論理回路入門（１１）

順序回路２：有限状態機械と順序回路設計

李　亜民

2024 年 12 月 03 日 (火)

順序回路

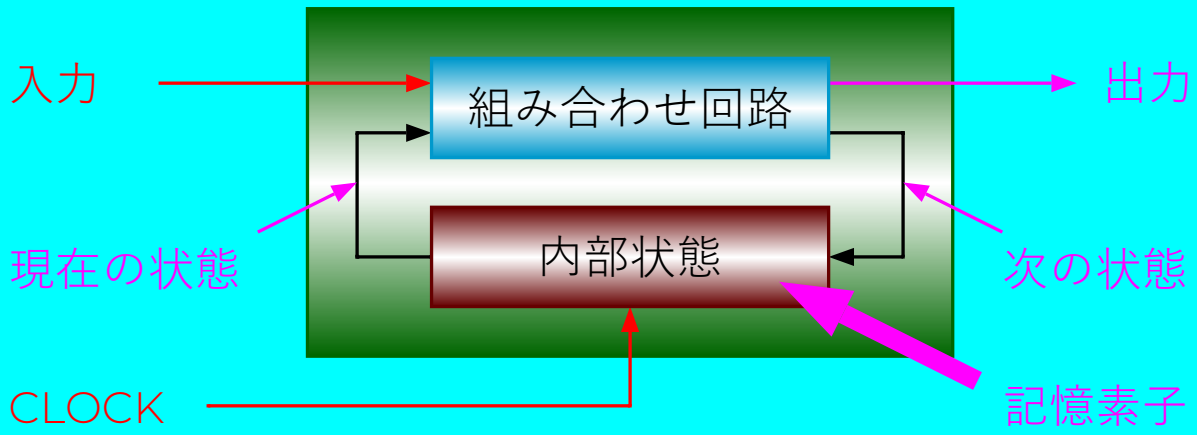
ポイント

- 順序回路の基本構造
- 順序回路の分析
- Mealy 型と Moore 型順序回路
- 順序回路設計の手順
- 状態遷移図
- 次の状態の真理値表、論理式、および回路
- 出力関数の真理値表、論理式、および回路
- 有限状態機械

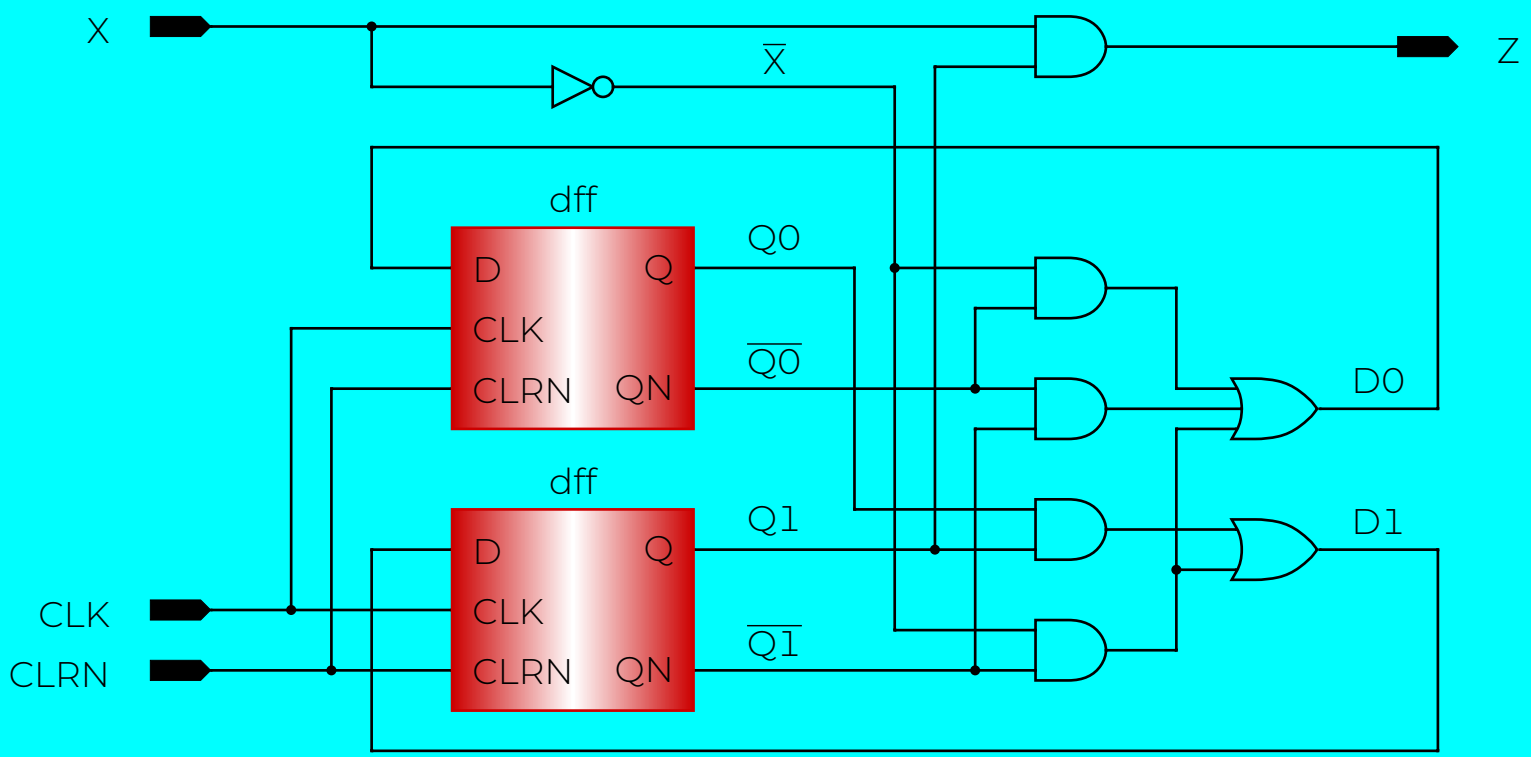
論理回路の種類

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。



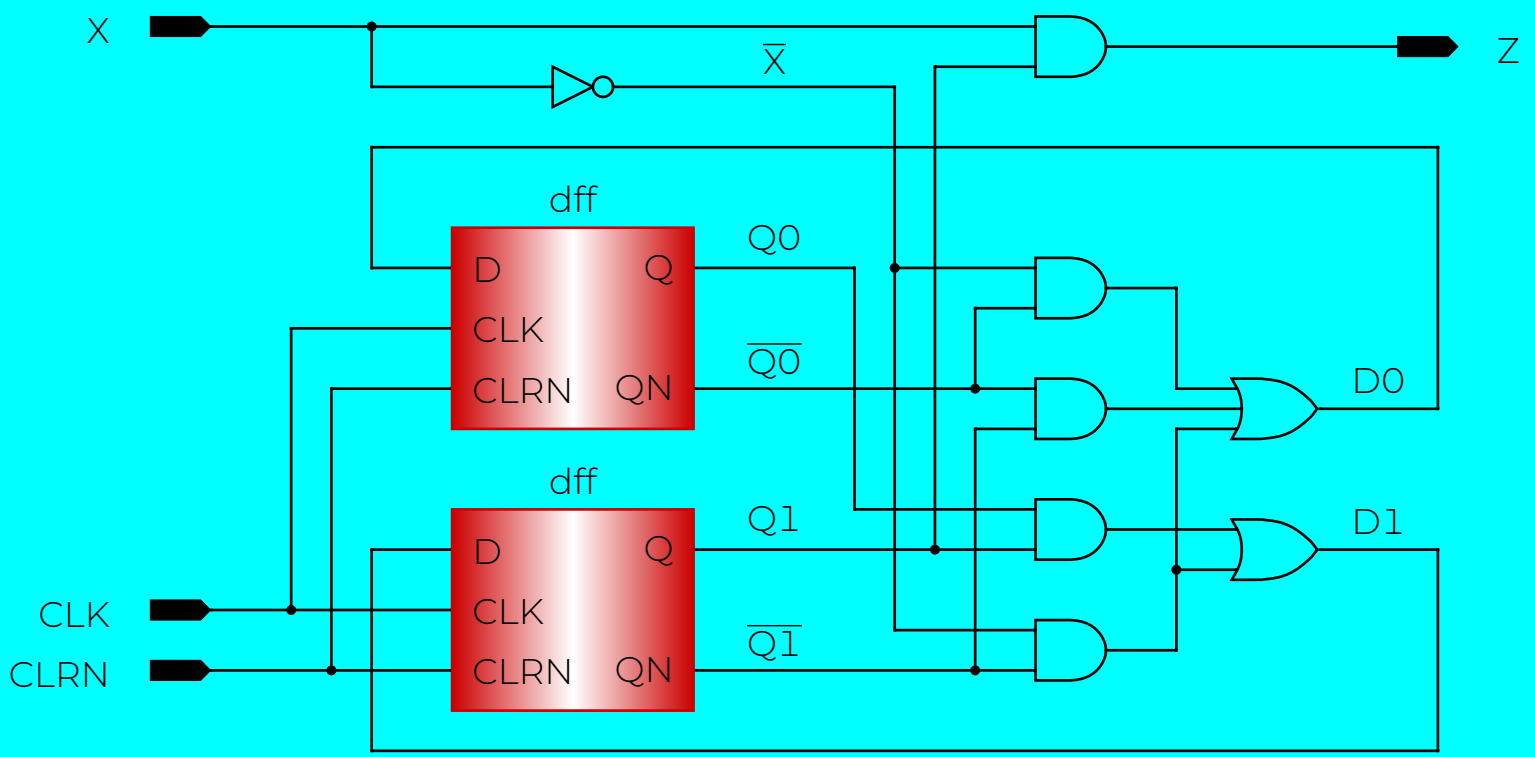
順序回路の分析



もし、こうゆうふうな回路があれば、

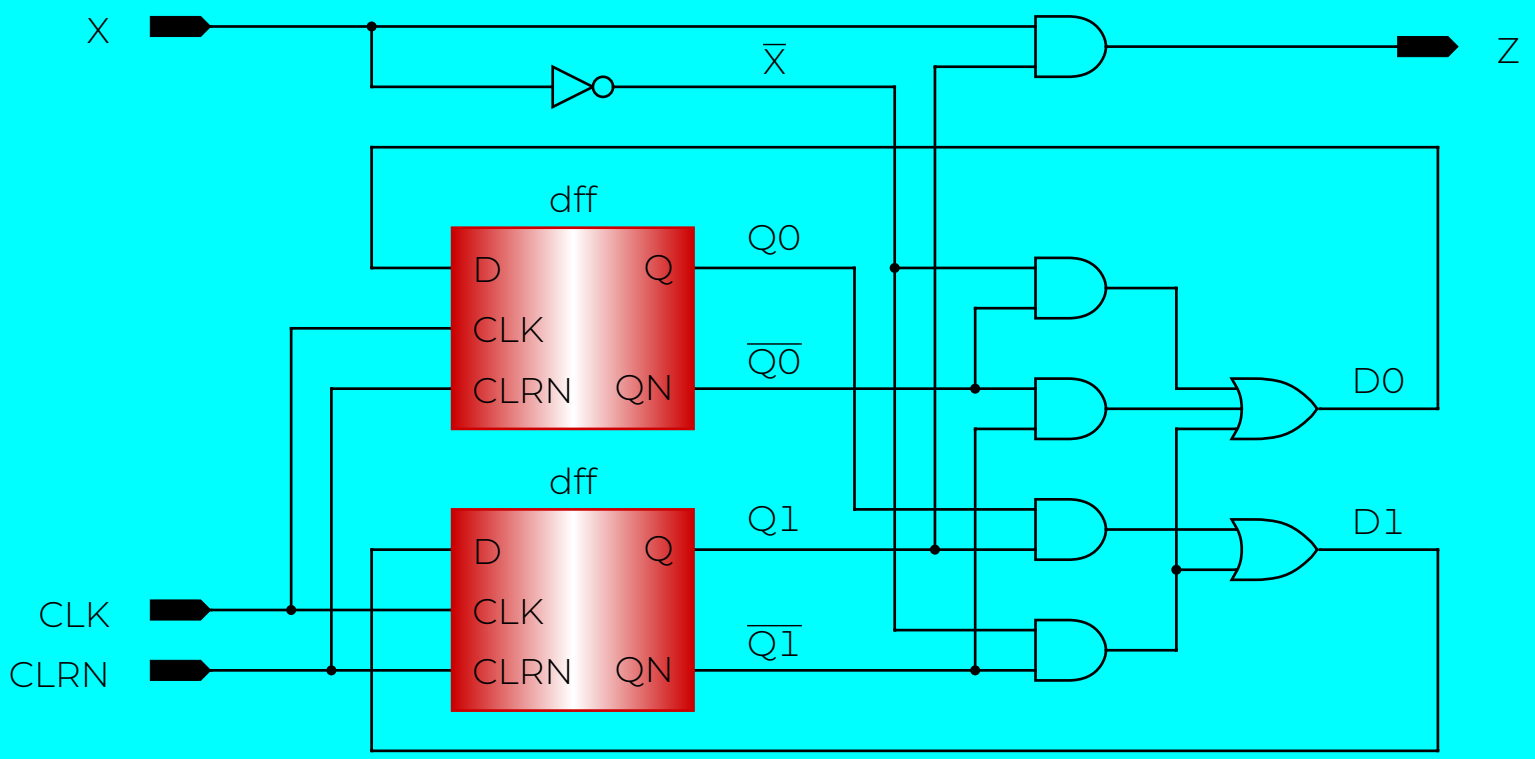
D1, D0, Z の論理式？　真理値表？　状態遷移図？

順序回路の分析 — 論理式



D1 =

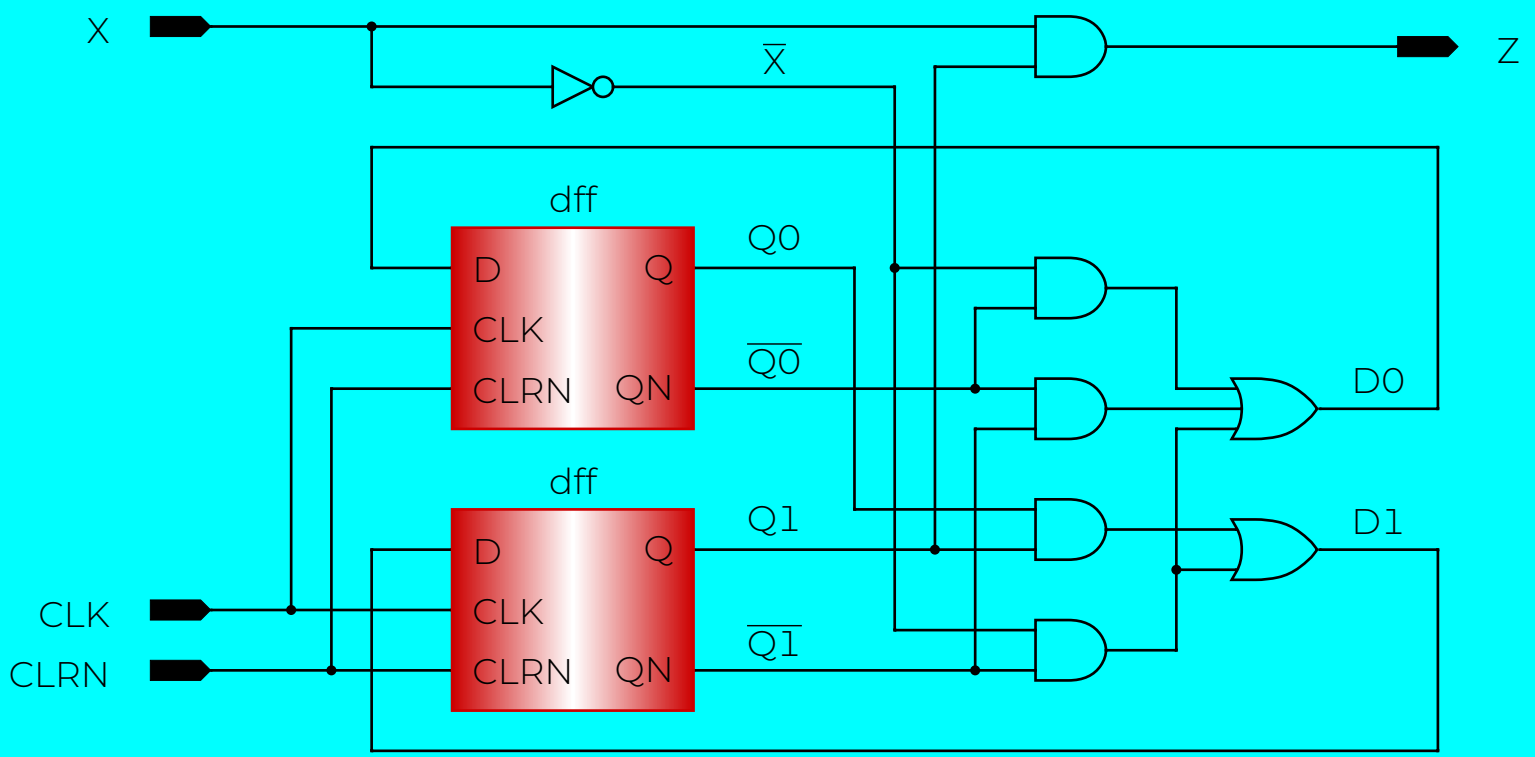
順序回路の分析 — 論理式



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

$$D0 =$$

順序回路の分析 — 論理式

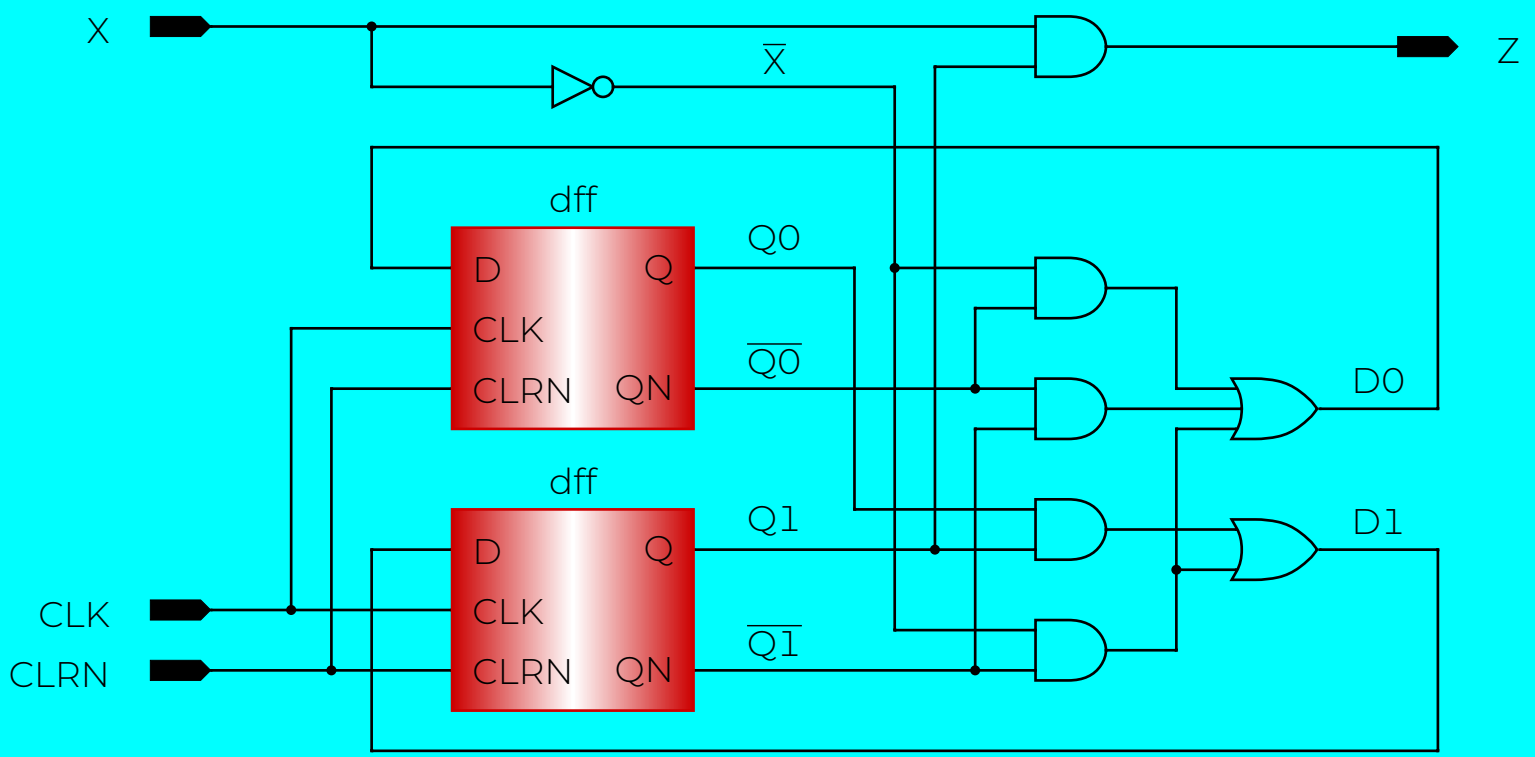


$$D1 = Q1 \cdot Q0 + \overline{Q1} \cdot \overline{X}$$

$$D0 = \overline{Q0} \cdot \overline{X} + \overline{Q1} \cdot \overline{Q0} + \overline{Q1} \cdot \overline{X}$$

$$Z =$$

順序回路の分析 — 論理式

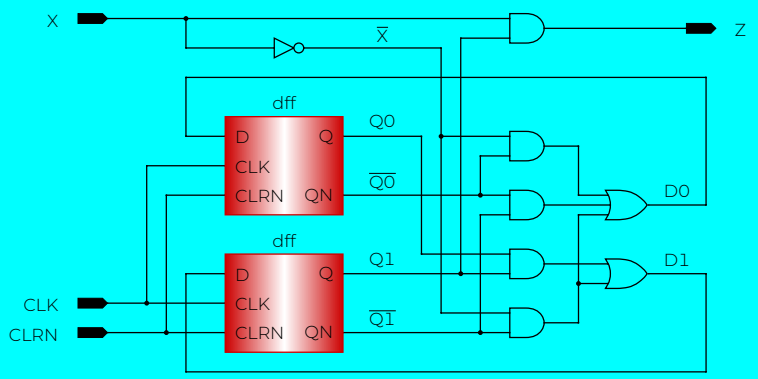


$$D1 = Q1 \cdot Q0 + \overline{Q1} \cdot \overline{X}$$

$$D0 = \overline{Q0} \cdot \overline{X} + \overline{Q1} \cdot \overline{Q0} + \overline{Q1} \cdot \overline{X}$$

$$Z = Q1 \cdot X$$

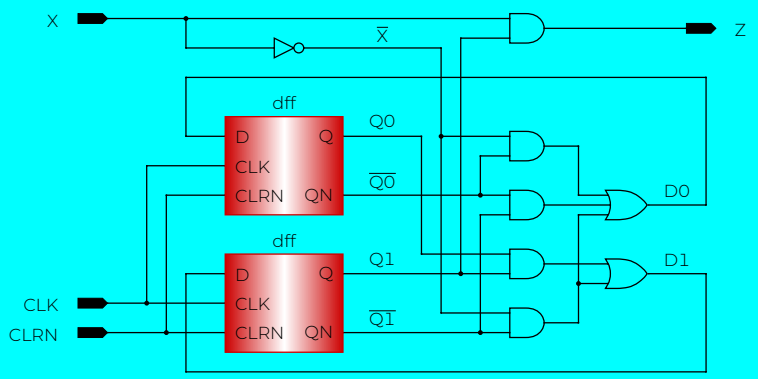
順序回路の分析 — 真理値表



$$D1 = Q1 \cdot Q0 + \overline{Q1} \cdot \overline{X}$$
$$D0 = \overline{Q0} \cdot \overline{X} + \overline{Q1} \cdot \overline{Q0} + \overline{Q1} \cdot \overline{X}$$
$$Z = Q1 \cdot X$$

現在の状態		入力	次の状態		出力
Q1	Q0	X	D1	D0	Z
S0 0	0	0			

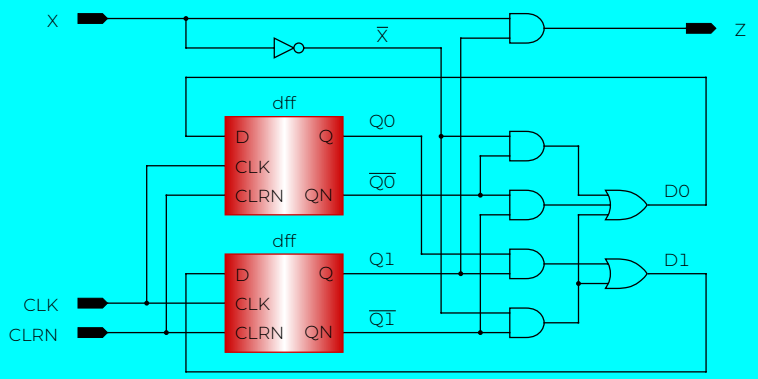
順序回路の分析 — 真理値表



$$D1 = Q1 \cdot Q0 + \overline{Q1} \cdot \overline{X}$$
$$D0 = \overline{Q0} \cdot \overline{X} + \overline{Q1} \cdot \overline{Q0} + \overline{Q1} \cdot \overline{X}$$
$$Z = Q1 \cdot X$$

現在の状態		入力	次の状態		出力
Q1	Q0	X	D1	D0	Z
S0 (0	0)	0	S3 (1	1)	0
		1			

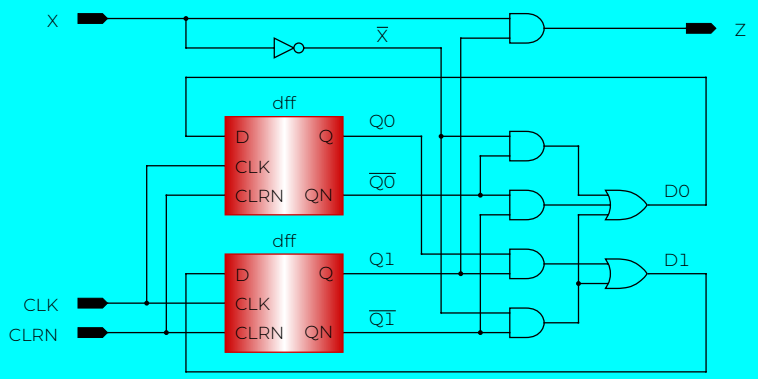
順序回路の分析 — 真理値表



$$D1 = Q1 \cdot Q0 + \overline{Q1} \cdot \overline{X}$$
$$D0 = \overline{Q0} \cdot \overline{X} + \overline{Q1} \cdot \overline{Q0} + \overline{Q1} \cdot \overline{X}$$
$$Z = Q1 \cdot X$$

現在の状態			入力	次の状態			出力
	Q1	Q0	X		D1	D0	Z
S0	0	0	0	S3	1	1	0
			1	S1	0	1	0
S1	0	1	0				

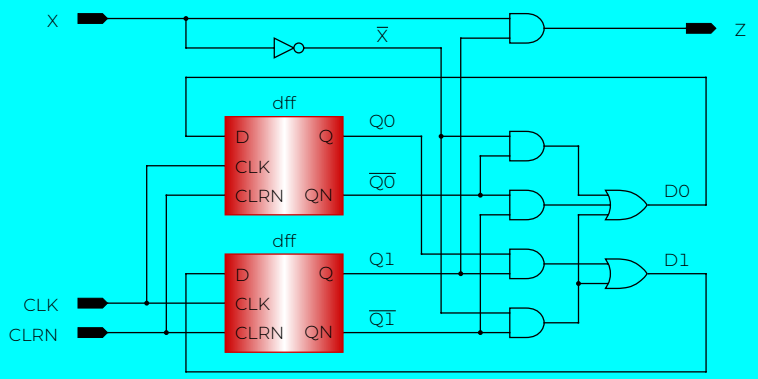
順序回路の分析 — 真理値表



$$D1 = Q1 \cdot Q0 + \overline{Q1} \cdot \overline{X}$$
$$D0 = \overline{Q0} \cdot \overline{X} + \overline{Q1} \cdot \overline{Q0} + \overline{Q1} \cdot \overline{X}$$
$$Z = Q1 \cdot X$$

現在の状態	Q1	Q0	入力 X	次の状態	D1	D0	出力 Z
S0	0	0	0	S3	1	1	0
			1	S1	0	1	0
S1	0	1	0	S3	1	1	0
			1				

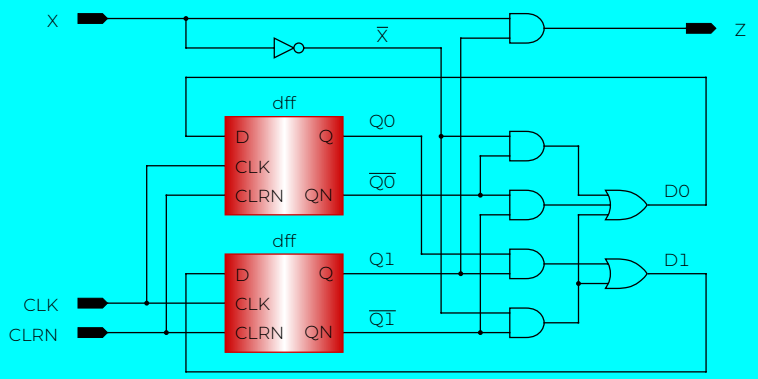
順序回路の分析 — 真理値表



$$D1 = Q1 \cdot Q0 + \overline{Q1} \cdot \overline{X}$$
$$D0 = \overline{Q0} \cdot \overline{X} + \overline{Q1} \cdot \overline{Q0} + \overline{Q1} \cdot \overline{X}$$
$$Z = Q1 \cdot X$$

現在の状態	Q1	Q0	入力 X	次の状態	D1	D0	出力 Z
S0	0	0	0	S3	1	1	0
			1	S1	0	1	0
S1	0	1	0	S3	1	1	0
			1	S0	0	0	0
S2	1	0	0				

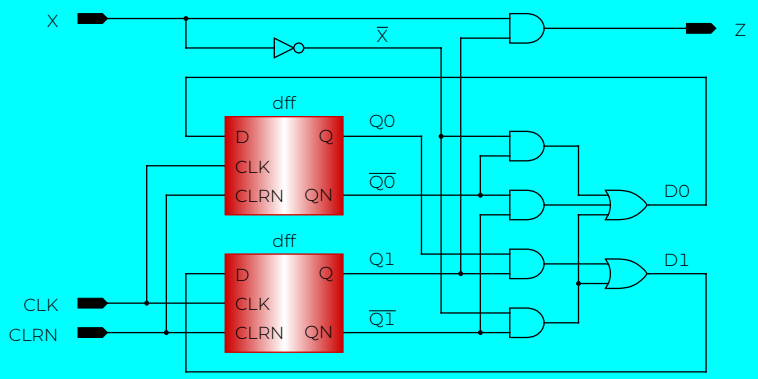
順序回路の分析 — 真理値表



$$D1 = Q1 \cdot Q0 + \overline{Q1} \cdot \overline{X}$$
$$D0 = \overline{Q0} \cdot \overline{X} + \overline{Q1} \cdot \overline{Q0} + \overline{Q1} \cdot \overline{X}$$
$$Z = Q1 \cdot X$$

現在の状態	Q1	Q0	入力 X	次の状態	D1	D0	出力 Z
S0	0	0	0	S3	1	1	0
			1	S1	0	1	0
S1	0	1	0	S3	1	1	0
			1	S0	0	0	0
S2	1	0	0	S1	0	1	0
			1				

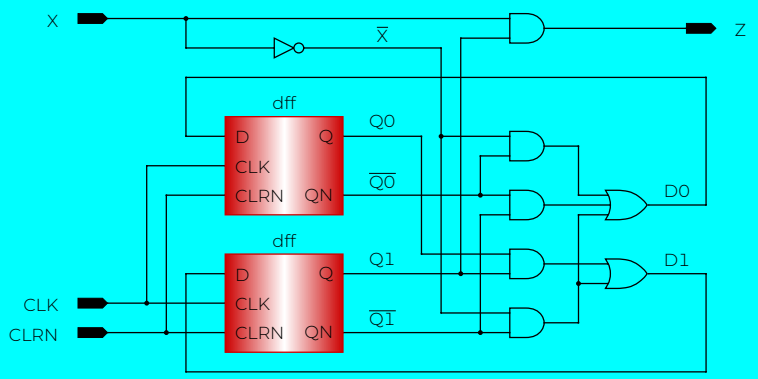
順序回路の分析 — 真理値表



$$D1 = Q1 \cdot Q0 + \overline{Q1} \cdot \overline{X}$$
$$D0 = \overline{Q0} \cdot \overline{X} + \overline{Q1} \cdot \overline{Q0} + \overline{Q1} \cdot \overline{X}$$
$$Z = Q1 \cdot X$$

現在の状態	Q1	Q0	入力 X	次の状態	D1	D0	出力 Z
S0	0	0	0	S3	1	1	0
			1	S1	0	1	0
S1	0	1	0	S3	1	1	0
			1	S0	0	0	0
S2	1	0	0	S1	0	1	0
			1	S0	0	0	1
S3	1	1	0				

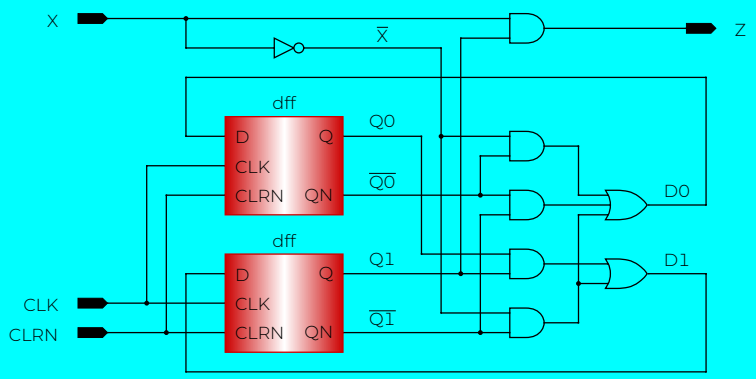
順序回路の分析 — 真理値表



$$D1 = Q1 \cdot Q0 + \overline{Q1} \cdot \overline{X}$$
$$D0 = \overline{Q0} \cdot \overline{X} + \overline{Q1} \cdot \overline{Q0} + \overline{Q1} \cdot \overline{X}$$
$$Z = Q1 \cdot X$$

現在の状態	Q1	Q0	入力 X	次の状態	D1	D0	出力 Z
S0	0	0	0	S3	1	1	0
			1	S1	0	1	0
S1	0	1	0	S3	1	1	0
			1	S0	0	0	0
S2	1	0	0	S1	0	1	0
			1	S0	0	0	1
S3	1	1	0	S2	1	0	0
			1				

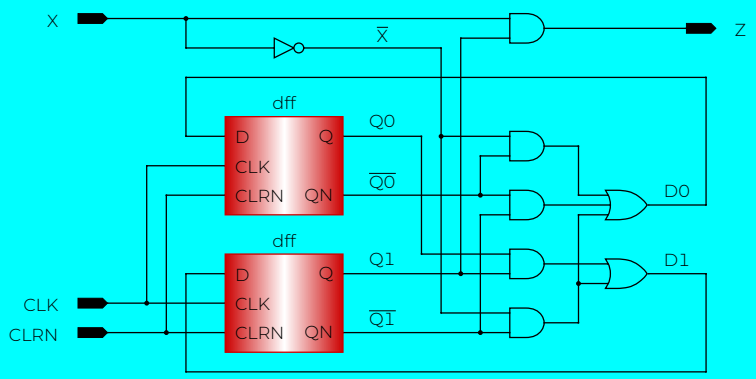
順序回路の分析 — 真理値表



$$D1 = Q1 \cdot Q0 + \overline{Q1} \cdot \overline{X}$$
$$D0 = \overline{Q0} \cdot \overline{X} + \overline{Q1} \cdot \overline{Q0} + \overline{Q1} \cdot \overline{X}$$
$$Z = Q1 \cdot X$$

現在の状態			入力	次の状態			出力
	Q1	Q0	X		D1	D0	Z
S0	0	0	0	S3	1	1	0
			1	S1	0	1	0
S1	0	1	0	S3	1	1	0
			1	S0	0	0	0
S2	1	0	0	S1	0	1	0
			1	S0	0	0	1
S3	1	1	0	S2	1	0	0
			1	S2	1	0	1

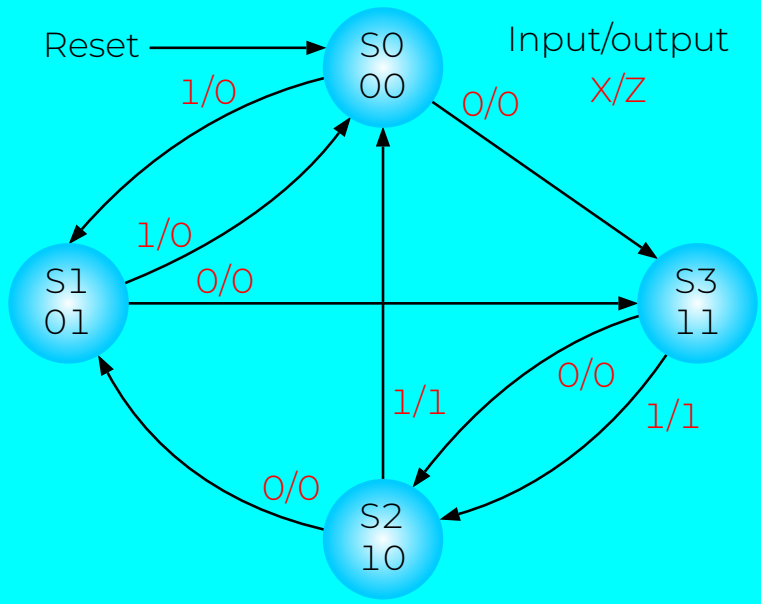
順序回路の分析 — 状態遷移図



$$D1 = Q1 \cdot Q0 + \overline{Q1} \cdot \overline{X}$$
$$D0 = \overline{Q0} \cdot \overline{X} + \overline{Q1} \cdot \overline{Q0} + \overline{Q1} \cdot \overline{X}$$
$$Z = Q1 \cdot X$$

現在の状態			入力	次の状態			出力
	Q1	Q0	X		D1	D0	Z
S0	0	0	0	S3	1	1	0
			1	S1	0	1	0
S1	0	1	0	S3	1	1	0
			1	S0	0	0	0
S2	1	0	0	S1	0	1	0
			1	S0	0	0	1
S3	1	1	0	S2	1	0	0
			1	S2	1	0	1

真理値表 ⇒ 状態遷移図 (分析の手順):



順序回路の実装 — 回路

$$
\begin{aligned}
D1 &= Q1 \cdot Q0 + \overline{Q1} \cdot \overline{X} \\
D0 &= \overline{Q0} \cdot \overline{X} + \overline{Q1} \cdot \overline{Q0} + \overline{Q1} \cdot \overline{X} \\
Z &= Q1 \cdot X
\end{aligned}
$$

```
‘timescale 1ns/1ns
module seq_ex (CLK, CLRN, X, Z, Q);
    input        CLK, CLRN, X;
    output       Z; // output
    output [1:0] Q;

    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] & ~X;
    assign D[0] = ~Q[0] & ~X    | ~Q[1] & ~Q[0] | ~Q[1] & ~X;
    // 2. outputs --------------------------------------- 2
    assign Z    =  Q[1] &  X;
    // 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
```

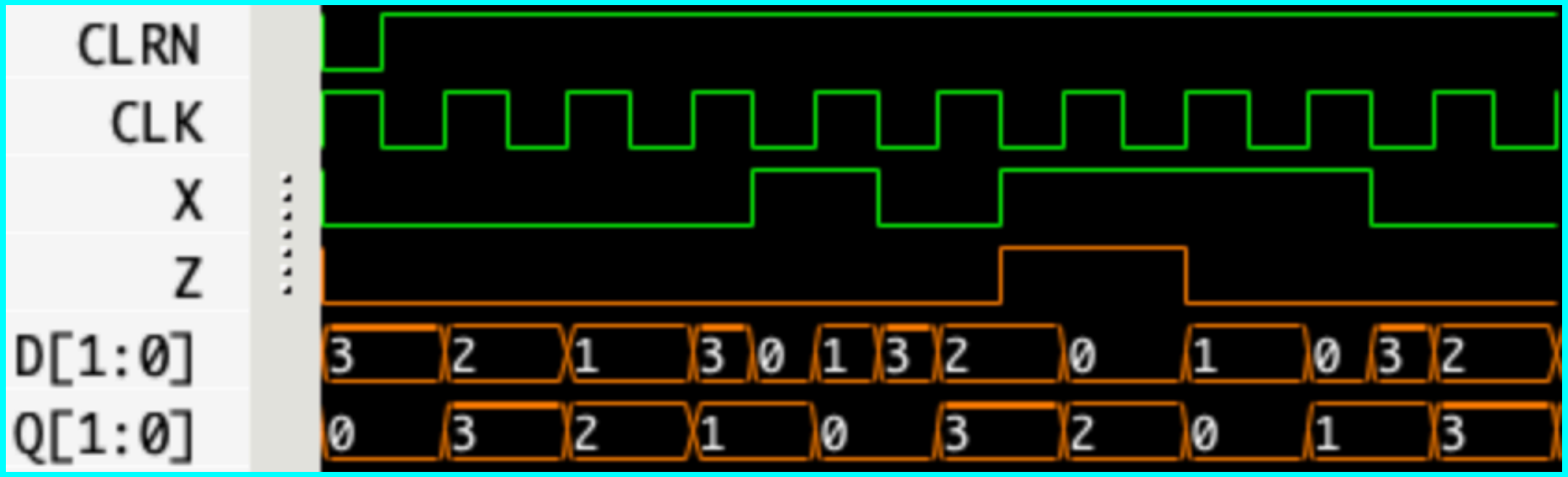
seq_ex.v

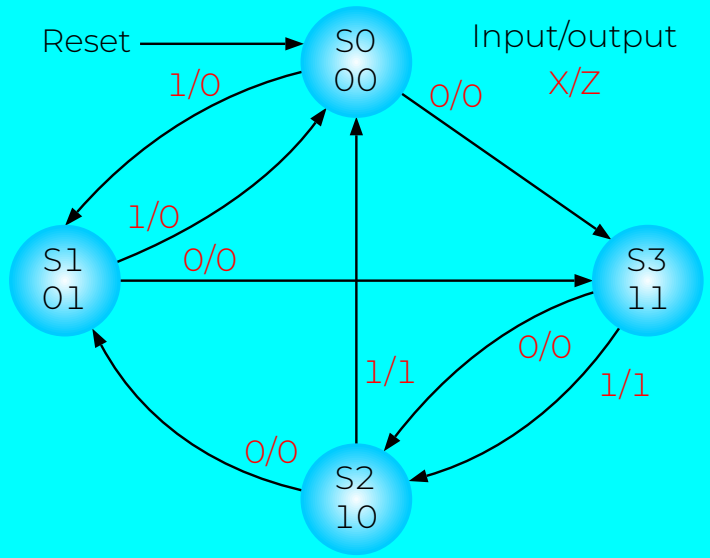
順序回路の検証 — テストベンチ

```
‘timescale 1ns/1ns
module seq_ex_tb;
    reg        X, CLK, CLRN;
    wire       Z;
    wire [1:0] Q;

    seq_ex i0 (CLK, CLRN, X, Z, Q);

    initial begin
        #0 CLRN = 0; CLK = 1; X = 0;
        #1 CLRN = 1;
        #6 X = 1;
        #2 X = 0;
        #2 X = 1;
        #6 X = 0;
        #3 $finish;
    end

    always #1 CLK = ~CLK;

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

endmodule
```

seq_ex_tb.v

順序回路の検証 — 波形

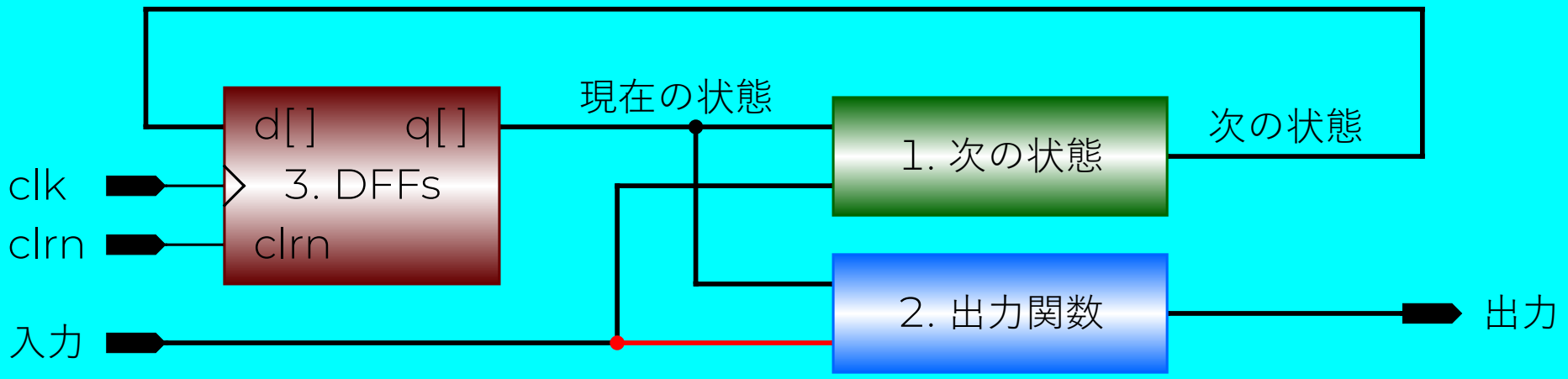




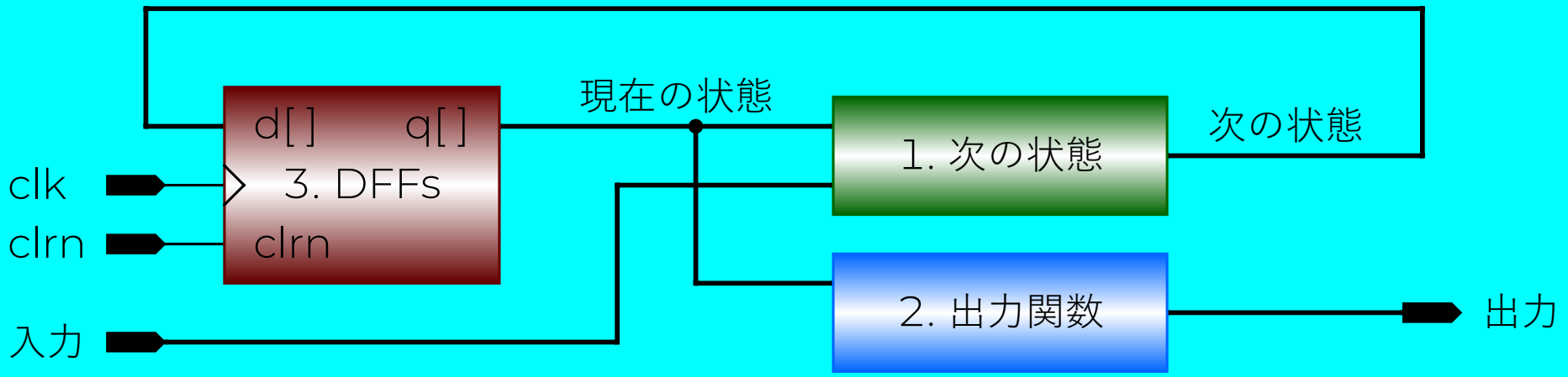
$$D1 = Q1 \cdot Q0 + \overline{Q1} \cdot \overline{X}$$
$$D0 = \overline{Q0} \cdot \overline{X} + \overline{Q1} \cdot \overline{Q0} + \overline{Q1} \cdot \overline{X}$$
$$Z = Q1 \cdot X$$

テストベンチを作成するときは、
状態遷移図のすべての状態と遷移
を確認しなければならない。

順序回路 — Mealy 型と Moore 型



(a) Mealy 型



(b) Moore 型

順序回路設計の手順

1. 問題を理解する。

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。

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)。

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。

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。

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。

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)。

順序回路設計の手順

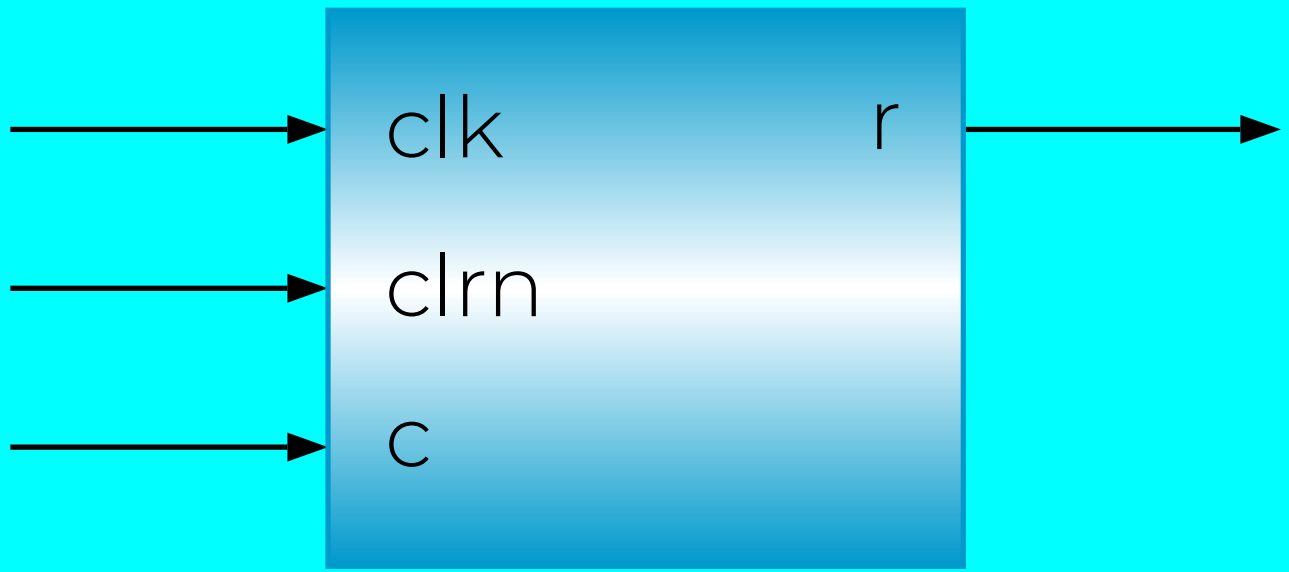
1. 問題を理解する。
2. 状態遷移図をつくる。
3. FF の数を決める ($n = \lceil \log_2 N \rceil$、N は状態の数)。
4. 各状態に n ビットの番号を付け、真理値表をつくる。
   - 次の状態の真理値表をつくる。
   - 出力関数の真理値表をつくる。
5. 真理値表から論理式をつくる (どの条件で出力が 1 になるか)。
   - カルノー図を用いて論理式を簡単化する。
6. 論理式から回路をつくる。
7. テストベンチをつくる (シミュレーションするため)。
8. 回路をシミュレーションする (回路設計の正当性検証)。

順序回路設計の手順

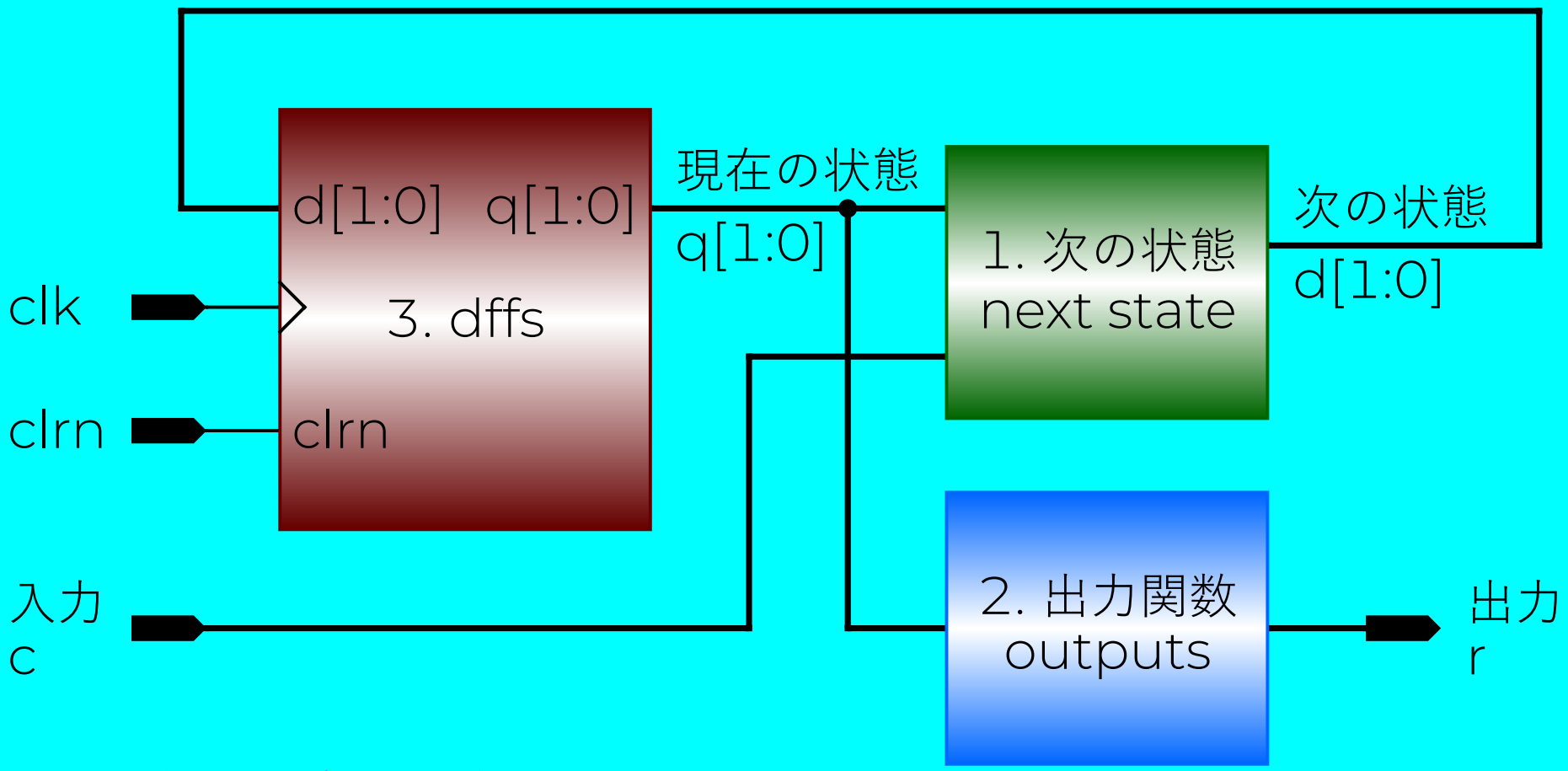
1. 問題を理解する。
2. 状態遷移図をつくる。
3. FF の数を決める ($n = \lceil \log_2 N \rceil$、N は状態の数)。
4. 各状態に n ビットの番号を付け、真理値表をつくる。
   - 次の状態の真理値表をつくる。
   - 出力関数の真理値表をつくる。
5. 真理値表から論理式をつくる (どの条件で出力が 1 になるか)。
   - カルノー図を用いて論理式を簡単化する。
6. 論理式から回路をつくる。
7. テストベンチをつくる (シミュレーションするため)。
8. 回路をシミュレーションする (回路設計の正当性検証)。
9. 与えられた回路の動作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

順序回路の例：問題

問題：入力信号として１ビットｃを持ち、ｃ＝１の数が３で割り切れるｃの入力列のみを受け入れる (例えば、011010, 1010100, 01111110, 000 は受け入れるが、01000, 101 は受け入れない) 有限状態機械を構築せよ。受け入れる状態のみ出力ｒは１である。



順序回路の例：回路



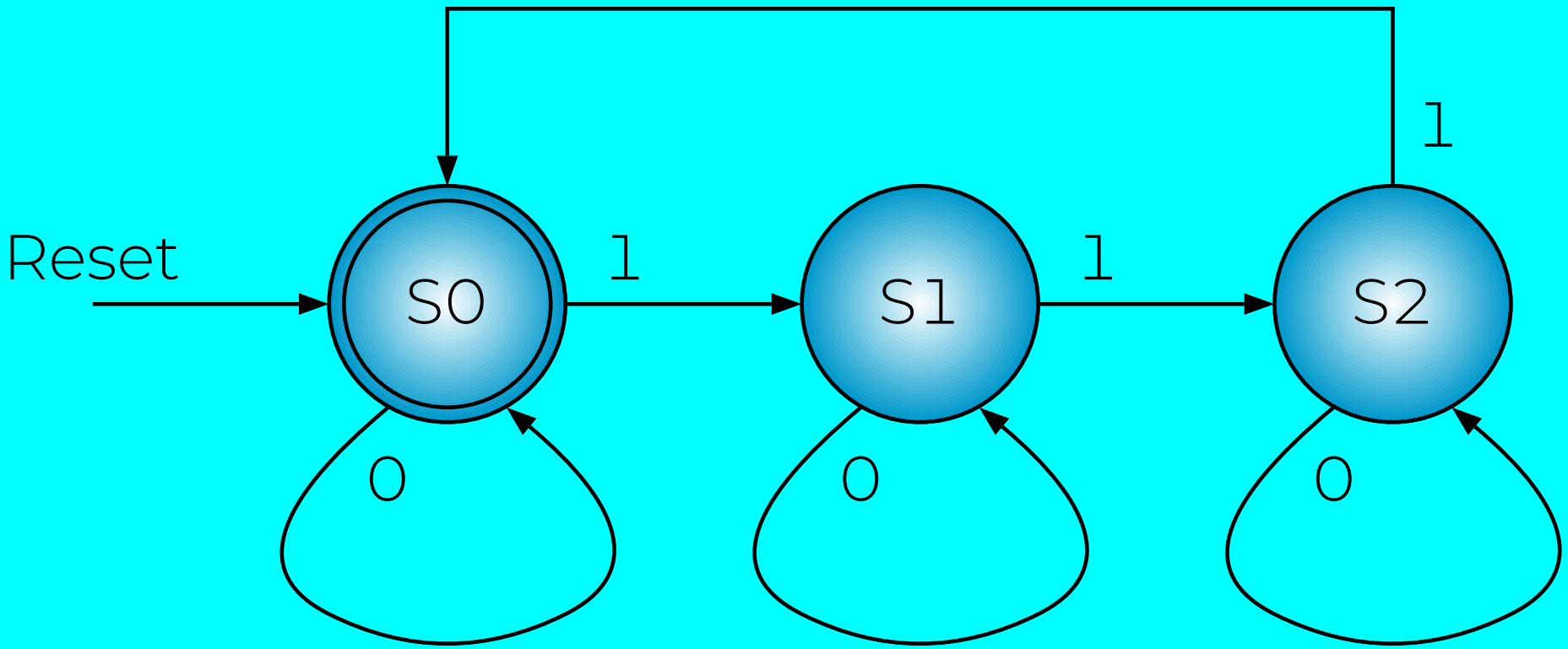
次の状態：組合せ回路

出力関数：組合せ回路

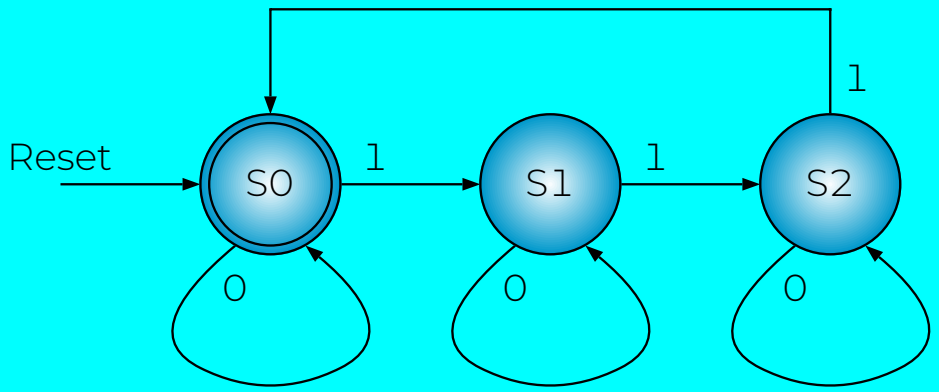
全体：順序回路

順序回路の例：状態遷移図

3 個状態が必要である。
S0：0 個 1 または 3 個 1(受け入れる)
S1：1 個 1
S2：2 個 1

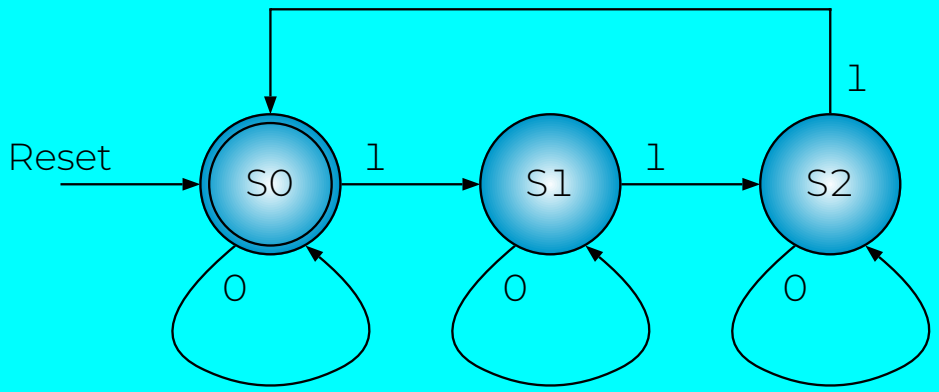


順序回路の例：次の状態の真理値表



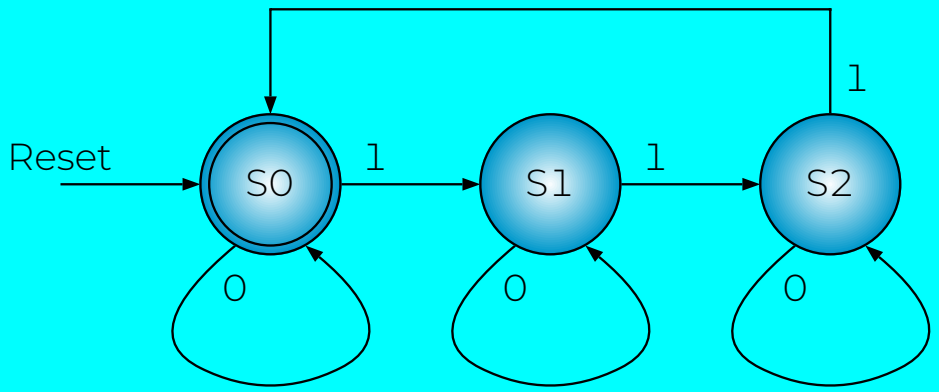
現在の状態			入力	次の状態	
	q[1]	q[0]	c	d[1]	d[0]
S0	0	0	0		

順序回路の例：次の状態の真理値表



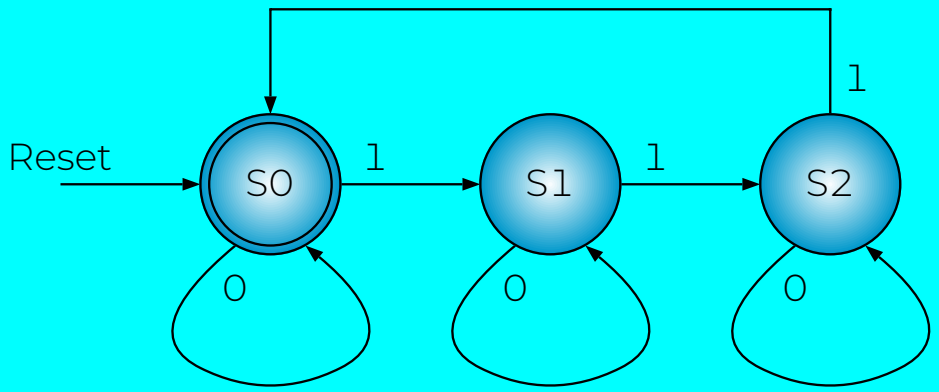
現在の状態			入力	次の状態		
	q[1]	q[0]	c		d[1]	d[0]
S0	0	0	0	S0	0	0
			1			

順序回路の例：次の状態の真理値表



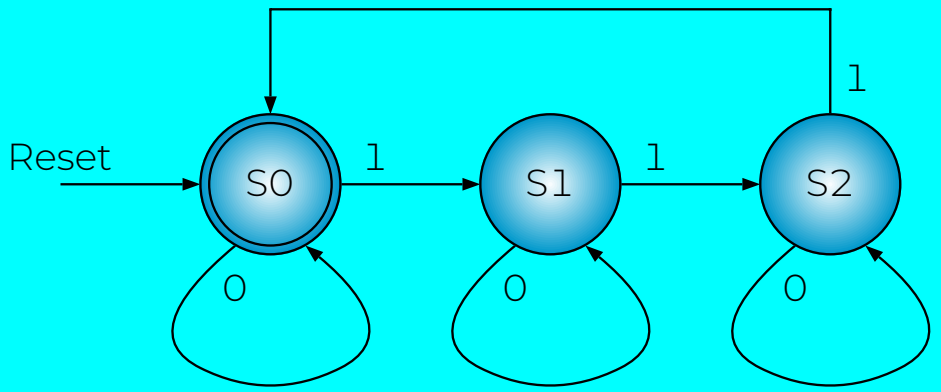
現在の状態			入力	次の状態		
	q[1]	q[0]	c		d[1]	d[0]
S0	0	0	0	S0	0	0
			1	S1	0	1
S1	0	1	0			

順序回路の例：次の状態の真理値表



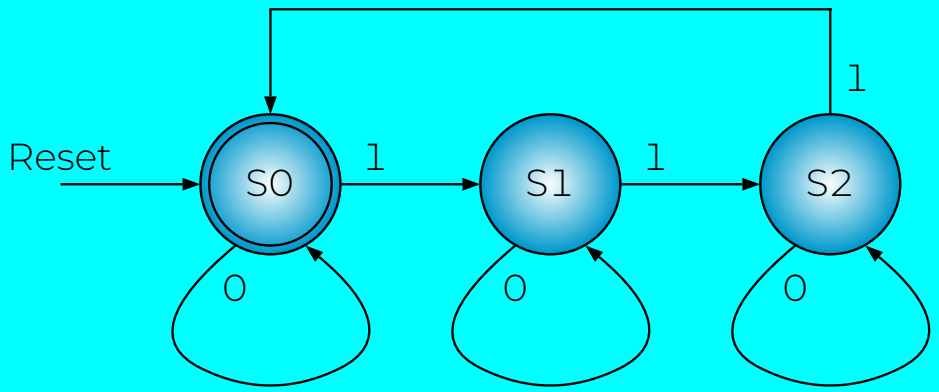
現在の状態			入力	次の状態		
	q[1]	q[0]	c		d[1]	d[0]
S0	0	0	0	S0	0	0
			1	S1	0	1
S1	0	1	0	S1	0	1
			1			

順序回路の例：次の状態の真理値表



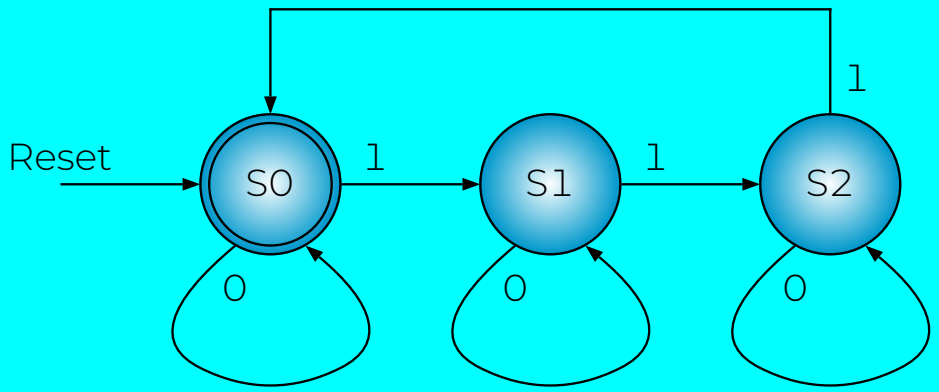
現在の状態			入力	次の状態		
	q[1]	q[0]	c		d[1]	d[0]
S0	0	0	0	S0	0	0
			1	S1	0	1
S1	0	1	0	S1	0	1
			1	S2	1	0
S2	1	0	0			

順序回路の例：次の状態の真理値表



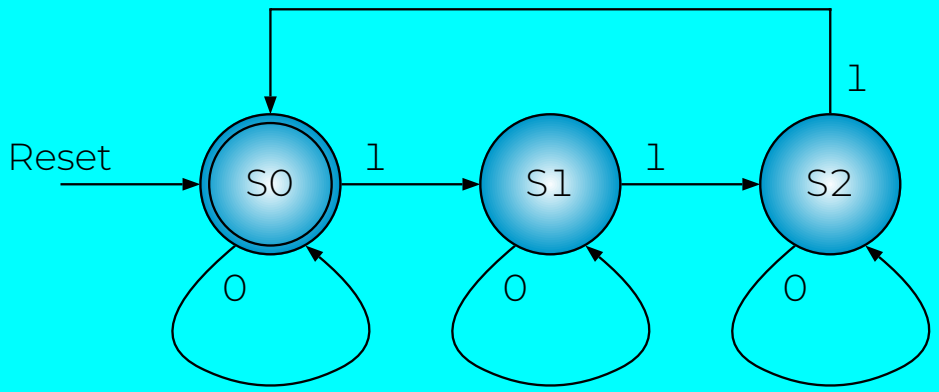
現在の状態			入力	次の状態		
	q[1]	q[0]	c		d[1]	d[0]
S0	0	0	0	S0	0	0
			1	S1	0	1
S1	0	1	0	S1	0	1
			1	S2	1	0
S2	1	0	0	S2	1	0
			1			

順序回路の例：次の状態の真理値表



現在の状態			入力	次の状態		
	q[1]	q[0]	c		d[1]	d[0]
S0	0	0	0	S0	0	0
			1	S1	0	1
S1	0	1	0	S1	0	1
			1	S2	1	0
S2	1	0	0	S2	1	0
			1	S0	0	0
存在しない	1	1	x			

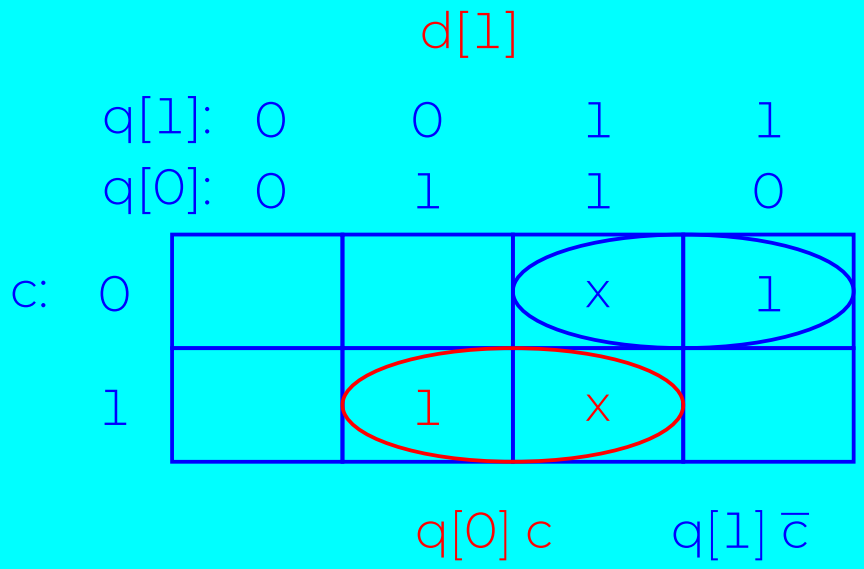
順序回路の例：次の状態の真理値表

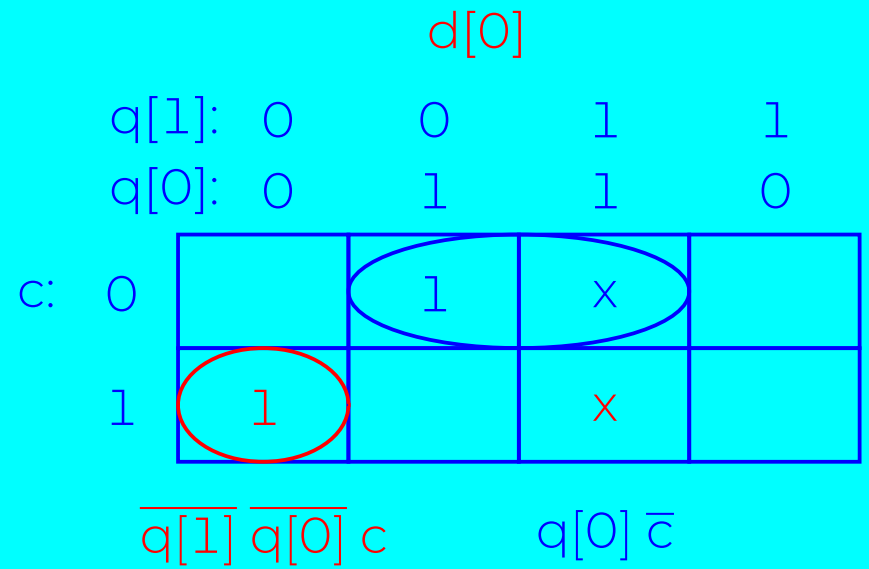


現在の状態			入力	次の状態		
	q[1]	q[0]	c		d[1]	d[0]
S0	0	0	0	S0	0	0
			1	S1	0	1
S1	0	1	0	S1	0	1
			1	S2	1	0
S2	1	0	0	S2	1	0
			1	S0	0	0
存在しない	1	1	X	ドントケア	X	X

順序回路の例：次の状態の論理式

カルノー図：





次の状態の論理式：

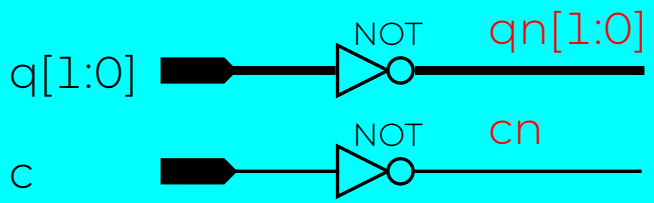
$$d[1] = q[0]\,c + q[1]\,\bar{c}$$

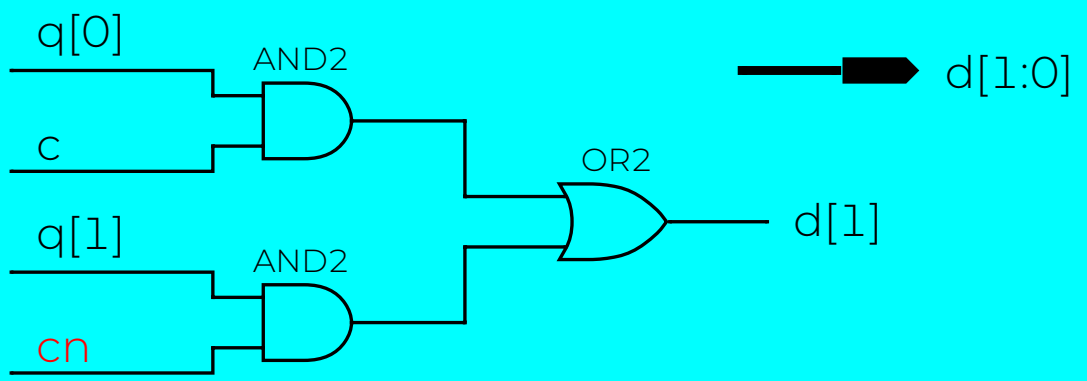
$$d[0] = \overline{q[1]}\;\overline{q[0]}\,c + q[0]\,\bar{c}$$

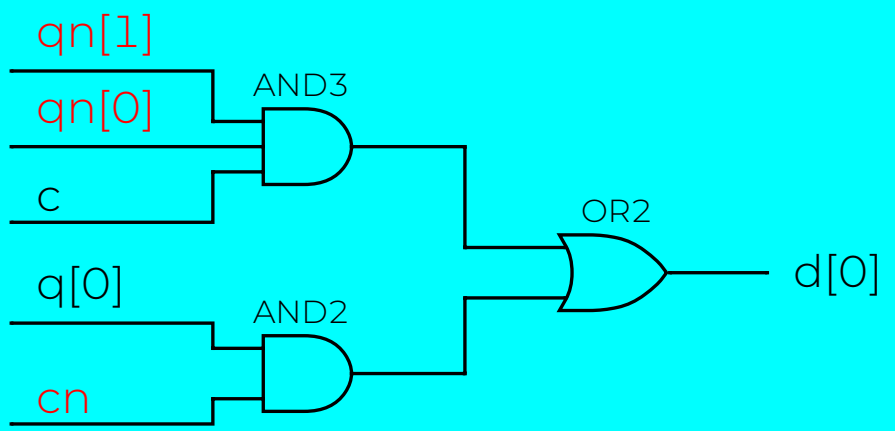
順序回路の例：次の状態の回路

$$d[1] = q[0]\,c + q[1]\,\overline{c}$$

$$d[0] = \overline{q[1]}\;\overline{q[0]}\,c + q[0]\,\overline{c}$$

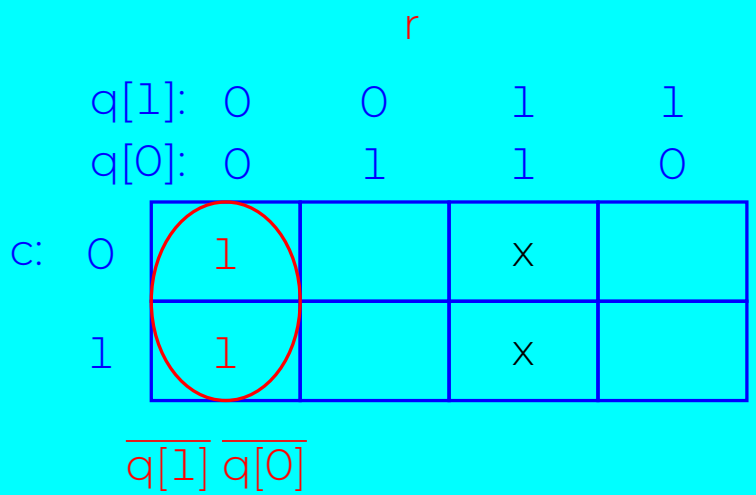






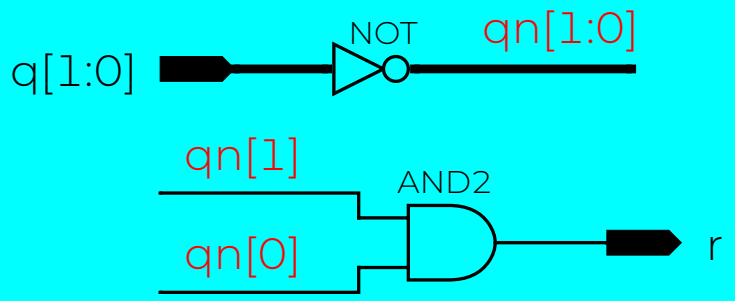
順序回路の例：出力関数の回路

現在の状態			入力	出力 r
	q[1]	q[0]	c	r
S0	0	0	x	1
S1	0	1	x	0
S2	1	0	x	0
存在しない	1	1	x	x

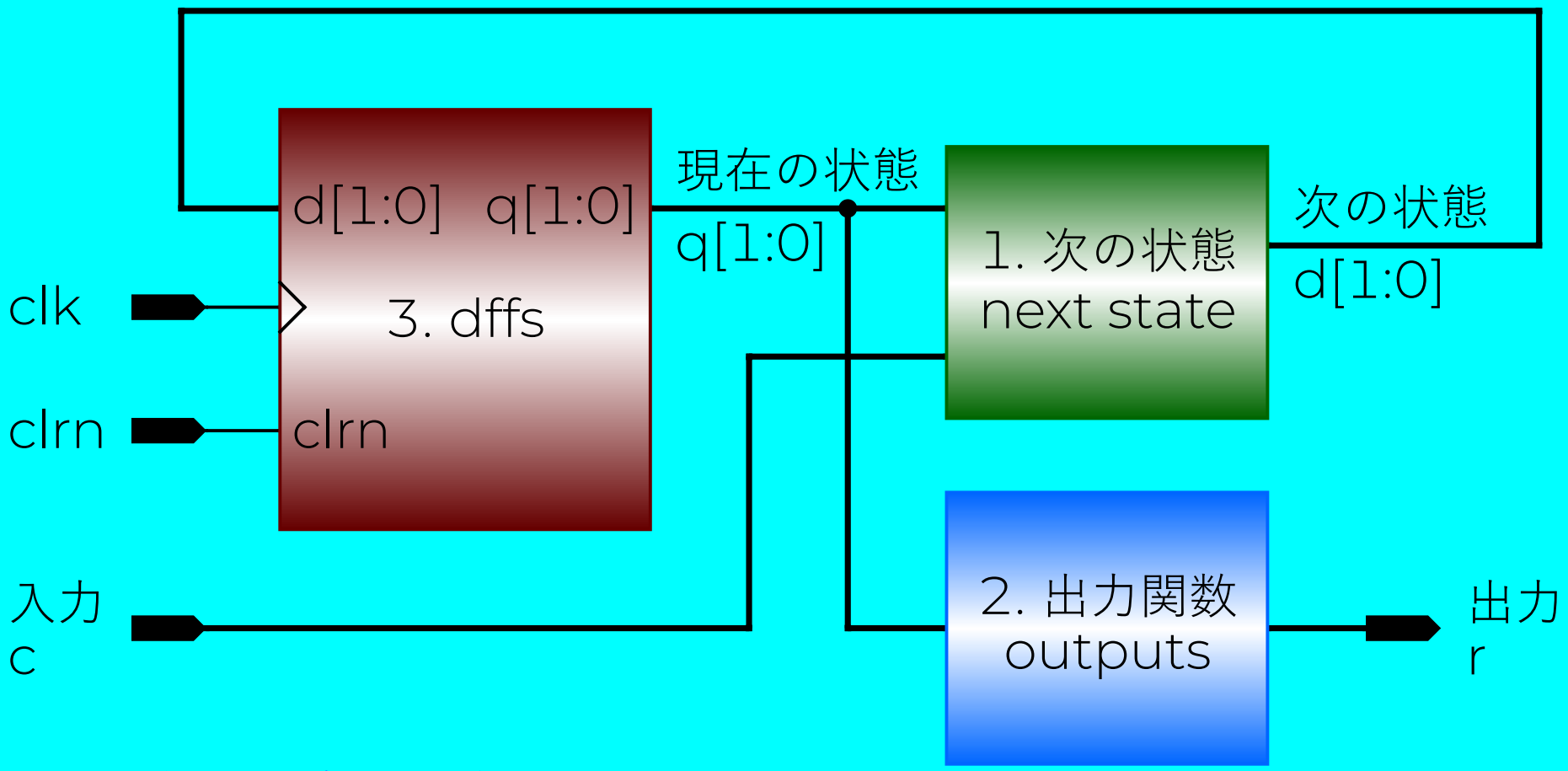


$$r = \overline{q[1]}\,\overline{q[0]}$$

Moore 型



順序回路の例：回路



次の状態：組合せ回路

出力関数：組合せ回路

全体：順序回路

順序回路の例：回路

```
‘timescale 1ns/1ns
module number3 (clk, clrn, c, r);
    input  clk, clrn, c;
    output r;

    reg  [1:0] q; // current state
    wire [1:0] d; // next state

    // 1. next state ------------------------------------- 1
    assign d[1] =          q[0] & c | q[1] & ~c;
    assign d[0] = ~q[1] & ~q[0] & c | q[0] & ~c;

    // 2. outputs ---------------------------------------- 2
    assign r = ~q[1] & ~q[0]; // state 0

    // 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
```

number3.v

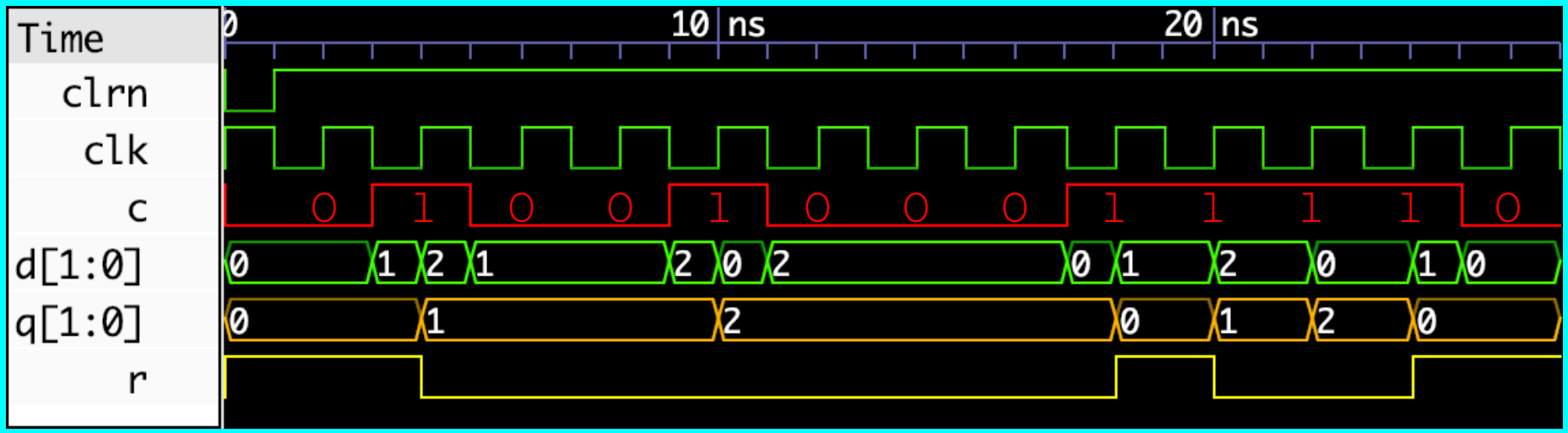
順序回路の例：テストベンチ

```
‘timescale 1ns/1ns
module number3_tb;
    reg clk, clrn, c;
    wire r;

    number3 i0 (clk, clrn, c, r);
    initial begin
        #0  clk = 1; clrn = 0; c = 0;
        #1  clrn = 1;
        #2  c = 1;
        #2  c = 0;
        #4  c = 1;
        #2  c = 0;
        #6  c = 1;
        #8  c = 0;
        #2 $finish;
    end
    always #1 clk = ~clk;

    initial begin
        $dumpfile ("number3.vcd");
        $dumpvars;
    end
endmodule
```

number3_tb.v

順序回路の例：波形

```
% iverilog -Wall -o number3 number3_tb.v number3.v
% vvp number3
% gtkwave number3.vcd
```



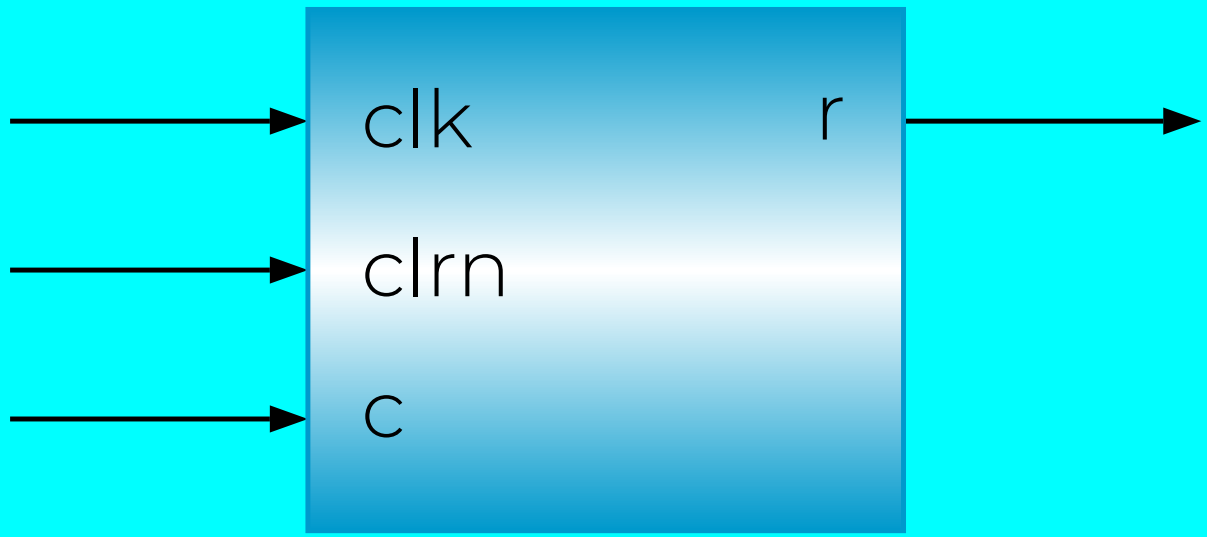
c = 0100100011110

MAC 対応: ターミナルで gtkwave 起動後 SST の number3_tb の左側にある三角形のボタンを押し、表示された i0 を選択することで機械の中の信号も全て摘出できる。

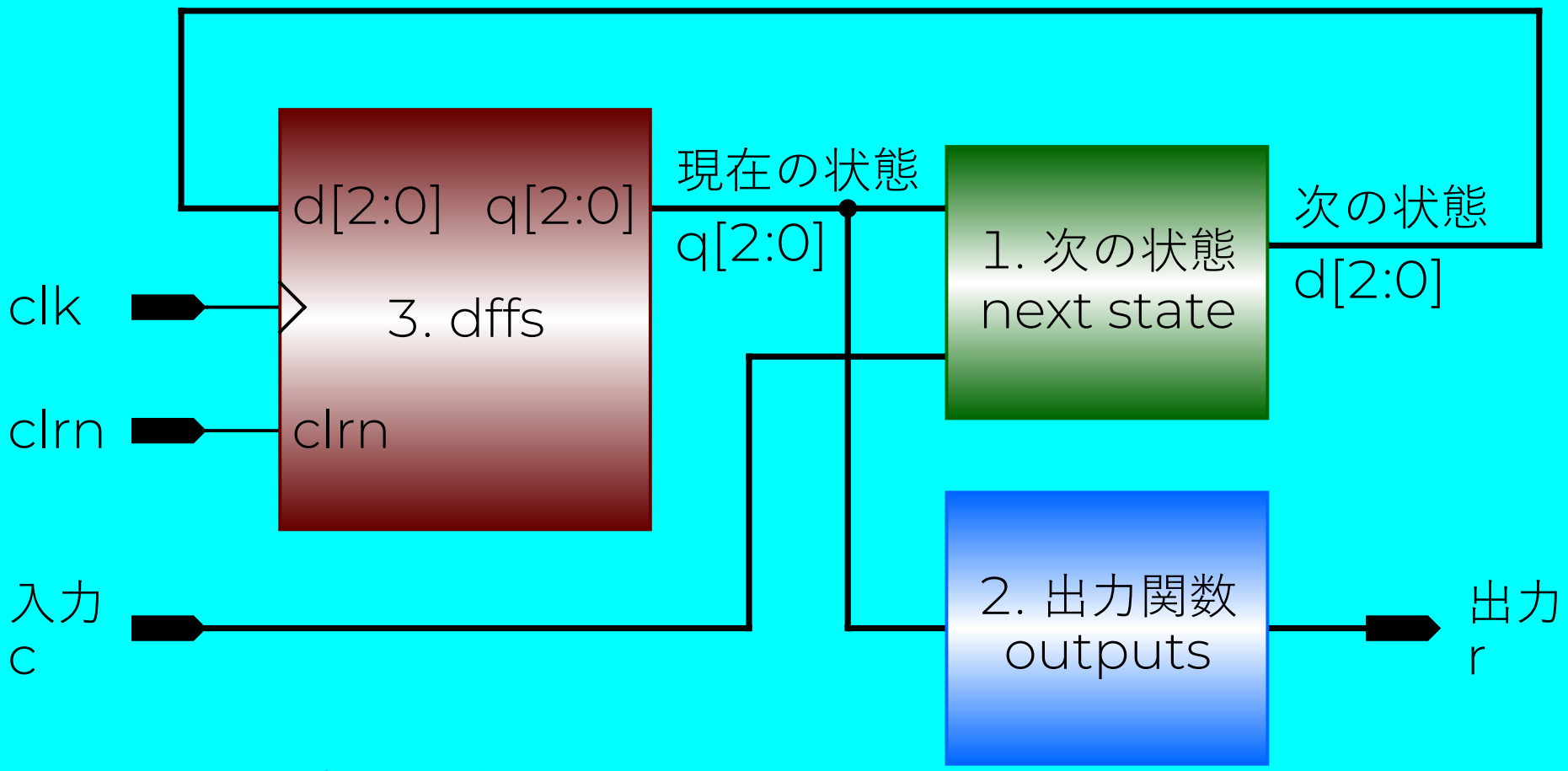
Windows/ModelSim 対応: add wave -r /*

課題 XI (100 点)

問題：入力信号として１ビットｃを持ち、ｃ＝０の数が偶数で、ｃ＝１の数が３で割り切れるｃの入力列のみを受け入れる (例えば、010010010, 1010001, 01111110, 00 は受け入れるが、01000, 1011 は受け入れない) 有限状態機械を構築せよ。受け入れる状態のみ出力ｒは１である (6 個状態が必要である)。



課題 XI (100 点)



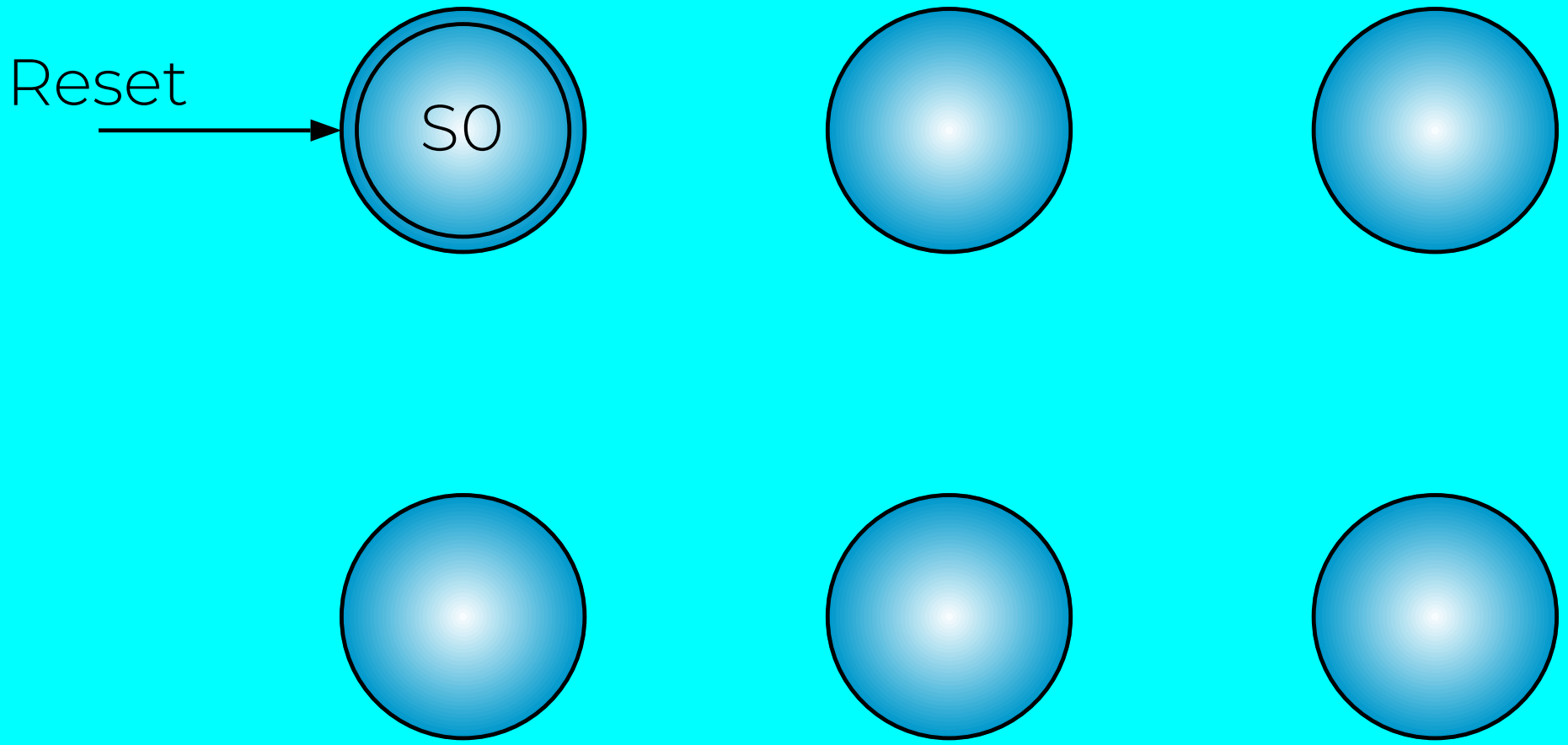
次の状態：組合せ回路

出力関数：組合せ回路

全体：順序回路

課題 XI (100 点)

状態遷移図：



課題 XI (100 点)

次の状態の真理値表：

現在の状態				入力	次の状態		
	q[2]	q[1]	q[0]	c	d[2]	d[1]	d[0]
S0	0	0	0	0	S		
				1	S		
S1	0	0	1	0	S		
				1	S		
S2	0	1	0	0	S		
				1	S		
S3	0	1	1	0	S		
				1	S		
S4	1	0	0	0	S		
				1	S		
S5	1	0	1	0	S		
				1	S		
存在しない	1	1	X	X	ドントケア	X	X	X

課題 XI (100 点)

カルノー図：

d[2]

q[0] c \ q[2] q[1]	0 0	0 1	1 1	1 0
0 0				
0 1				
1 1				
1 0				

d[1]

q[0] c \ q[2] q[1]	0 0	0 1	1 1	1 0
0 0				
0 1				
1 1				
1 0				

d[0]

q[0] c \ q[2] q[1]	0 0	0 1	1 1	1 0
0 0				
0 1				
1 1				
1 0				

次の状態の論理式：

d[2] =

d[1] =

d[0] =

課題 XI (100 点)

次の状態の論理式：

$d[2] =$

$d[1] =$

$d[0] =$

次の状態の回路図：

課題 XI (100 点)

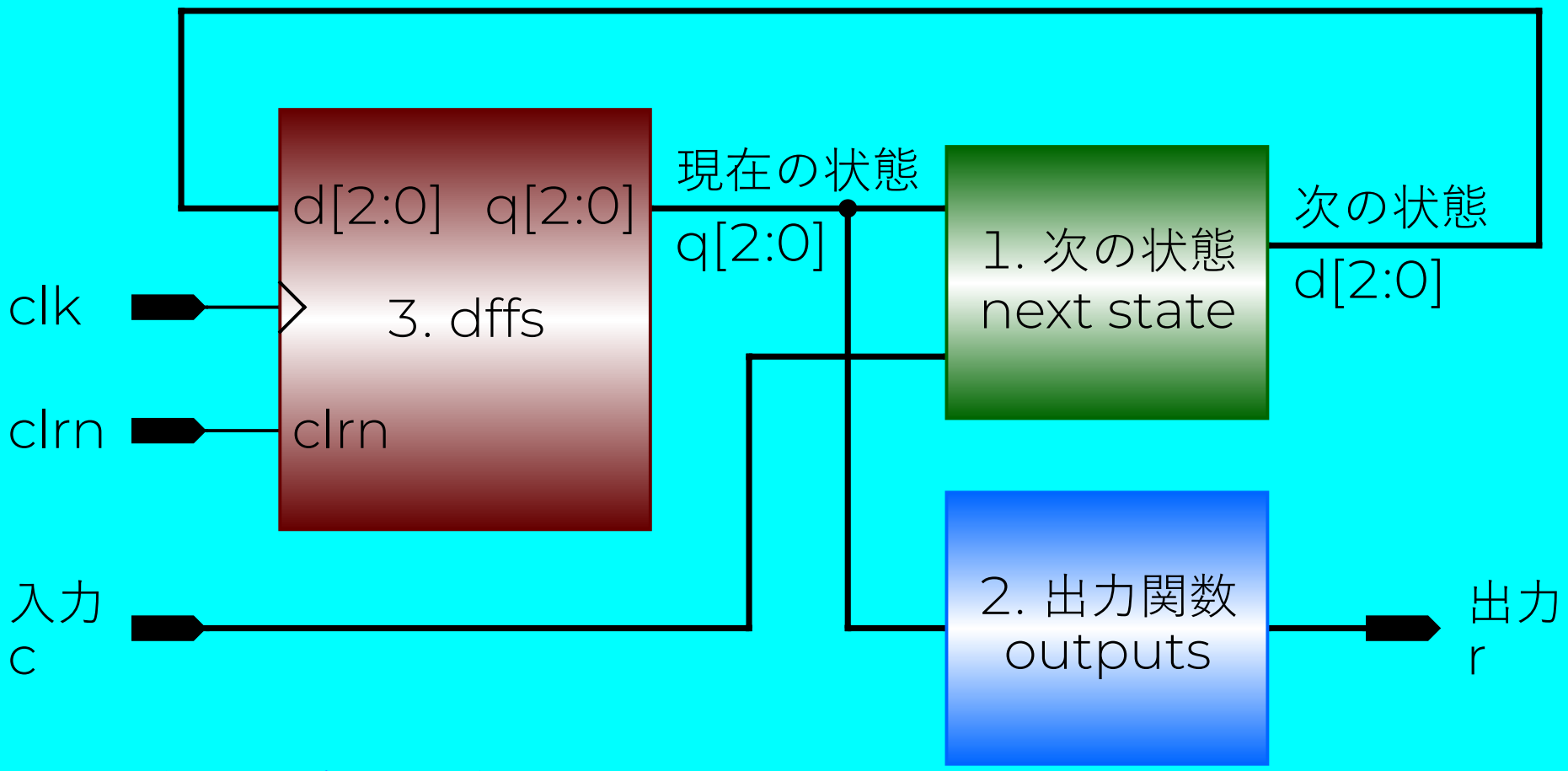
出力関数の真理値表：

	現在の状態			入力	出力 r
	q[2]	q[1]	q[0]	c	r
S0	0	0	0	x	1
S1	0	0	1	x	0
S2	0	1	0	x	0
S3	0	1	1	x	0
S4	1	0	0	x	0
S5	1	0	1	x	0
存在しない	1	1	x	x	x

$r = \overline{q[2]}\ \overline{q[1]}\ \overline{q[0]}$

出力関数の回路：

課題 XI (100 点)



次の状態：組合せ回路

出力関数：組合せ回路

全体：順序回路

課題 XI (100 点)

```
‘timescale 1ns/1ns
module even031 (clk, clrn, c, r);
    input  clk, clrn, c;
    output r;

    reg  [2:0] q; // current state
    wire [2:0] d; // next state

    // 1. next state ------------------------------------ 1
    assign d[2] = ;
    assign d[1] = ;
    assign d[0] = ;

    // 2. outputs --------------------------------------- 2
    assign r = ~q[2] & ~q[1] & ~q[0]; // state 0

    // 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
```

even031.v

課題 XI (100 点)

```
‘timescale 1ns/1ns
module even031_tb;
    reg  clk, clrn, a, c;
    wire r;
    even031 i0 (clk, clrn, c, r);
    initial begin
        #0  c = 0; clrn = 0; clk = 1; // S0
        #1  c = 1; clrn = 1;
        #2  c = 0;
        #2  c = 1;
        #2  c = 0;
        #2  c = 1;
        #2  c = 0;
        #2  c = 1;
        #2  c = 0;
        #2  c = 1;
        #2  c = 0;
        #2  c = 1;
        #2  c = 0;
        #2  c = 1;
        #6  c = 0;
        #2  c = 1;
        #4  c = 0;
        #2  c = 1;
        #4 $finish;
    end
    always #1 clk = ~clk;
    initial begin
        $dumpfile ("even031.vcd");
        $dumpvars;
    end
endmodule
```

even031_tb.v

課題 XI (100 点)

```
% iverilog -Wall -o even031 even031_tb.v even031.v
% vvp even031
% gtkwave even031.vcd
```

波形：

波形の説明：
入力列 c = 10101010101011101101l