2011年4月19日作成
2017年6月7日更新

マテリアルデザイン
Materials Design

名古屋工業大学 先進セラミックス研究センター
井田　隆

はじめに
Preface

材料設計 materials design の本来の意味は，どのような材料を組み合わせるか，どのようなプロセス（温度や時間，混合のしかた，撹拌のしかた）で目的の材料を作製するか，材料をつくるための原料をどのように配合するかなどの計画を立てることです。目的の性能・品質の材料を作るために，なるべく安い原料・設備・エネルギーで，要求される納期のうちに必要な量の製品を生産することが目標になります。

材料設計は，経済合理性とも省エネルギー，環境負荷の軽減とも直結しています。最近では，材料設計の段階で，既に将来製品を廃棄するためのコストが意識される場合もあります。しかし，このような意味での材料設計の具体的な方法は，対象となる製品，社会状況や経済状況によっても変わりますから，大学での授業としてふさわしい内容にすることは困難です。

この講義では材料設計のためにコンピュータ computer を利用する技術を学びます。特に，いわゆるコンピュータシミュレーション（模擬実験）に内容を限定します。本当の実験の一部でもシミュレーションで代用することができれば，コストが安くついて，エネルギーの節約になり，安全性を確保あるいは危険性を回避する事ができて，廃棄物の心配もせずに済みようになります。

はじめの４回の授業ではコンピュータの基礎（論理演算，記憶，算術演算，数値計算）について勉強します（理系の大学生なら最低限知っておいてほしいレベルのことです）。それ以降はコンピュータを利用した材料（物質）シミュレーションを利用するために必要になる基礎的なことがらを勉強します。市販の分子動力学計算ソフトウェアを使うことができるような知識（何が計算できるのか，どう解釈したら良いか），あるいは（自分が直接使わないとしても）使うことに価値があるかを判断できるような知識，結果の妥当性を判断できる知識を身につけることを目標とします。現時点では分子動力学シミュレーションのソフトウェアは高価なものですが，無料で公開されているソフトウェアもあり，これか

らは市販のソフトウェアがさらに普及し，無料ソフトウェアもより使いやすいものになることが予想されます。

この授業は環境材料工学科の学生を主な対象としています。コンピュータや情報，通信，電気，電子回路技術などに苦手意識を持つ人も少なくないかもしれませんが，「無理をしてでも」コンピュータの勉強をしておくことをおすすめします。コンピュータや情報通信技術，計算技術は急速に発展していますし，将来さらに重要になることが予測されます。どうせいつか勉強しなければいけないのですから，若いうちから勉強を始めた方が有利です。大学の学部３年は，そのような勉強をはじめるのに適した学年です。

第１部　コンピュータの基礎 Fundamentals about computer

１．コンピュータの基礎（１）–論理演算– Logical operations

現在市販されているコンピュータの主要な部分はディジタル回路で作られています。「ディジタル信号処理」と「論理演算」とは「まったく同じ」という訳ではないのですが，コンピュータの情報処理が「論理演算」に基づいているとすれば考えやすくなります。 現在使われている論理演算の体系は G. Boole という数学者がまとめたものに基づいており，ブール演算とも呼ばれます。

ブール演算は，加減乗除の四則演算に比べればずっと簡単です。 まず，扱う数は 0 と 1 の２種類しかありません。この２つの値が論理値であり，それぞれ「偽」=“false”=0,「真」= “true” = 1 と名前を付けて呼ばれます。ただし，必ずしも普通の意味で「真偽」を表している訳ではありません。 0 や 1 という数を，普通と少し違う意味で使うので，紛らわしくならないように別の名前で呼ぶようにしていると考えれば良いでしょう。

１–１　記号論理とブール代数　Symbolic logic and Boolean algebra

記号論理あるいは論理代数には，「論理を数学的に表現するための方法」という面もあるのですが，普通の代数学で「いろいろな値をとりうる数のようなもの」を x という文字で表すのと同じように，「0 か 1 の論理値をとりうる変数」を x や A などの文字（論理変数）を使って表すだけのことです。

ディジタル回路では，回路上の特定の場所の電圧や，特定の導線を流れる電流，特定のキャパシタに蓄えられる電荷などが「0 とみなせる場合」と「0 以外とみなせる場合」の 2 種類あり，この二つの状態が頻繁に入れ替わります。ディジタル回路を使って信号処理をする場合に，回路上の接点や導線，キャパシタなど「目に見えて実在するもの」は「論理変数」に対応し，むしろ「論理値」の方が「普通には目に見えず仮想的な存在」に対応します。

高校までに習う算数・数学では「数値」に具体的なイメージが結びつき，「変数」は抽象的な概念だったかもしれませんが，電子回路を使ったディジタル回路は，それとはかなり様子が違います。

さて，ブール代数で使われる演算は NOT 演算，AND 演算，OR 演算の 3 種類だけです。

(1) NOT 演算（$\overline{X}$）の規則

$$\overline{0}=1$$

$$\overline{1}=0$$

(2) AND 演算（$A \wedge B$）の規則

$$0 \wedge 0 = 0$$

$$0 \wedge 1 = 0$$

$$1 \wedge 0 = 0$$

$$1 \wedge 1 = 1$$

(3) OR 演算（$A \vee B$）の規則

$$0 \vee 0 = 0$$

$$0 \vee 1 = 1$$

$$1 \vee 0 = 1$$

$$1 \vee 1 = 1$$

この $\overline{X}$, $A \wedge B$, $A \vee B$ という演算は，上の規則以外は $+, -, \times, \div$ などの計算とまったく同じように扱えるとします。これらの演算を「論理演算」といいます。

論理演算の演算規則を表現するなどの目的で「論理値表」と使うと便利な場合があります。表 1.1 に NOT, AND, OR 演算の論理値表を示します。

表 1.1　否定 NOT・論理積 AND・論理和 OR 演算の論理値表

X	$\overline{X}$
0	1
1	0

A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

「AND 演算」$A \wedge B$ は，0 と 1 の値にしか使わないことを除けば，「かけ算」 $A \times B$ とまったく同じことです。「$0 \times 0 = 0$」，「$0 \times 1 = 0$」，「$1 \times 0 = 0$」，「$1 \times 1 = 1$」ですね。そこで， AND 演算のこと，あるいは AND 演算の結果のことを，論理的なかけ算という意味で「論理積」という言葉で表します。

「OR 演算」$A \vee B$ は「たし算」 $A + B$ と似ています。$1+1=2$ に対して $1 \vee 1 = 1$ というところが違っていますが，論理的な足し算という意味で「論理和」という言葉が使われます。

任意の論理値（つまり 0 か 1 の値）を取る変数を論理変数と呼びます。たとえば，「今日が休講だということが正しいかどうか」ということを A という記号で表して，「正しい」ということを「A は 1 という値を取る」と言い換えて，「正しくない」ということを「A は 0 という値を取る」言い換えることにします。

この場合，今日が休講なら $A = 1$，休講でなければ $A = 0$ と表現することができます。また，「今日は休講だ」という記述を否定すれば「今日は休講でない」ということになりますから，「今日は休講でないということが正しいかどうか」ということが $\overline{A}$ と表されて，もし今日が休講なら $A = 1$ だから $\overline{A} = 0$，休講でなければ $A = 0$ だから $\overline{A} = 1$ になるということは 1 番目の「 NOT の規則」から確かめることができるでしょう。

今度は，「来週は試験だということが正しいかどうか」ということを B という記号で表せば，「今日は休講でなおかつ来週は試験だということが正しいかどうか」ということが $A \wedge B$ と表されるということになり，また，「今日が休講であるか来週が試験であるかのどちらかだということが正しいかどうか」ということが $A \vee B$ と表されることになります。

簡単なことをわざと難しくしているように思えるかもしれませんが，実際に使う論理演算は，（論理と言う名前はついていますが）必ずしも普通の意味での「論理」と対応づけられなければいけないという必然性はありません。

必要なことは，普通の（ $+, -, \times, \div$ などの）計算の規則ではなく，上にあげた「論理演算の規則」に従うということです。もし頭が混乱してきたら，論理演算の規則に戻って考えるようにしてください。

1－2　ド・モルガンの法則　De Morgan's law

「法則」というほどおおげさなことではないと思いますが,

$$\overline{A \wedge B} = \overline{A} \vee \overline{B}$$

$$\overline{A \vee B} = \overline{A} \wedge \overline{B}$$

という二つの関係のことをド・モルガンの法則と言います。「論理積の否定は否定の論理和と等しく，論理和の否定は否定の論理積に等しい」という関係です。

前節の「論理演算の規則」を使えば簡単に導くことができますが，実際に論理演算の中で良く出てくるのでド・モルガンの法則と呼んで使うことにします。

ド・モルガンの法則から，以下の関係も導かれます。

$$A \wedge B = \overline{\overline{A} \vee \overline{B}}$$

$$A \vee B = \overline{\overline{A} \wedge \overline{B}}$$

これらの関係も，前節の「論理演算の規則」を使えば簡単に導くことができます。規則から $\overline{0} = 0, \overline{1} = 1$ だから，一般的に $\overline{\overline{A}} = A$ と書けることを確認しておいてください。

この法則に何か意味を持たせるためには，たとえば次のような記述を想像してみたら

良いでしょう。「Q 君がタワケでもないしアホでもないということはない」この記述は，「Q 君はタワケであるかアホであるかのどちらかである」と同じことを言おうとしています。「A ならば B である」という命題（論理包含）が否定と論理和の組み合わせ「(A でない) または B である」と等価だということは理解しにくいところですが，上の記述が「Q 君はタワケでなければアホである」とも言い換えられることに対応します。

さて，どうして当たり前のことにわざわざ「ド・モルガンの法則」と名前を付けて呼ぶことが多いのでしょうか？

これはコンピュータや周辺回路をデザインしたり作る人にとって，（あたりまえなことだとしても）この言葉の示す内容が非常に便利で役に立ち，頻繁に用いられるからです。現実の論理演算は電子回路を使って実現されています。ド・モルガンの法則の意味する内容を利用すれば，必要な機能を実現するための素子の種類や数，配線の本数を減らすことができて，回路の動作を速くしたり，製造コストを下げたり，信頼性を向上させたり，資源やエネルギーの消費を抑えられるからなのです。

1－3　電子回路による論理演算　Logical operation with electronic circuits

この節では，電子回路によって論理演算を実現するための具体的な方法について説明します。

論理演算には NOT と AND，OR の三種類があるのですが，ド・モルガンの法則を使えば，すべての論理演算を NOT 演算と AND 演算だけの二種類の組み合わせ，あるいは NOT 演算と OR 演算だけの二種類の組み合わせでも表現できることがわかります。

実際には NAND 演算 = NOT (*A* AND *B*) という演算や NOR 演算 = NOT (*A* OR *B*) という演算を実現する回路が良く使われます。この節でも，はじめに NOT 演算を実現する回路，つぎに NAND 演算を実現する回路について説明します。

論理値 1 を +5 V，論理値 0 を 0 V という電圧で表現することにします。この +5 V という電圧を選んだのは，トランジスタを使う時にはこのくらいの電圧を選んでおくのが都合が良いからで，実際に少し前までは多くの論理回路に +5 V という電圧が使われていました。ただし，最近のコンピュータでは発熱を抑えるために +3.3 V とか +3 V という値が使われるようになってきています。

1－3－1　否定回路　NOT circuit

NOT 演算を電子回路で実現するためには，「入力が 0 V なら +5 V を出力し，入力が +5 V なら 0 V を出力する」回路を作ればよいことになります。そのような回路のことを NOT（ノット）回路と呼びます。NOT 回路を作る方法は一通りではなく，いろいろな作り方があります。例えばリレー（電磁石でスイッチを開閉する素子）や真空管，トランジスタと抵抗器の組み合わせでも NOT 回路を作ることができます。しかし実際に VLSI（超大規模集積回路，very large-scale integration, 数十万個以上の素子が搭載されている IC チップ）で使われている NOT 回路ではCMOS（シーモス） という素子が使われていて，記号で表現すると，図 1.1 のようになります。

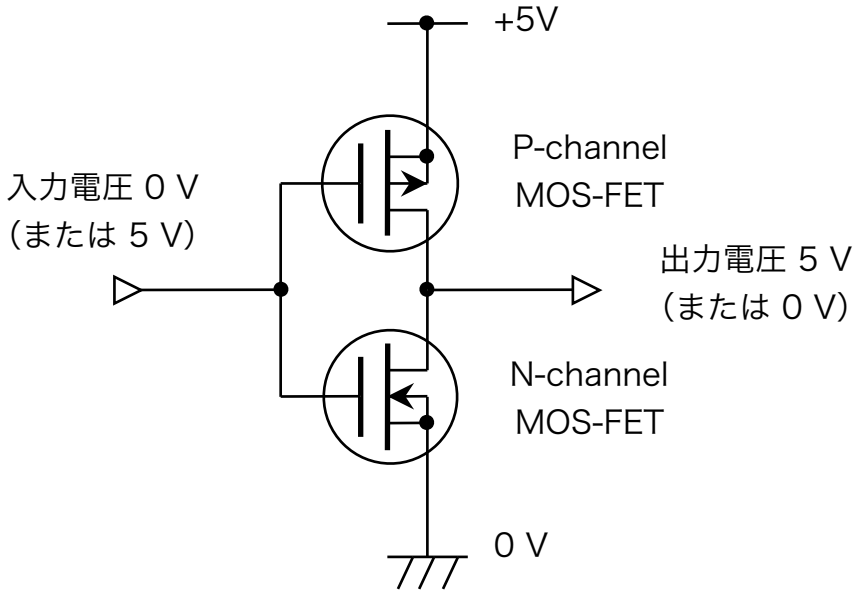
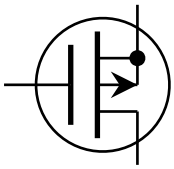
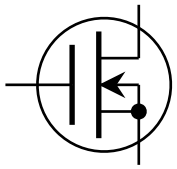


図 1.1　CMOS による NOT 回路

図 1.1 の中で 

の記号は，+5 V の電源（のプラス側の端子）に接続することを表していて，また 

の記号はアース（地球）earth とかグランド（地面）ground と呼ばれるもので，+5 V の電源のマイナス側の端子に接続することを意味しています。図 1.1 中で  と  の記号は MOS-FET（モスフェット） (金属酸化物半導体電界効果型トランジスタ metal-oxide semiconductor field-effect transistor) を表しています（図 1.2）。

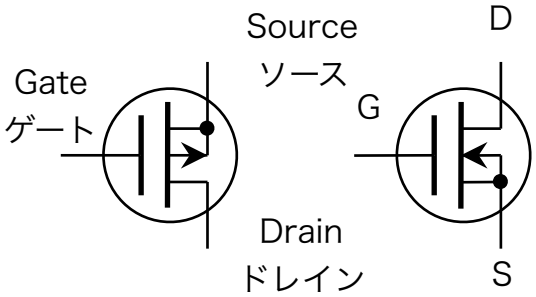


図 1.2　MOS-FET の回路記号。P-channel MOS-FET = PMOS（左）と N-channel MOS-FET = NMOS（右）

MOS-FET には P-チャンネルと N-チャンネルの２種類があり，それぞれ PMOS, NMOS と呼ばれます。一つの基板上に両方のタイプのトランジスタを作り込んだものを相補型金属酸化膜半導体 CMOS (complementary MOS)（シーモス）と呼び，図 1.1 の NOT 回路は CMOS を使った構成を示しています。実際の MOS-FET の構造は図 1.3 のようになっています。

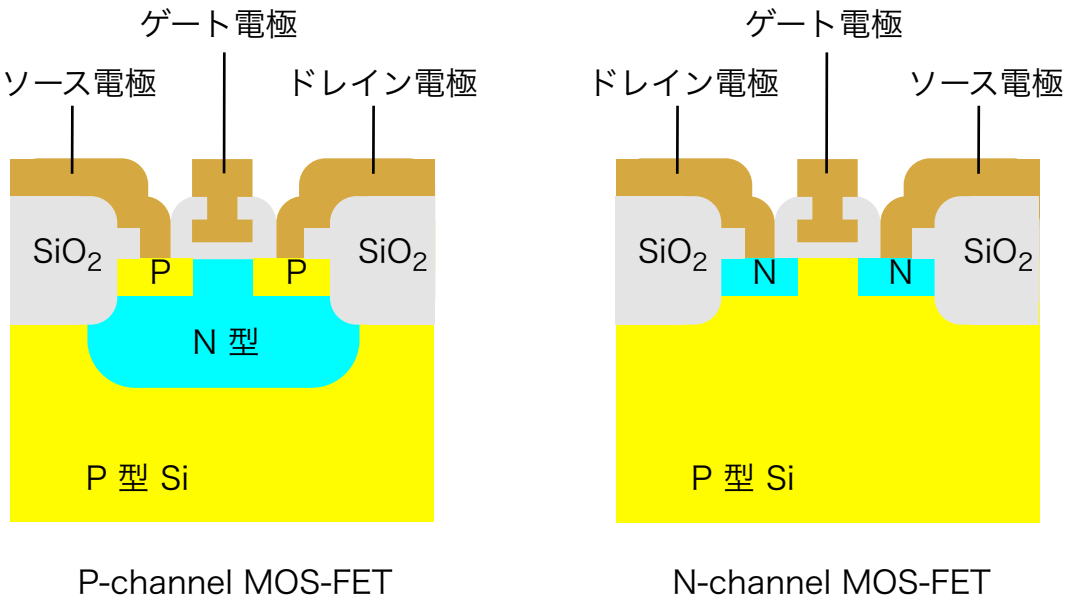


図 1.3　MOS-FET の構造。P 型 Si 基板上に作製される場合。

MOS-FET は金属電極と半導体基板の間を薄い酸化物層で遮った構造を持ちます。FET から出ている 3 本の線にはそれぞれ名前がついていて，ソース source (水源)，ドレイン drain (排水管)，ゲート gate (水門) と呼ばれます。

FET は，ソースからドレインへの荷電担体（キャリヤ carrier）の移動を，ゲートにかける電圧で on/off するスイッチの働きをもっています。PMOS では荷電担体が正孔なのでソースからドレインに電流が流れますが，NMOS では荷電担体が電子なのでドレインからソースに電流が流れます。図 1.1 の NOT 回路では入力電圧が 0 V のとき PMOS が on，NMOS が off になるので出力電圧は +5 V となり，入力電圧が +5 V のとき PMOS は off，NMOS が on になるので出力電圧は 0 V となります。

FET のはたらきを，図 1.4 のようにスイッチに置き換えれば理解しやすいでしょう。たとえば，入力電圧が 0 V のとき，PMOS のソースとゲートの間に電圧がかかりますから，PMOSのソースとゲートの間に電流が流れることができるようになりますが，NMOSのゲートとソースの間には電圧がかからないので，NMOS のソースとゲートの間には電流が流れられません。出力端子から見ると 5 V 端子との間のスイッチが閉じて，0 V 端子との間のスイッチが開くのと同じことになるので出力が 5 V となります。入力電圧が 5 V のときには逆に PMOS が絶縁，NMOS が導通状態になるので，出力側から見ると 5 V との間に挿入されているスイッチが off，0 V との間に挿入されているスイッチが on になるので，出力電圧は 0 V になります。

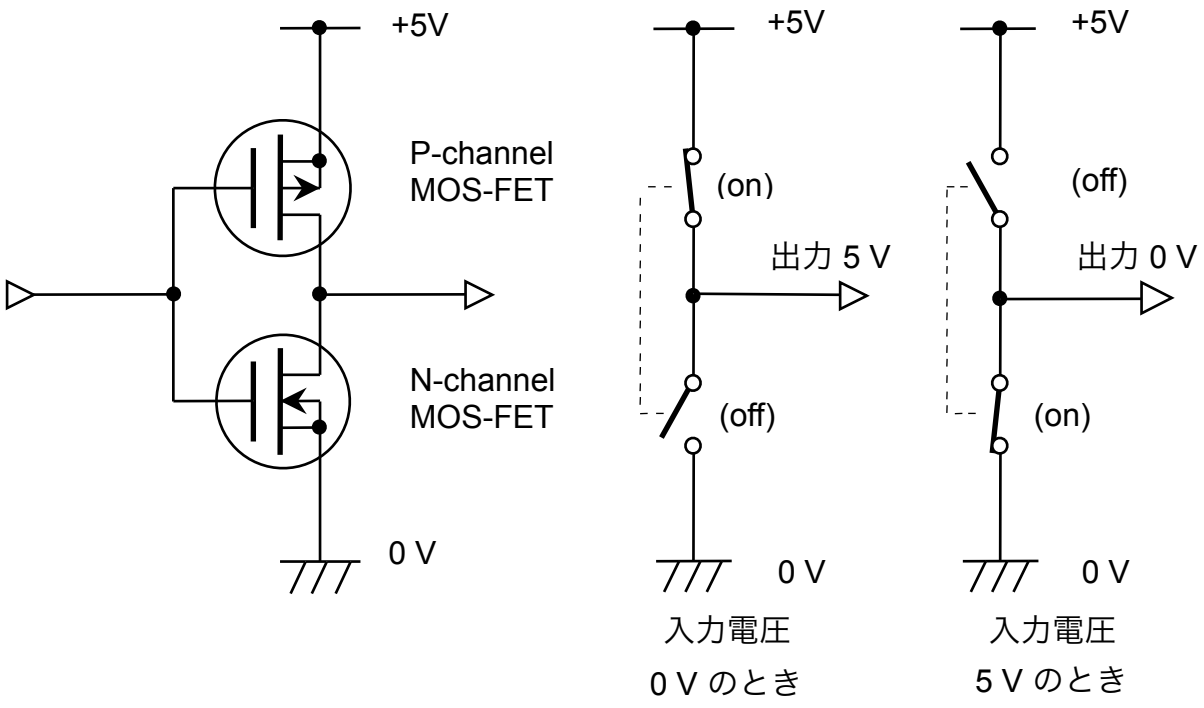


図 1.4　NOT 回路の MOS-FET をスイッチに置き換えて考える

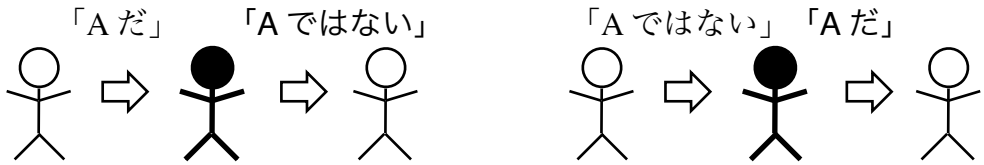


図 1.5　NOT 回路のはたらき。つねに聞いたことと反対のことを言う人と同じこと。

NOT 回路のはたらきは，図 1.5 のように，「つねに聞いたことと反対のことを言う人」と同じことです。

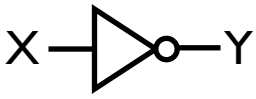


図 1.6　NOT 素子（ゲート）の MIL 記号

一つの NOT 回路ごとに図 1.1 のような回路図を描くと複雑になりすぎるので，実際の論理回路では，NOT 回路をひとまとまりとして，図 1.6 のような記号であらわします。現在一般的に使われている記号は MIL（ミル） (Military) 記号と呼ばれるものです。実際の回路では NOT 素子は電源の +5 V と 0 V の端子に線でつながないと働かないのですが，MIL 記号を使って論理回路を描くときには電源の +5 V と 0 V に接続する線は省略します。

1－3－2　否定論理積 NAND 回路　NAND circuit

二つの入力 A, B に対して，"not (A and B)"， $Y = \overline{A \wedge B}$ を出力する回路を NAND（ナンド）回路と呼びます。図 1.7 のように PMOS を並列に，NMOS を直列に接続すれば NAND 回路が実現されます。

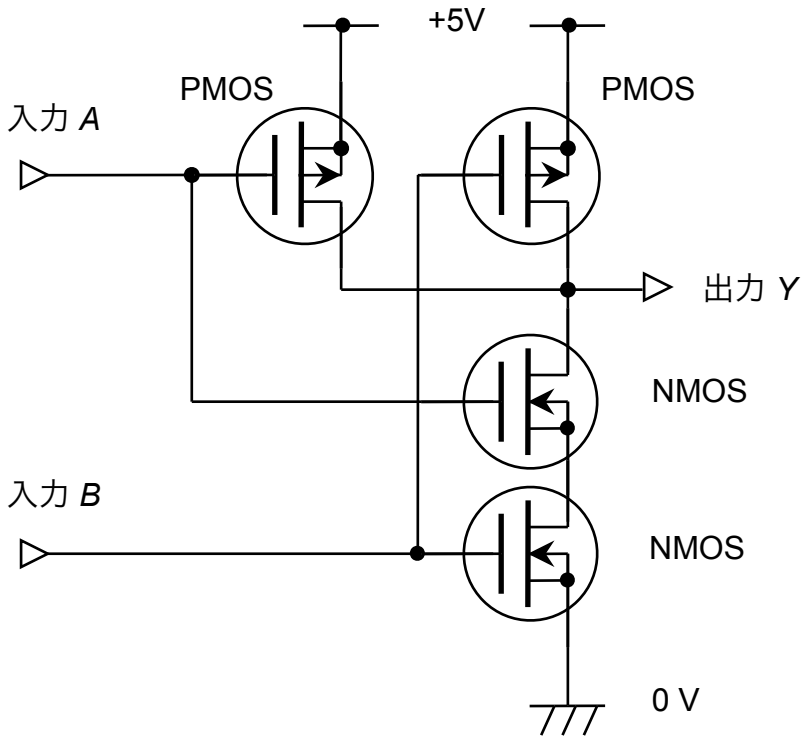


図 1.7　MOS-FET による NAND 回路

NAND 回路の入力が両方とも 0 V のときは，２つの PMOS が両方導通，２つの NMOS が両方絶縁状態になるので，出力は 5 V となります。入力のうち片方が 0 V で片方が 5 V のとき，PMOS も NMOS も片方が導通，片方が絶縁状態になりますが，PMOS が並列で NMOS は直列なので，出力側から見ると 5 V 端子とは導通していて，０V 端子とは絶縁している状態になり，やはり出力は 5 V です。入力が両方とも 5 V のときだけ出力が 0 V になります。0 V → 0，5 V → 1 として，表で示せば以下のような関係があります。

表 1.2　否定論理積 NAND 演算の論理値表

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

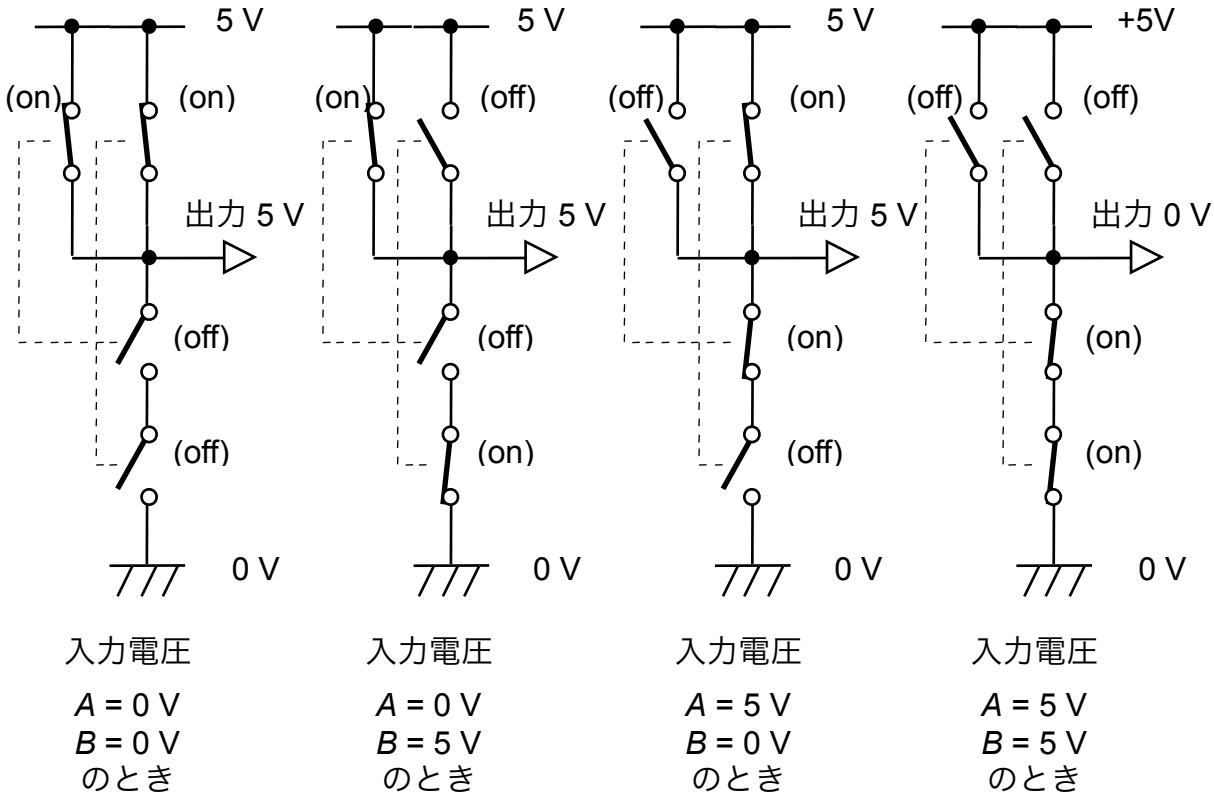


図 1.8　NAND 回路のMOS-FET をスイッチにおきかえて考える

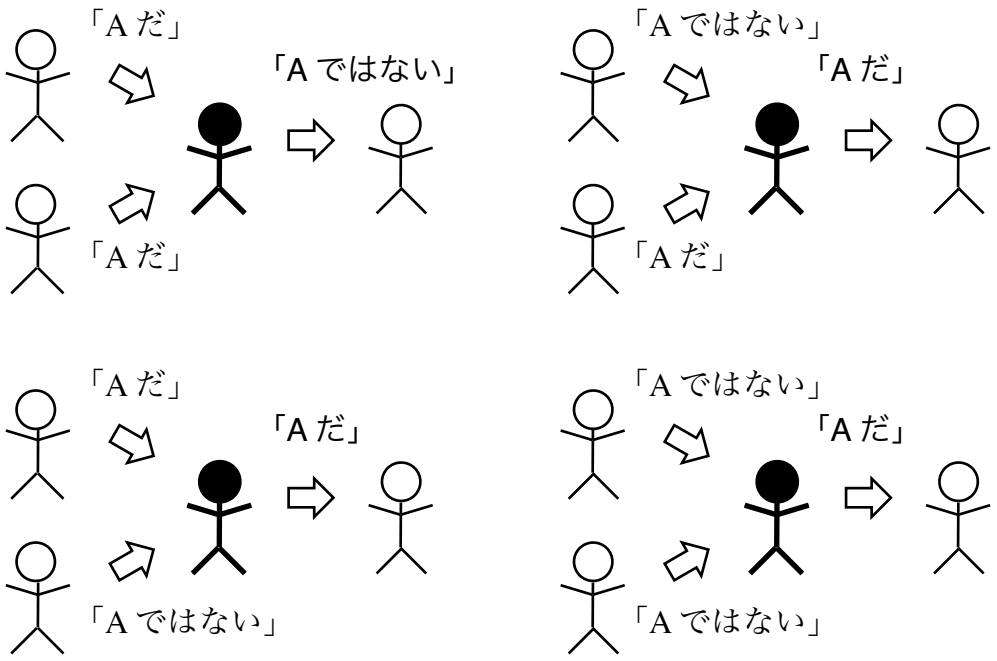


図 1.9　NAND 回路のはたらき

論理回路で NAND 演算を実現するための素子（NAND ゲート）は，図 1.10 のような MIL 記号であらわされます。

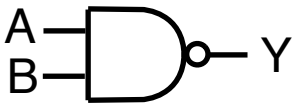


図 1.10　NAND ゲートの MIL 記号

1－3－3　論理積 AND 回路　AND circuit

NAND ゲートの出力に NOT ゲートを接続すれば，AND 回路が得られます（図 1.11, 1.12）。

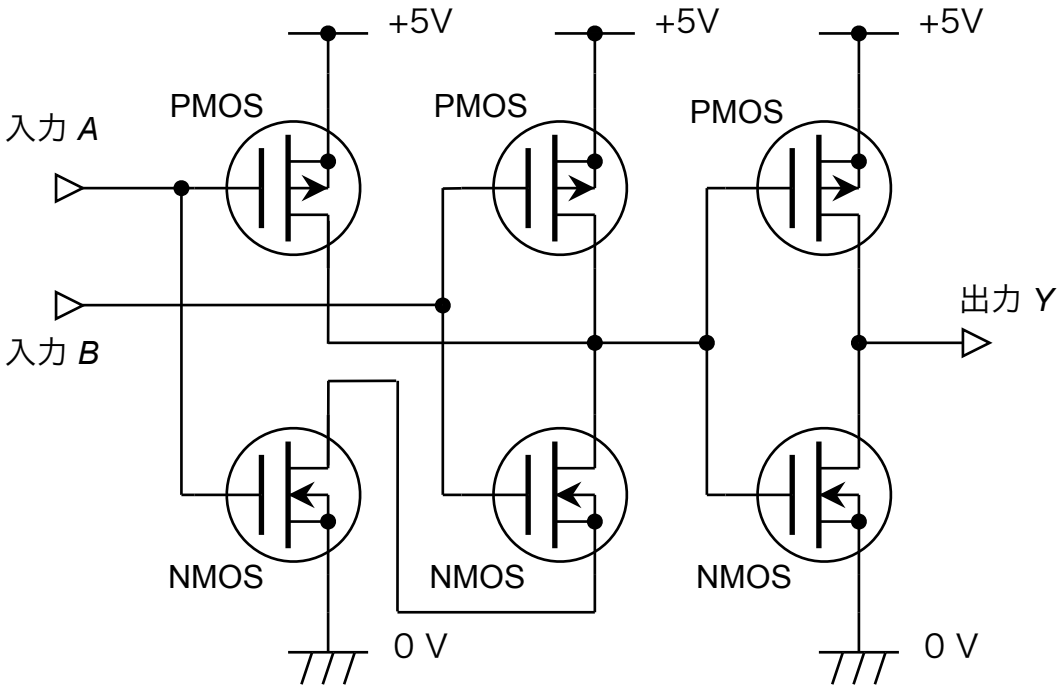


図 1.11 MOS-FET による AND 回路

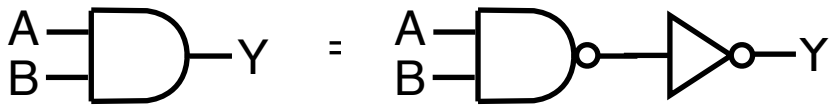


図 1.12　AND ゲートの MIL 記号（左）。

NAND ゲートとNOT ゲートの組み合わせで実現される（右）。

1－3－4　否定論理和 NOR 回路と論理和 OR 回路　NOR circuit & OR circuit

NAND 回路と逆に，PMOS を直列に，NMOS を並列に接続すれば NOR（ノア）回路が得られます（図1.13）。またこれを NOT ゲートと接続すれば OR（オア）回路になります（図1.14)。NOR ゲートと OR ゲートの MIL 記号を図 1.15 に示します。

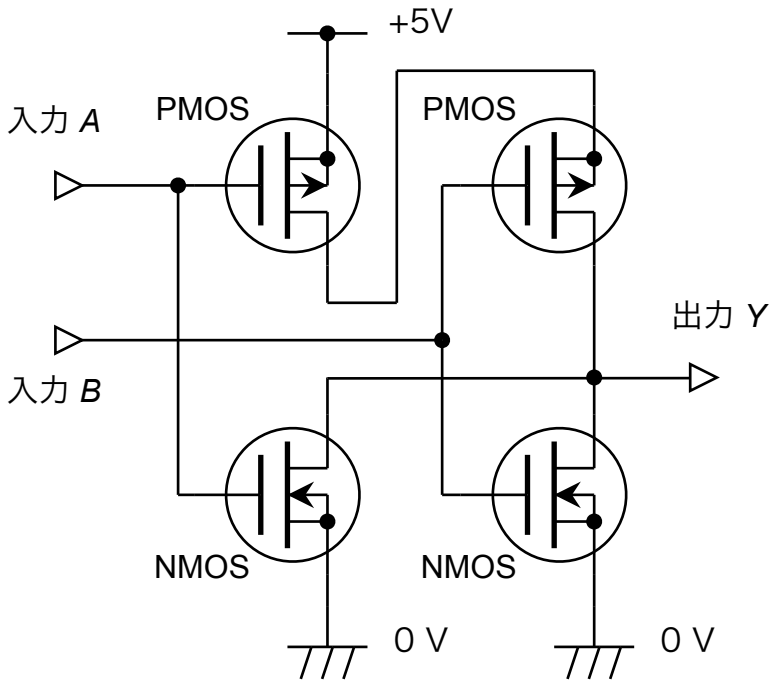


図 1.13　MOS-FET による NOR 回路

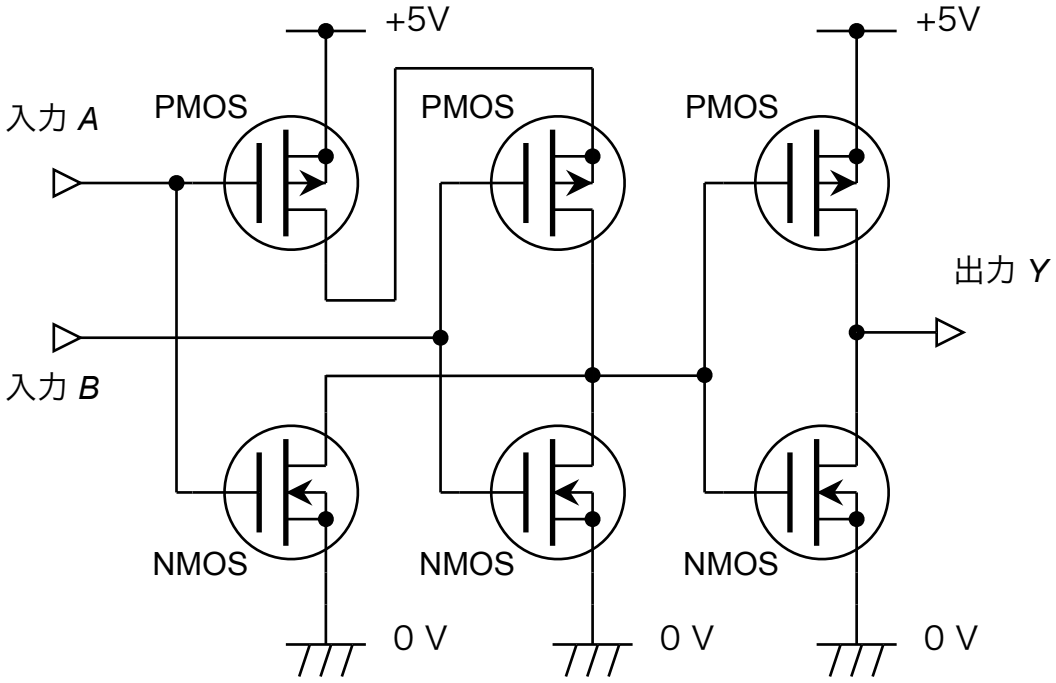


図 1.14　MOS-FET による OR 回路

図 1.15　NOR ゲート（左）と OR ゲート（右）の MIL 記号

1－3－5　排他的論理和 XOR 回路　Exclusive OR circuit

二つの入力 A, B が食い違っているときにだけ 1 を出力する演算

$$Y = (A \vee B) \wedge \overline{A \wedge B} = \left(A \wedge \overline{B}\right) \vee \left(\overline{A} \wedge B\right)$$

を排他的論理和，exclusive OR（エクスクルーシブ・オア），XOR（エックス・オア），EOR（イー・オア）と呼び，意外に良く使われます。そのうちの一例は第 3 章で扱う「算術演算」です。MOS-FET では図 1.16 のような回路で排他的論理和が実現されます。XOR ゲートの MIL 記号を図 1.17 に示します。

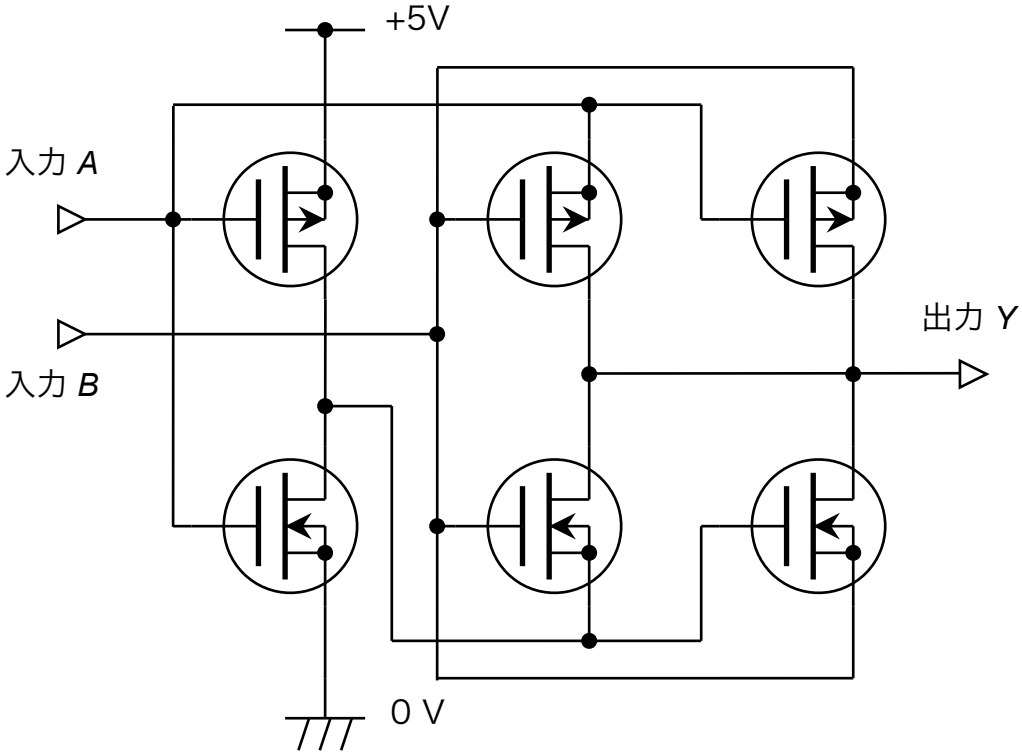


図 1.16　MOS-FET による XOR 回路

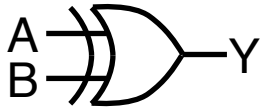


図 1.17　XOR ゲートの MIL 記号

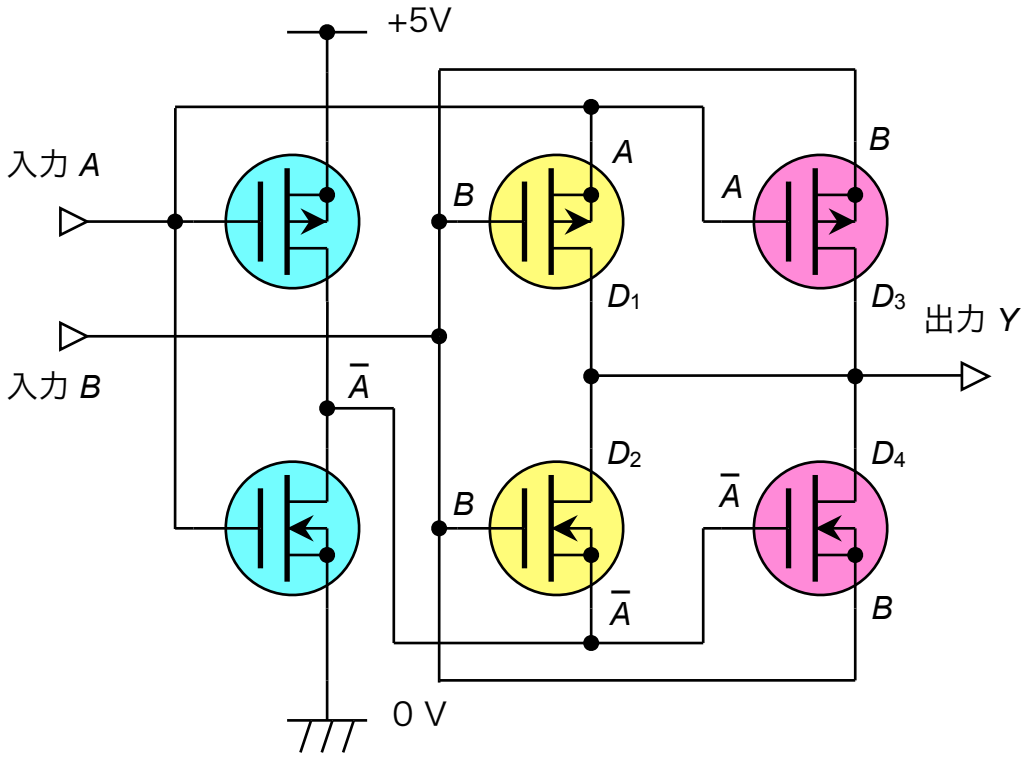


図 1.18　XOR 回路のしくみを説明する図。

水色，黄色，ピンクの 3 組の CMOS で構成されている。

図 1.18 には，３組の CMOS を水色，黄色，ピンクに色分けして示します。図 1.18 中，水色の CMOS は NOT 回路（1 – 3 – 1 節）であり，入力 A がこの NOT 回路の入力として接続されていることを確認してください。この回路の出力は $\overline{A}$ と表されます。

残りの２組の CMOS の出力が並列に接続されたものが XOR 回路全体の出力 Y となっていますが，さらに PMOS（上段）と NMOS（下段）とに分けて一つ一つの MOS-FET の動作を確かめます。

図 1.18 中，黄色の PMOS (上) と NMOS (下) のゲートはいずれも入力 B に接続されていますが，PMOS のソースは A に，NMOS のソースは $\overline{A}$ に接続されています。PMOS はゲート電圧 (B) がソース電圧 (A) より低い場合 ($B < A$) だけソース (A) とドレイン (D_1) の間が導通状態になるので，“$A = 1, B = 0$” の時にはドレイン電圧が $D_1 = A = 1$ と確定します。それ

以外の場合にはソースとドレインの間は絶縁状態になるので，ドレイン電圧 D_1 は確定しません。表 1.4 の D_1 の値を確認してください。一方，NMOS はゲート電圧 (B) がソース電圧 ($\overline{A}$) より高い場合 ($\overline{A} < B$) だけソース ($\overline{A}$) とドレイン (D_2) の間が導通状態になるので，“$A = 1, B = 1$” の時にドレイン電圧が $D_2 = \overline{A} = 0$ と確定し，それ以外の場合 D_2 は不定です。表 1.4 の D_2 の値を確認してください。

図 1.18 中，ピンクの PMOS (上) のゲートは A に，NMOS (下) のゲートは $\overline{A}$ に接続され，ソースはいずれも入力 B に接続されています。PMOS はゲート電圧 (A) がソース電圧 (B) より低い場合 ($A < B$) だけソース (B) とドレイン (D_3) の間が導通状態になるので，“$A = 0, B = 1$” の時にはドレイン電圧が $D_3 = A = 0$ と確定し，それ以外の場合には D_3 は不定です。表 1.4 の D_3 の値を確認してください。最後に，ピンクの NMOS はゲート電圧 ($\overline{A}$) がソース電圧 (B) より高い場合 ($B < \overline{A}$) だけソース (B) とドレイン (D_4) の間が導通状態になるので，“$A = 1, B = 0$” の時にドレイン電圧が $D_4 = B = 1$ と確定し，それ以外の場合 D_4 は不定です。表 1.4 の D_4 の値を確認してください。

XOR 回路全体の出力 Y には４つのMOS-FET のドレイン端子 D_1, D_2, D_3, D_4 が並列に接続されていますが，常にそのうちの一つしか導通状態に無いので，出力が一通りに決まります。

表 1.4　XOR 回路の入力 (A, B) と中間出力 $\bar{A}, D_1, D_2, D_3, D_4$，全体の出力 ($Y$) の関係

A	B	$\overline{A}$	D_1	D_2	D_3	D_4	Y
0	0	1	不定	不定	不定	0	0
0	1	1	不定	不定	1	不定	1
1	0	0	1	不定	不定	不定	1
1	1	0	不定	0	不定	不定	0