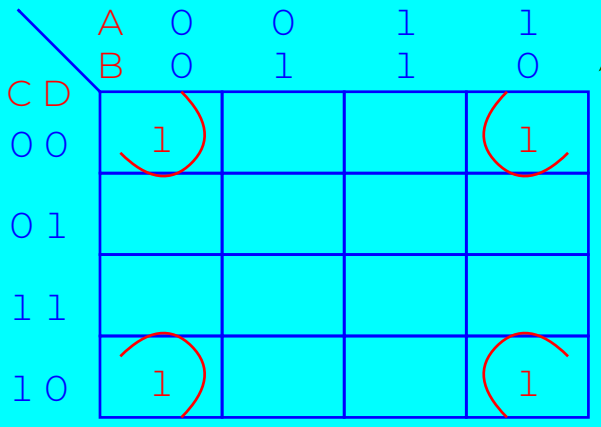
カルノー図の上下の端、
左右の端はそれぞれ連
続していると考える。



$F = \overline{B}\,\overline{D}$

⇐

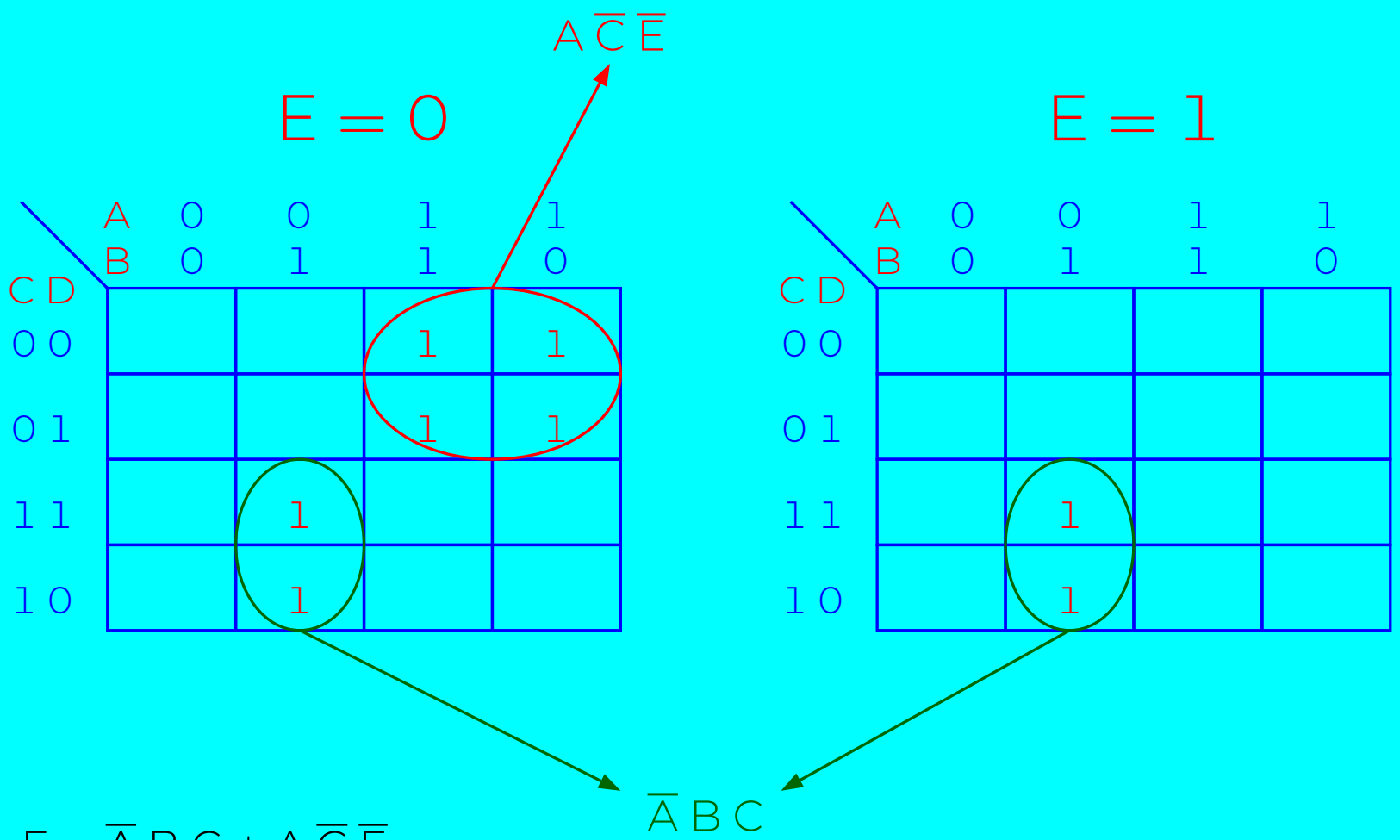
グレイコードの順序を
変更すると



$F = \overline{B}\,\overline{D}$

カルノー図による論理式を簡単化 5 入力

カルノー図 (3 次元：4 列 × 4 行 × 2 層)

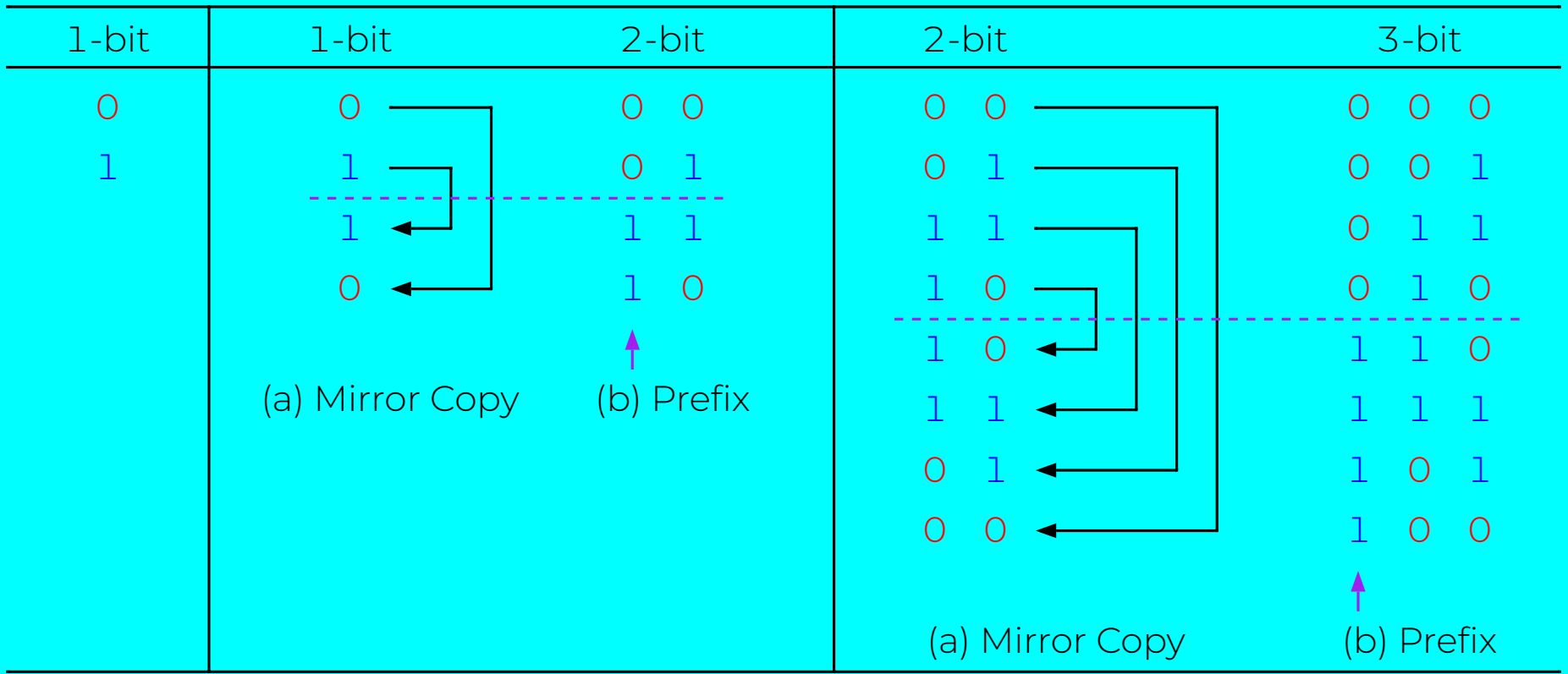


$F = \overline{A}BC + A\overline{C}\,\overline{E}$

グレイコード（Gray Code）

グレイコード: 隣接する 2 進数のハミング距離が常に 1 となる

ハミング距離: 2 つの文字列の対応する位置にある異なった文字の個数

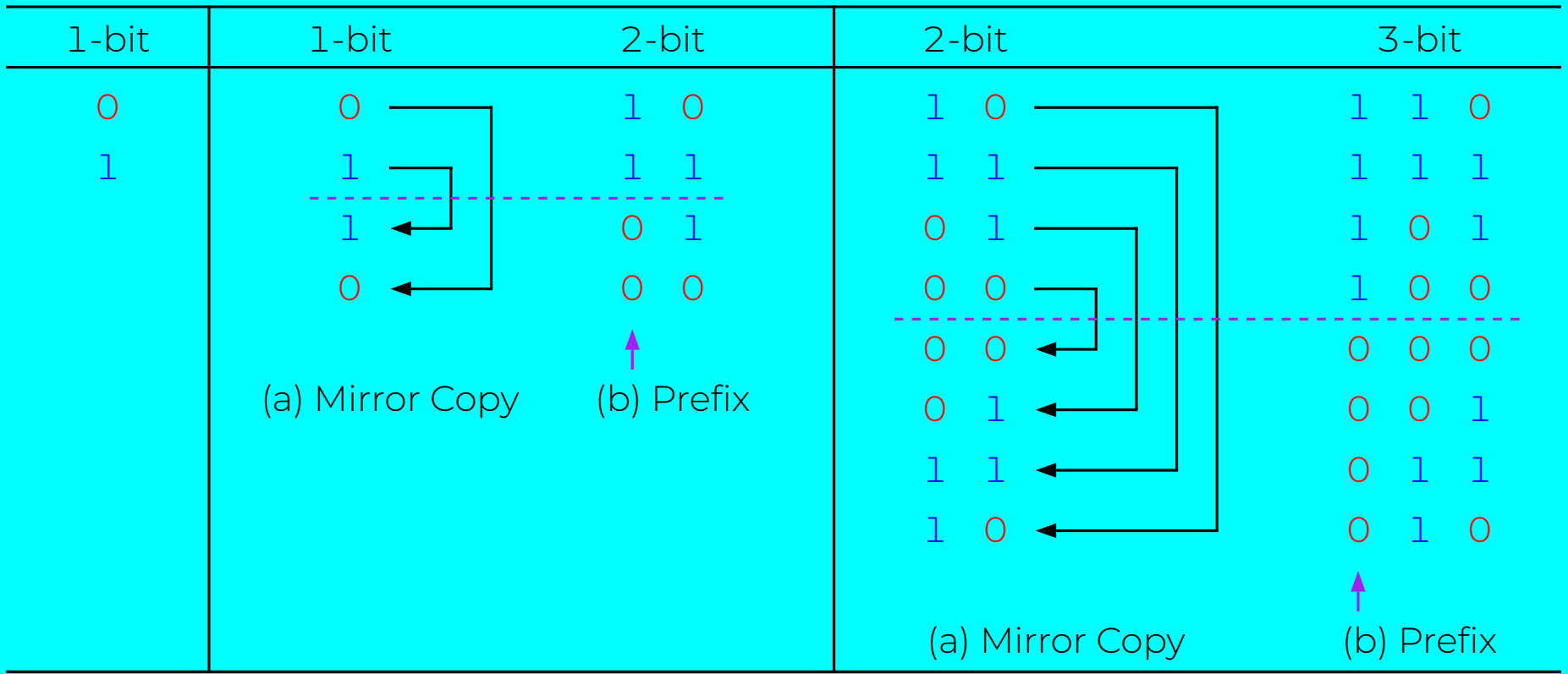


練習：4 ビットのグレイコードを作ってください。

グレイコード（Gray Code）

グレイコード: 隣接する 2 進数のハミング距離が常に 1 となる

ハミング距離: 2 つの文字列の対応する位置にある異なった文字の個数

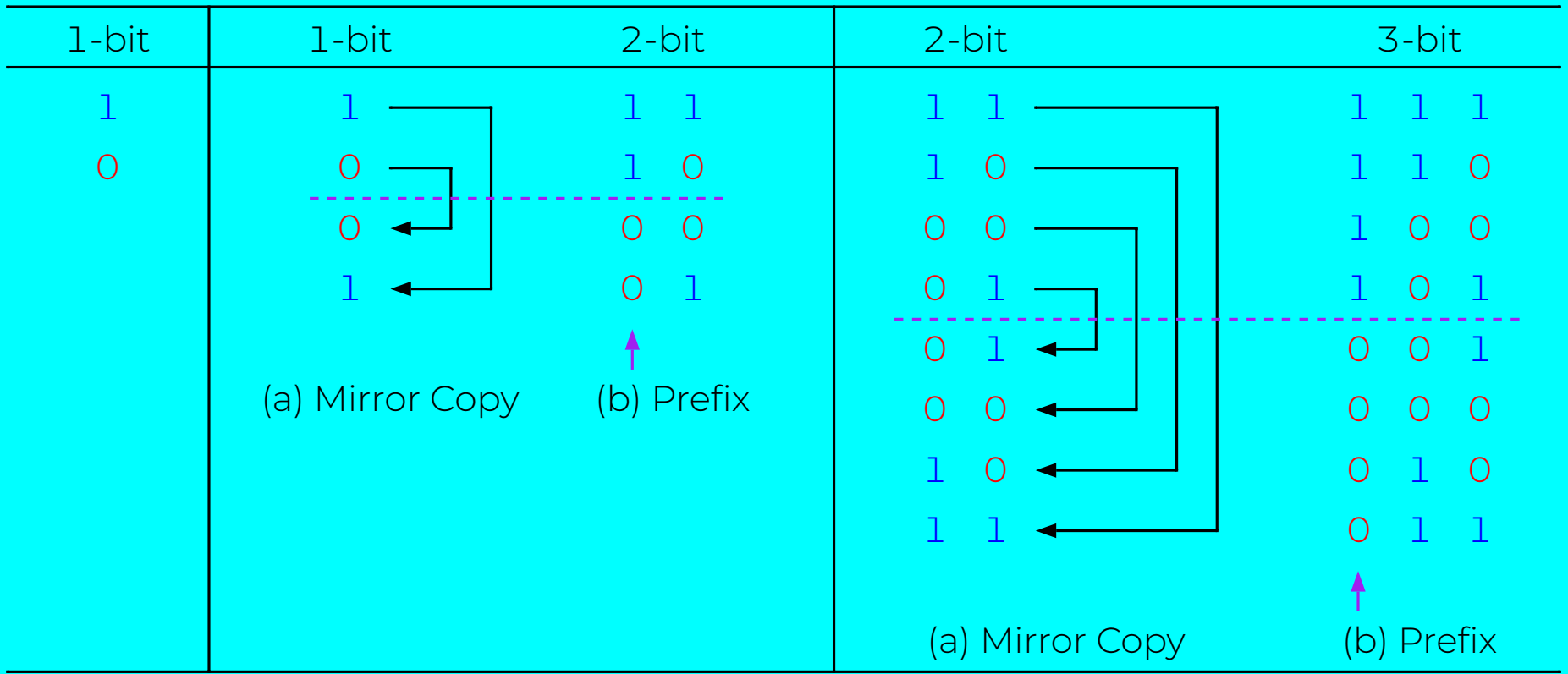


練習：4 ビットのグレイコードを作ってください。

グレイコード（Gray Code）

グレイコード: 隣接する 2 進数のハミング距離が常に 1 となる

ハミング距離: 2 つの文字列の対応する位置にある異なった文字の個数

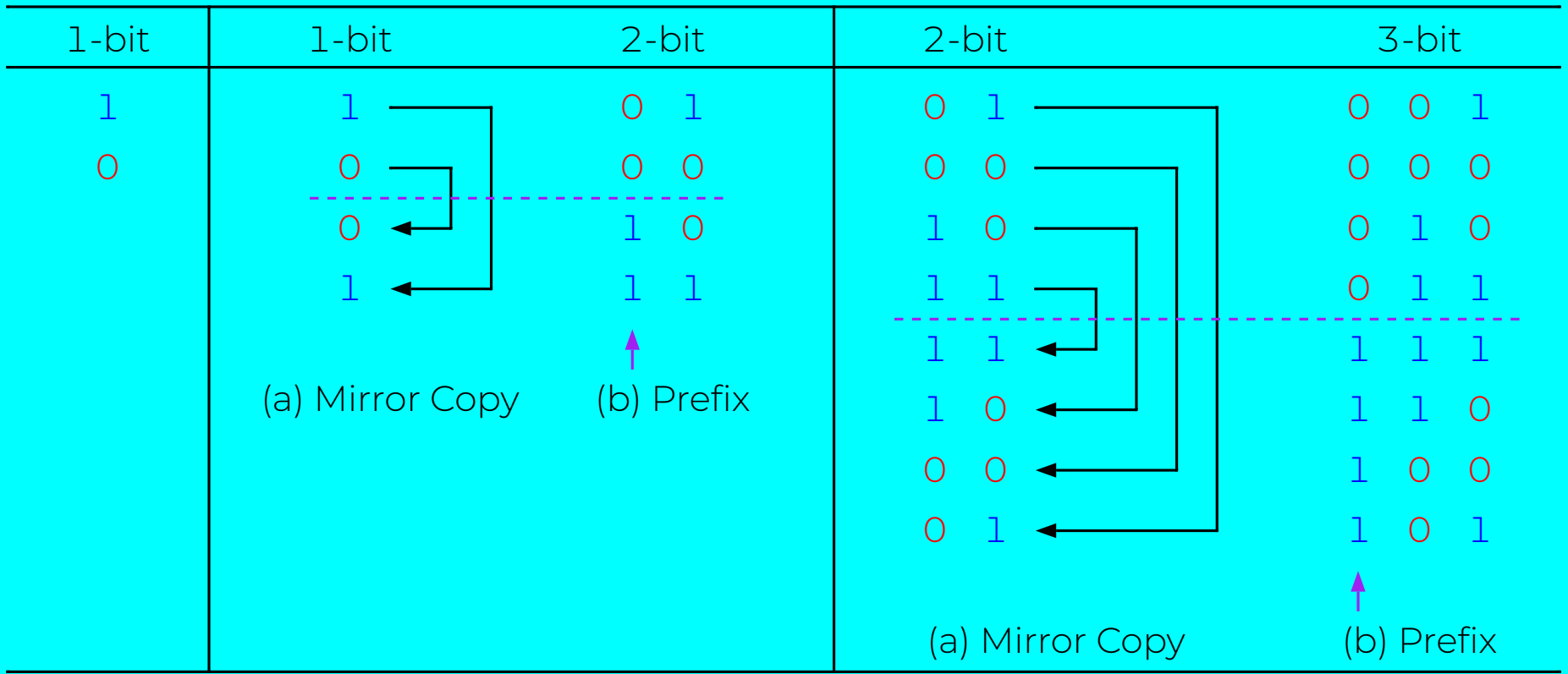


練習：4 ビットのグレイコードを作ってください。

グレイコード（Gray Code）

グレイコード: 隣接する 2 進数のハミング距離が常に 1 となる

ハミング距離: 2 つの文字列の対応する位置にある異なった文字の個数



練習：4 ビットのグレイコードを作ってください。

2 進数

- 10 進数 (Decimal Number) は、0 から 9 までの 10 個の数字を使って数を表現する。例：127,619,000
- 2 進数 (Binary Number) は、数字 0, 1 の 2 個の数字を使って数を表現する。
- 例：32 ビット 2 進数:
  - 01001111000111011000011110100101
  - ビット: Bit (Binary digit)、2 進数の桁
- 質問: 上記の 32 ビット 2 進数はどんな意味を持っているか？

32 ビット 2 進数 — すべてのパターン

0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0001
0000 0000 0000 0000 0000 0000 0000 0010
0000 0000 0000 0000 0000 0000 0000 0011
0000 0000 0000 0000 0000 0000 0000 0100

… …

1111 1111 1111 1111 1111 1111 1111 1011
1111 1111 1111 1111 1111 1111 1111 1100
1111 1111 1111 1111 1111 1111 1111 1101
1111 1111 1111 1111 1111 1111 1111 1110
1111 1111 1111 1111 1111 1111 1111 1111

質問: 全部で何パターンあるか？

32 ビット 2 進数 — すべてのパターン

0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0001
0000 0000 0000 0000 0000 0000 0000 0010
0000 0000 0000 0000 0000 0000 0000 0011
0000 0000 0000 0000 0000 0000 0000 0100

… …

1111 1111 1111 1111 1111 1111 1111 1011
1111 1111 1111 1111 1111 1111 1111 1100
1111 1111 1111 1111 1111 1111 1111 1101
1111 1111 1111 1111 1111 1111 1111 1110
1111 1111 1111 1111 1111 1111 1111 1111

質問: 全部で何パターンあるか？ 答：2^{32} パターン

覚えてください

2^0	=	1	2^9	=	512
2^1	=	2	2^{10}	=	1,024
2^2	=	4	2^{11}	=	2,048
2^3	=	8	2^{12}	=	4,096
2^4	=	16	2^{13}	=	8,192
2^5	=	32	2^{14}	=	16,384
2^6	=	64	2^{15}	=	32,768
2^7	=	128	2^{16}	=	65,536
2^8	=	256			
2^{32}	=	4,294,967,296 (覚えてなくてもよい)			

2 進数

- 質問: 次の 2 進数はどんな意味を持っているか？
  01001111001110110000111101001O1
- 答え: わからない。場合によって違う意味を持っている。
  持ちうる意味:
  - CPU の命令 (Instruction)
  - データ (Data)
    - 整数 (Integer)
    - 固定小数点数 (Fixed Point Number)
    - 浮動小数点数 (Floating Point Number)
    - IP アドレス (IP Address)
    - デジタル写真の中のピクセル (Pixel)
    - ... ...

2 進数整数 (Integer)

- 質問: 次の 4 ビット 2 進数整数はどんな意味を持っているか？

$$1001_2$$

- 答え: わからない。整数の表現する方法によって違う意味を持っている:
  - 符号なし (絶対値) $\cdots\cdots\cdots\cdots\cdots\cdots (1001_2 = \ \ 9_{10})$
  - 符号付き (正の整数と負の整数を表現可能)
    - ★ 1 の補数表現する方法 $\cdots\cdots\cdots (1001_2 = -6_{10})$
    - ★ 2 の補数表現する方法 $\cdots\cdots\cdots (1001_2 = -7_{10})$
    - ★ 符号–絶対値表現する方法 $\cdots\cdots (1001_2 = -1_{10})$
    - ★ Biased 表現する方法 $\cdots\cdots\cdots (1001_2 = +2_{10})$
    - ★ $\cdots$ $\cdots$

2 進数整数 — 符号なし数（絶対値）

- n ビット符号なし数の大きさは

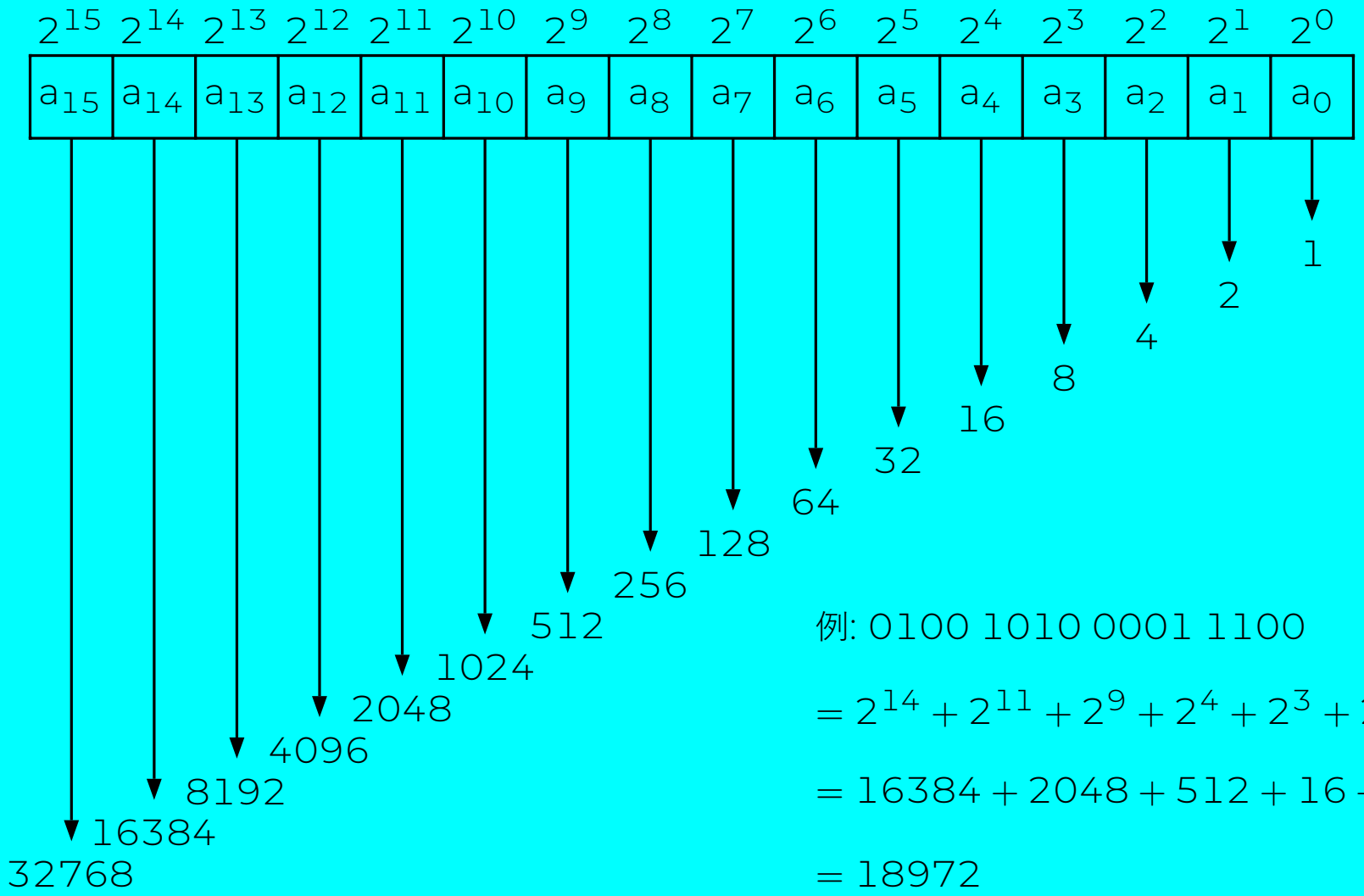
$$a_{n-1}a_{n-2}\cdots a_1a_0 = a_{n-1}\times 2^{n-1} + a_{n-2}\times 2^{n-2} + \cdots + a_1\times 2^1 + a_0\times 2^0$$

- 32 ビット符号なし数:

0000 0000 0000 0000 0000 0000 0000 0000 = 0_{10}
0000 0000 0000 0000 0000 0000 0000 0001 = 1_{10}
0000 0000 0000 0000 0000 0000 0000 0010 = 2_{10}
... ...
1111 1111 1111 1111 1111 1111 1111 1101 = $4{,}294{,}967{,}293_{10}$
1111 1111 1111 1111 1111 1111 1111 1110 = $4{,}294{,}967{,}294_{10}$
1111 1111 1111 1111 1111 1111 1111 1111 = $4{,}294{,}967{,}295_{10}$

- n ビット符号なし数 (絶対値) のうち最小値は 0、最大値は $2^n - 1$ である。
例：$n = 4$ のとき、最小値は 0、最大値は 15 である。

2 進数整数 — 符号なし数（絶対値）



例: 0100 1010 0001 1100

$= 2^{14} + 2^{11} + 2^9 + 2^4 + 2^3 + 2^2$

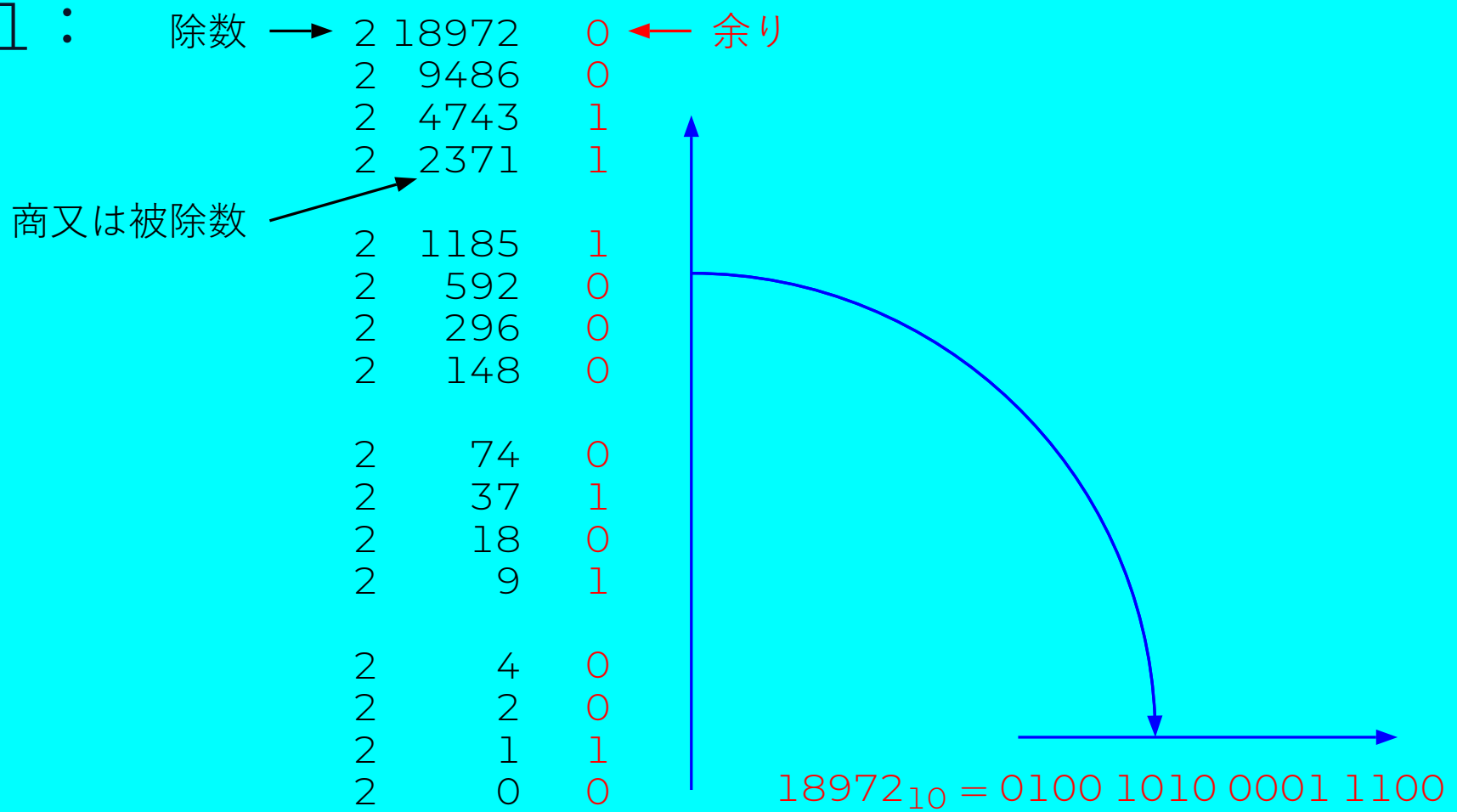
$= 16384 + 2048 + 512 + 16 + 8 + 4$

$= 18972$

2 進数整数 — 符号なし数（絶対値）

問題：10 進数の 18972 を 16 ビットの 2 進数に直してください。

解答 1：



2 進数整数 — 符号なし数（絶対値）

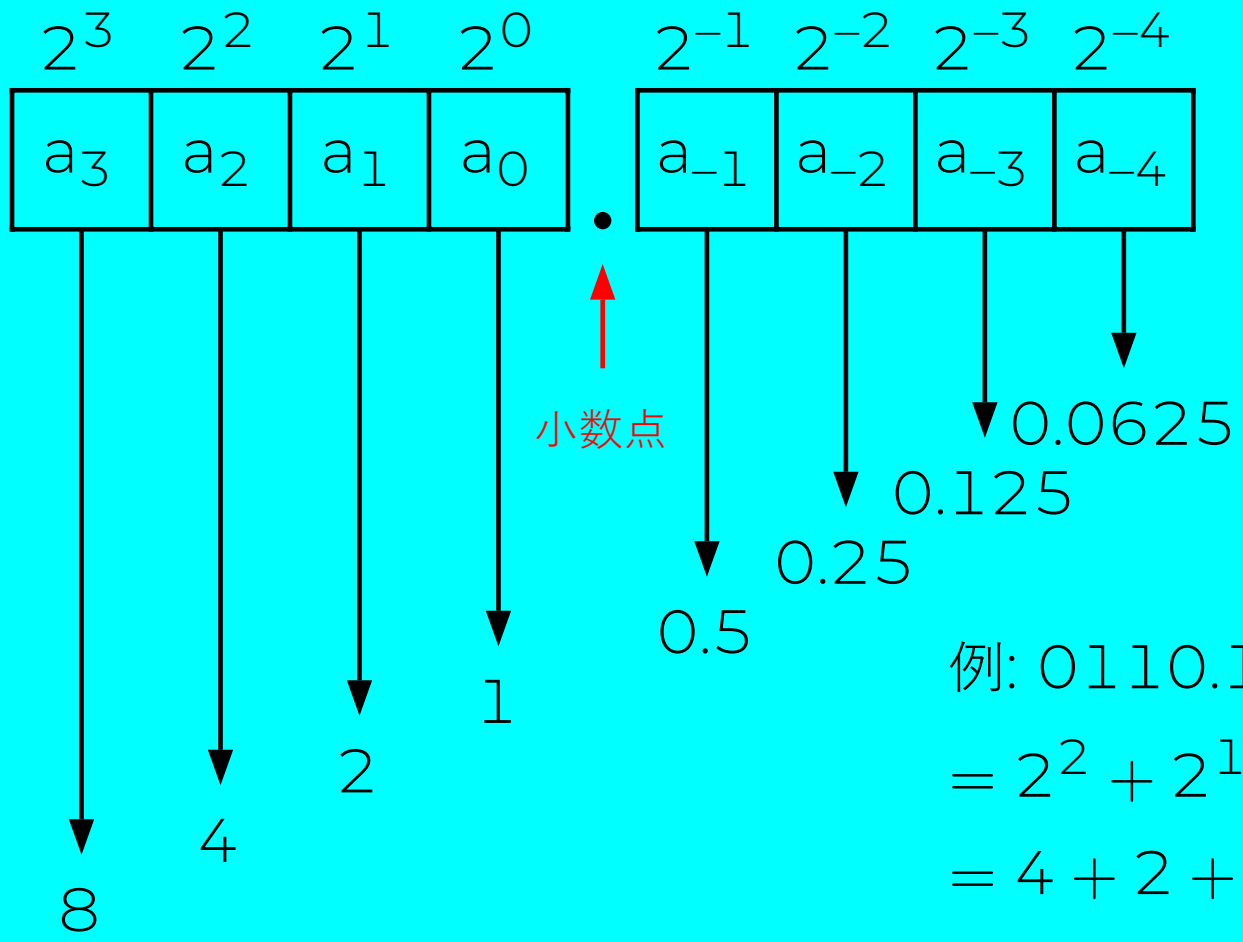
問題：10 進数の 18972 を 16 ビットの 2 進数に直してください。

解答 2：

$$18972 - 16384 = 2588 \cdots\cdots 2^{14}$$
$$2588 - 2048 = 540 \cdots\cdots 2^{11}$$
$$540 - 512 = 28 \cdots\cdots 2^{9}$$
$$28 - 16 = 12 \cdots\cdots 2^{4}$$
$$12 - 8 = 4 \cdots\cdots 2^{3}$$
$$4 - 4 = 0 \cdots\cdots 2^{2}$$

$$18972_{10} = 0100\ 1010\ 0001\ 1100_2$$

2 進数小数 — 符号なし数（絶対値）



例: 0110.1010

$= 2^2 + 2^1 + 2^{-1} + 2^{-3}$

$= 4 + 2 + 0.5 + 0.125$

$= 6.625$

Or: $(64 + 32 + 8 + 2)/16 = 106/16 = 6.625$

16 進数

- 16 進数 (Hexadecimal Number) は、0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F の 16 個の数字を使って数を表現する。
- 4 ビットの 2 進数 ⇒ 1 桁の 16 進数
- 2 進法より短く表記できる (2 進数はビットが多くて読みにくい)。
- 例: $0011\ 1111_2 = 3F_{16}$

2 進数	16 進数	8 進数	絶対値
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	8	10	8
1001	9	11	9
1010	A	12	10
1011	B	13	11
1100	C	14	12
1101	D	15	13
1110	E	16	14
1111	F	17	15

2 進数を 16 進数に

- 2 進数を下位から 4 ビット毎に区切って、4 ビット単位で 16 進数に置きかえる。
- 30 ビットの 2 進数の例:

$$\begin{array}{cccccccccc} & 10 & 1010 & 0011 & 1110 & 0111 & 0110 & 1111 & 1001_2 \\ = & 2 & A & 3 & E & 7 & 6 & F & 9_{16} \end{array}$$

- 注意：16 進数は本質的に 2 進数と同じで、表記する方法が違うだけである。

2 進数の加算

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

繰り上がり (carry out)

変数で表す:

```
    1  ←― a
+   1  ←― b
-----------
  1 0
```

入力信号

出力信号

```
  ↑   ↑
  c   s
```

c: carry out (上位桁への繰り上がり)

s: sum (和)

組み合わせ回路設計の手順

1. 問題を理解する。

組み合わせ回路設計の手順

1. 問題を理解する。
2. 入力と出力信号の名前を任意に決める。

組み合わせ回路設計の手順

1. 問題を理解する。
2. 入力と出力信号の名前を任意に決める。
3. 入力のあらゆる組み合わせを列挙し，真理値表をつくる。

組み合わせ回路設計の手順

1. 問題を理解する。
2. 入力と出力信号の名前を任意に決める。
3. 入力のあらゆる組み合わせを列挙し，真理値表をつくる。
4. 真理値表から論理式をつくる。

組み合わせ回路設計の手順

1. 問題を理解する。
2. 入力と出力信号の名前を任意に決める。
3. 入力のあらゆる組み合わせを列挙し，真理値表をつくる。
4. 真理値表から論理式をつくる。
   (どの組み合わせで出力が 1 になるか。)

組み合わせ回路設計の手順

1. 問題を理解する。
2. 入力と出力信号の名前を任意に決める。
3. 入力のあらゆる組み合わせを列挙し，真理値表をつくる。
4. 真理値表から論理式をつくる。
   (どの組み合わせで出力が 1 になるか。)
5. カルノー図を用いて論理式を簡単化する。

組み合わせ回路設計の手順

1. 問題を理解する。
2. 入力と出力信号の名前を任意に決める。
3. 入力のあらゆる組み合わせを列挙し，真理値表をつくる。
4. 真理値表から論理式をつくる。
   (どの組み合わせで出力が 1 になるか。)
5. カルノー図を用いて論理式を簡単化する。
6. 論理式から回路をつくる。

組み合わせ回路設計の手順

1. 問題を理解する。
2. 入力と出力信号の名前を任意に決める。
3. 入力のあらゆる組み合わせを列挙し，真理値表をつくる。
4. 真理値表から論理式をつくる。
   (どの組み合わせで出力が 1 になるか。)
5. カルノー図を用いて論理式を簡単化する。
6. 論理式から回路をつくる。
7. テストベンチをつくる (シミュレーションするため)。

組み合わせ回路設計の手順

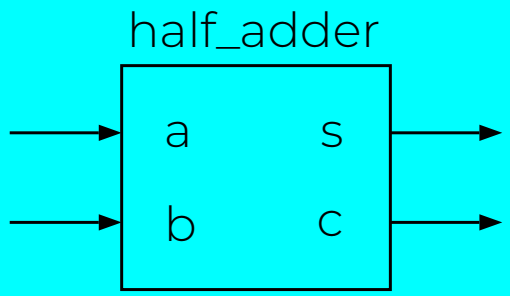
1. 問題を理解する。
2. 入力と出力信号の名前を任意に決める。
3. 入力のあらゆる組み合わせを列挙し，真理値表をつくる。
4. 真理値表から論理式をつくる。
   (どの組み合わせで出力が 1 になるか。)
5. カルノー図を用いて論理式を簡単化する。
6. 論理式から回路をつくる。
7. テストベンチをつくる (シミュレーションするため)。
8. 回路をシミュレーションする (回路設計の正当性検証)。

組み合わせ回路設計の手順

1. 問題を理解する。
2. 入力と出力信号の名前を任意に決める。
3. 入力のあらゆる組み合わせを列挙し，真理値表をつくる。
4. 真理値表から論理式をつくる。
   (どの組み合わせで出力が 1 になるか。)
5. カルノー図を用いて論理式を簡単化する。
6. 論理式から回路をつくる。
7. テストベンチをつくる (シミュレーションするため)。
8. 回路をシミュレーションする (回路設計の正当性検証)。
9. 与えられた回路の動作を理解する (波形の説明)。

半加算器回路設計

input　a, b;　// a + b　(加算)
output　c, s;　// c: carry out (上位桁への繰り上がり), s: sum (和)



真理値表

a	b	c	s	コメント
0	0	0	0	0 + 0 = 0 0　(加算)
0	1	0	1	0 + 1 = 0 1　(加算)
1	0	0	1	1 + 0 = 0 1　(加算)
1	1	1	0	1 + 1 = 1 0　(加算)

出力の c、s は a + b の結果 (和) を 2 進数で表した数値。

$00_2 = 0_{10}$　　$01_2 = 1_{10}$　　$10_2 = 2_{10}$

半加算器回路設計

真理値表

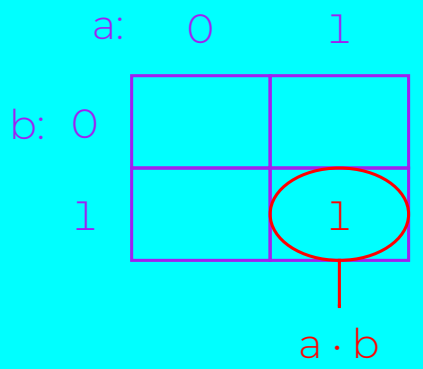
a	b	c	s	コメント
0	0	0	0	$0+0=0\,0$ (加算)
0	1	0	1	$0+1=0\,1$ (加算)
1	0	0	1	$1+0=0\,1$ (加算)
1	1	1	0	$1+1=1\,0$ (加算)

論理式

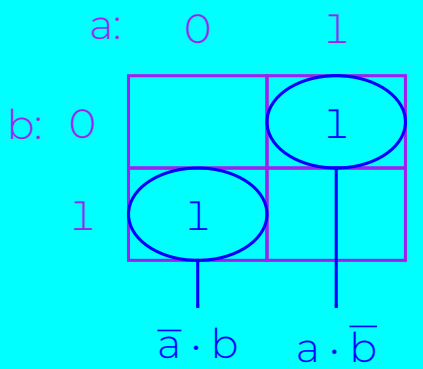
$c = a \cdot b$

$s = \overline{a} \cdot b + a \cdot \overline{b} = a \oplus b$

c のカルノー図



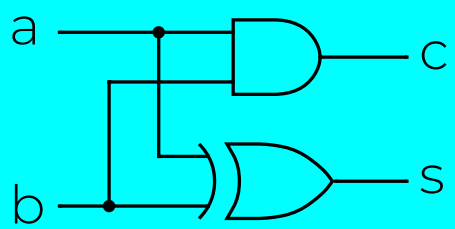
s のカルノー図



半加算器回路設計 (回路)

論理式 $c = a \cdot b$

$$s = \overline{a} \cdot b + a \cdot \overline{b} = a \oplus b$$



Verilog HDL による回路（half_adder.v）

```
‘timescale 1ns/1ns

module half_adder (a, b, c, s);
    input  a, b;
    output c, s;

    assign c = a & b; // AND
    assign s = a ^ b; // XOR

endmodule
```

half_adder.v

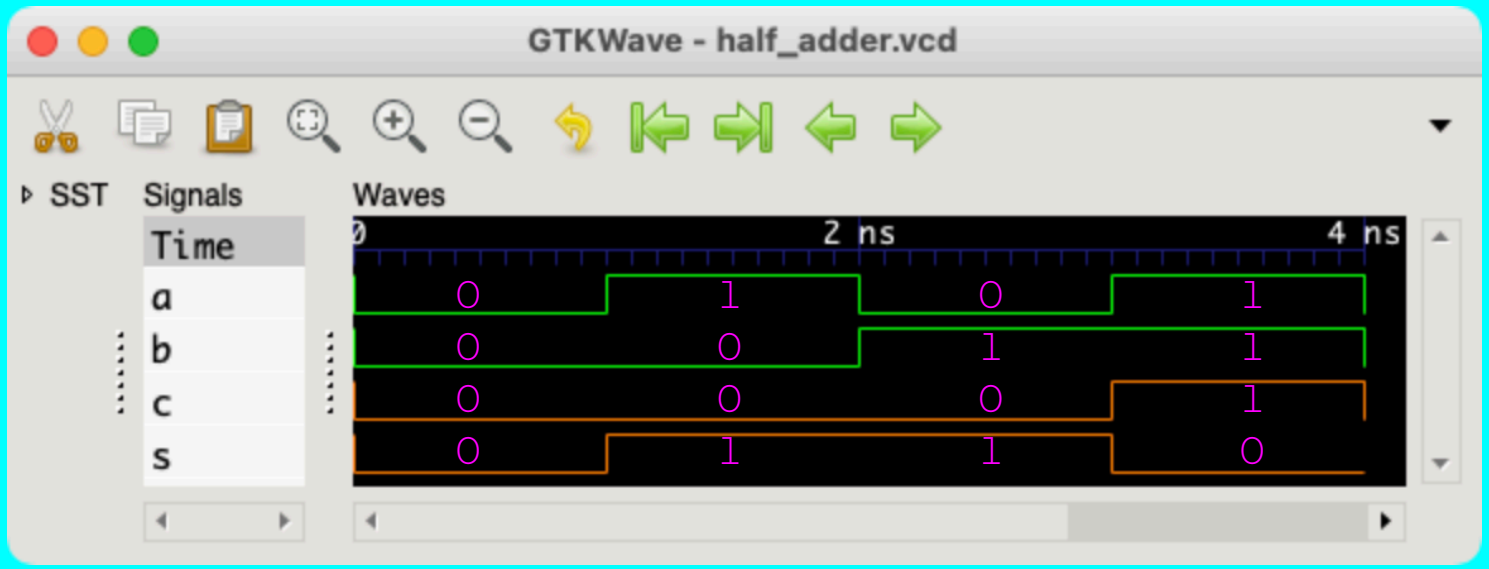
半加算器回路設計 (テストベンチ)

```
‘timescale 1ns/1ns
module half_adder_tb;
    reg  a, b;
    wire c, s;
    half_adder ha (a, b, c, s);
    initial begin
        #0 a = 0;
        #0 b = 0;
        #4 $finish;
    end
    always #1 a = ~a;
    always #2 b = ~b;
    initial begin
        $dumpfile ("half_adder.vcd");
        $dumpvars;
    end
endmodule
```

half_adder_tb.v

半加算器回路設計 (波形)

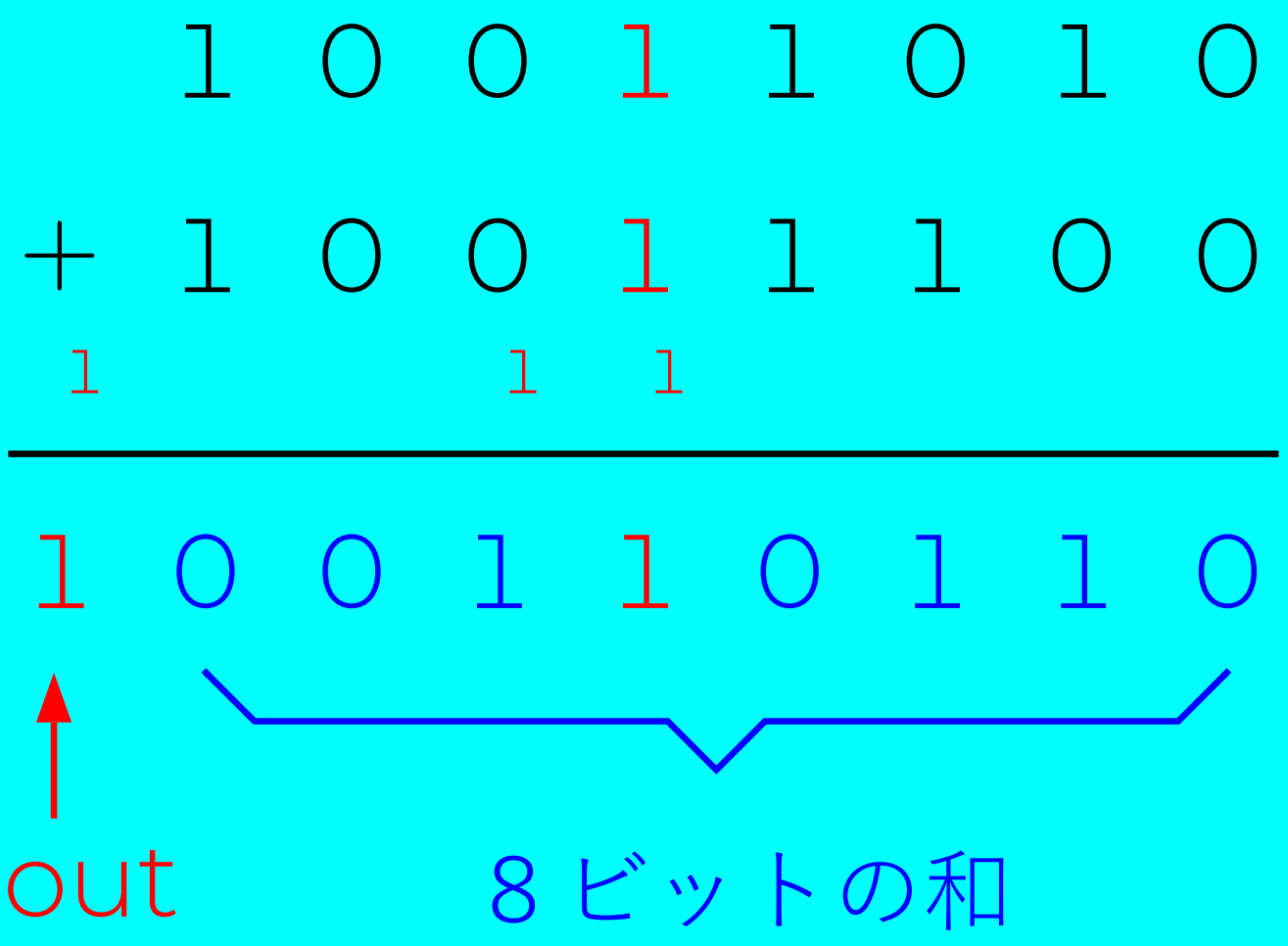
```
% iverilog -Wall -o half_adder \
  half_adder_tb.v half_adder.v
% vvp half_adder
% gtkwave half_adder.vcd
```



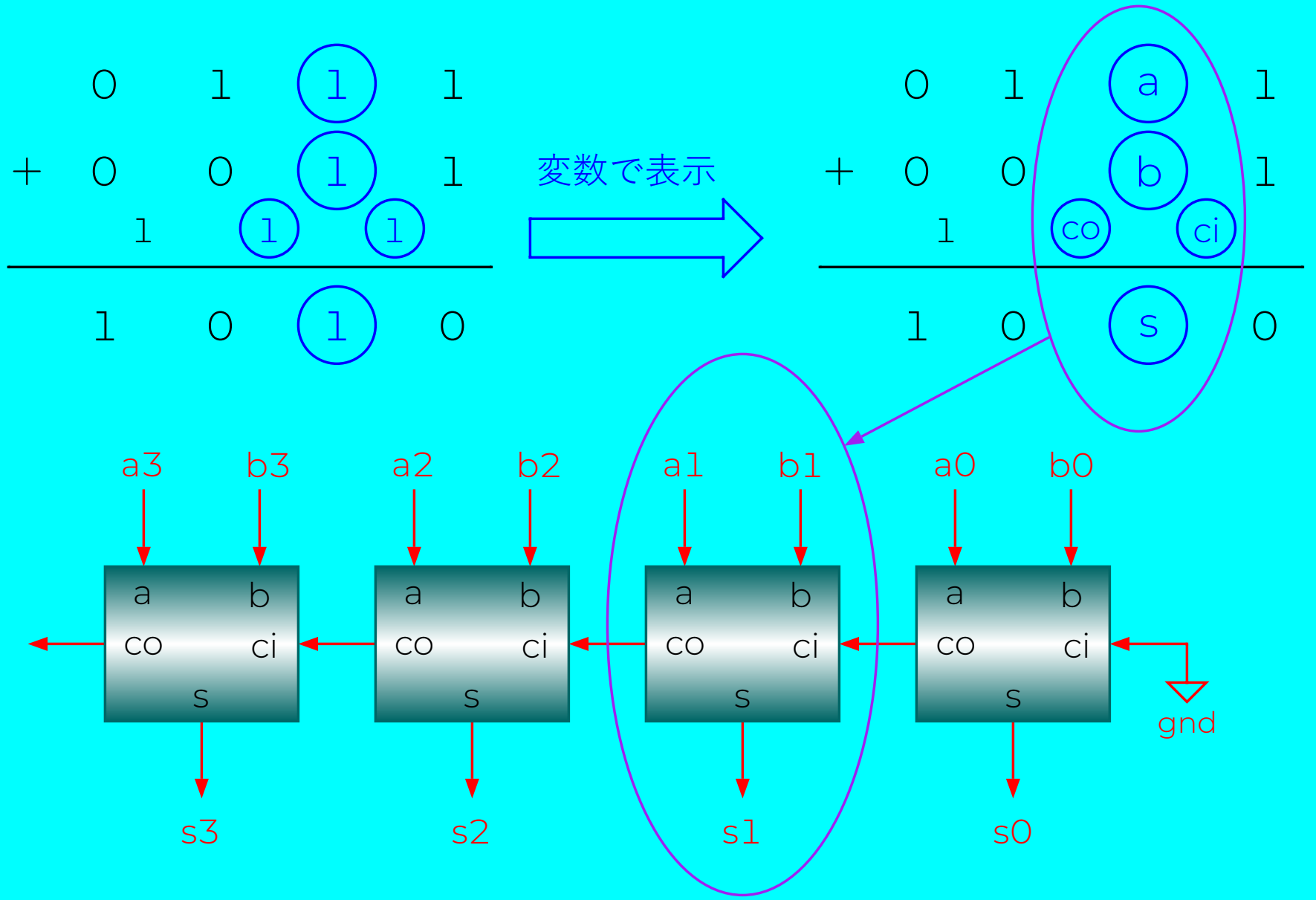
a b cs　a b cs　a b cs　a b cs

0+0=00　1+0=01　0+1=01　1+1=10

2 進数の加算 (8 ビットの例)



全加算器回路設計


0 1 1 1
+ 0 0 1 1
1 1 1
1 0 1 0
変数で表示
0 1 a 1
+ 0 0 b 1
1 co ci
1 0 s 0
a3 b3 a2 b2 a1 b1 a0 b0
a b
co ci
s
gnd
s3 s2 s1 s0

全加算器の真理値表

input a, b, ci; // ci: carry in (下位桁からの繰り上がり)
output co, s; // co: carry out (上位桁への繰り上がり), s: sum (和)

a	b	ci	co	s	コメント
0	0	0	0	0	$0+0+0=0\,0$ (加算)
0	0	1	0	1	$0+0+1=0\,1$ (加算)
0	1	0	0	1	$0+1+0=0\,1$ (加算)
0	1	1	1	0	$0+1+1=1\,0$ (加算)
1	0	0	0	1	$1+0+0=0\,1$ (加算)
1	0	1	1	0	$1+0+1=1\,0$ (加算)
1	1	0	1	0	$1+1+0=1\,0$ (加算)
1	1	1	1	1	$1+1+1=1\,1$ (加算)

$s = \overline{a}\,\overline{b}\,ci + \overline{a}\,b\,\overline{ci} + a\,\overline{b}\,\overline{ci} + a\,b\,ci$ 最小項の論理和
$co = \overline{a}\,b\,ci + a\,\overline{b}\,ci + a\,b\,\overline{ci} + a\,b\,ci$ 最小項の論理和

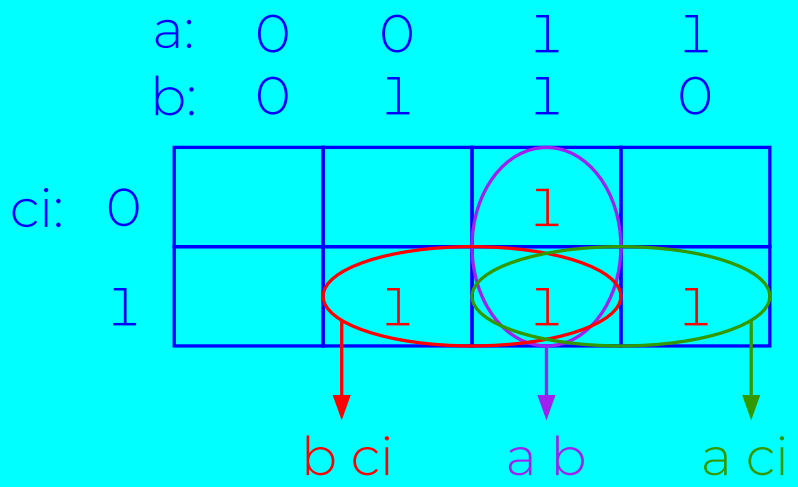
全加算器回路設計

真理値表から論理式をつくる

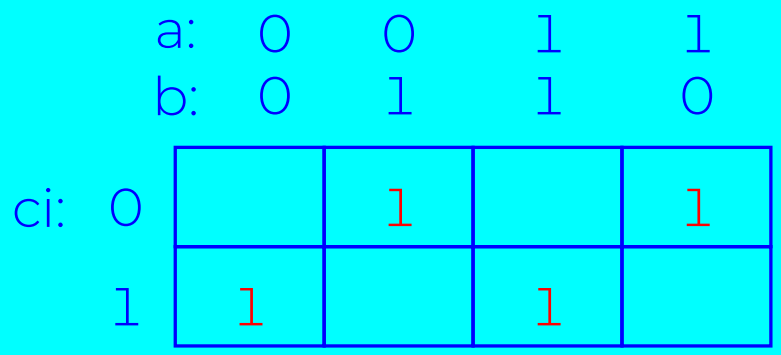
— どの組み合わせで出力が 1 になるか

カルノー図による論理式を簡単化する (co)

co のカルノー図



s のカルノー図



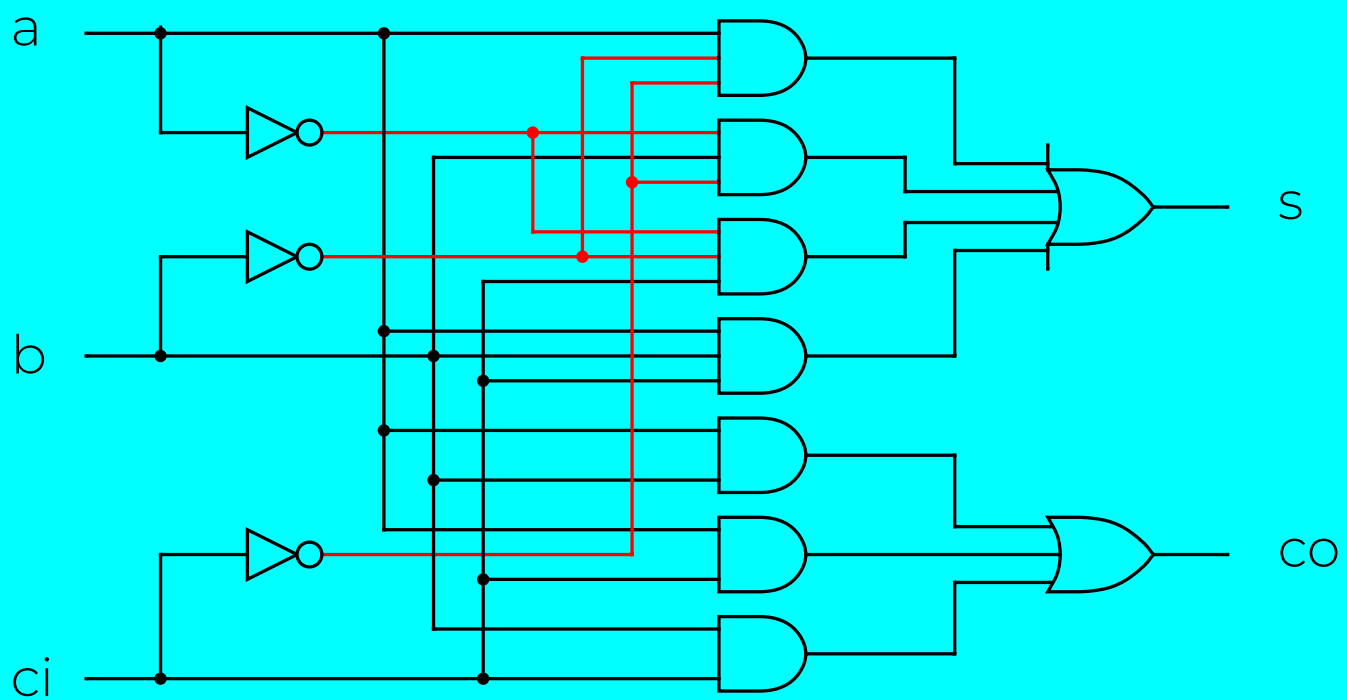
$co = a\,b + a\,ci + b\,ci$ (二つの入力が 11 であれば)

$s = a\,\overline{b}\,\overline{ci} + \overline{a}\,b\,\overline{ci} + \overline{a}\,\overline{b}\,ci + a\,b\,ci$

全加算器回路設計 (回路)

$$co = a\,b + a\,ci + b\,ci$$ (二つの入力が 1 1 であれば)

$$s = a\,\overline{b}\,\overline{ci} + \overline{a}\,b\,\overline{ci} + \overline{a}\,\overline{b}\,ci + a\,b\,ci$$



全加算器回路設計 (回路)

$co = a\,b + a\,ci + b\,ci$ (二つの入力が 1 であれば)

$s = a\,\overline{b}\,\overline{ci} + \overline{a}\,b\,\overline{ci} + \overline{a}\,\overline{b}\,ci + a\,b\,ci$

```
‘timescale 1ns/1ns

module fa (a, b, ci, co, s);
    input  a, b, ci;
    output co, s;

    assign co = a & b | a & ci | b & ci;

    assign s =  a & ~b & ~ci |
               ~a &  b & ~ci |
               ~a & ~b &  ci |
                a &  b &  ci;

endmodule
```

fa.v

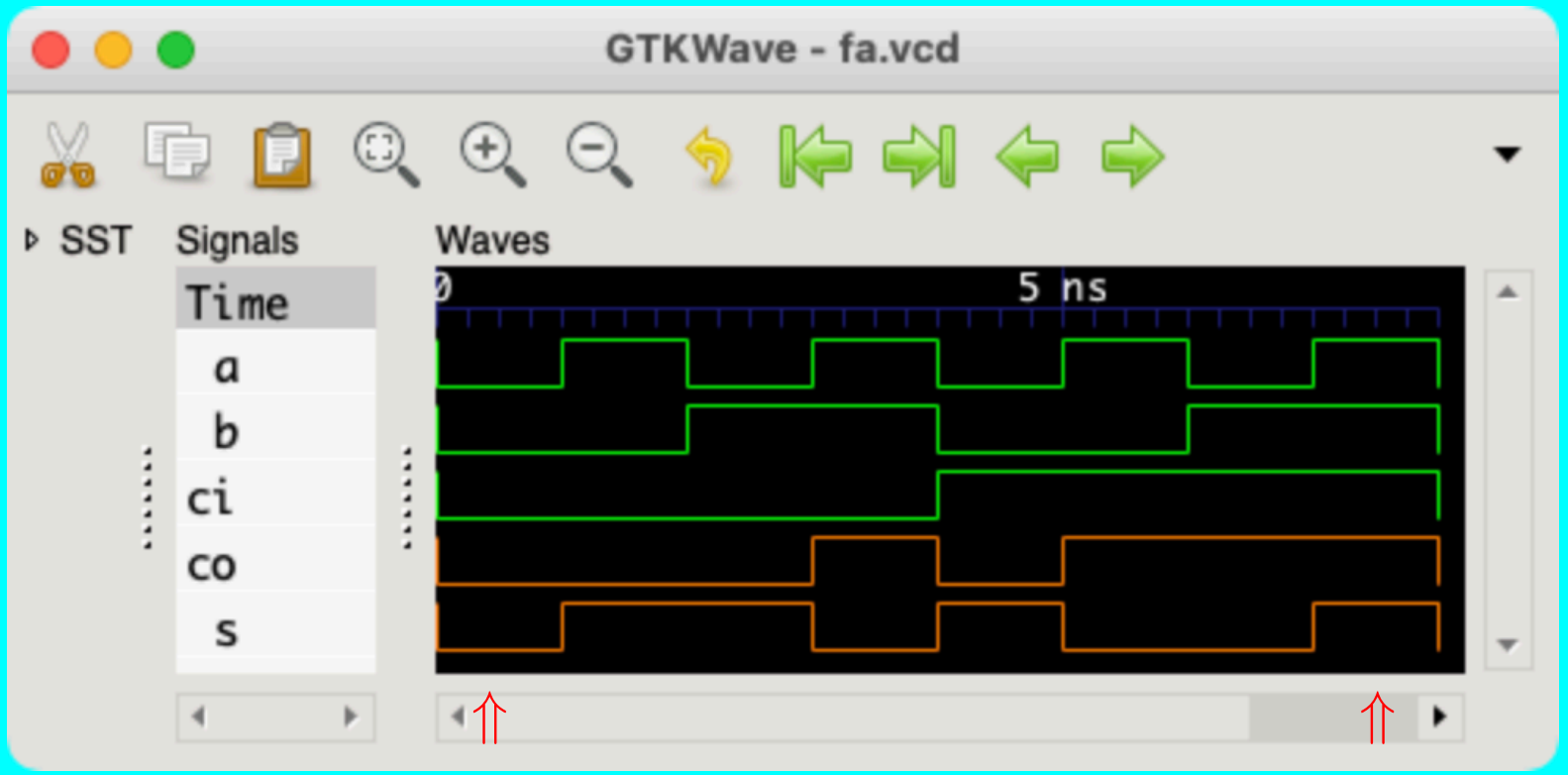
全加算器回路設計 (テストベンチ)

```
‘timescale 1ns/1ns
module fa_tb;
    reg  a, b, ci;
    wire co, s;
    fa fadder (a, b, ci, co, s);
    initial begin
        #0 a  = 0; b  = 0; ci = 0;
        #8 $finish;
    end
    always #1 a  = ~a;
    always #2 b  = ~b;
    always #4 ci = ~ci;
    initial begin
        $dumpfile ("fa.vcd");
        $dumpvars;
    end
endmodule
```

fa_tb.v

全加算器回路設計 (波形)

```
% iverilog -Wall -o fa fa_tb.v fa.v
% vvp fa
% gtkwave fa.vcd
```



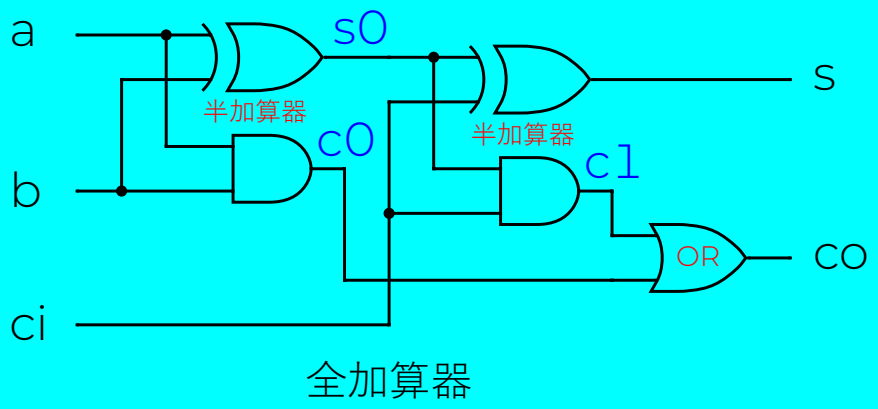
$0+0+0=00_2$ $1+1+1=11_2$

半加算器を用いた全加算器の設計

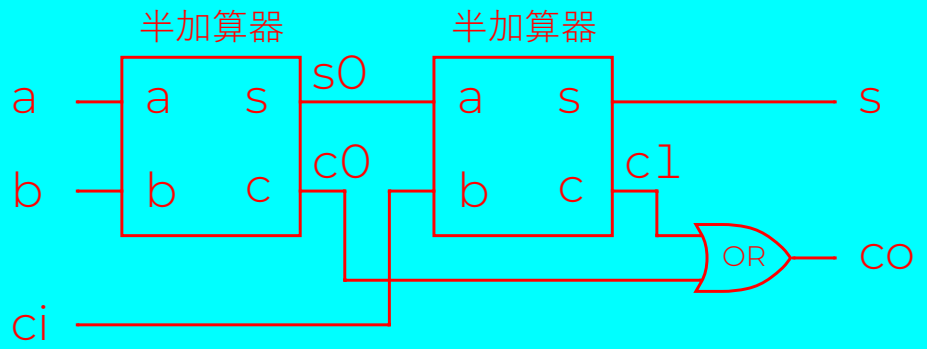
全加算器は、２つの半加算器と１つのOR回路を用いて構成することができる。

$$
\begin{aligned}
s &= a\,\overline{b}\,\overline{ci} + \overline{a}\,b\,\overline{ci} + \overline{a}\,\overline{b}\,ci + a\,b\,ci \\
&= (a \oplus b) \oplus ci = s0 \oplus ci \\
co &= \overline{a}\,b\,ci + a\,\overline{b}\,ci + a\,b\,\overline{ci} + a\,b\,ci \\
&= (\overline{a}\,b + a\,\overline{b})\,ci + a\,b\,(\overline{ci} + ci) \\
&= (a \oplus b)\,ci + a\,b = s0\,ci + a\,b = c1 + c0
\end{aligned}
$$

$$
\begin{aligned}
s0 &= a \oplus b \\
c0 &= a\ \ b \\
c1 &= s0\ \ ci
\end{aligned}
$$



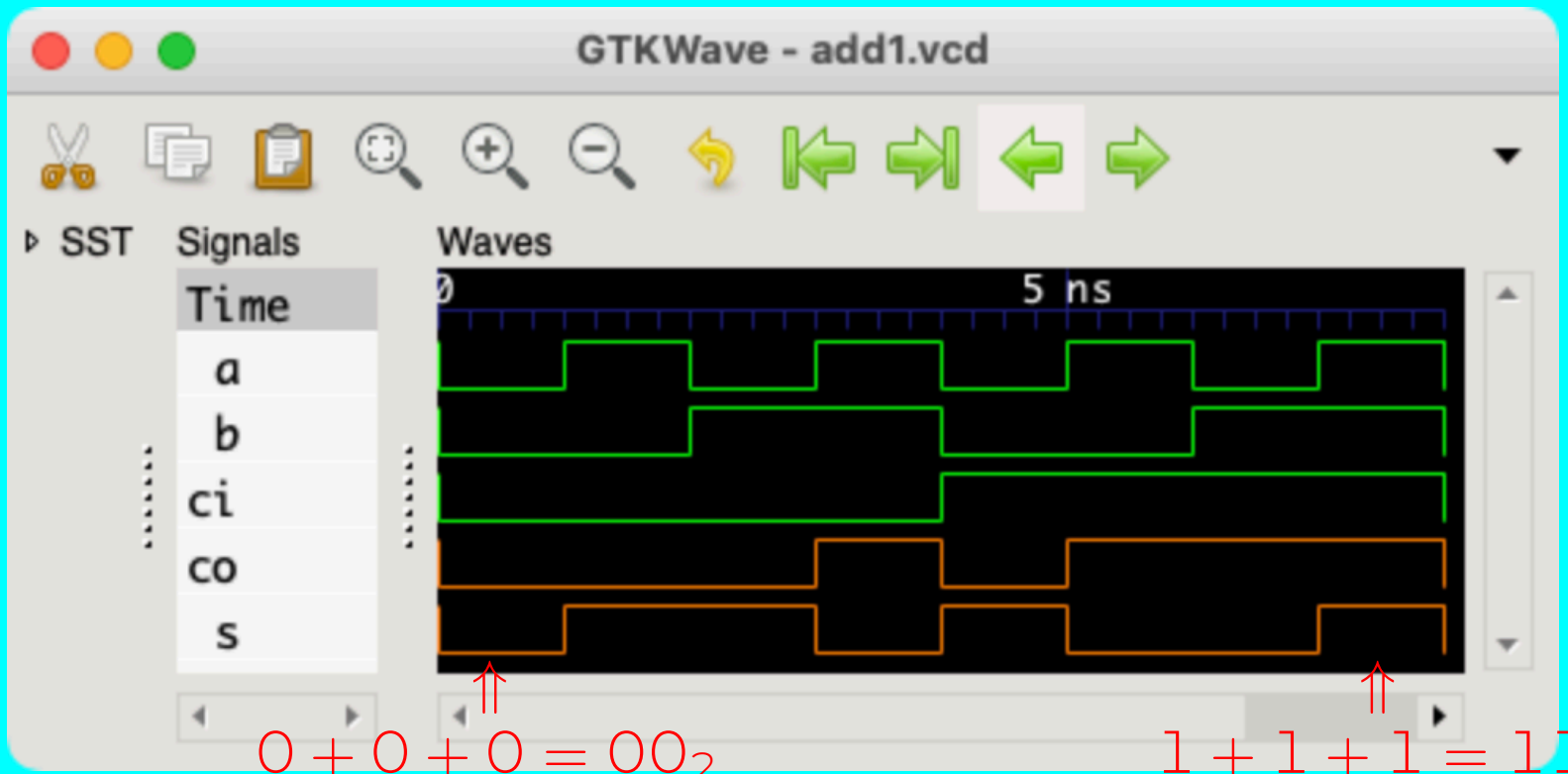
全加算器



全加算器

半加算器を用いた全加算器の波形

```
% iverilog -Wall -o add1 \
  add1_tb.v add1.v half_adder.v
% vvp add1
% gtkwave add1.vcd
```



カルノー図と全加算器

まとめ

- 組み合わせ回路
- 真理値表
- 論理式
- カルノー図
- グレイコード
- 2 進数
- 半加算器
- 全加算器

課題 V (200 点)

問題 1 : $FO = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,B\,\overline{C}\,\overline{D} + A\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,B\,C\,D + \overline{A}\,\overline{B}\,C\,\overline{D} + \overline{A}\,B\,C\,\overline{D} + A\,\overline{B}\,C\,\overline{D}$

をカルノー図で簡単化し、FO と簡単化された論理式 FS の回路を設計し動作検証シミュレーションして下さい。モジュール名は ex5 にすること。テストベンチ ex5_tb.v を使って下さい。

```
‘timescale 1ns/1ns
module ex5 (A, B, C, D, FO, FS);
    input  A, B, C, D;
    output FO, FS;
    assign FO = ~A & ~B & ~C & ~D | ~A &  B & ~C & ~D |
                 A & ~B & ~C & ~D | ~A &  B &  C &  D |
                ~A & ~B &  C & ~D | ~A &  B &  C & ~D |
                 A & ~B &  C & ~D;
    assign FS =                                ; // simplified
endmodule
```

ex5.v

課題 V (200 点)

問題 2 ： 半加算器half_adder.vを設計し動作検証シミュレーションして下さい (P35-P39 を参照)。
テストベンチhalf_adder_tb.vを使って下さい。

そして、half_adder を使用し、全加算器add1.vを設計し動作検証シミュレーションして下さい (P47-P48 を参照)。
テストベンチadd1_tb.vを使って下さい。

```
% iverilog -Wall -o add1 add1_tb.v add1.v half_adder.v
% vvp add1
% gtkwave add1.vcd &
```