論理回路入門（２）

論理ゲートのCMOS型実装

李　亜民

2023年10月3日(火)

論理ゲートのCMOS型実装

ポイント

- 基本的な論理演算と論理ゲート: AND、OR、NOT
- ほかのゲート: NAND、NOR、XOR、XNOR
- CMOSトランジスタ
- NOTゲートのCMOS型実装
- NANDゲートのCMOS型実装
- ANDゲートのCMOS型実装
- NORゲートのCMOS型実装
- ORゲートのCMOS型実装
- NANDゲートのトランジスター型実装
- NANDゲートのNMOS型実装

基本的な論理演算 1. AND

1. AND (アンド)
   論理積 (かつ)
   『例』 $F = A \cdot B = A\ B$

(Verilog HDL の表現)
(F = A & B)

0 · 0 = 0 偽
0 · 1 = 0 偽
1 · 0 = 0 偽
1 · 1 = 1 ..... 真 かつ 真 = 真

論理積は、入力値がすべて 1 のときに 1 を出力する。
それ以外の入力値のときは 0 を出力する。

基本的な論理演算 2. OR

2. OR (オア)
   論理和 (または)
   『例』 $F = A + B$

(Verilog HDL の表現)
(`F = A | B`)

$0 + 0 = 0$	..... 偽 または 偽 = 偽	
$0 + 1 = 1$		真
$1 + 0 = 1$		真
$1 + 1 = 1$		真

論理和は、入力値がすべて 0 のときに 0 を出力する。
それ以外の入力値のときは 1 を出力する。

基本的な論理演算 3. NOT

3. NOT (ノット)
否定
『例』 $F = \overline{A}$

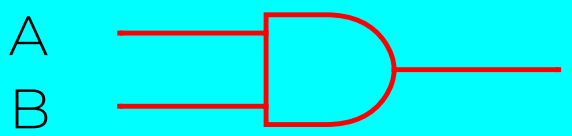
(Verilog HDL の表現)
(F = ~A)

$\overline{0} = 1$ ..... 偽の否定 = 真
$\overline{1} = 0$ ..... 真の否定 = 偽

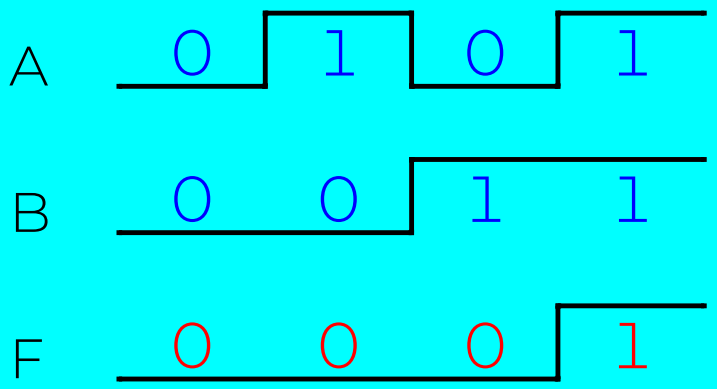
論理否定は、入力された値が 0 なら 1 に、
1 なら 0 に反転する。

基本的な論理ゲート — 1. ANDゲート

論理積



波形



論理式 $F = A \cdot B = AB$

Verilog HDLの表現：`F = A & B`

論理積 (AND) は、入力値がすべて1のときに1を出力する。それ以外の入力値のときは0を出力する。

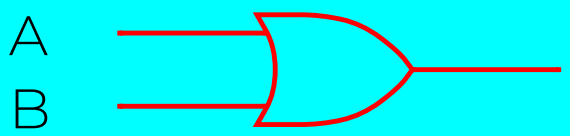
真理値表

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

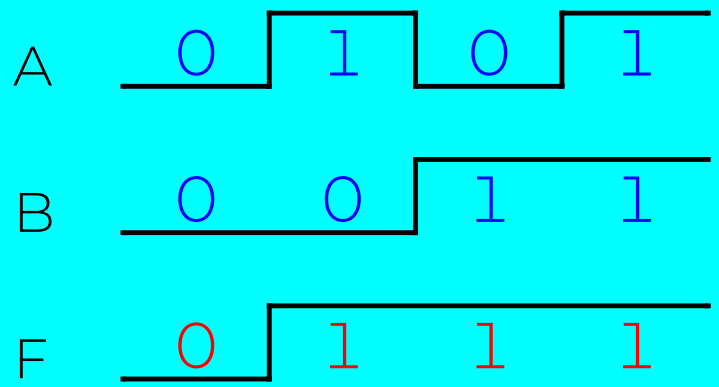
論理積は、入力値がすべて1のときに1を出力する。それ以外の入力値のときは0を出力する。

基本的な論理ゲート — 2. ORゲート

論理和



波形



論理式 $F = A + B$

Verilog HDL の表現：`F = A | B`

論理和 (OR) は、入力値にいずれか 1 が入力されたときに 1 を出力する。それ以外の入力値のときは 0 を出力する。

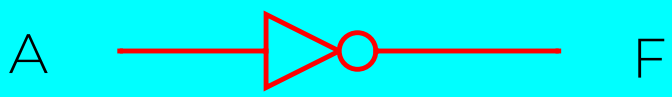
論理和は、入力値がすべて 0 のときに 0 を出力する。それ以外の入力値のときは 1 を出力する。

真理値表

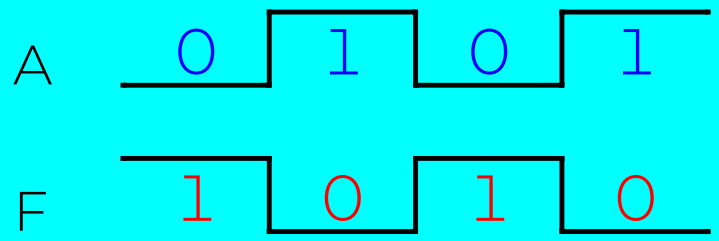
A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

基本的な論理ゲート — 3. NOTゲート

否定



波形



論理式 $F = \overline{A}$

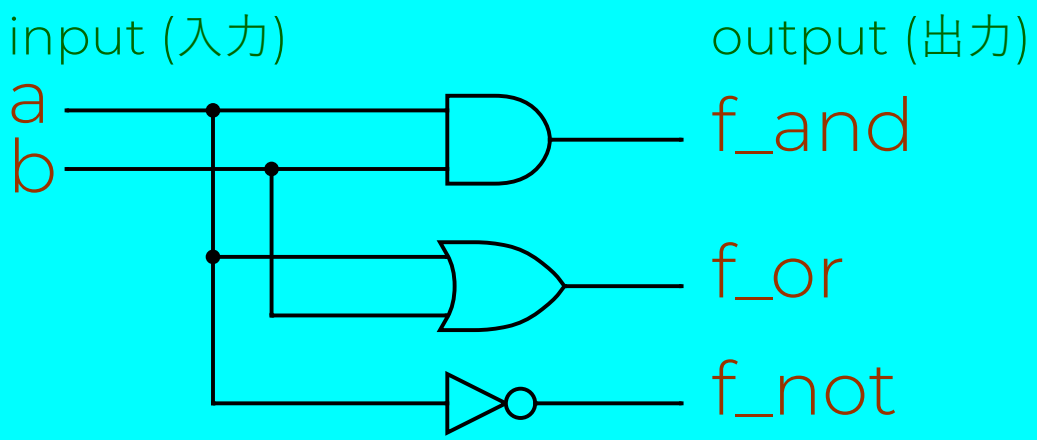
Verilog HDL の表現：`F = ~A`

否定 (NOT) は、入力された値が 1 なら 0 に、0 なら 1 に反転する。

真理値表

A	F
0	1
1	0

回路と Verilog HDL による実装



and_or_not.v

```
`timescale 1ns/1ns

module and_or_not (a, b, f_and, f_or, f_not);
    input  a, b;
    output f_and, f_or, f_not;

    assign f_and = a & b; // AND
    assign f_or  = a | b; // OR
    assign f_not = ~a;    // NOT

endmodule
-:---  and_or_not.v   All L10    (Verilog)
```

and_or_not.v

データフロー記述
(論理式のレベル)

テストベンチ — シミュレーションのため

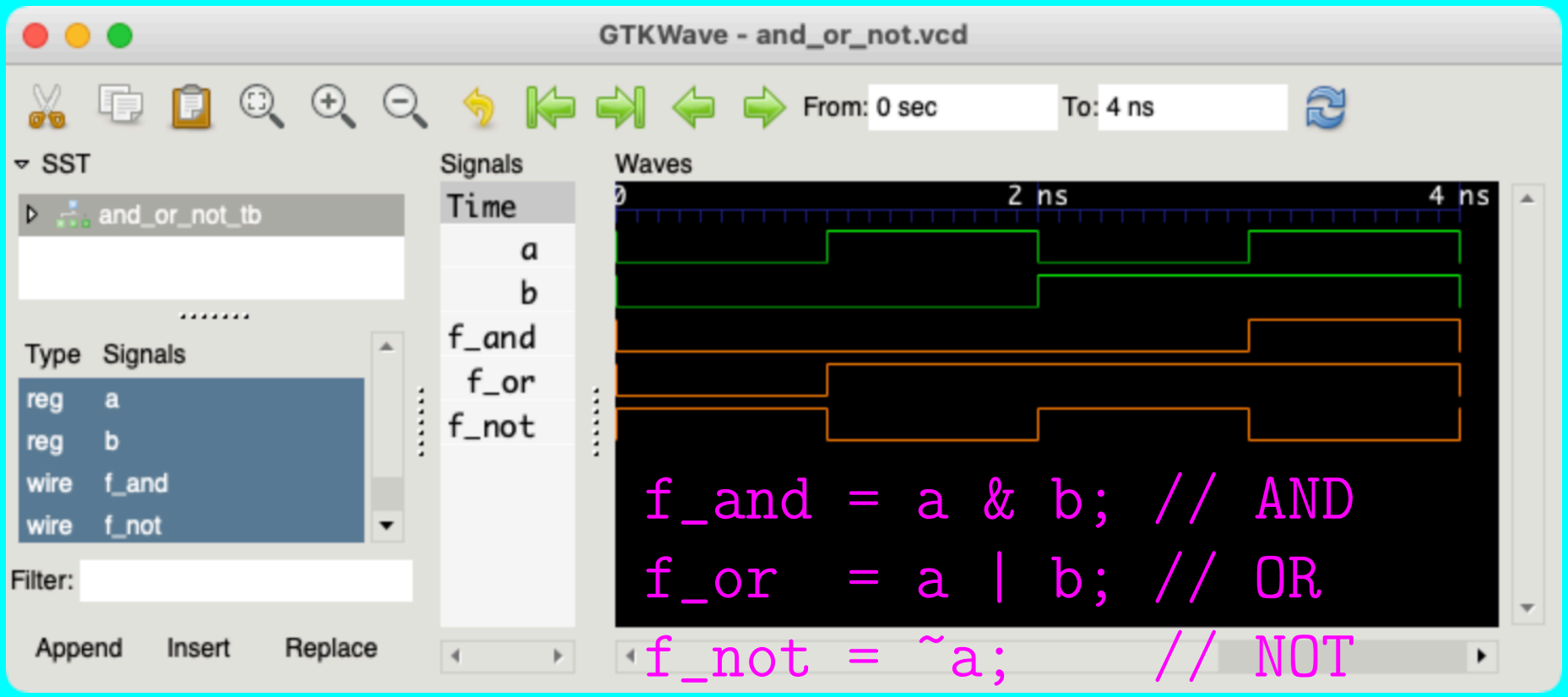
and_or_not_tb.v

```
`timescale 1ns/1ns

module and_or_not_tb;
    reg  a, b;
    wire f_and, f_or, f_not;

    and_or_not i0 (a, b, f_and, f_or, f_not);

    initial begin
        #0 a = 0;
        #0 b = 0;
        #4 $finish;
    end

    always #1 a = ~a;
    always #2 b = ~b;

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

endmodule
-:---  and_or_not_tb.v   All L22    (Verilog)
Beginning of buffer
```

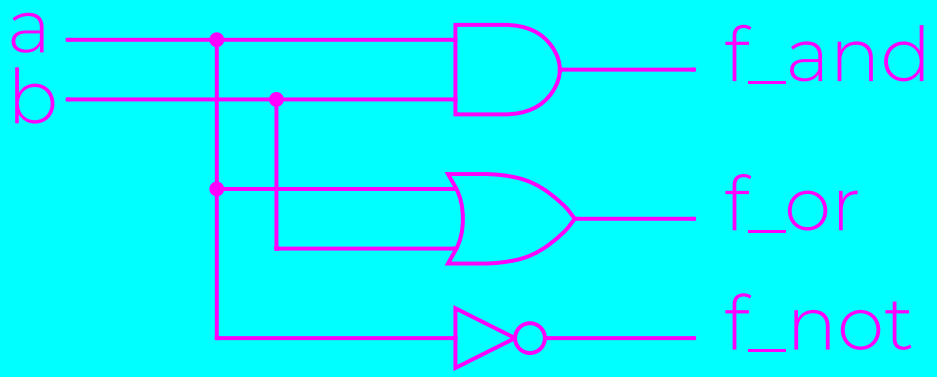
[and_or_not_tb.v]

シミュレーションと波形

```
[yamin@mac logic % iverilog -Wall -o and_or_not and_or_not_tb.v and_or_not.v ]
[yamin@mac logic % ./and_or_not                                              ]
 VCD info: dumpfile and_or_not.vcd opened for output.
[yamin@mac logic % gtkwave and_or_not.vcd                                    ]
```



構造記述Verilog HDLによる実装



異なる記述による実装:

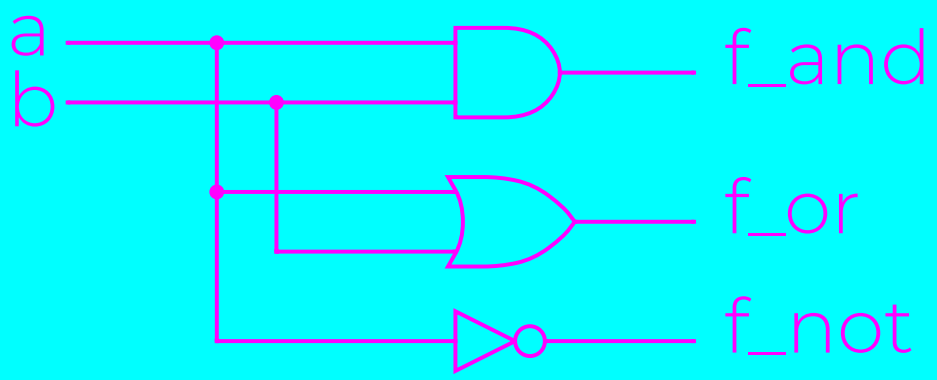
データフロー記述 (論理式のレベル):

```
assign f_and = a & b; // AND
assign f_or  = a | b; // OR
assign f_not = ~a;    // NOT
```

構造記述 (回路図をそのまま):

```
and i0 (f_and, a, b); // and (output, input1, input2)
or  i1 (f_or,  a, b); // or  (output, input1, input2)
not i2 (f_not, a);    // not (output, input)
```

構造記述Verilog HDLによる実装




```
‘timescale 1ns/1ns
module and_or_not_struct (a, b, f_and, f_or, f_not);
    input  a;
    input  b;
    output f_and;
    output f_or;
    output f_not;           // Structural Style (Schematic):
    and i0 (f_and, a, b); // and (output, input1, input2)
    or  i1 (f_or,  a, b); // or  (output, input1, input2)
    not i2 (f_not, a);    // not (output, input)
endmodule
```

構造記述 (回路図をそのまま)

and_or_not_struct.v

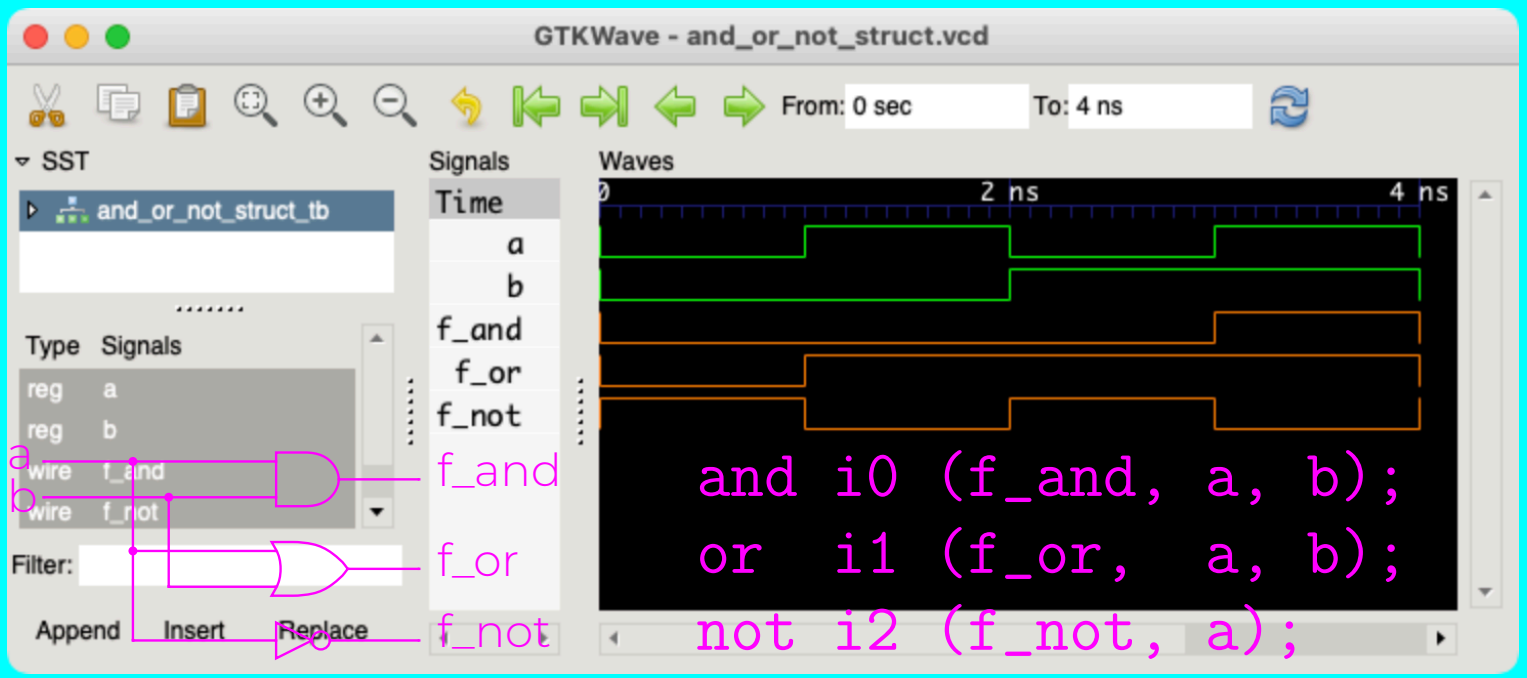
テストベンチ — シミュレーションのため

```
‘timescale 1ns/1ns
module and_or_not_struct_tb;
    reg  a, b;
    wire f_and, f_or, f_not;
    and_or_not_struct i0 (a, b, f_and, f_or, f_not);
    initial begin
        #0 a = 0;
        #0 b = 0;
        #4 $finish;
    end
    always #1 a = ~a;
    always #2 b = ~b;
    initial begin
        $dumpfile ("and_or_not_struct.vcd");
        $dumpvars;
    end
endmodule
```

and_or_not_struct_tb.v

シミュレーションと波形

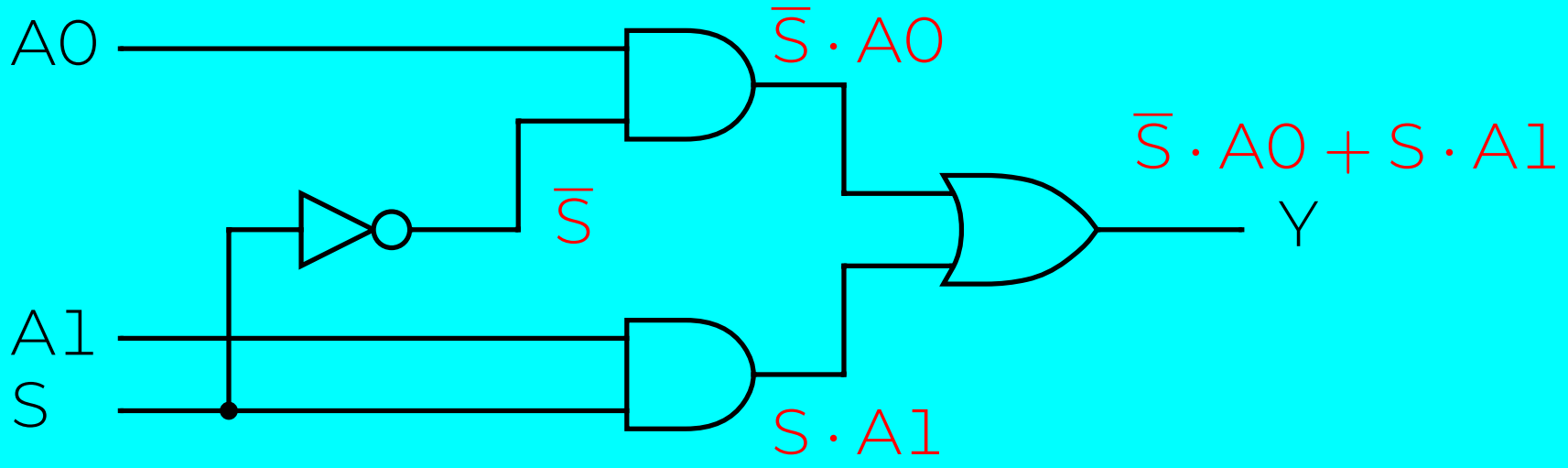
```
% iverilog -Wall -o and_or_not_struct \
  and_or_not_struct_tb.v and_or_not_struct.v
% ./and_or_not_struct
% gtkwave and_or_not_struct.vcd
```



論理式と論理回路の例 (回路図)

論理式の例 $Y = \overline{S} \cdot A0 + S \cdot A1$

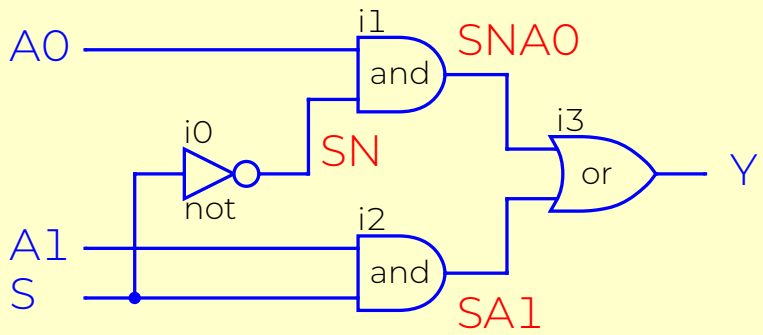
その論理式の論理回路



優先順位: (高い) 否定 → 論理積 → 論理和 (低い)

論理式と論理回路の例 (Verilog HDL)

```
‘timescale 1ns/1ns

module mux2x1_struct (A0, A1, S, Y); // Structural Style
    input  A0;
    input  A1;
    input  S;
    output Y;

    wire   SN;
    wire   SNA0;
    wire   SA1;

    not i0 (SN, S);           // NOT (output, input)
    and i1 (SNA0, SN,  A0); // AND (output, input1, input2)
    and i2 (SA1,  S,   A1); // AND (output, input1, input2)
    or  i3 (Y, SNA0,  SA1); // OR  (output, input1, input2)
endmodule
```



構造記述 (回路図をそのまま)

mux2x1_struct.v

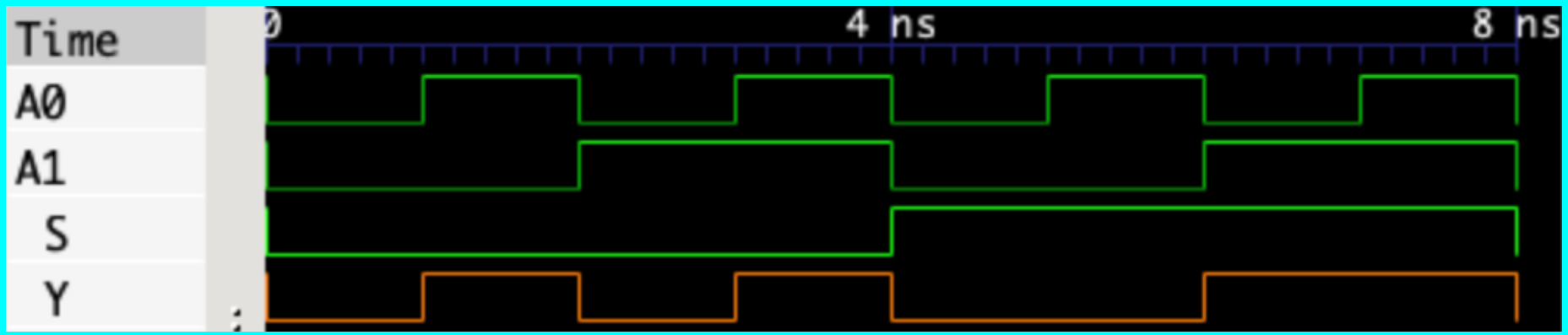
論理式と論理回路の例 (テストベンチ)

```
‘timescale 1ns/1ns
module mux2x1_struct_tb;
    reg  A0, A1, S;
    wire Y;
    mux2x1_struct i0 (A0, A1, S, Y);
    initial begin
        #0 A0 = 0; A1 = 0; S = 0;
        #8 $finish;
    end
    always #1 A0 = ~A0;
    always #2 A1 = ~A1;
    always #4 S  = ~S;
    initial begin
        $dumpfile ("mux2x1_struct.vcd");
        $dumpvars;
    end
endmodule
```

mux2x1_struct_tb.v

論理式と論理回路の例 (波形)

```
% iverilog -Wall -o mux2x1_struct \
  mux2x1_struct_tb.v mux2x1_struct.v
% ./mux2x1_struct
% gtkwave mux2x1_struct.vcd
```

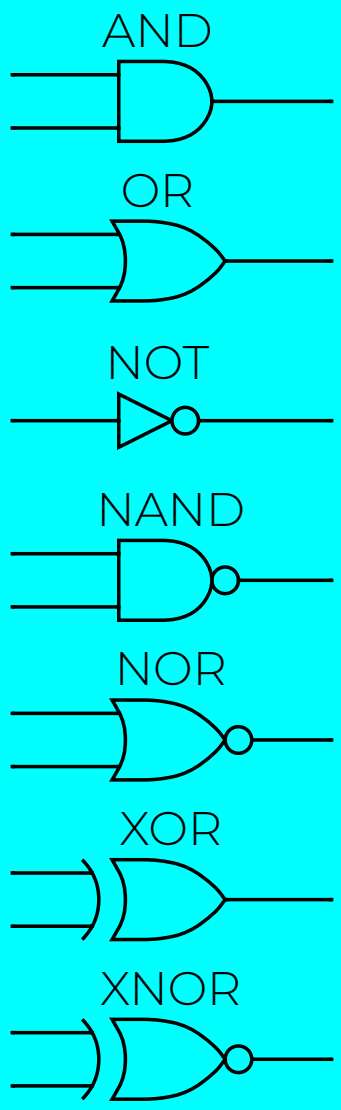


S = 0 の時、$Y = \overline{S} \cdot A0 + S \cdot A1 = 1 \cdot A0 + 0 \cdot A1 = A0 + 0 = A0$

S = 1 の時、$Y = \overline{S} \cdot A0 + S \cdot A1 = 0 \cdot A0 + 1 \cdot A1 = 0 + A1 = A1$

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)`

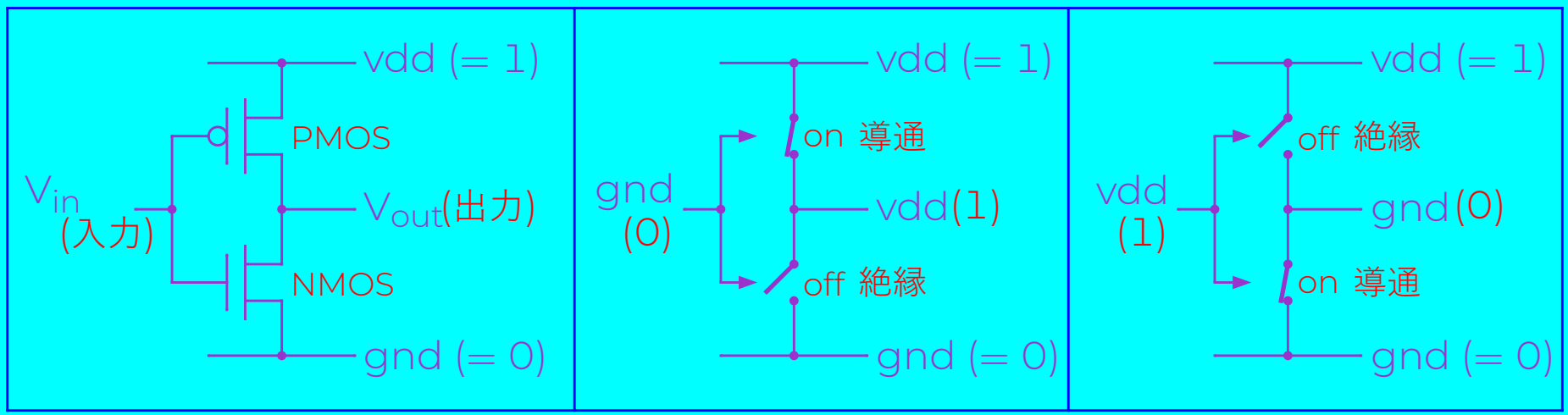


CMOS トランジスタ

コンピュータの中には「1」と「0」のデータしかない。

その「1」と「0」を作り出すのが CMOS トランジスタである。

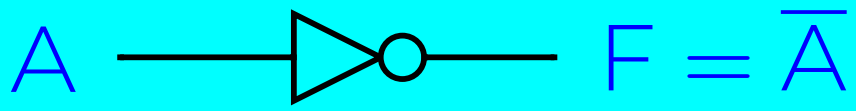
トランジスタには、P 型トランジスタ（PMOS）と N 型トランジスタ（NMOS）があり、この 2 つのトランジスタを組み合わせて使うのが CMOS トランジスタである。



入力が gnd のとき、PMOS が導通、NMOS が絶縁状態になるので、出力は vdd となる。

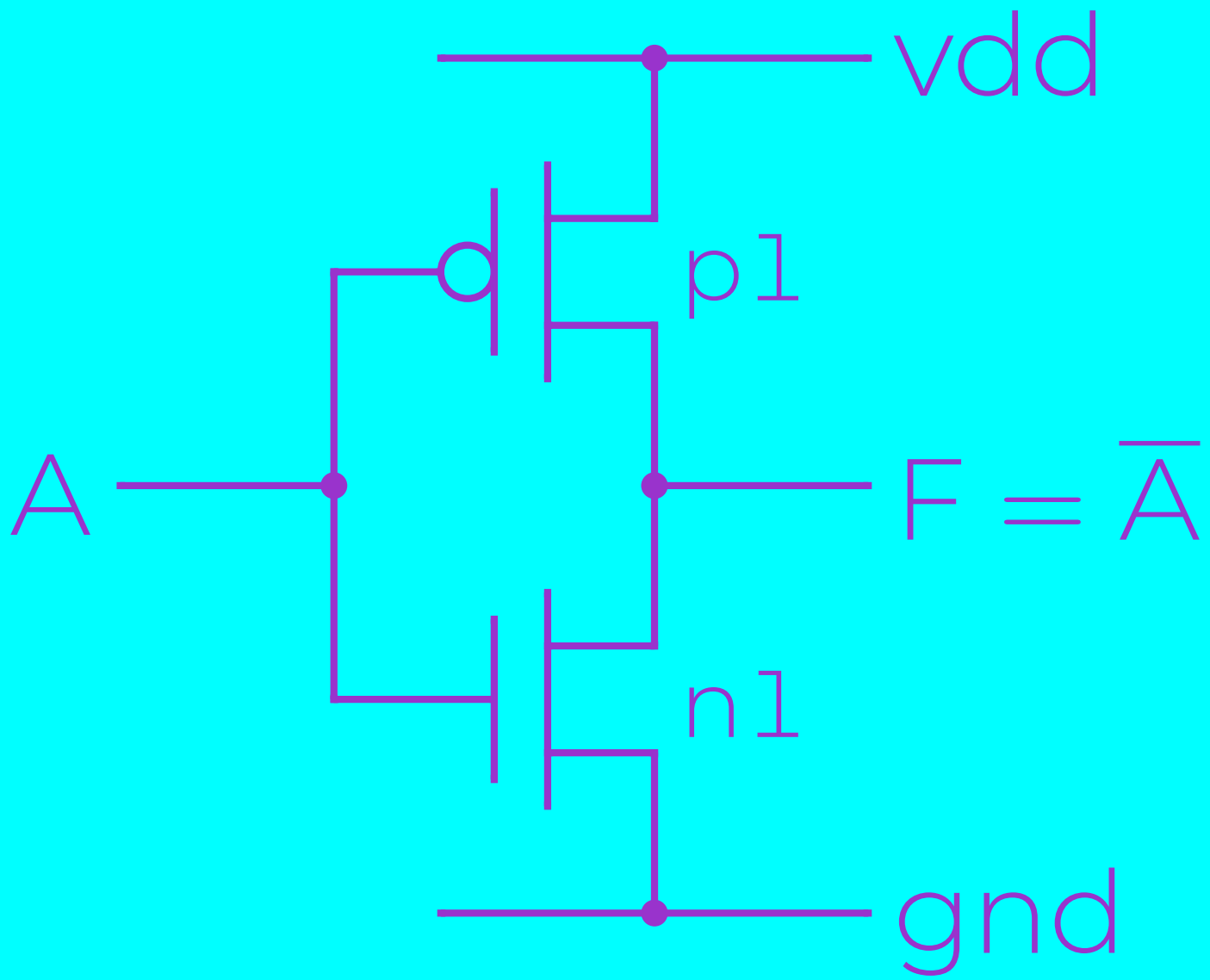
入力が vdd のとき、PMOS が絶縁、NMOS が導通状態になるので、出力は gnd となる。

NOT ゲート



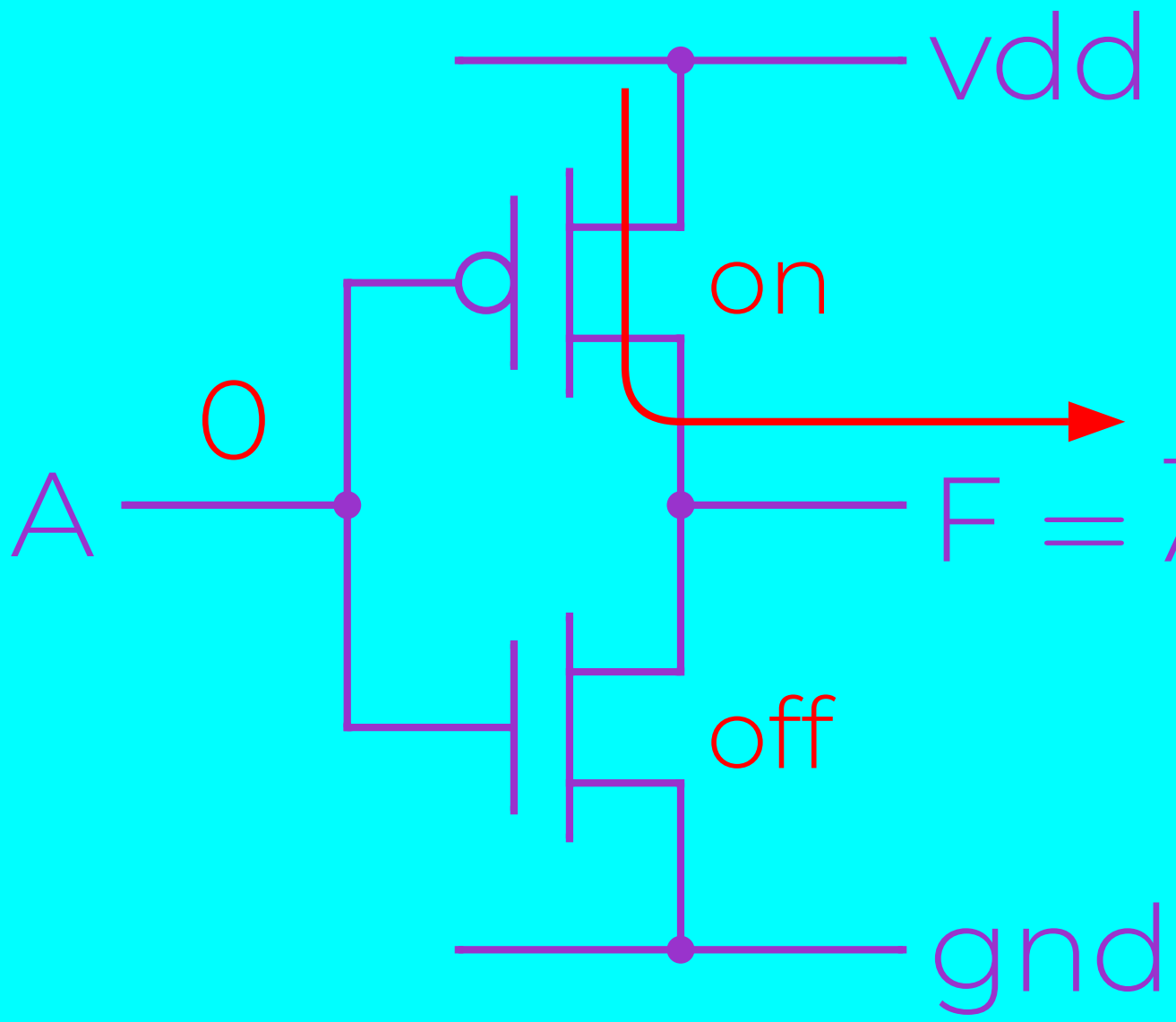
A	$\overline{A}$
0	1
1	0

NOT ゲートの CMOS 型実装



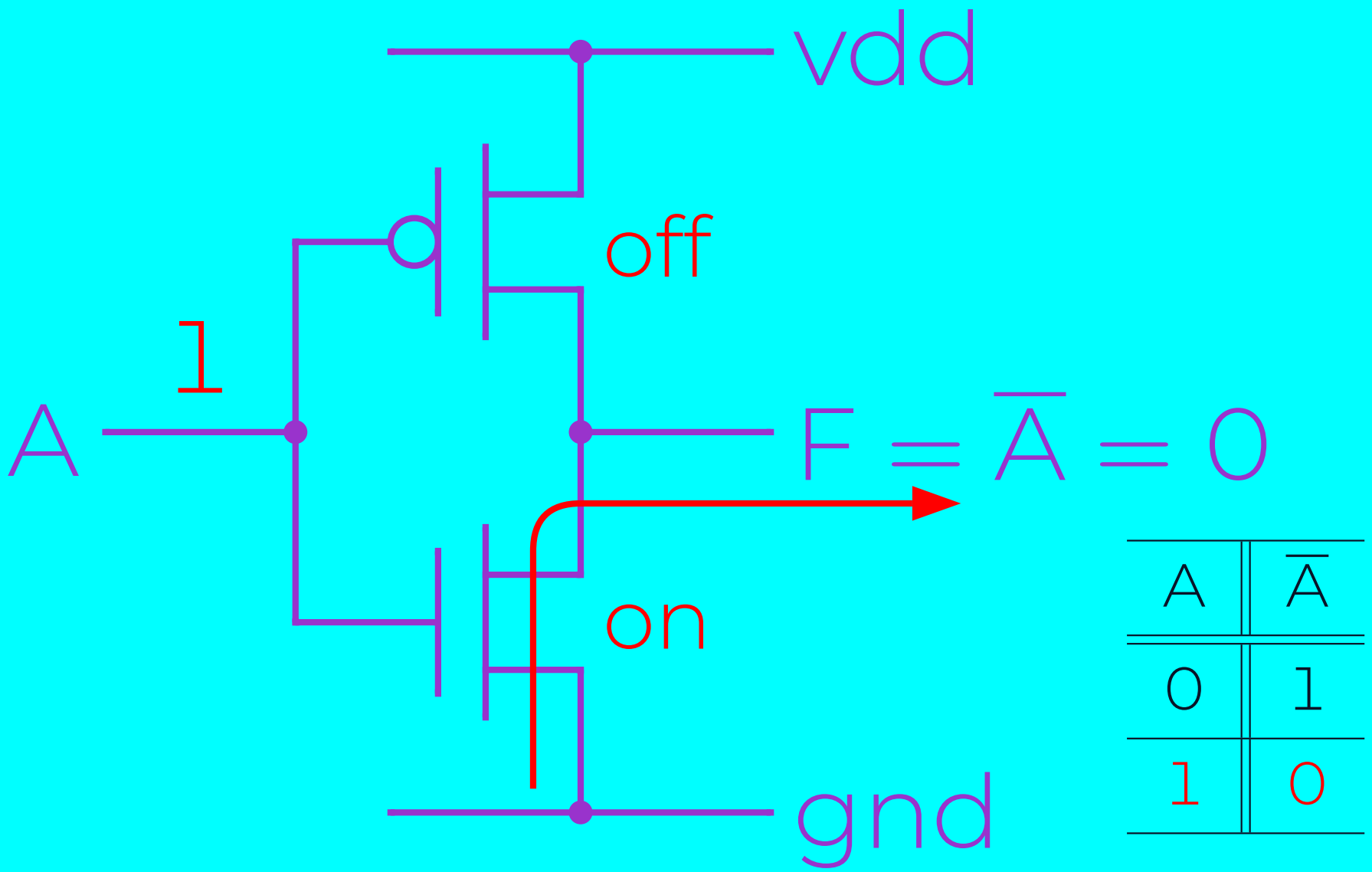
A	$\overline{A}$
0	1
1	0

NOT ゲートの CMOS 型実装



A	$\overline{A}$
0	1
1	0

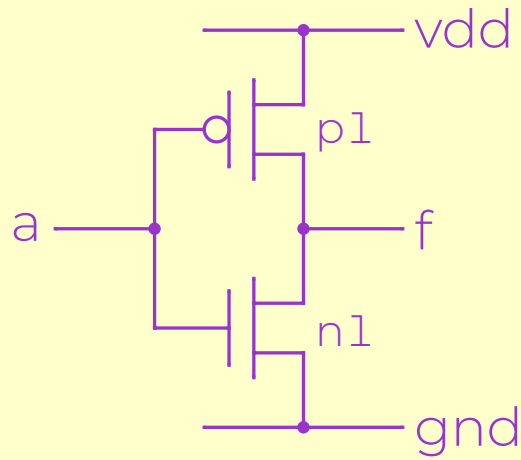
NOT ゲートの CMOS 型実装



A	$\overline{A}$
0	1
1	0

NOT ゲートの CMOS 型実装

```
‘timescale 1ns/1ns

module cmosnot (f, a); // a circuit
    input   a;   // inputs
    output  f;   // output

    supply1 vdd; // vdd
    supply0 gnd; // gnd

    // pmos (drain, source, gate);
    pmos p1 (f,      vdd,    a);

    // nmos (drain, source, gate);
    nmos n1 (f,      gnd,    a);

endmodule
```



構造記述 (回路図をそのまま)

cmosnot.v

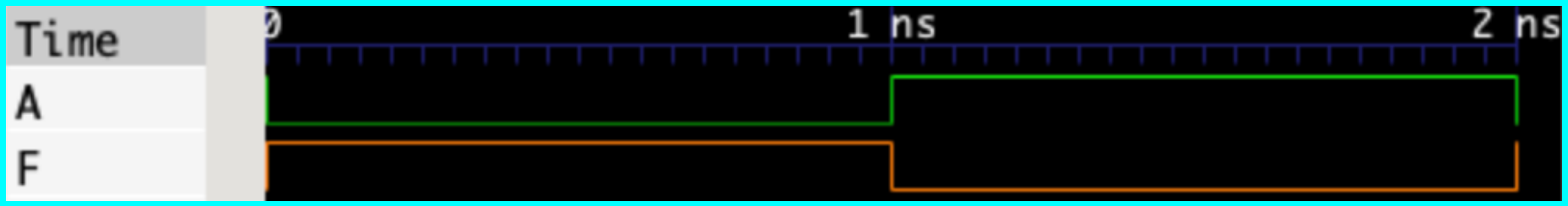
NOT ゲートの CMOS 型実装

```
‘timescale 1ns/1ns

module cmosnot_tb; // a test bench, not a circuit
    reg  A;
    wire F;
    cmosnot i0 (F, A); // invoke cmosnot
    initial begin
        A = 0;
        #2 $finish;
    end
    always #1 A = ~A;
    initial begin
        $dumpfile ("cmosnot.vcd");
        $dumpvars;
    end
endmodule
```

cmosnot_tb.v

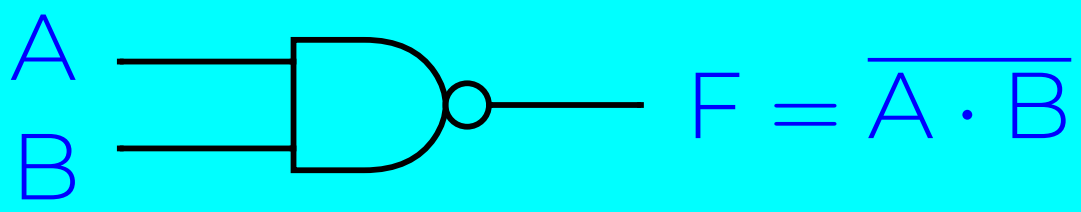
NOT ゲートの CMOS 型実装

```
% iverilog -Wall -o cmosnot \
  cmosnot_tb.v cmosnot.v
% ./cmosnot
% gtkwave cmosnot.vcd
```



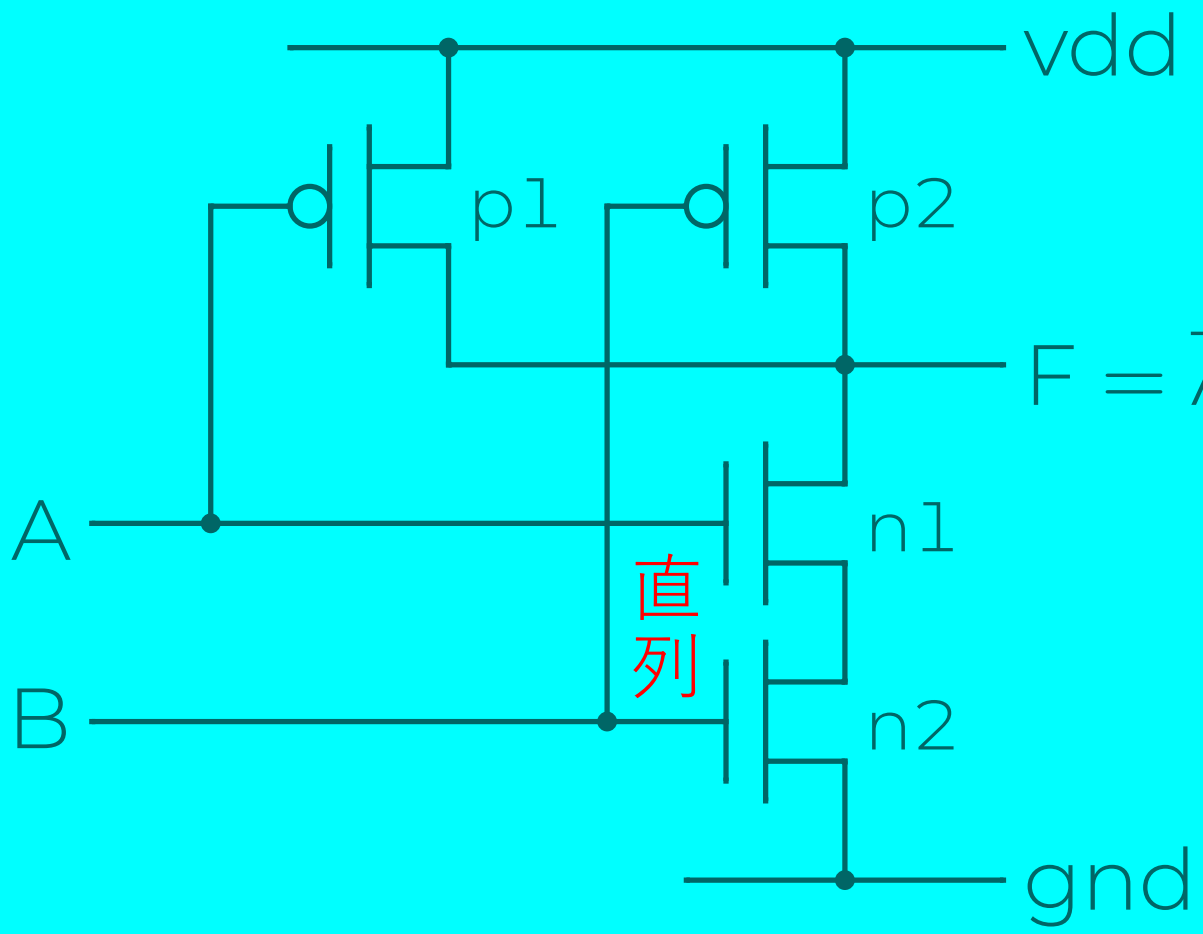
A	$F = \overline{A}$
0	1
1	0

NAND ゲート



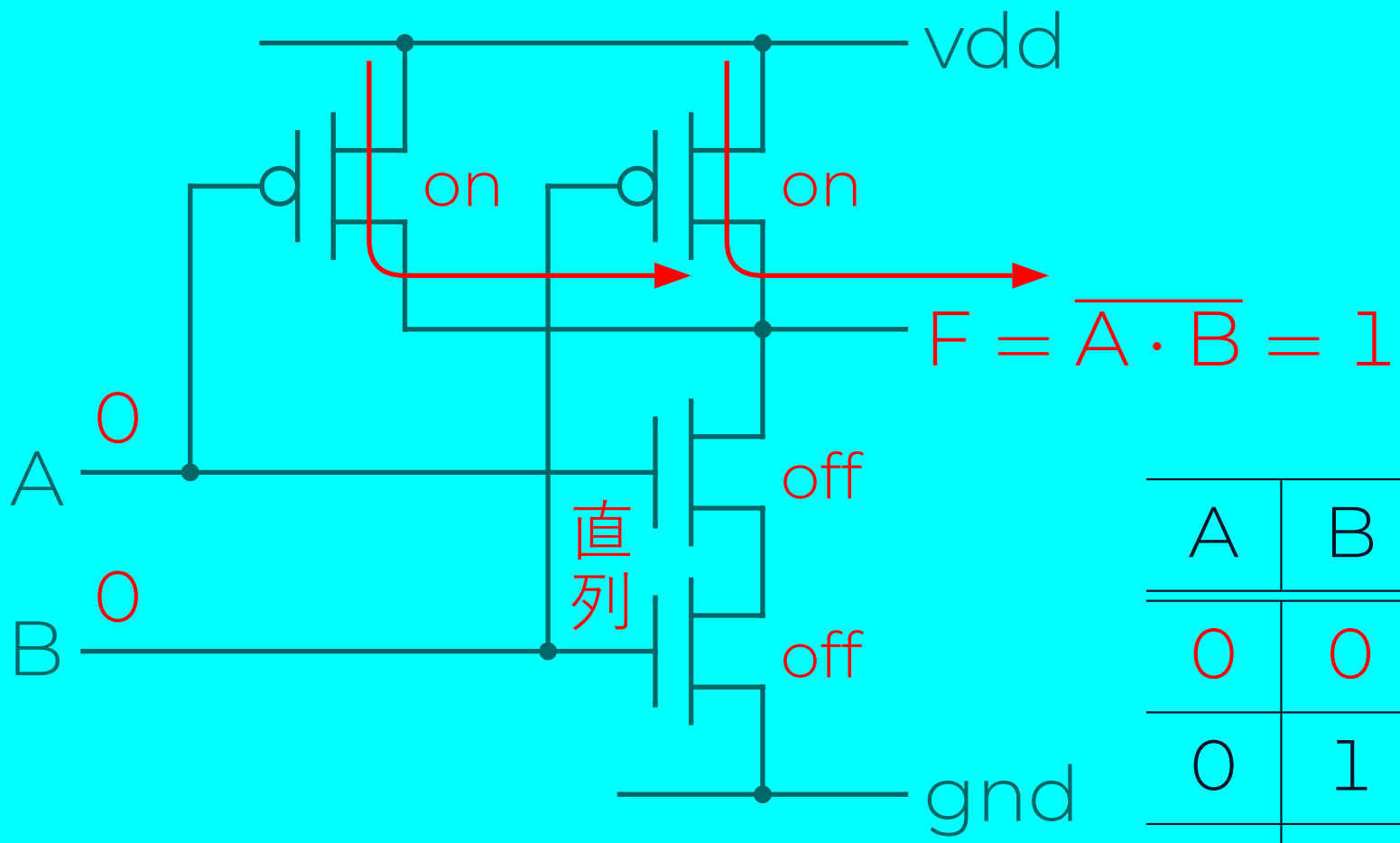
A	B	$A \cdot B$	$\overline{A \cdot B}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

NAND ゲートの CMOS 型実装



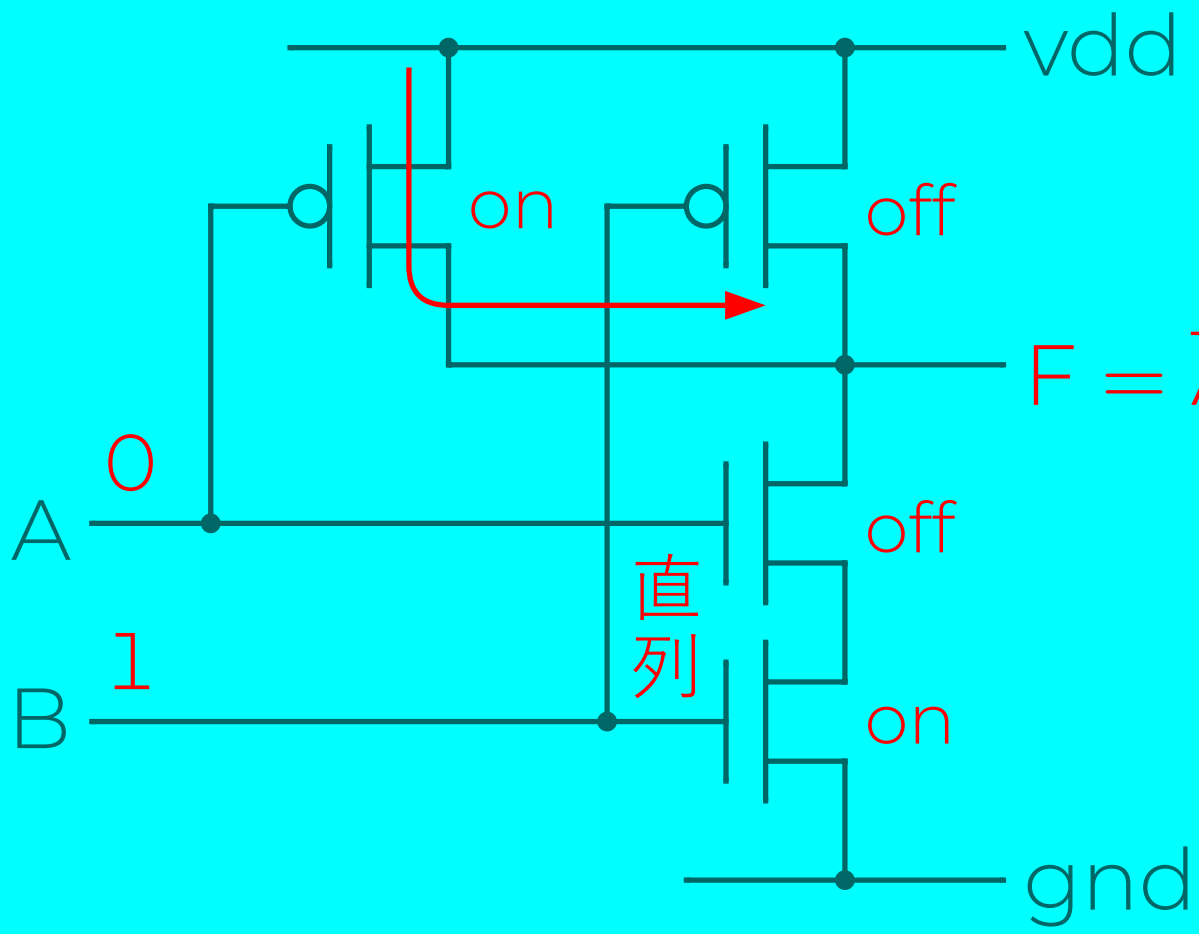
A	B	$\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

NAND ゲートの CMOS 型実装



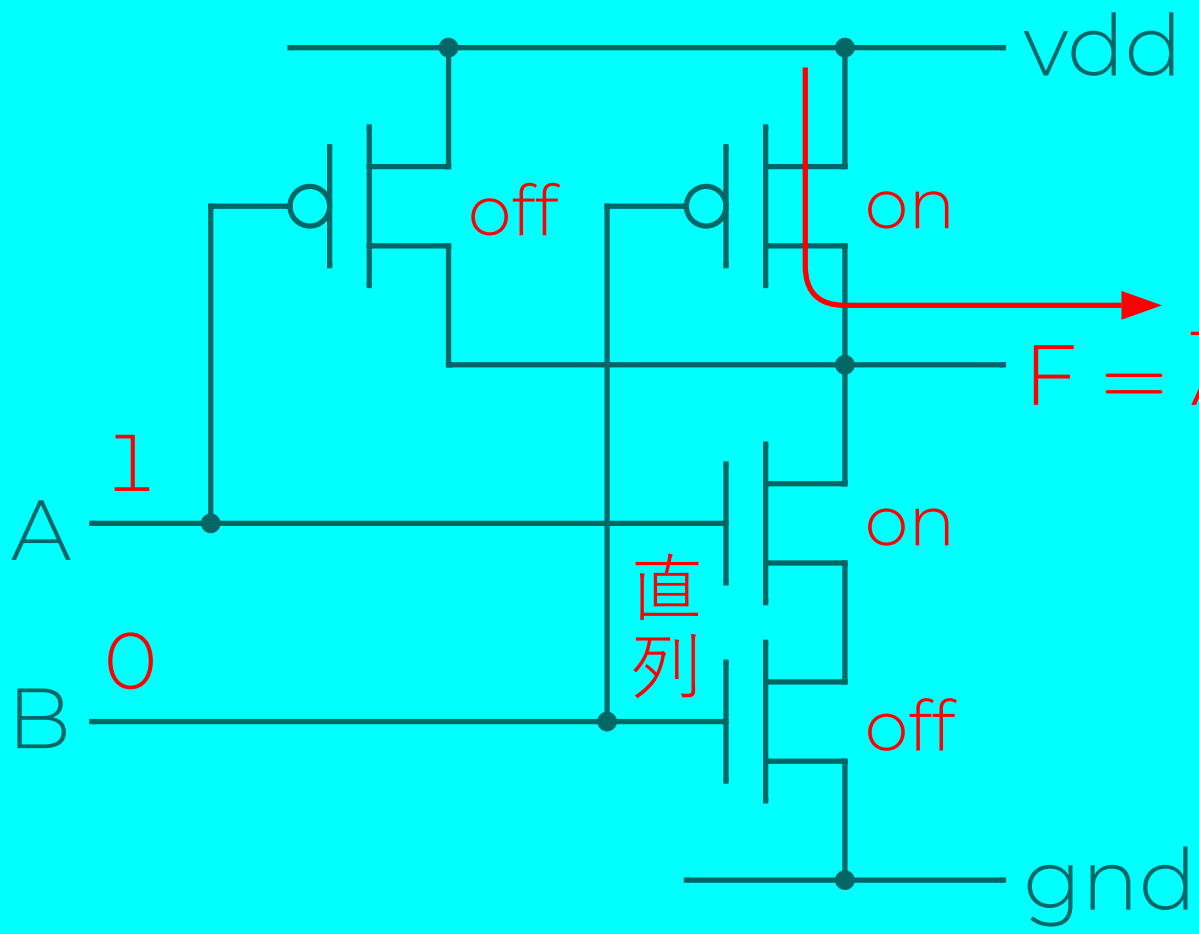
A	B	$\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

NAND ゲートの CMOS 型実装



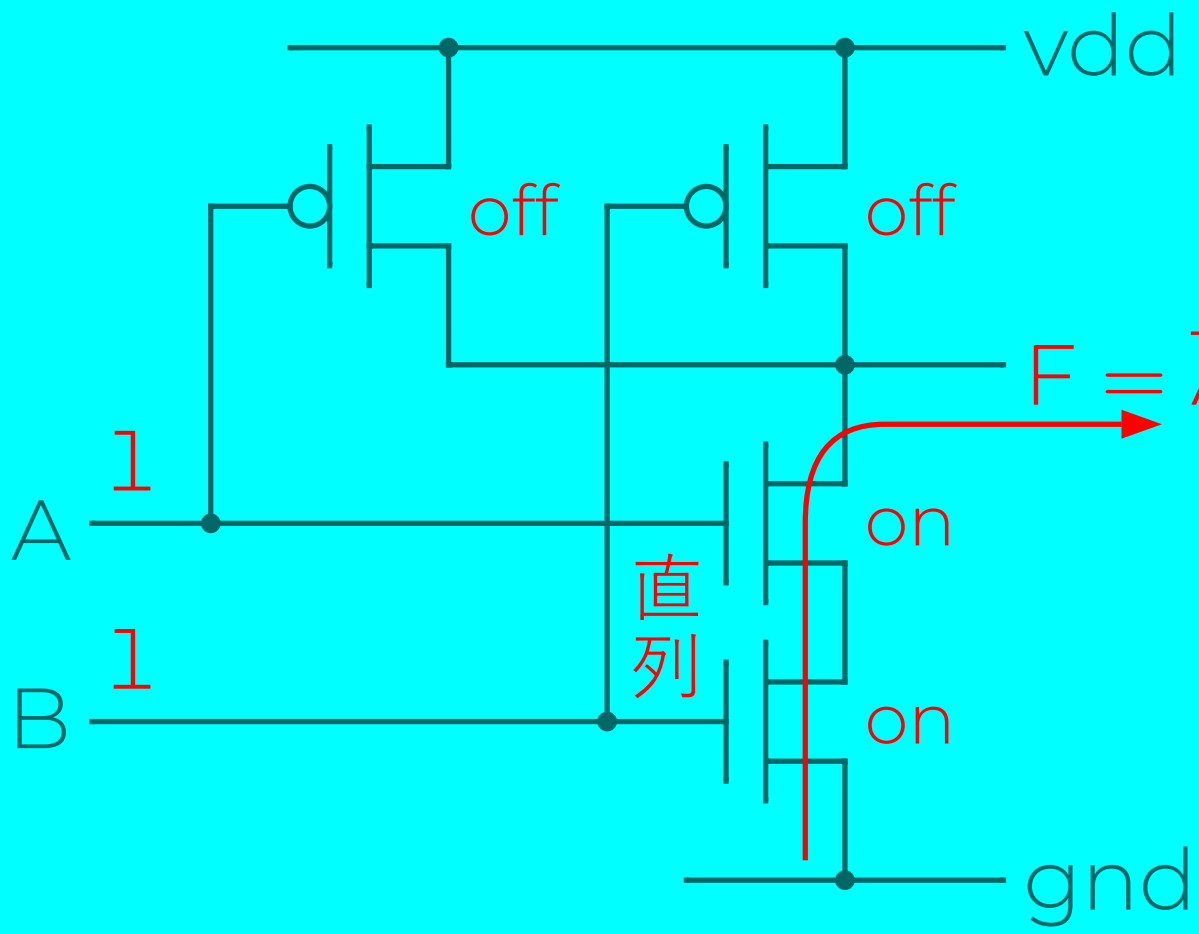
A	B	$\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

NAND ゲートの CMOS 型実装



A	B	$\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

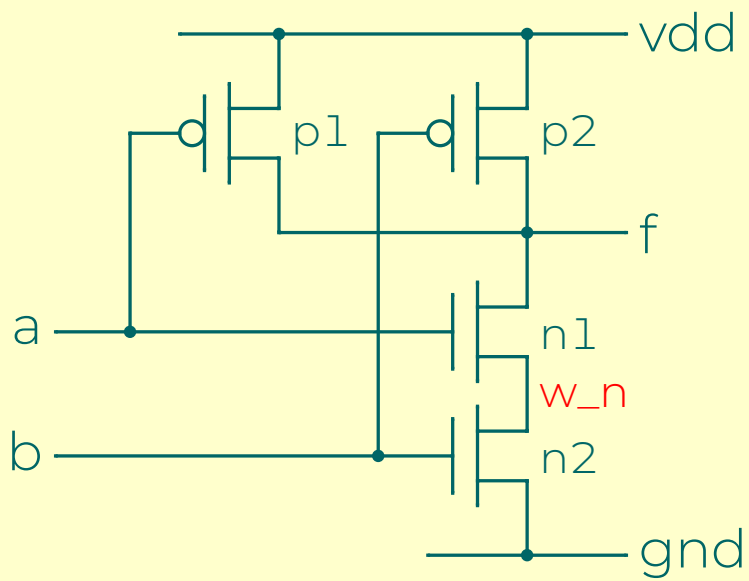
NAND ゲートの CMOS 型実装



A	B	$\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

NAND ゲートの CMOS 型実装

```
‘timescale 1ns/1ns
module cmosnand (f, a, b); // a circuit
    input   a, b; // inputs
    output  f;    // output

    supply1 vdd;  // vdd
    supply0 gnd;  // gnd
    wire    w_n;  // internal wire

    // pmos (drain, source, gate);
    pmos p1 (f,     vdd,    a);
    pmos p2 (f,     vdd,    b);

    // nmos (drain, source, gate);
    nmos n1 (f,     w_n,    a);
    nmos n2 (w_n,   gnd,    b);
endmodule
```



構造記述 (回路図をそのまま)

cmosnand.v

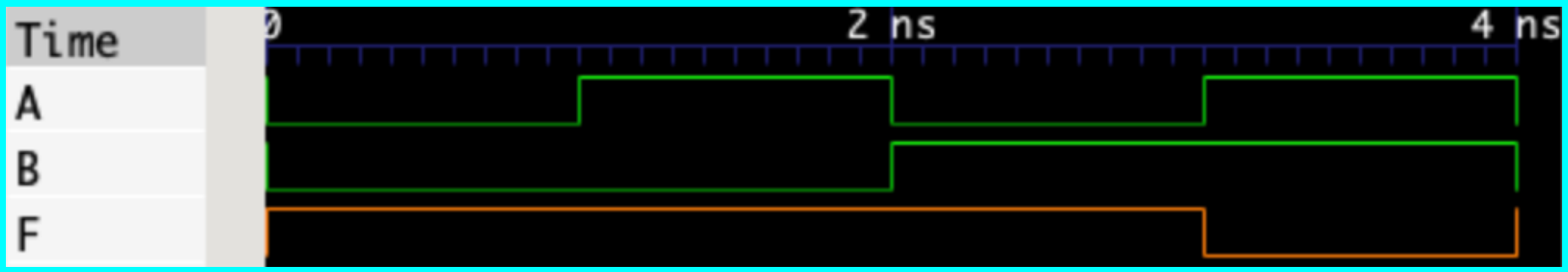
NAND ゲートの CMOS 型実装

```
‘timescale 1ns/1ns
module cmosnand_tb; // a test bench, not a circuit
    reg  A, B;
    wire F;
    cmosnand not_and (F, A, B); // invoke cmosnand
    initial begin
        A = 0;
        B = 0;
        #4 $finish;
    end
    always #1 A = ~A;
    always #2 B = ~B;
    initial begin
        $dumpfile ("cmosnand.vcd");
        $dumpvars;
    end
endmodule
```

cmosnand_tb.v

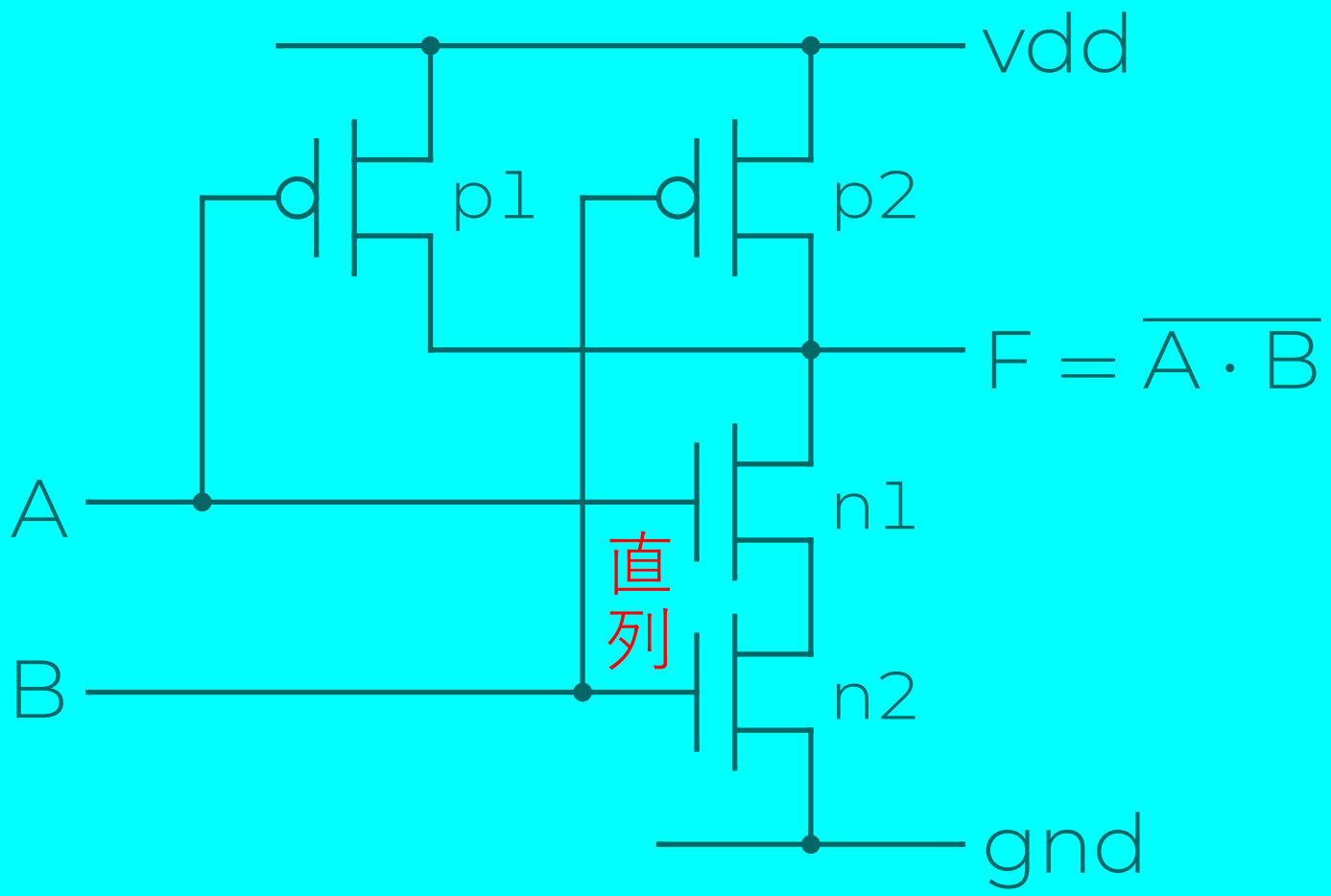
NAND ゲートの CMOS 型実装

```
% iverilog -Wall -o cmosnand \
  cmosnand_tb.v cmosnand.v
% ./cmosnand
% gtkwave cmosnand.vcd
```

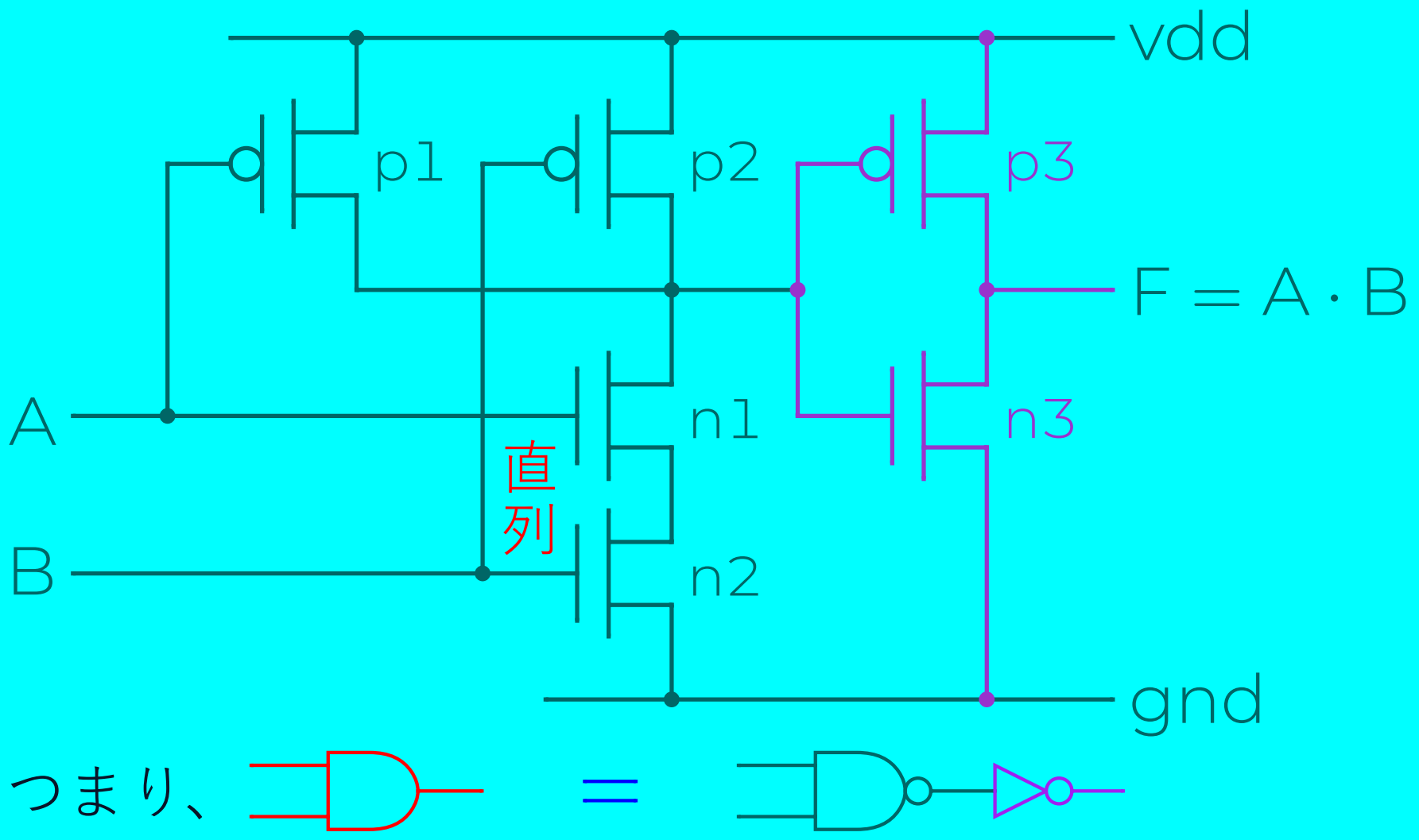


A	B	$A \cdot B$	$F = \overline{A \cdot B}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

NAND ゲートの CMOS 型実装



AND ゲートの CMOS 型実装



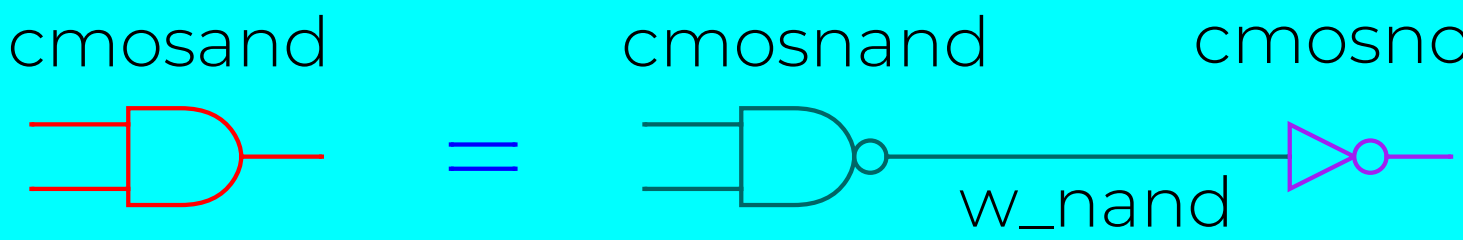
つまり、

Note: A PMOS is connected to the vdd and NMOS is connected to the ground.

AND ゲートの CMOS 型実装

```
‘timescale 1ns/1ns

module cmosand (f, a, b); // a circuit
    input   a, b;   // inputs
    output  f;      // output

    wire    w_nand; // internal wire

    cmosnand i0 (w_nand, a, b);
    cmosnot  i1 (f, w_nand);

endmodule
```

cmosand.v



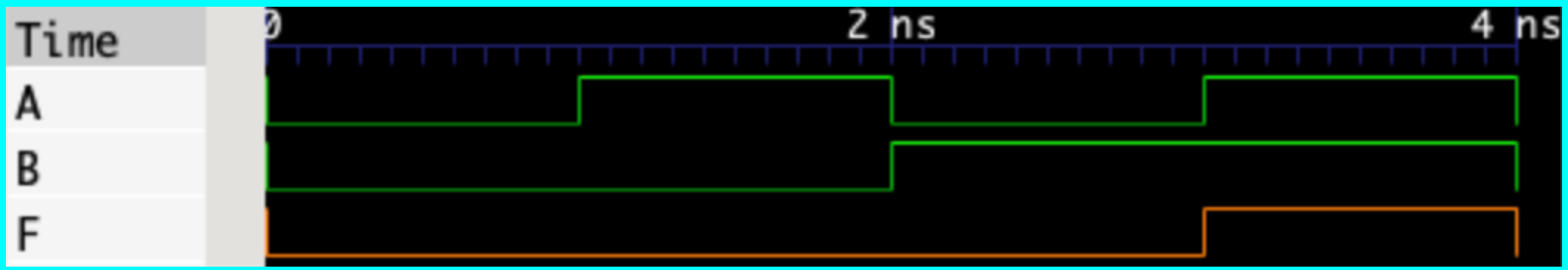
AND ゲートの CMOS 型実装

```
‘timescale 1ns/1ns
module cmosand_tb; // a test bench, not a circuit
    reg  A, B;
    wire F;
    cmosand i0 (F, A, B); // invoke cmosand
    initial begin
        A = 0;
        B = 0;
        #4 $finish;
    end
    always #1 A = ~A;
    always #2 B = ~B;
    initial begin
        $dumpfile ("cmosand.vcd");
        $dumpvars;
    end
endmodule
```

cmosand_tb.v

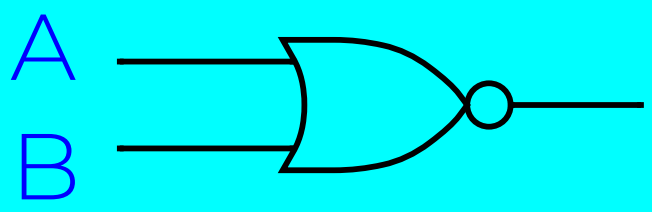
AND ゲートの CMOS 型実装

```
% iverilog -Wall -o cmosand \
  cmosand_tb.v cmosand.v cmosnand.v cmosnot.v
% ./cmosand
% gtkwave cmosand.vcd
```



A	B	F = A · B
0	0	0
0	1	0
1	0	0
1	1	1

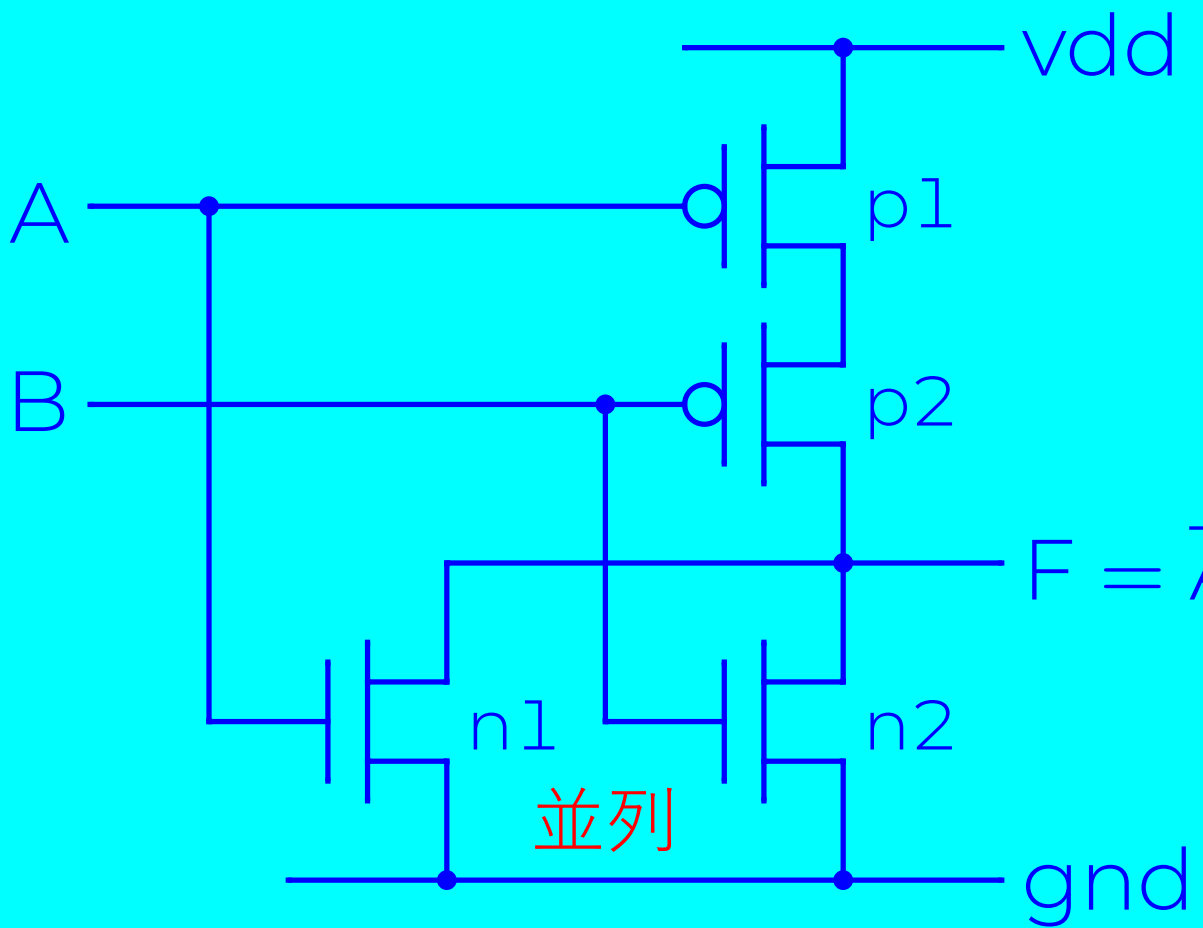
NOR ゲート



$$F = \overline{A + B}$$

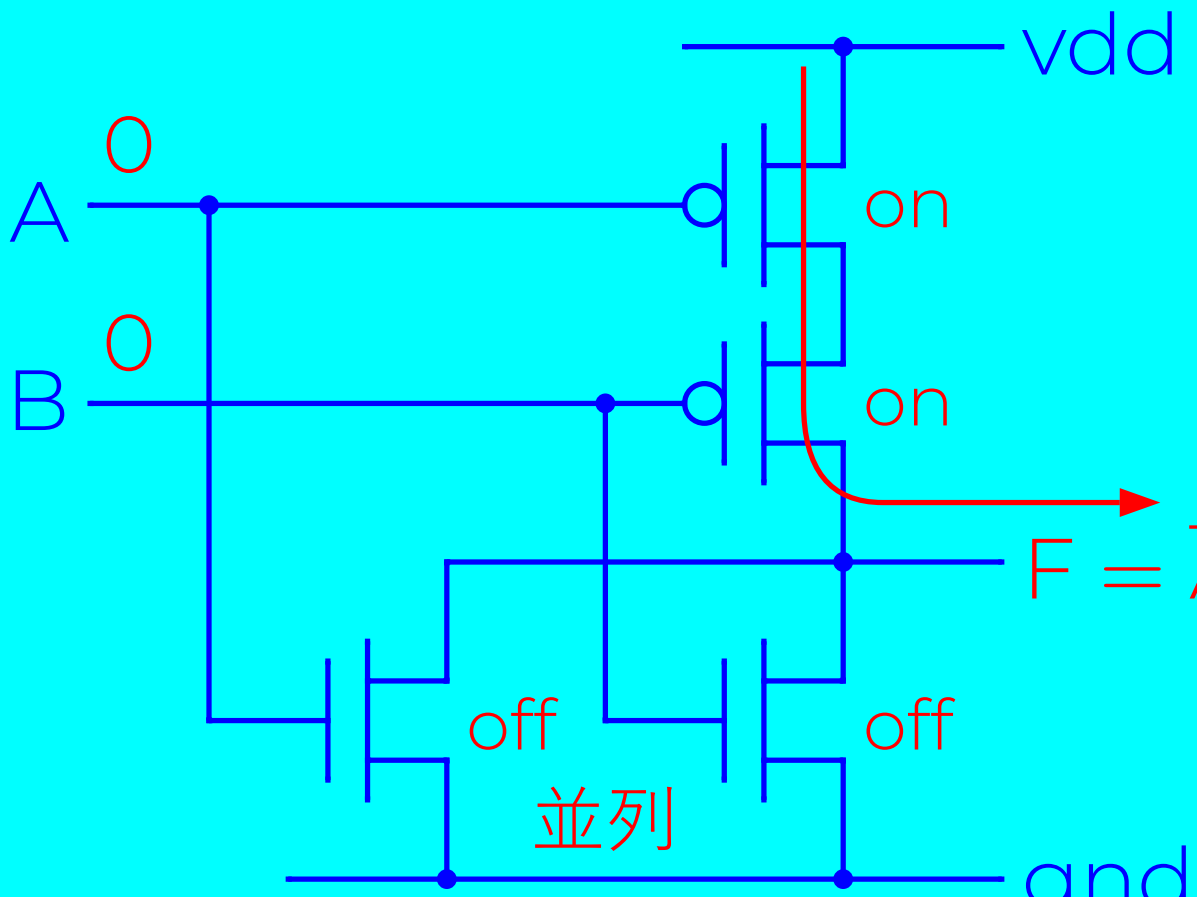
A	B	$A + B$	$\overline{A + B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

NOR ゲートの CMOS 型実装



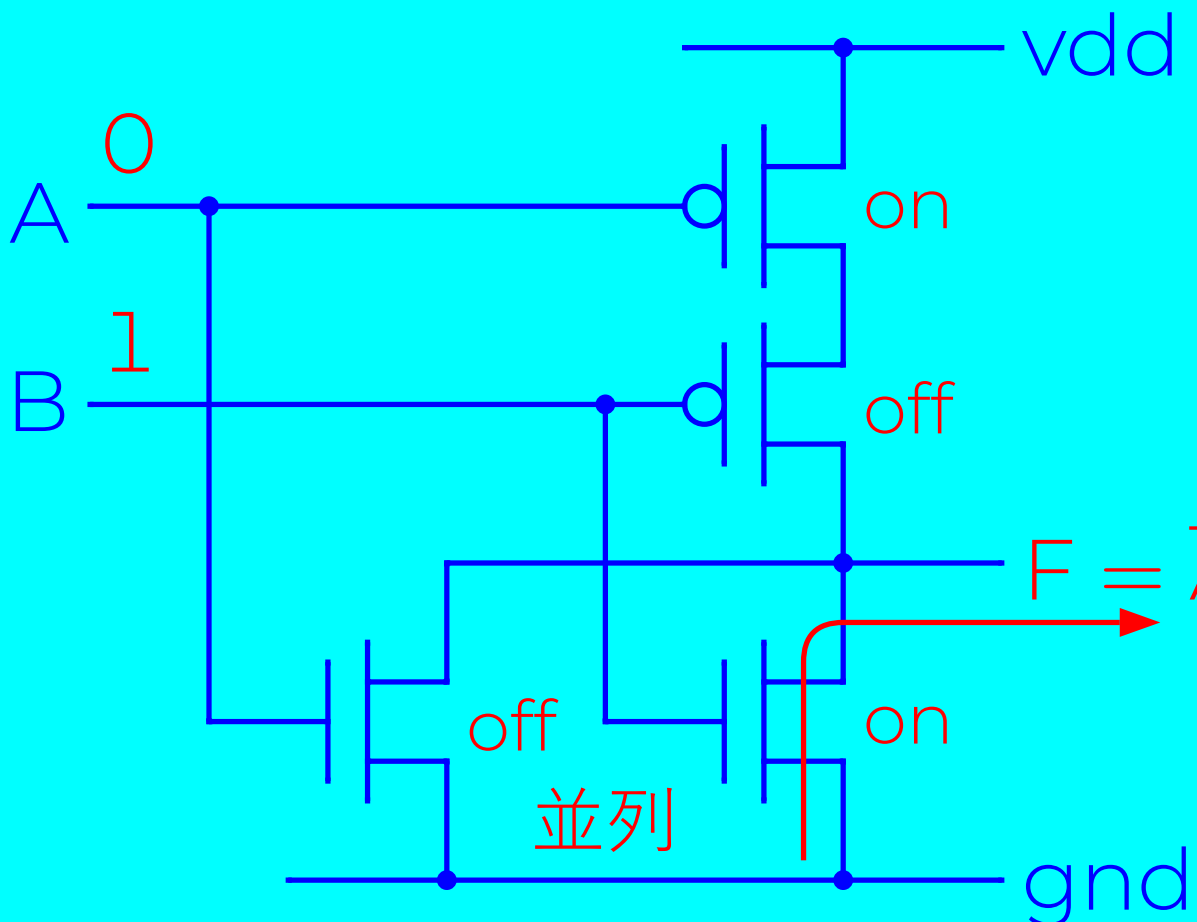
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

NOR ゲートの CMOS 型実装



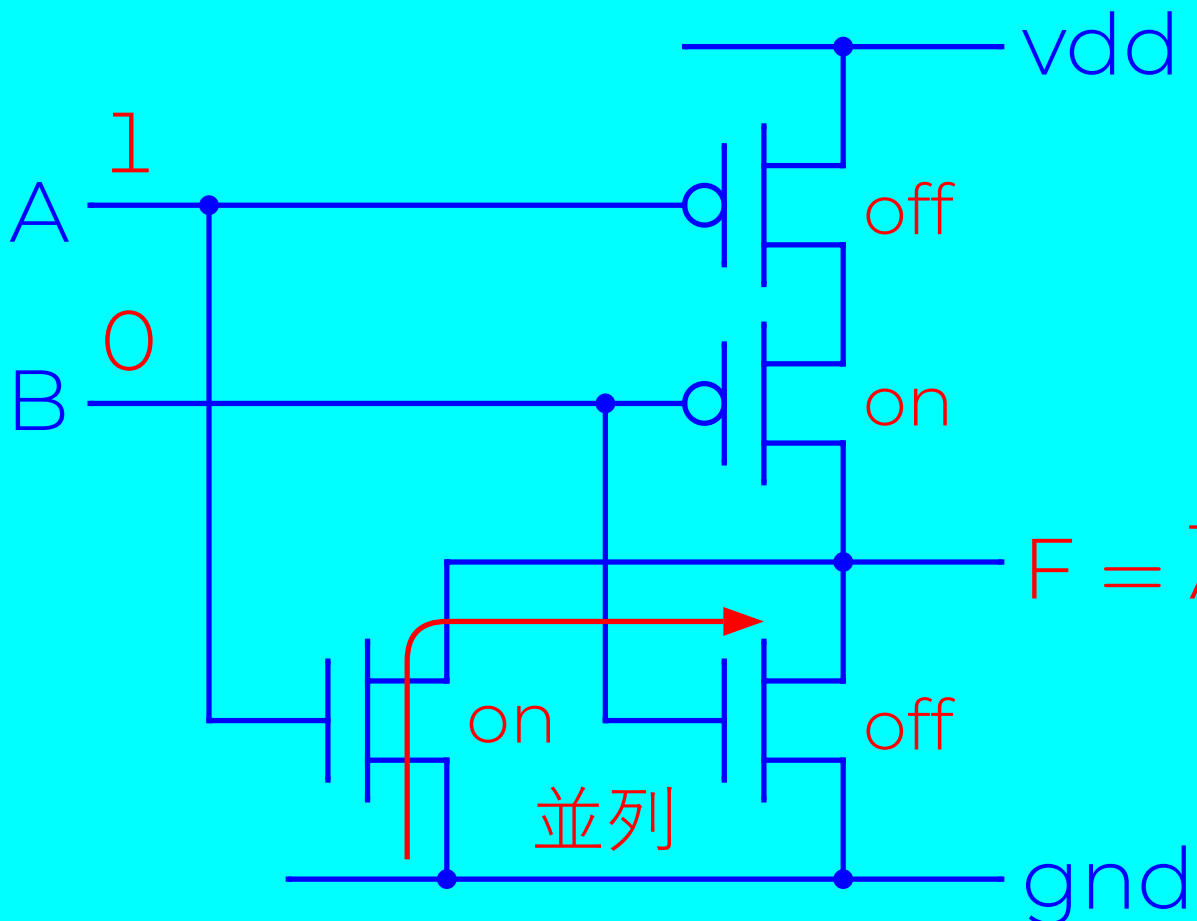
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

NOR ゲートの CMOS 型実装



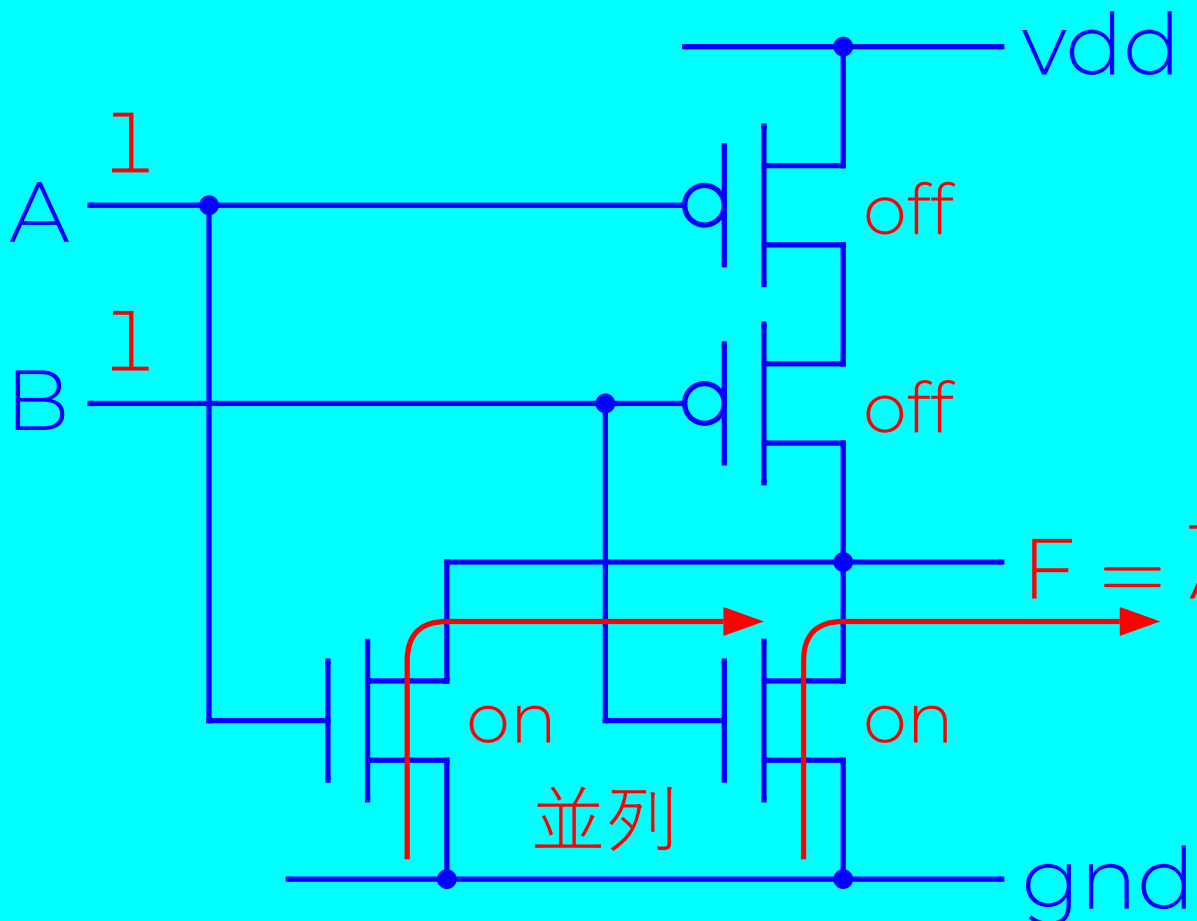
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

NOR ゲートの CMOS 型実装



A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

NOR ゲートの CMOS 型実装

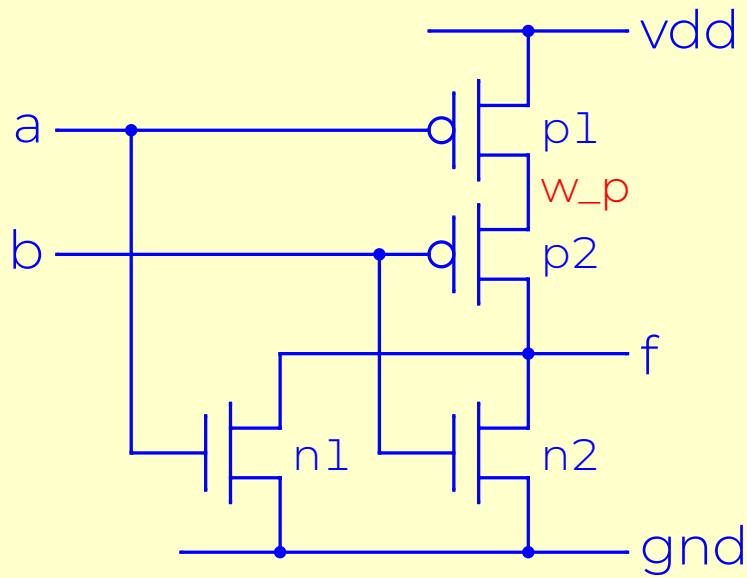


$F = \overline{A + B} = 0$

A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

NOR ゲートの CMOS 型実装

```
‘timescale 1ns/1ns
module cmosnor (f, a, b); // a circuit
    input   a, b; // inputs
    output  f;    // output

    supply1 vdd;  // vdd
    supply0 gnd;  // gnd
    wire    w_p;  // internal wire

    // pmos (drain, source, gate);
    pmos p1 (w_p,   vdd,    a);
    pmos p2 (f,     w_p,    b);

    // nmos (drain, source, gate);
    nmos n1 (f,     gnd,    a);
    nmos n2 (f,     gnd,    b);
endmodule
```



構造記述 (回路図をそのまま)

cmosnor.v

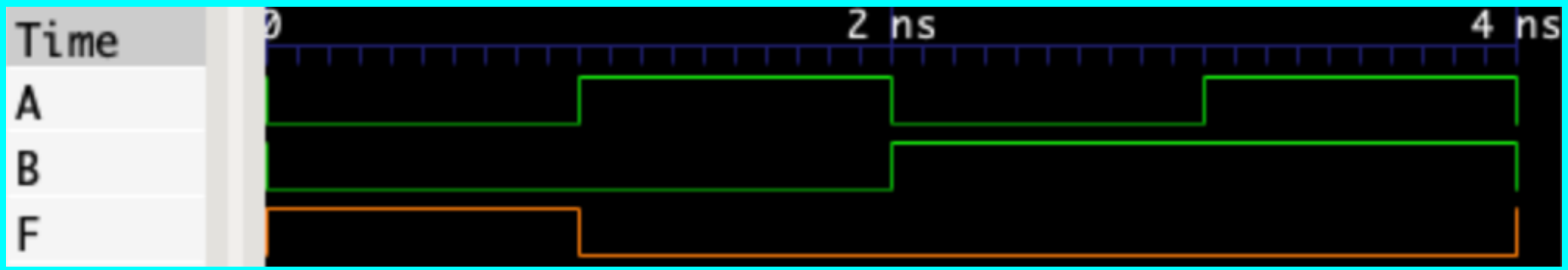
NOR ゲートの CMOS 型実装

```
‘timescale 1ns/1ns
module cmosnor_tb; // a test bench, not a circuit
    reg  A, B;
    wire F;
    cmosnor not_or (F, A, B); // invoke cmosnor
    initial begin
        A = 0;
        B = 0;
        #4 $finish;
    end
    always #1 A = ~A;
    always #2 B = ~B;
    initial begin
        $dumpfile ("cmosnor.vcd");
        $dumpvars;
    end
endmodule
```

cmosnor_tb.v

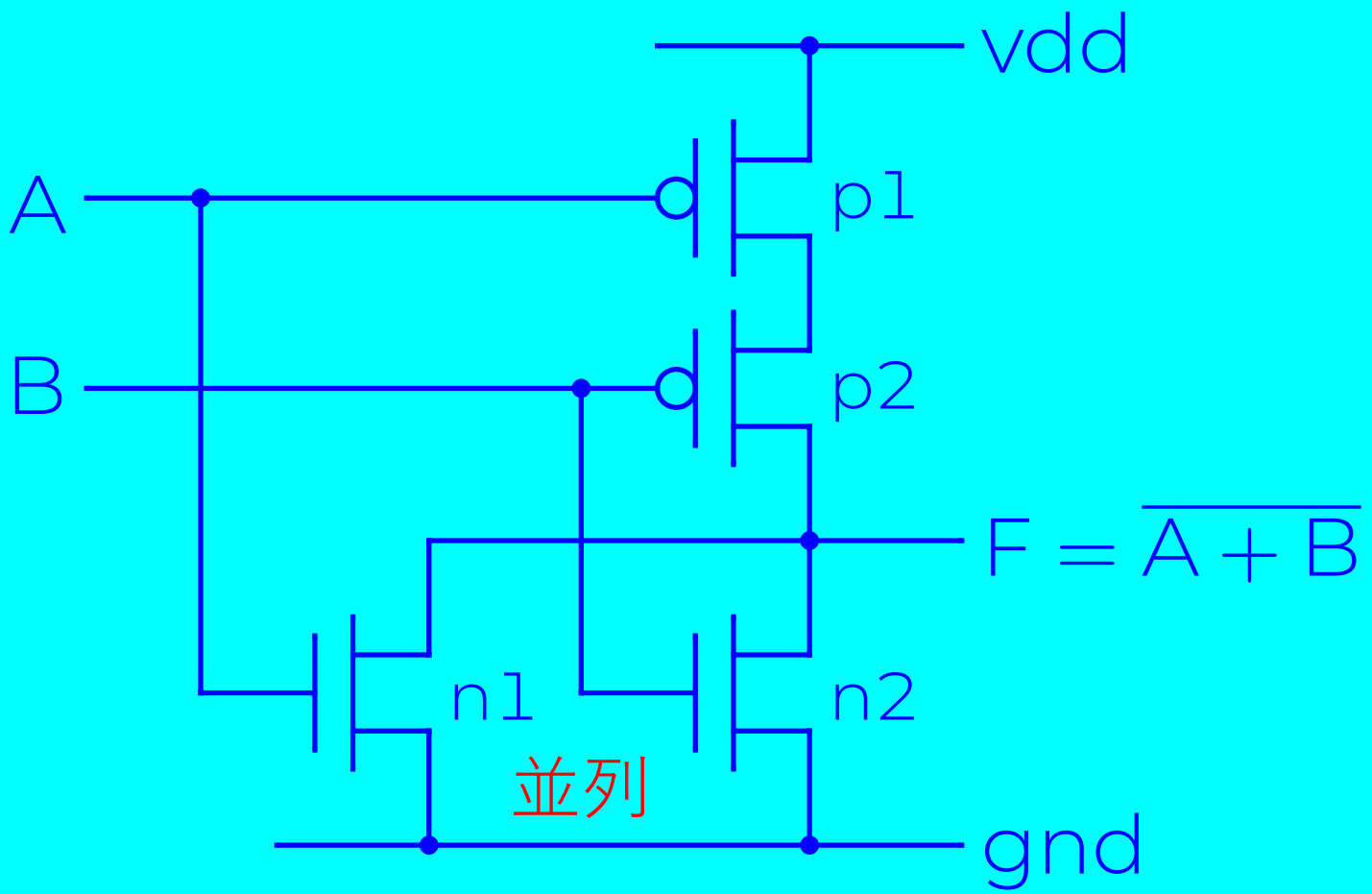
NOR ゲートの CMOS 型実装

```
% iverilog -Wall -o cmosnor \
  cmosnor_tb.v cmosnor.v
% ./cmosnor
% gtkwave cmosnor.vcd
```

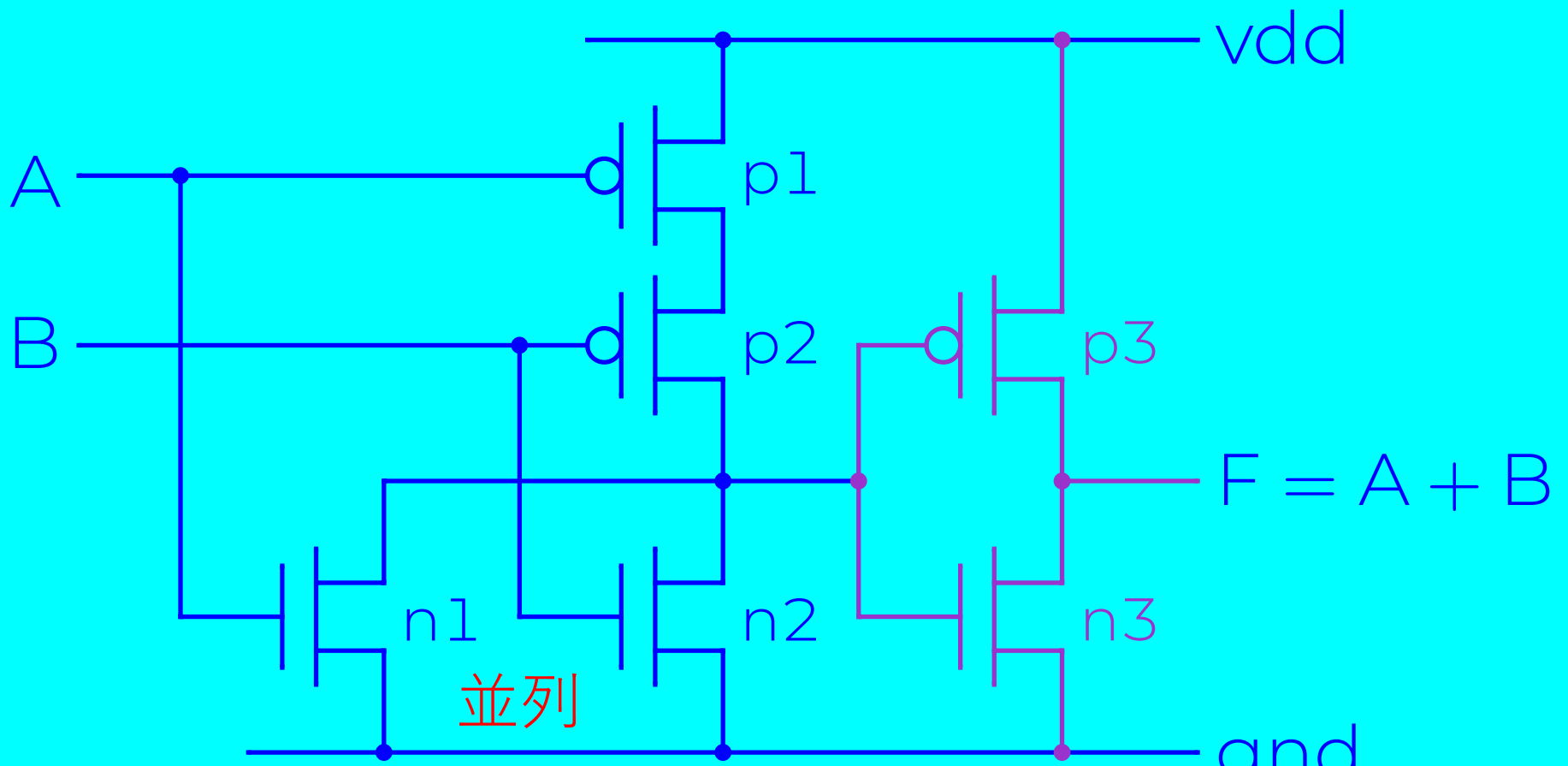


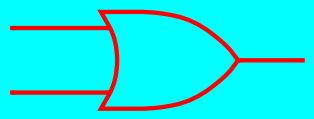
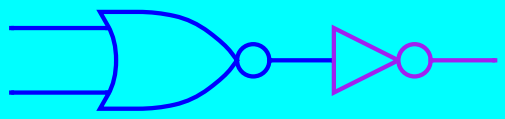
A	B	$A + B$	$F = \overline{A + B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

NOR ゲートの CMOS 型実装



OR ゲートの CMOS 型実装



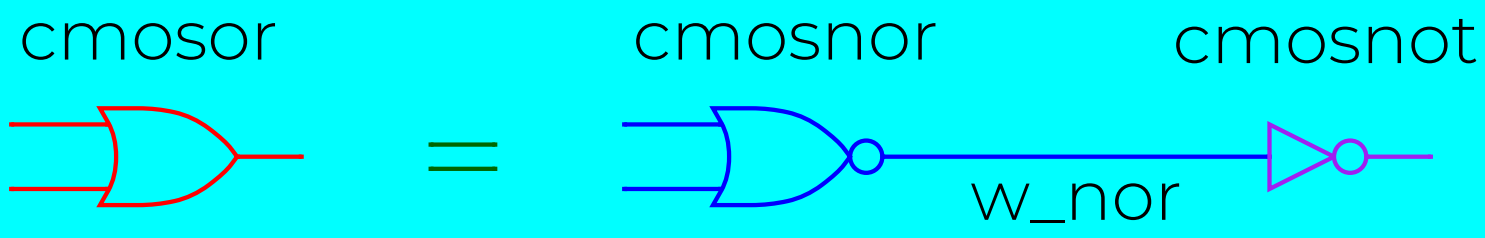
つまり、  ＝ 

Note: A PMOS is connected to the vdd and NMOS is connected to the ground.

OR ゲートの CMOS 型実装

```
‘timescale 1ns/1ns

module cmosor (f, a, b); // a circuit
    input   a, b;  // inputs
    output  f;     // output

    wire    w_nor; // internal wire

    cmosnor i0 (w_nor, a, b);
    cmosnot i1 (f, w_nor);

endmodule
```

cmosor.v



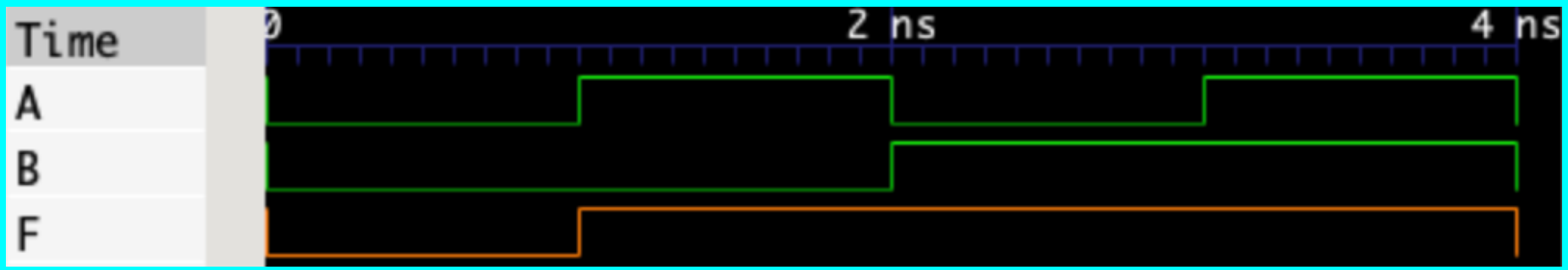
OR ゲートの CMOS 型実装

```
‘timescale 1ns/1ns
module cmosor_tb; // a test bench, not a circuit
    reg  A, B;
    wire F;
    cmosor i0 (F, A, B); // invoke cmosor
    initial begin
        A = 0;
        B = 0;
        #4 $finish;
    end
    always #1 A = ~A;
    always #2 B = ~B;
    initial begin
        $dumpfile ("cmosor.vcd");
        $dumpvars;
    end
endmodule
```

cmosor_tb.v

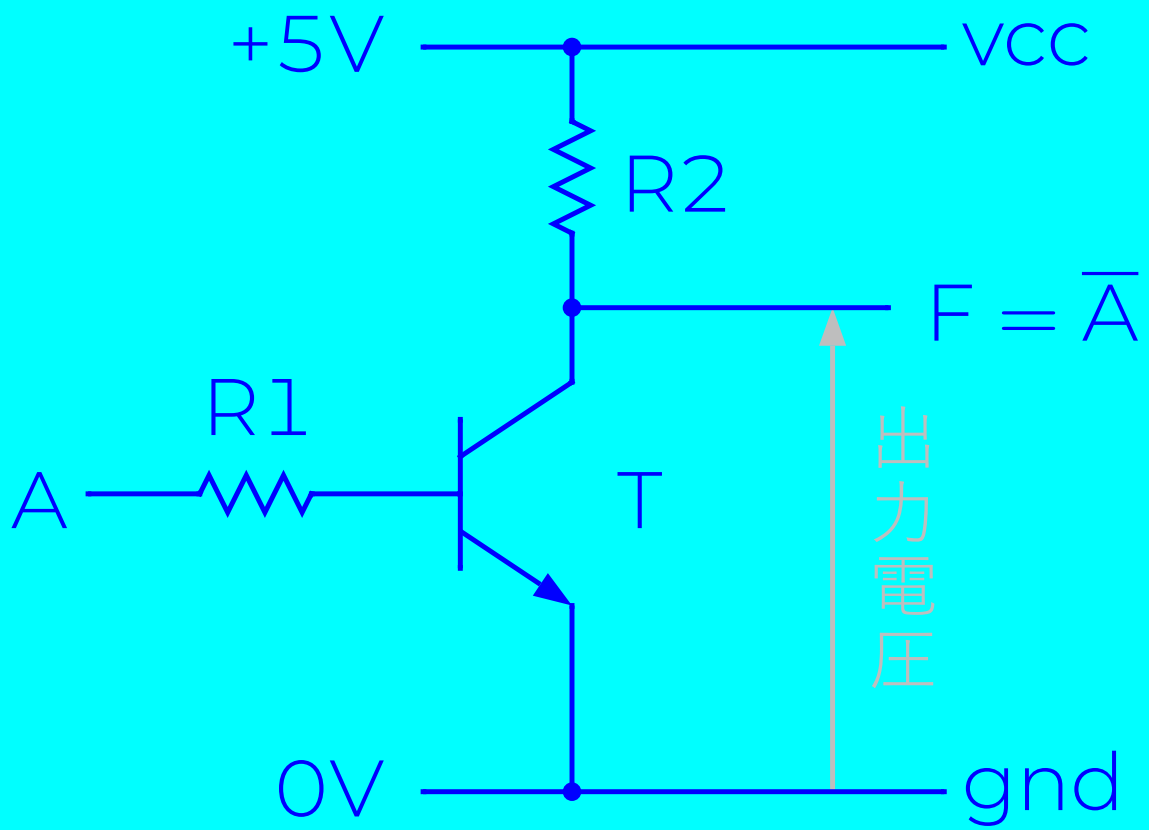
OR ゲートの CMOS 型実装

```
% iverilog -Wall -o cmosor \
  cmosor_tb.v cmosor.v cmosnor.v cmosnot.v
% ./cmosor
% gtkwave cmosor.vcd
```



A	B	F = A + B
0	0	0
0	1	1
1	0	1
1	1	1

トランジスター (Transistor)



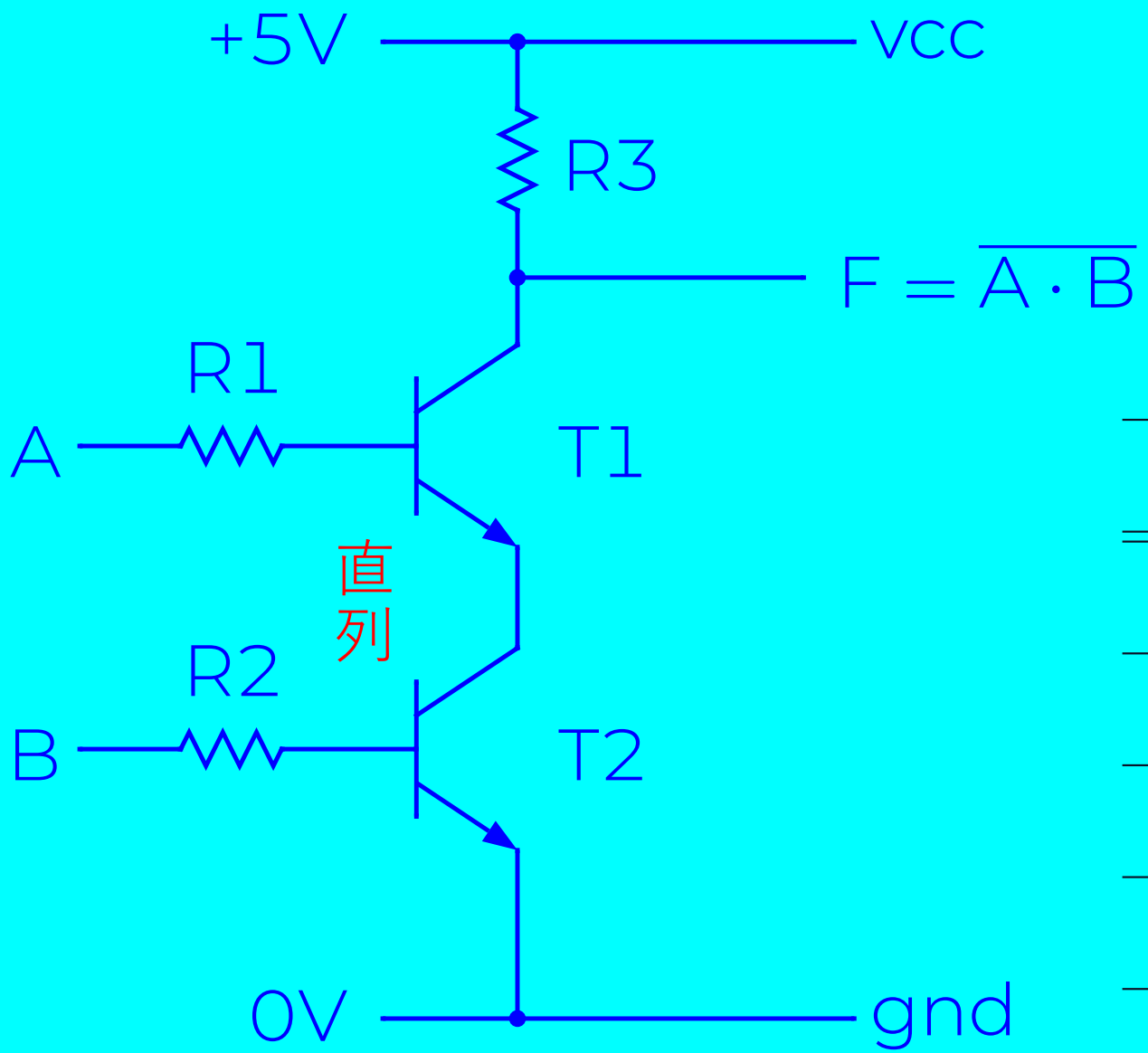
入力	出力
A	F
L	H
H	L

L: 低電位 (0)

H: 高電位 (1)

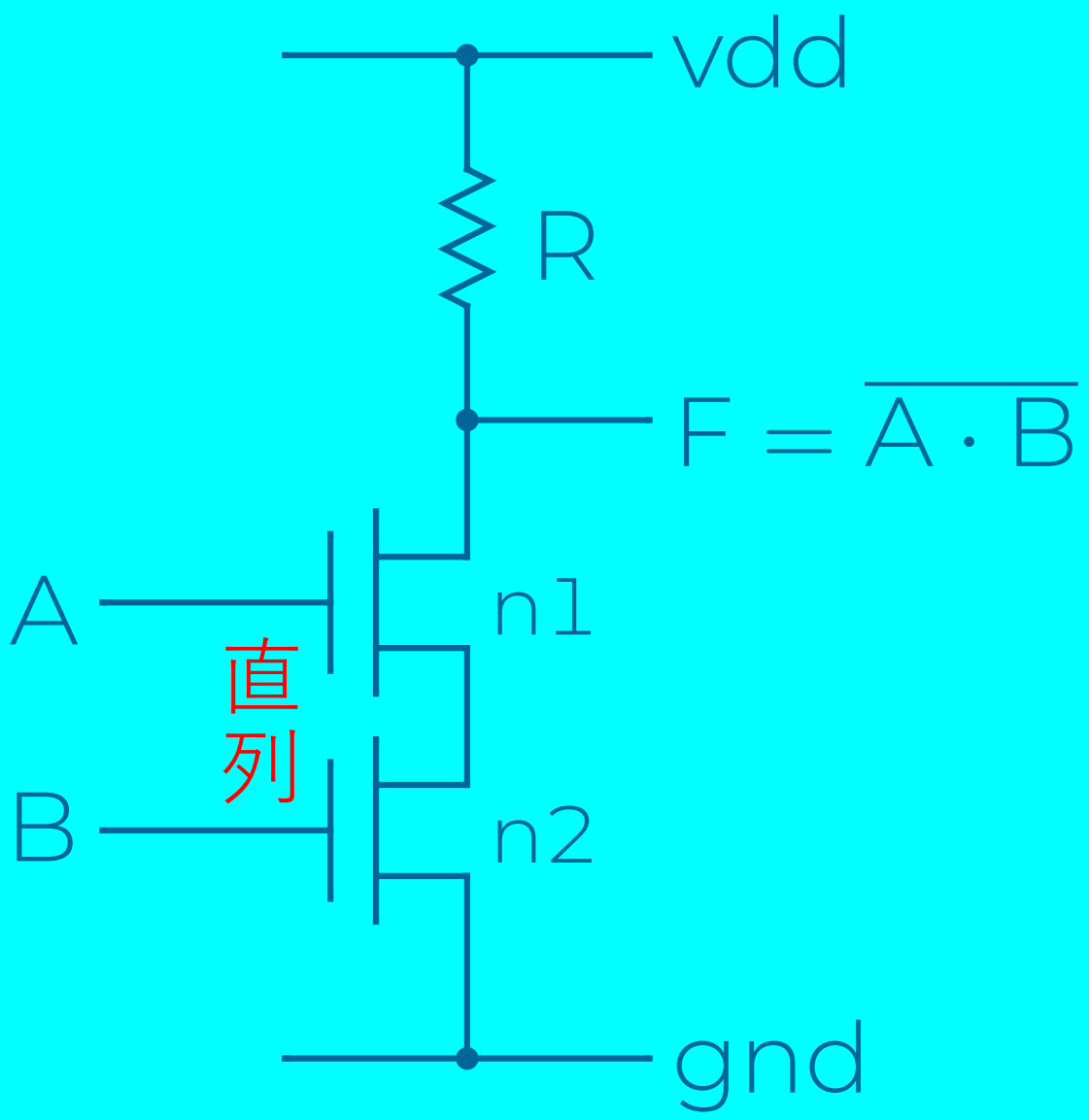
TTL — Transistor-transistor logic
(トランジスタ-トランジスタロジック)

NANDゲートのトランジスター型実装



A	B	$\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

NANDゲートの NMOS 型実装



A	B	$\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

論理ゲートのCMOS型実装

まとめ

- 基本的な論理演算と論理ゲート: AND、OR、NOT
- ほかのゲート: NAND、NOR、XOR、XNOR
- CMOSトランジスタ
- NOTゲートのCMOS型実装
- NANDゲートのCMOS型実装
- ANDゲートのCMOS型実装
- NORゲートのCMOS型実装
- ORゲートのCMOS型実装
- NANDゲートのトランジスター型実装
- NANDゲートのNMOS型実装

課題 II (100 点)

以下の論理式を CMOS 回路として合成して下さい。

(1) $F = \overline{A \cdot B \cdot C}$ (P29 - P37 の $F = \overline{A \cdot B}$ を参照)

モジュール名：cmosnand3

テストベンチ：cmosnand3_tb.v

(2) $F = \overline{A + B + C}$ (P43 - P51 の $F = \overline{A + B}$ を参照)

モジュール名：cmosnor3

テストベンチ：cmosnor3_tb.v

1. CMOS 回路の回路図を書け
2. CMOS 回路の Verilog HDL コードを書け
3. CMOS 回路の真理値表を書け
4. CMOS 回路をシミュレーションせよ
5. シミュレーション波形について説明せよ

発展：自由練習

下図の CMOS 回路の真理値表と F の論理式を書いて、さらに回路を実装しシミュレーションしてください。