論理回路入門（14）

論理回路の復習

李　亜民

2024 年 1 月 9 日 (火)

論理回路入門

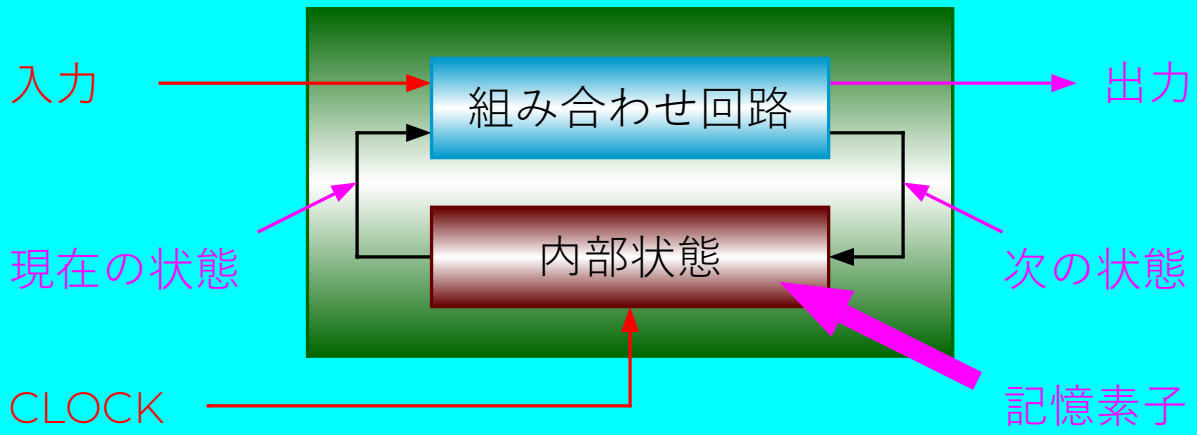
勉強した内容

- 論理回路の基本: ブール代数、真理値表、カルノー図、論理式の積和形と和積形、回路の設計と動作検証の方法
- 組合せ回路: 半加算器、全加算器、リップルキャリーアダー、キャリールックアヘッドアダー、ツリー型桁上げ先見加算器、減算器、乗算器、ウォレス ツリー乗算器、マルチプレクサ、7 セグメント LED、ALU、デコーダ、エンコーダ
- 順序回路: RS ラッチ、D ラッチ、D フリップフロップ、T フリップフロップ、JK フリップフロップ、レジスタ・ファイル、状態遷移図、交通信号機制御システム、N 進カウンター

論理回路の種類

1. 組み合わせ回路 (Combinational Circuit): 現在の入力のみで出力が決まる回路である。

2. 順序回路 (Sequential Circuit): 内部状態と入力信号で出力が決まる回路である。有限状態機械 (Finite state machine — FSM) とも呼ばれる。



基本的な論理演算

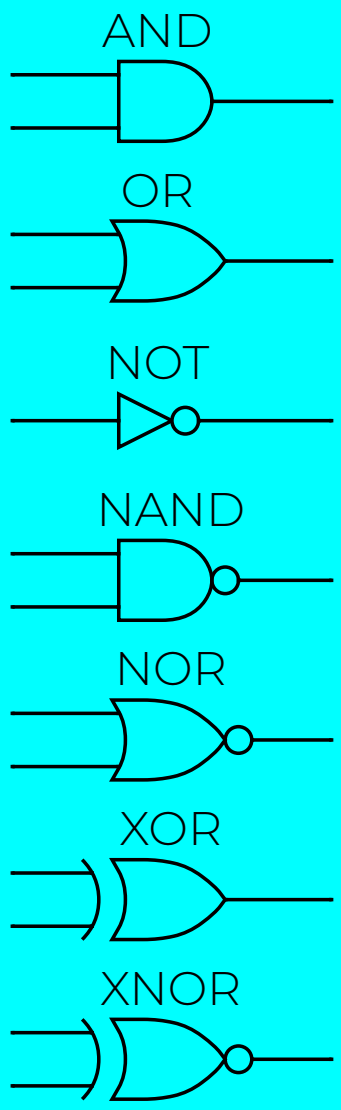
1. AND (アンド) — 論理積 (かつ)
   - 『例』 $F = A \text{ AND } B = A \cdot B = A\ B$
   - 『例』 Verilog HDL の表現：`F = A & B`

2. OR (オア) — 論理和 (または)
   - 『例』 $F = A \text{ OR } B = A + B$
   - 『例』 Verilog HDL の表現：`F = A | B`

3. NOT (ノット) — 否定
   - 『例』 $F = \text{NOT } A = \overline{A}$
   - 『例』 Verilog HDL の表現：`F = ~A`

よく使われた 7 ゲート

(1)	AND:	$F = A \cdot B$	`F = A & B`
(2)	OR:	$F = A + B$	`F = A | B`
(3)	NOT:	$F = \overline{A}$	`F = ~A`
(4)	NAND:	$F = \overline{A \cdot B}$	`F = ~(A & B)`
(5)	NOR:	$F = \overline{A + B}$	`F = ~(A | B)`
(6)	XOR:	$F = A \oplus B$	`F = A ^ B`
(7)	XNOR:	$F = \overline{A \oplus B}$	`F = ~(A ^ B)`



ブール代数の定理

1. $A \cdot A = A$ ……………………… 冪等律
2. $A + A = A$ ……………………… 冪等律
3. $A \cdot B = B \cdot A$ ……………………… 交換律
4. $A + B = B + A$ ……………………… 交換律
5. $(A \cdot B) \cdot C = A \cdot (B \cdot C)$ ……………… 結合律
6. $(A + B) + C = A + (B + C)$ ……………… 結合律
7. $A \cdot (B + C) = (A \cdot B) + (A \cdot C)$ ………… 分配律
8. $A + (B \cdot C) = (A + B) \cdot (A + C)$ ………… 分配律
9. $A \cdot (A + B) = A$ ……………………… 吸収律
10. $A + (A \cdot B) = A$ ……………………… 吸収律

ブール代数の定理

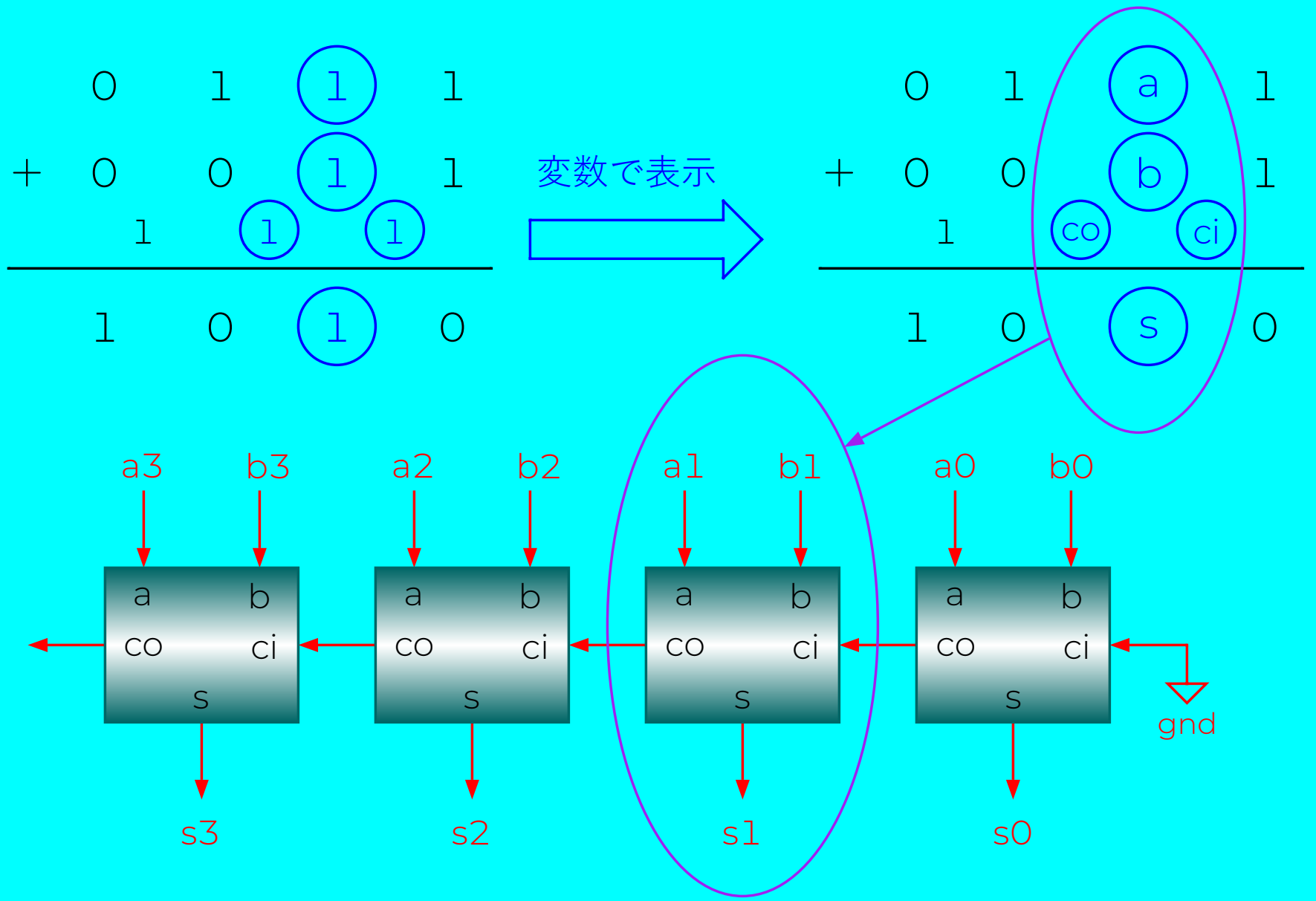
11. $A \cdot 0 = 0$ ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・支配律
12. $A + 1 = 1$ ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・支配律
13. $A \cdot 1 = A$ ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・同一律
14. $A + 0 = A$ ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・同一律
15. $A \cdot \overline{A} = 0$ ・・・・・・・・・・・・・・・・・・・・・・・・・・・補元律
16. $A + \overline{A} = 1$ ・・・・・・・・・・・・・・・・・・・・・・・・・・・補元律
17. $\overline{\overline{A}} = A$ ・・・・・・・・・・・・・・・・・・・・・・・・・・・・対合律

18. $\overline{A \cdot B} = \overline{A} + \overline{B}$ ・・・・・・・・・・・・・ド・モルガンの法則
19. $\overline{A + B} = \overline{A} \cdot \overline{B}$ ・・・・・・・・・・・・・ド・モルガンの法則

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)。

例：全加算器設計


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

例：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

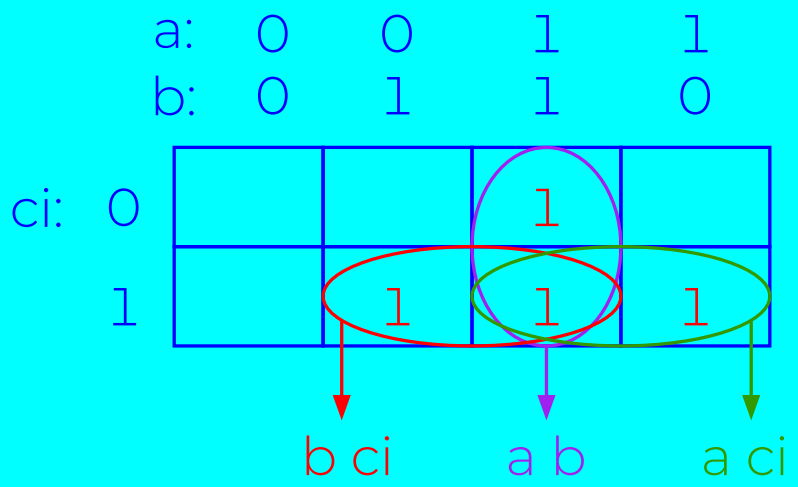
例：全加算器設計

真理値表から論理式をつくる

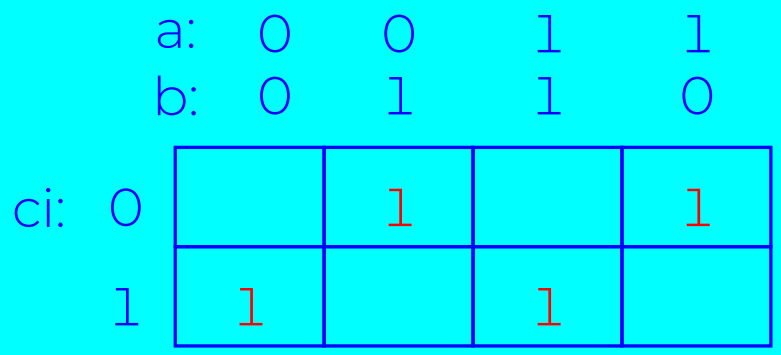
— どの組み合わせで出力が 1 になるか

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



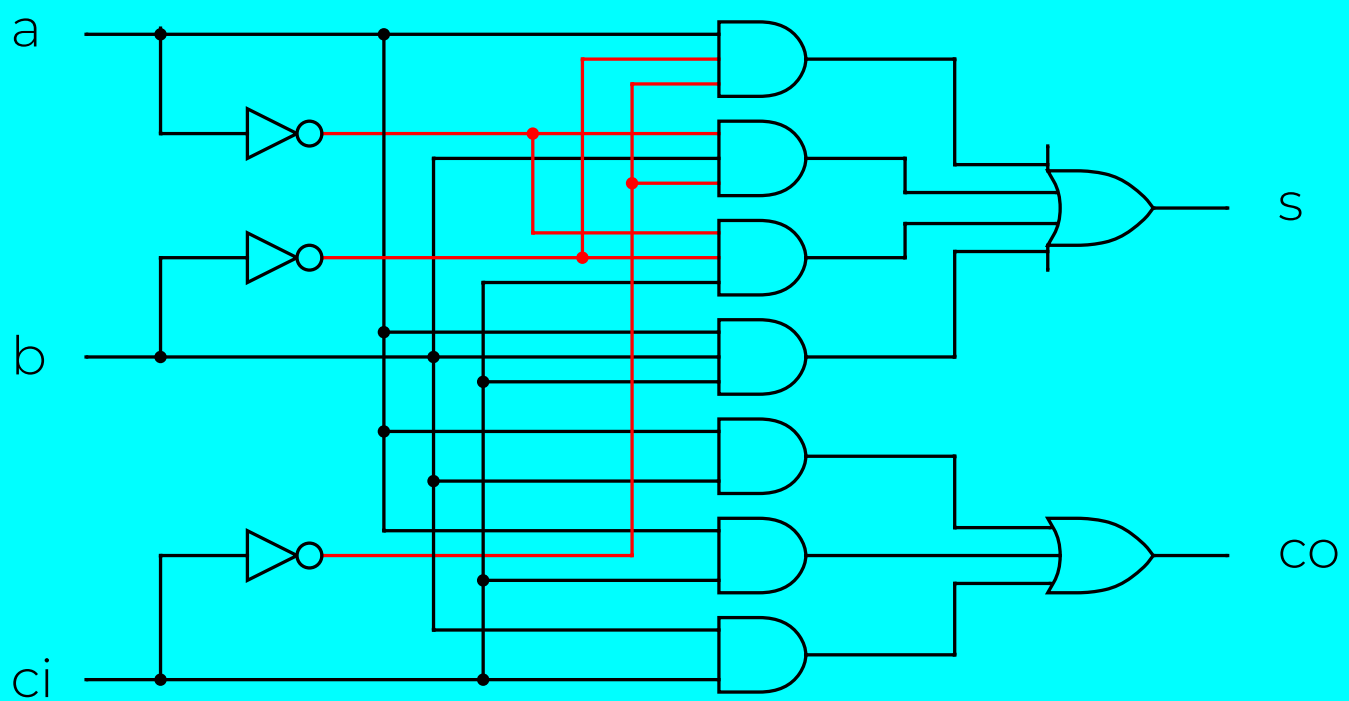
$co = a\,b + a\,ci + b\,ci$ (二つの入力が 1 1 であれば)

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

例：全加算器設計 (回路)

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

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



例：全加算器設計 (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

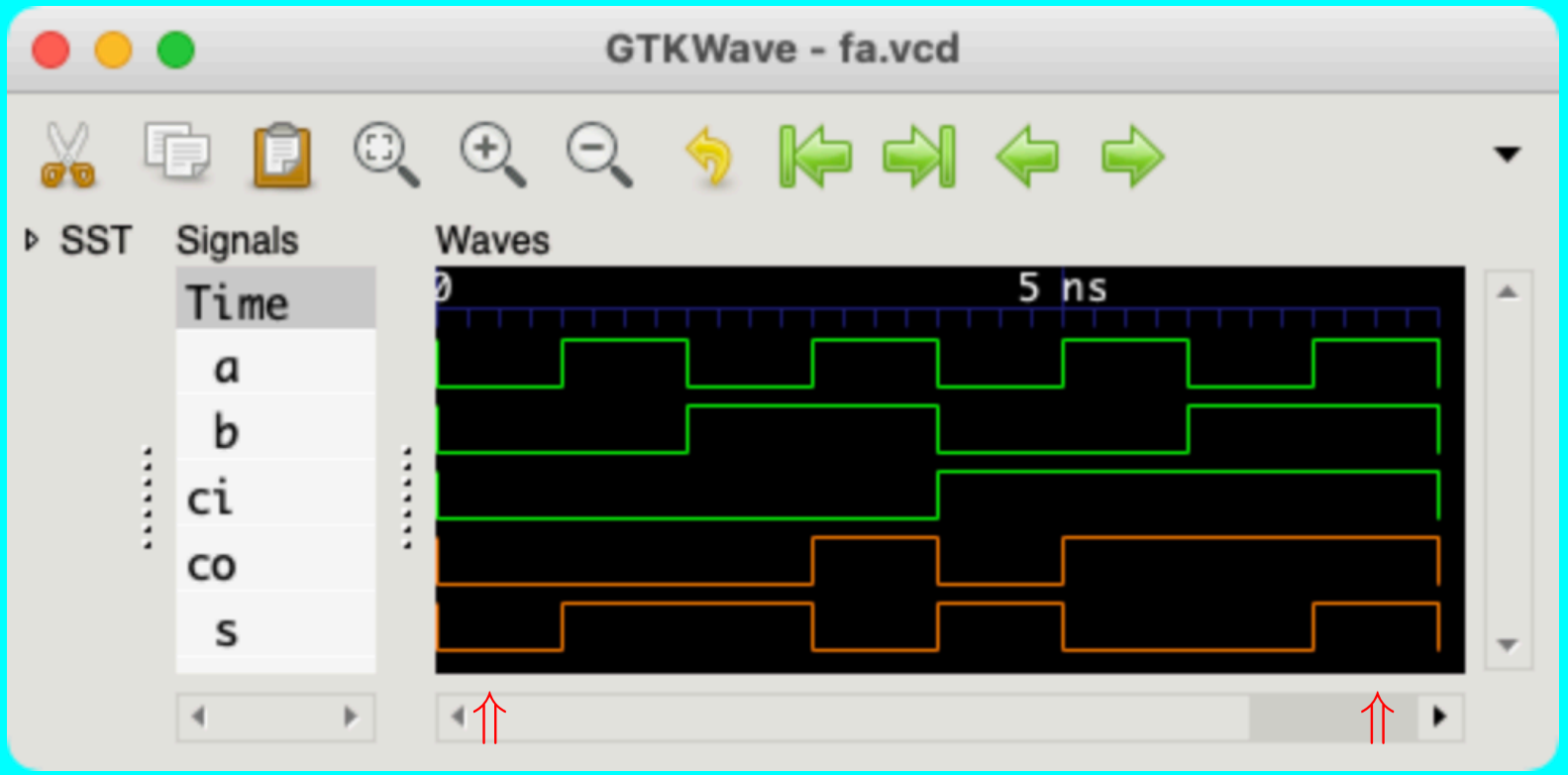
例：全加算器設計 (テストベンチ)

```
‘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

例：全加算器設計 (波形)

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



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

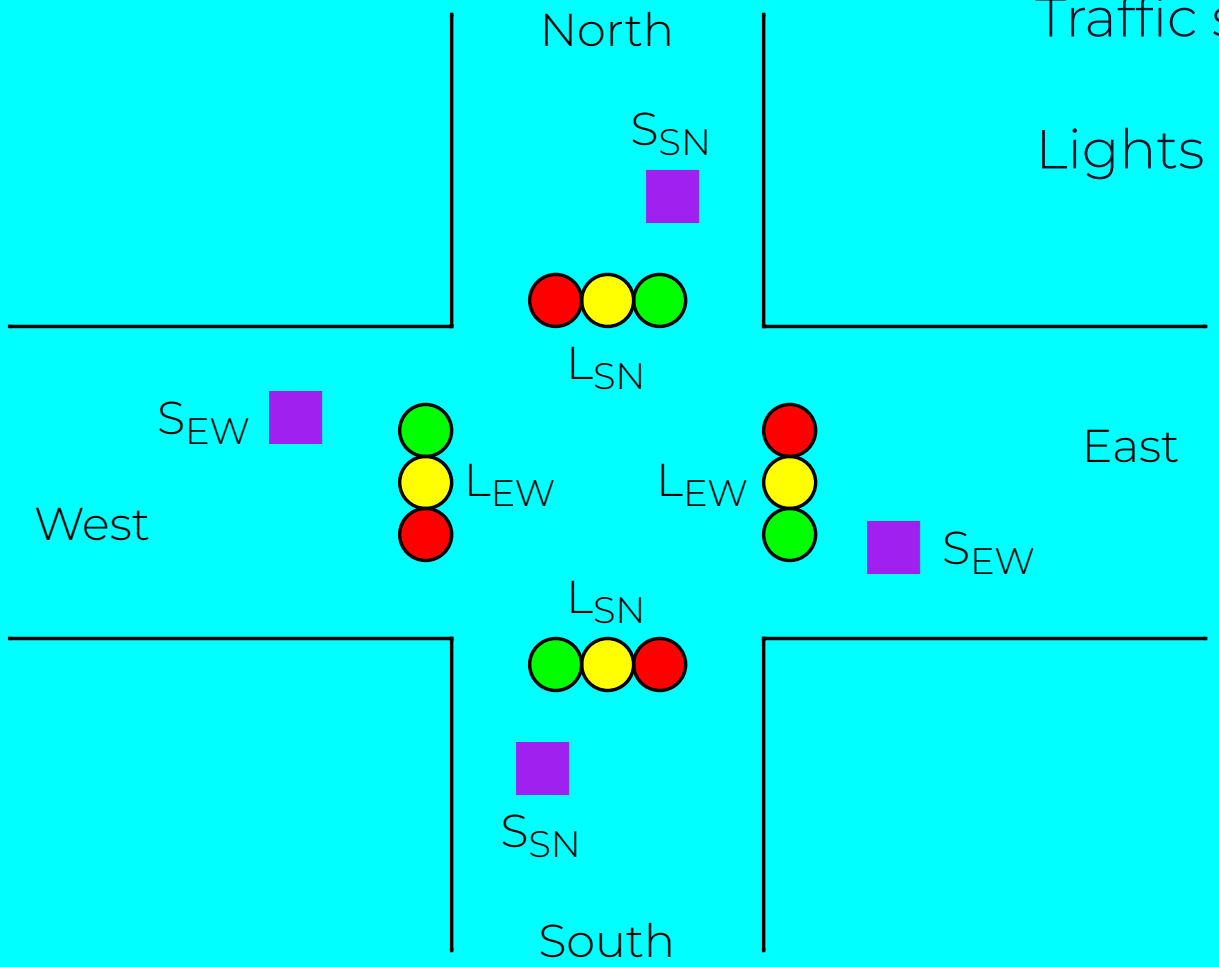
他の組合せ回路

1. 半加算器
2. リップルキャリーアダー
3. キャリールックアヘッドアダー
4. ツリー型桁上げ先見加算器
5. 減算器
6. 乗算器
7. ウォレス ツリー乗算器
8. マルチプレクサ
9. 7 セグメント LED
10. ALU
11. デコーダ（2-4、3-8、4-16）
12. エンコーダ（4-2、8-3、16-4）、プライオリティエンコーダ

順序回路設計の手順

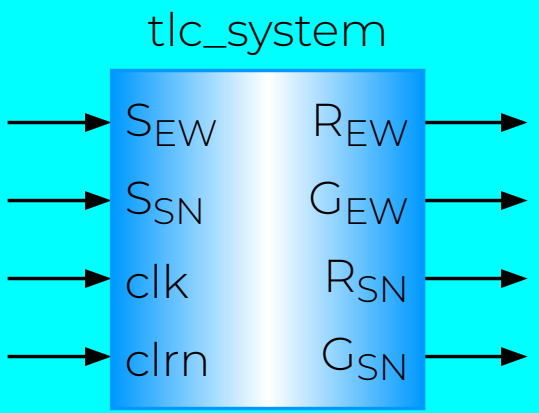
1. 問題を理解する。
2. 状態遷移図をつくる。
3. FF の数を決める ($n = \lceil \log_2 N \rceil$、N は状態の数)。
4. 各状態に n ビットの番号を付け、真理値表をつくる。
   - 次の状態の真理値表をつくる。
   - 出力関数の真理値表をつくる。
5. 真理値表から論理式をつくる (どの条件で出力が 1 になるか)。
   - カルノー図を用いて論理式を簡単化する。
6. 論理式から回路をつくる。
7. テストベンチをつくる (シミュレーションするため)。
8. 回路をシミュレーションする (回路設計の正当性検証)。
9. 与えられた回路の動作を理解する (波形の説明)。

例：交通信号機制御システム（問題）



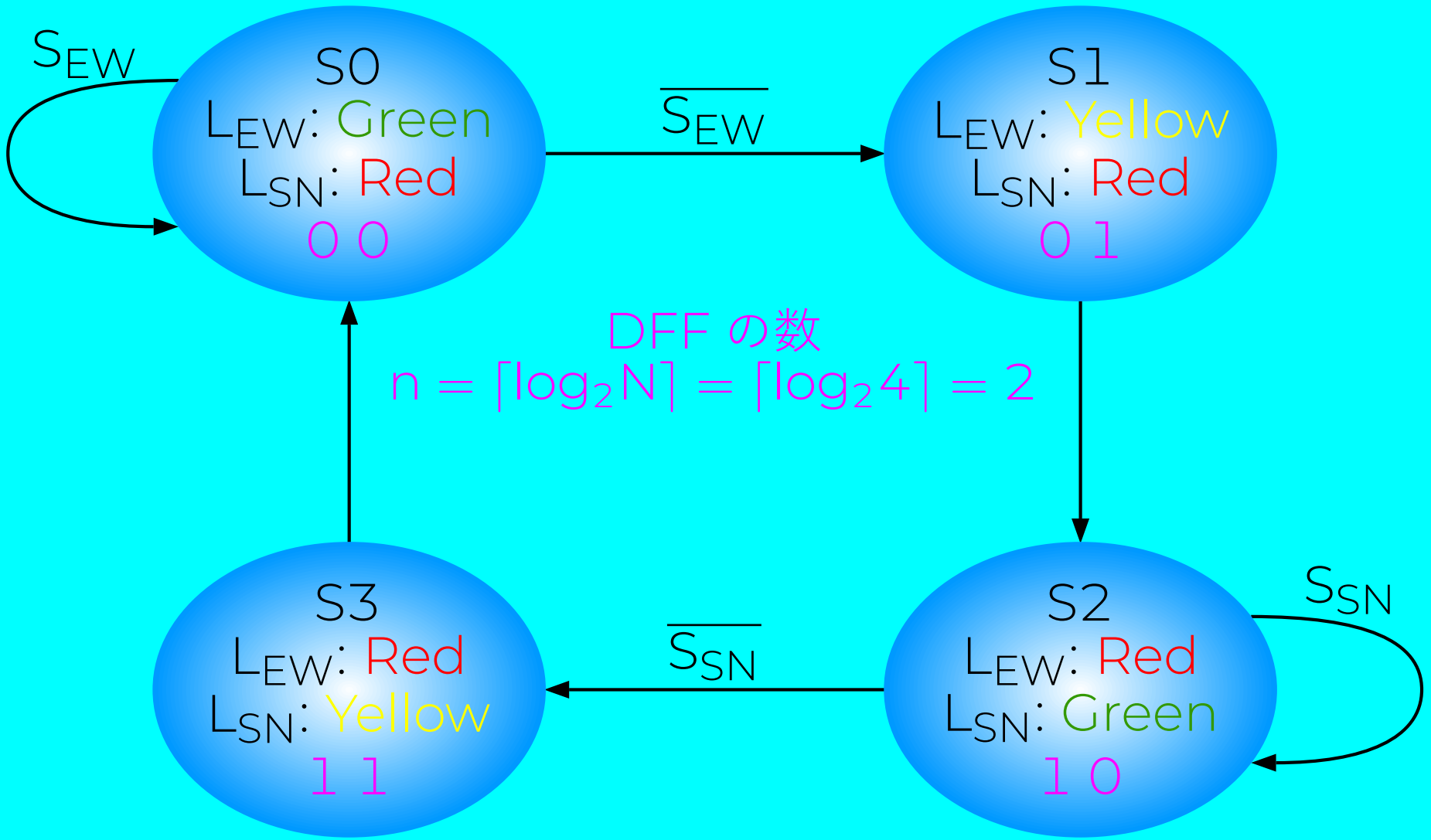
Traffic sensors S_{EW} and S_{SN}
(1 if there is traffic)
Lights L_{EW} and L_{SN}

Rxx	Gxx	Color
0	1:	Green
1	0:	Red
1	1:	Yellow



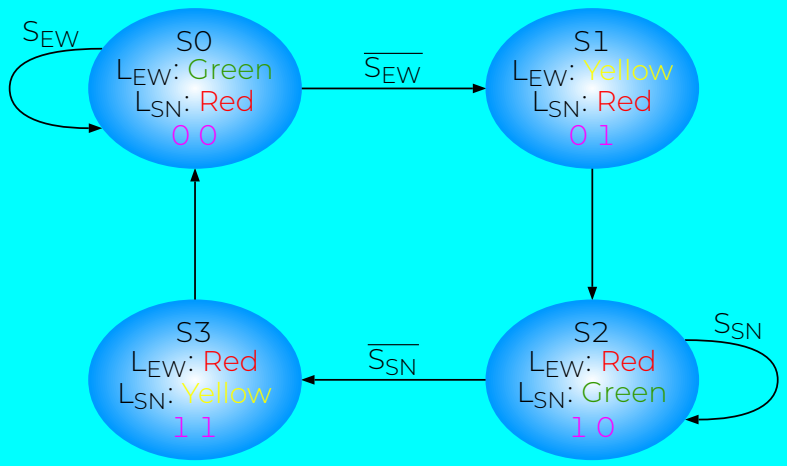
EW: East and West (東西)
SN: South and North (南北)

例：交通信号機制御（状態遷移図）



例：交通信号機制御（次の状態）

次の状態の真理値表
(状態遷移表)



現在の状態			入力		次の状態		
	Q1	Q0	S_{EW}	S_{SN}		D1	D0
S0	0	0	0	X	S1	0	1
			1	X	S0	0	0
S1	0	1	X	X	S2	1	0
S2	1	0	X	0	S3	1	1
			X	1	S2	1	0
S3	1	1	X	X	S0	0	0

例：交通信号機制御（次の状態）

現在の状態			入力		次の状態		
	Q1	Q0	S_{EW}	S_{SN}		D1	D0
S0	0	0	0	X	S1	0	1
			1	X	S0	0	0
S1	0	1	X	X	S2	1	0
S2	1	0	X	0	S3	1	1
			X	1	S2	1	0
S3	1	1	X	X	S0	0	0

次の状態の論理式

$$D1 = \overline{Q1} \cdot Q0 + Q1 \cdot \overline{Q0}$$

$$D0 = \overline{Q1} \cdot \overline{Q0} \cdot \overline{S_{EW}} + Q1 \cdot \overline{Q0} \cdot \overline{S_{SN}}$$

D1

Q1 Q0	0 0	0 1	1 1	1 0
S_{EW} S_{SN} 0 0		1		1
0 1		1		1
1 1		1		1
1 0		1		1

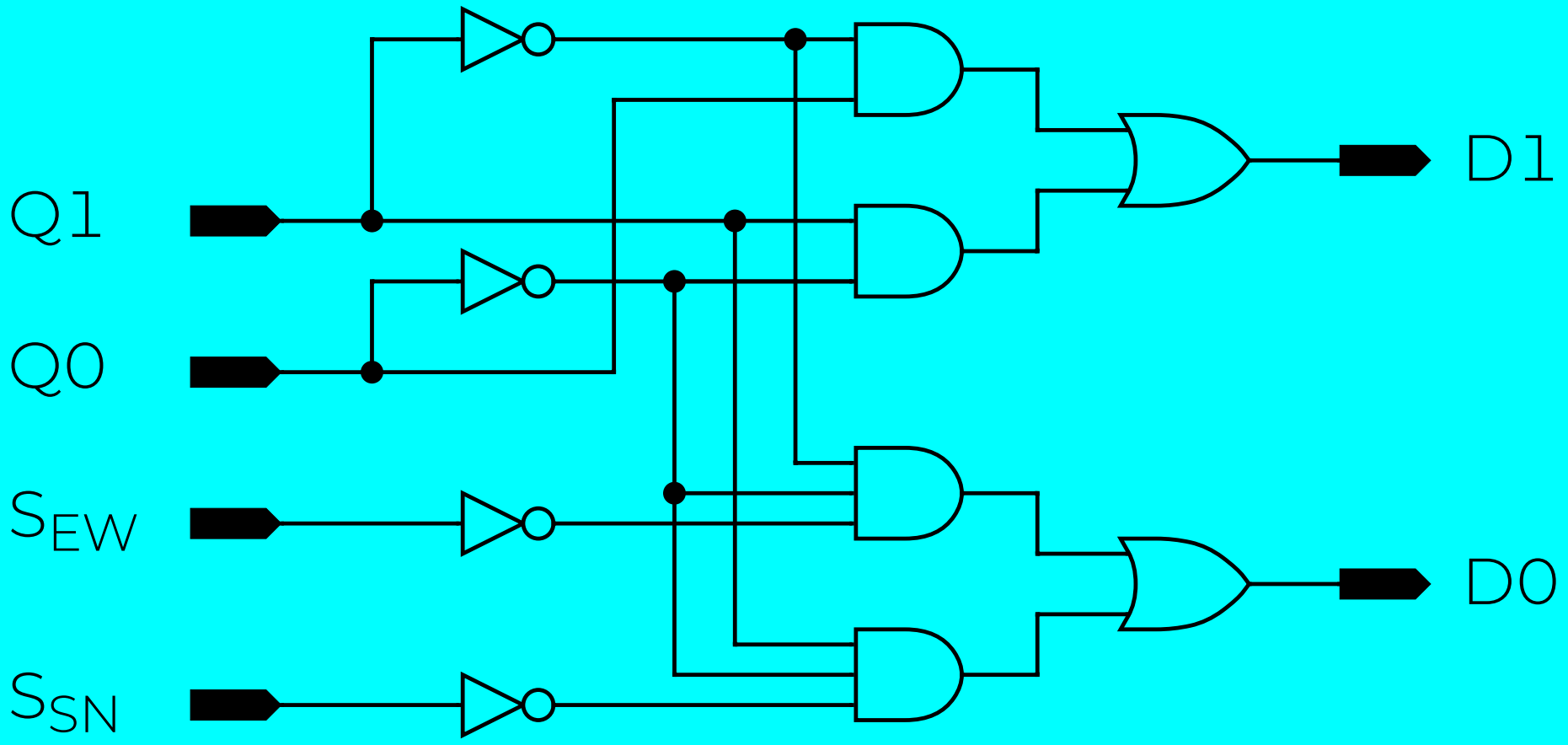
D0

Q1 Q0	0 0	0 1	1 1	1 0
S_{EW} S_{SN} 0 0	1			1
0 1	1			
1 1				
1 0				1

例：交通信号機制御（次の状態の回路）

$$D1 = \overline{Q1} \cdot Q0 + Q1 \cdot \overline{Q0}$$

$$D0 = \overline{Q1} \cdot \overline{Q0} \cdot \overline{S_{EW}} + Q1 \cdot \overline{Q0} \cdot \overline{S_{SN}}$$



例：交通信号機制御（出力関数）

出力関数の真理値表

S0: L_{EW}: Green; L_{SN}: Red
S1: L_{EW}: Yellow; L_{SN}: Red
S2: L_{EW}: Red; L_{SN}: Green
S3: L_{EW}: Red; L_{SN}: Yellow

Color encoding:

R	G	Color
0	1	Green
1	0	Red
1	1	Yellow

現在の状態			出力			
	Q1	Q0	R_{EW}	G_{EW}	R_{SN}	G_{SN}
S0	0	0	0	1	1	0
S1	0	1	1	1	1	0
S2	1	0	1	0	0	1
S3	1	1	1	0	1	1

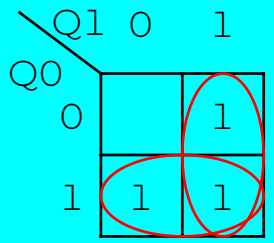
例：交通信号機制御（出力関数）

現在の状態			出力			
	Q1	Q0	R_{EW}	G_{EW}	R_{SN}	G_{SN}
S0	0	0	0	1	1	0
S1	0	1	1	1	1	0
S2	1	0	1	0	0	1
S3	1	1	1	0	1	1

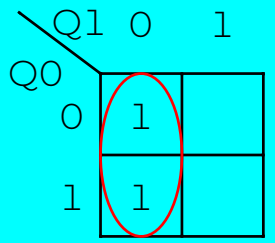
出力関数の論理式

$$R_{EW} = Q1 + Q0$$
$$G_{EW} = \overline{Q1}$$
$$R_{SN} = \overline{Q1} + Q0$$
$$G_{SN} = Q1$$

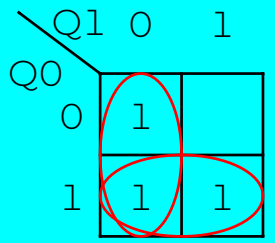
R_{EW}



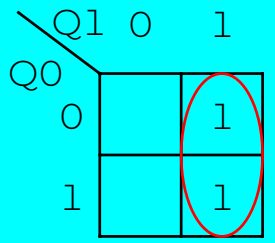
G_{EW}



R_{SN}



G_{SN}



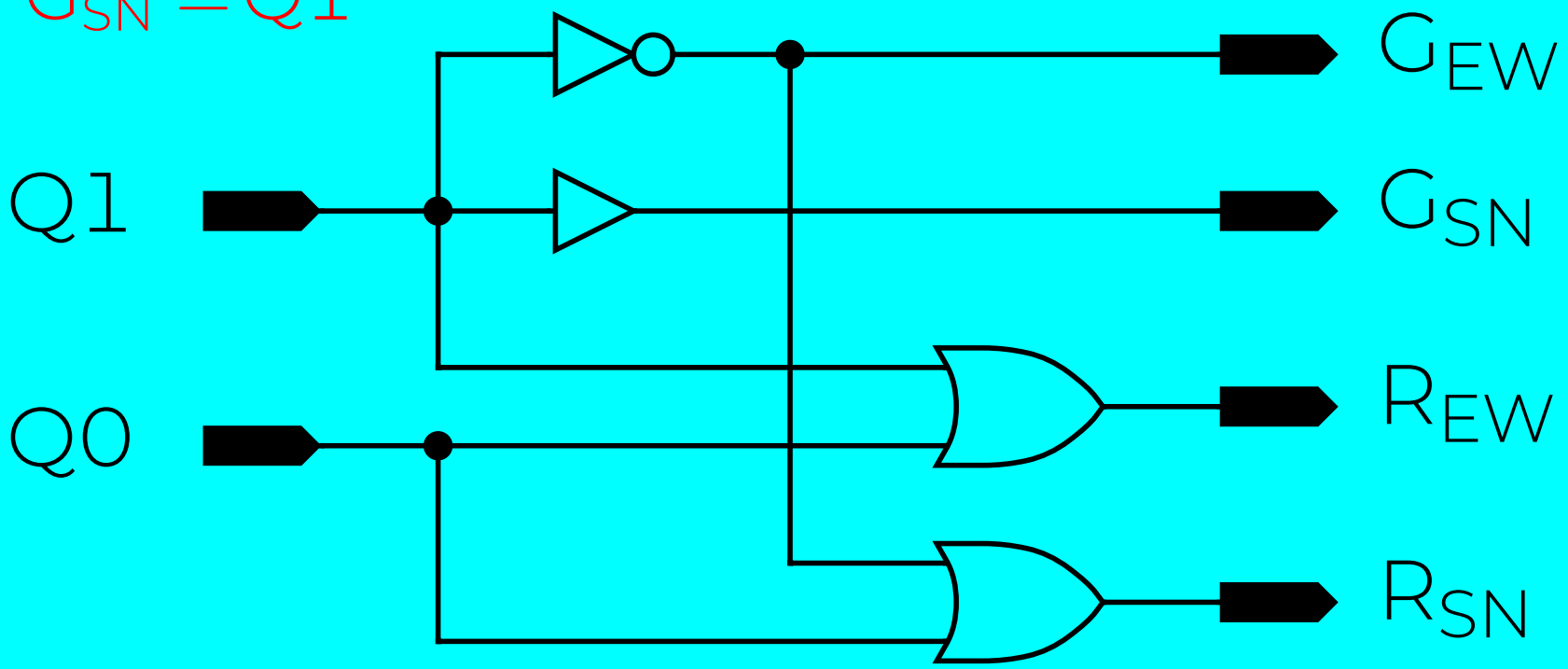
例：交通信号機制御（出力関数の回路）

$$R_{EW} = Q1 + Q0$$

$$G_{EW} = \overline{Q1}$$

$$R_{SN} = \overline{Q1} + Q0$$

$$G_{SN} = Q1$$



例：交通信号機制御（回路）

```
‘timescale 1ns/1ns
module tlc_system (CLK, CLRN, SEW, SSN, REW, GEW, RSN, GSN);
    input  CLK, CLRN, SEW, SSN;
    output REW, GEW, RSN, GSN; // outputs

    reg    [1:0] Q; // current state
    wire   [1:0] D; // next state
    // 1. next state ------------------------------------ 1
    assign D[1] = ~Q[1] &  Q[0] | Q[1] & ~Q[0];
    assign D[0] = ~Q[1] & ~Q[0] & ~SEW | Q[1] & ~Q[0] & ~SSN;
    // 2. outputs --------------------------------------- 2
    assign REW =  Q[1] | Q[0];
    assign GEW = ~Q[1];
    assign RSN = ~Q[1] | Q[0];
    assign GSN =  Q[1];
    // 3. DFFs ------------------------------------------ 3
    always @(posedge CLK or negedge CLRN) begin // DFFs
        if (CLRN == 0) begin
            Q <= 0;
        end else begin
            Q <= D;
        end
    end
endmodule
```

tlc_system.v

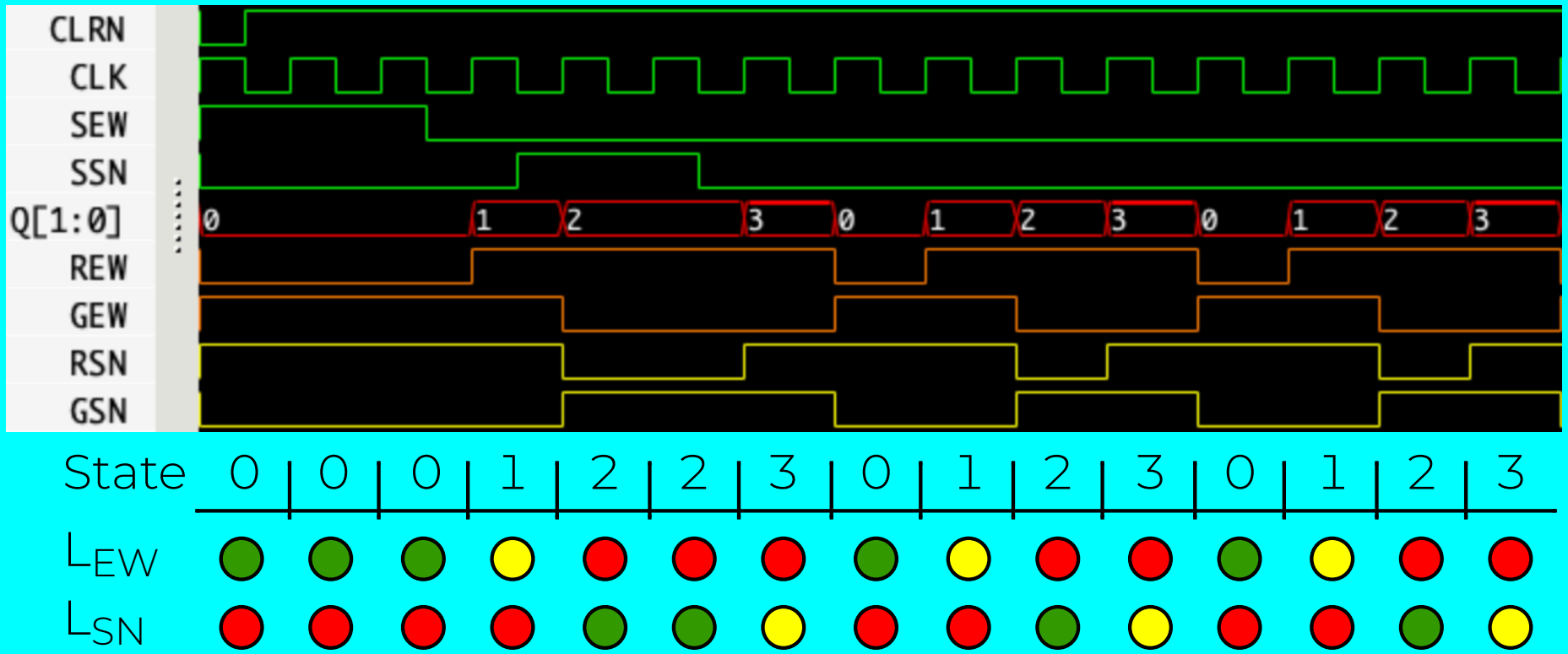
例：交通信号機制御（テストベンチ）

```
‘timescale 1ns/1ns
module tlc_system_tb;
    reg  CLK, CLRN, SEW, SSN;
    wire REW, GEW, RSN, GSN;

    tlc_system i0 (CLK, CLRN, SEW, SSN, REW, GEW, RSN, GSN);

    initial begin
        #0 CLK  = 1; CLRN = 0; SEW = 1; SSN = 0;
        #1 CLRN = 1;
        #4 SEW  = 0;
        #2 SSN  = 1;
        #4 SSN  = 0;
        #22 $finish;
    end

    always #1 CLK = ~CLK;

    initial begin
        $dumpfile ("tlc_system.vcd");
        $dumpvars;
    end

endmodule
```

tlc_system_tb.v

例：交通信号機制御（波形）

```
% iverilog -Wall -o tlc_system tlc_system_tb.v tlc_system.v
% ./tlc_system
% gtkwave tlc_system.vcd
```



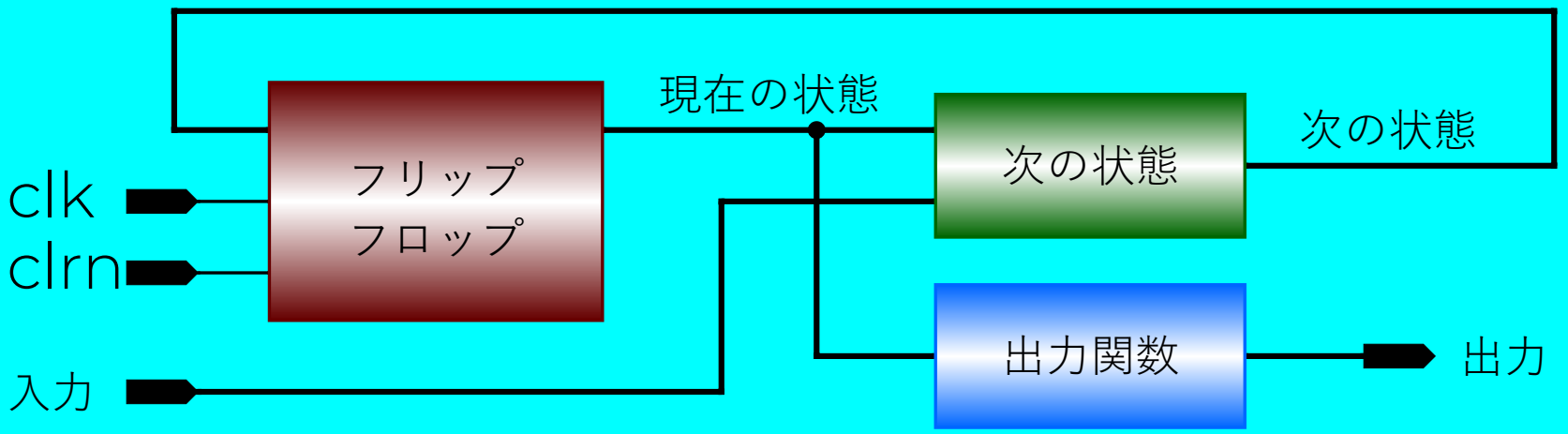
この回路では、どの方向からも車が来ないとき、信号機の色が変わり続けたり、一方から車が来続けると、たとえもう一方に多くの車があってもずっと信号が変わらなかったりするという問題があった。

同期式N進カウンターの設計

7セグメントLEDを使い同期式N進（0 ～ N − 1）カウントアップ (u = 1)/カウントダウン (u = 0) する回路をXFFを用いて設計せよ (Don't Care 項を利用)。7セグメントLED: 0: 点灯 1: 消灯。

N = 3, 4, 6, 8, 10, 16

X = D, JK, T



課題 XIV (200 点)

問題１：３入力多数決回路を設計して下さい（EDA ツールを使用しないこと）。

(1) ３入力多数決回路の真理値表

入力			出力	
A	B	C	Y	コメント
0	0	0	0	全員一致
0	0	1		多数決
0	1	0		多数決
0	1	1		多数決
1	0	0		多数決
1	0	1		多数決
1	1	0		多数決
1	1	1	1	全員一致

課題 XIV (200 点)

(2) Y の論理式 (最小項の論理和)

Y =

(3) Y のカルノー図

(4) Y の論理式 (簡単化された論理式)

Y =

(5) 回路図

(6) 回路の Verilog HDL code

(7) 予測波形

	0ns	1ns	2ns	3ns	4ns	5ns	6ns	7ns	8ns
A									
B									
C									
Y									

課題 XIV (200 点)

問題 2 ： 7 セグメント LED を使い同期式 16 進カウントアップ / カウントダウンする回路を、DFF を用いて設計して下さい（EDA ツールを使用しないこと）。

- 入力信号: clk, clrn, u
- カウント範囲: 0 ～ f
- u = 1 (カウントアップ):
  0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f, 0, 1, 2, 3, ···
- u = 0 (カウントダウン):
  0, f, e, d, c, b, a, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, f, e, d, ···
- 出力信号: seg[6:0] (7 セグメント LED, 0: 点灯; 1: 消灯)

課題 XIV (200 点)

(1) 状態遷移図

(2) 次の状態 next_state の真理値表

(3) 次の状態 next_state 各出力信号のカルノー図

(4) 次の状態 next_state の論理式

n[3] =

n[2] =

n[1] =

n[0] =

(5) 次の状態 next_state の回路図

(6) 次の状態 next_state 回路の Verilog HDL code

課題 XIV (200 点)

(7) 出力関数 seven_seg の真理値表

(8) 出力関数 seven_seg 各出力信号のカルノー図

(9) 出力関数 seven_seg の論理式

seg[0] =
seg[1] =
seg[2] =
seg[3] =
seg[4] =
seg[5] =
seg[6] =

(10) 出力関数 seven_seg の回路図

(11) 出力関数 seven_seg 回路の Verilog HDL code

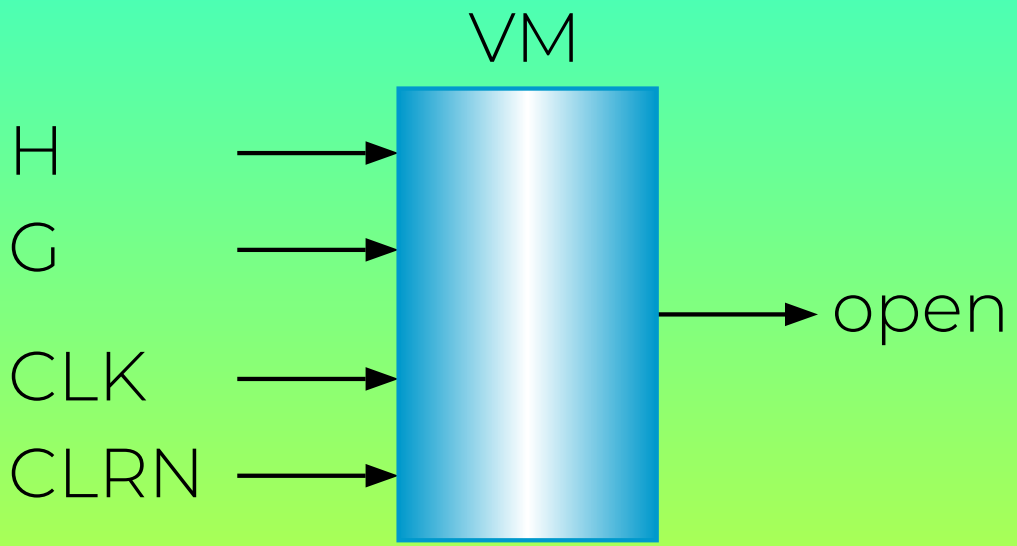
課題 XIV (200 点)

(12) 予測波形

- clrn
- clk
- u
- q (状態)
- seg[0]
- seg[1]
- seg[2]
- seg[3]
- seg[4]
- seg[5]
- seg[6]

発展：自由練習

自動販売機の制御回路を設計し動作検証シミュレーションして下さい（EDA ツールを使用すること）。



H: ひゃく

G: ごじゅう

(1) 150 円のジュースが買える (open=1)

(2) 50 円玉 (G=1) と 100 円玉 (H=1) のみ利用できる

(3) お釣りは出ない

終わり