

計算科学基礎

Computer Science

Fundamentals

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

この講義では主にコンピュータ computer を利用する技術を学びます。特に、コンピュータを使ったシミュレーション（模擬実験）のようなこと、コンピュータを使って実験データを解釈する方法や、実験データを正しく解釈するための考え方を身につけることを主な課題とします。リアルな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、コストとエネルギーの節約になり、安全性を確保し危険性を回避する事ができて、廃棄物を削減し不必要な環境負荷をかけずに済むようになる場合もあります。

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）についての計算シミュレーションや、コンピュータを利用して実験データの解釈をするために必要になる基礎的なことから勉強します。市販の分子動力学計算ソフトウェアなど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、計算結果（出力）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。現時点では市販される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存在し、近い将来に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。

この授業は生命・応用化学科の学生を主な対象とします。コンピュータや情報技術、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、無理をしてもコンピュータのことについて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も予

測されます。いつか勉強しなければいけないことなら、若い年齢から勉強を始めた方が将来有利になるでしょう。

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

コンピュータ (computer) という語が定着するようになった当初、コンピュータは機械式の計算機 (mechanical calculator) を用いる計算システムを意味するものだったらしいのですが、現在市販され利用されるコンピュータの主要な部分はデジタル電子回路 (digital electronic circuit) を組み合わせて作られています。 (補足 1.A) 「デジタル信号処理」 (digital signal processing) と「論理演算」 (logical operation) とは「まったく同じ」ではないのですが、コンピュータを使った情報処理 (information processing) 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。昔から長い間使われ続けている論理演算の体系は、英国の G. Boole という数学者が19世紀に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より簡単です。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 (真理値; 真偽値) と呼ばれ、0と1はそれぞれ「偽」= “false” = 0, 「真」= “true” = 1 と名前を付けて呼ばれます。ただし、「論理値」は普通の意味での「真偽」を表しているとは限りません。0や1という数を、普通と少し違う意味・文脈で使うので、紛らわしくならないように別の名前と呼ぶようにしているとも良いでしょう。

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」という面があります。代数学 (algebra) で「いろいろな値をとりうる数のようなもの」を x という文字で変数 variable として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) を使って表します。

デジタル回路では、回路上の特定の位置の電圧 (voltage) や、特定の導線 (lead) を流れる電流 (current), 特定のキャパシタ (capacitor) に蓄えられる電荷 (electric charge) などに「0 とみなせる場合」と「0 以外とみなせる場合」の2種類の意味を持たせ、この二つの状態を頻繁に入れ替えさせます。デジタル回路を使って信号処理をする場合に、回路の中の接点 (contact) 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、「論理値」の方が「普通には目に見えないもの」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、現実に使われているデジタル電子回路はそれとはかなり様子が違い、目に見えて実在する電気回路が「変数」の役割をし、目には直接見えない電圧や電流を「数値」に対応させます。

ブール代数で使われる演算は NOT 演算 (否定演算), AND 演算 (論理積演算), OR 演算 (論理和演算) の3種類だけです。

(1) NOT 演算 (\bar{X}) の規則

$$\bar{0} = 1$$

$$\bar{1} = 0$$

(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

$$0 \vee 0 = 0$$

$$0 \vee 1 = 1$$

$$1 \vee 0 = 1$$

$$1 \vee 1 = 1$$

\bar{X} , $A \wedge B$, $A \vee B$ のような記号で表される演算は、上の規則以外は $+$, $-$, \times , \div などの演算と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 (真理値表; truth table) を使うと便利な場合があります。[Table 1.1](#) に NOT, AND, OR 演算の論理値表を示します。

Table 1.1 否定 NOT・論理積 AND・論理和 OR 演算の論理値表

X	\bar{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 の値に対してしか使わないことを除けば、^{のぞ}「かけ算」
じょうざんマルティプリケーション
 (乗算; multiplication) $A \times B$ と同じことです。0×0=0, 1×0=0, 1×1=1と同じよ
 うに $0 \wedge 0 = 0$, $1 \wedge 0 = 0$, $1 \wedge 1 = 1$ です。AND 演算のことや AND 演算の結果のこ
 とが、論理的なかけ算あるいはその結果 (積) ^{せき} (product) ^{プロダクト} のようなことという意味で「論理
コンジャンクション
 積」(logical conjunction) と呼ばれます。

「OR 演算」 $A \vee B$ は「たし算」^{かさん} ^{アディション} (加算; addition) $A + B$ 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、論理的な足し算あるいはその結果 (和) ^{サム} (sum)
 のようなことという意味で「論理和」(logical sum) と呼ばれます。

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 ([補足 1.1.A](#)) 。

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、

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

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

と表される二つの関係のことがド・モルガンの法則 (de Morgan's law) と呼ばれます。「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。

[1-1 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、論理演算の中で多
 く出現する関係であり、ド・モルガンの法則と呼ばれ、デジタル電子回路設計をする
 きなどに実際に使います。

ド・モルガンの法則から、以下の関係も導かれます。

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

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

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 $\bar{0} = 1$, $\bar{1} = 0$ であり、一般的に $\bar{\bar{A}} = A$ のように書けることを確認しておくといひかもしれません ([補足 1.2.A](#))。

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

論理演算には NOT と AND, OR の三種類があるのですが、ド・モルガンの法則を知れば、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、接合型トランジスタを使ってデジタル回路を作る時にはこのくらいの電圧を選んでおくのが都合が良かったからです。実際に 1980 年代頃までには多くの論理回路に +5 V が使われていました。現在のコンピュータの演算処理回路 (プロセッサ) では、発熱を抑えるために +3.3 V や +3 V などが使われるようになっていますが、コンピュータの周辺回路 (USB など) には、今でも +5 V が使われます。

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; ナット) 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などで NOT 回路が作られました。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る IC チップ) で使われている NOT 回路では CMOS (シーモス) と呼ばれる素子が使われていて、記号で表現すると、[Fig. 1.3.1.1](#) のようになります。

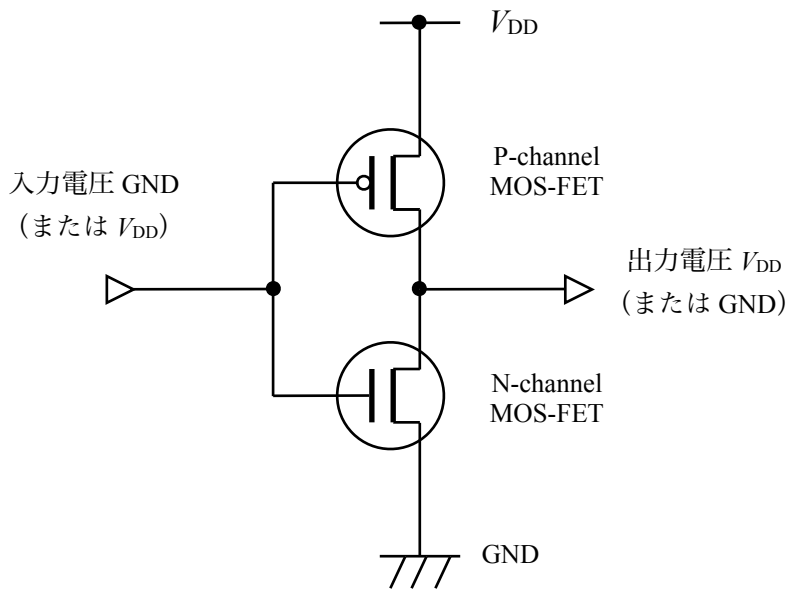


Fig. 1.3.1.1 CMOSによるNOT回路

Figure 1.3.1.1 の中で V_{DD} の記号は、例えば +5 V の直流電源のプラス側の端子に接続することを意味します。 V_{DD} の D はドレン drain (排水口) の意味ですが、ドレン電圧と区別するために DD と表記する習慣があります。(補足 1.3.1.A) GND の記号はグラウンド (地面) ground と呼ばれ、例えば +5 V の直流電源のマイナス側の端子に接続

することを意味します。 と の記号は MOS-FET (金属酸化半導体電界効果型トランジスタ metal oxide semiconductor field-effect transistor) を表します (Fig. 1.3.1.2)。

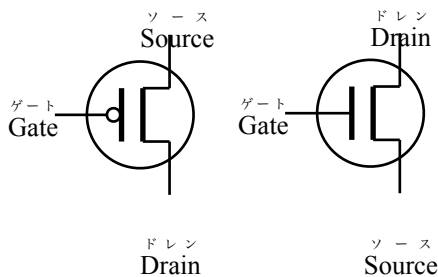


Fig. 1.3.1.2 MOS-FET の回路記号。P-channel MOS-FET = PMOS (左) と N-channel MOS-FET = NMOS (右)

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、[Fig. 1.3.1.1](#) 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[Fig. 1.3.1.3](#) のようになっています。

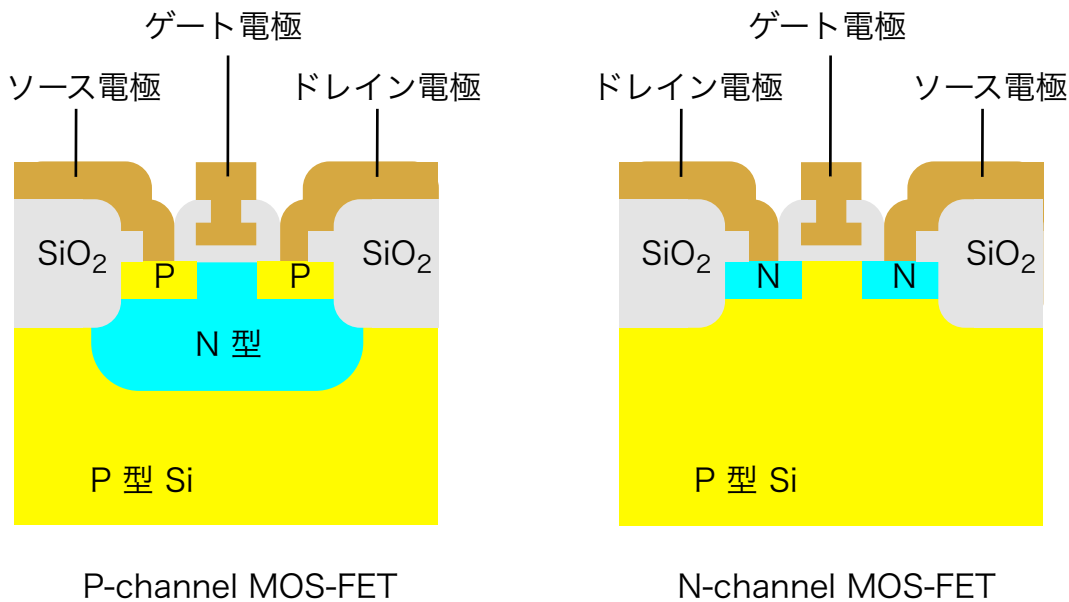


Fig. 1.3.1.3 MOS-FET の構造。P 型 Si 基板上に作製される場合。

MOS-FET は金属電極と半導体基板の間を薄い酸化物層^{さえぎ}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おり、ソース source (水源)、ドレイン drain (排水口)、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 (キャリア 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ちます。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、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。[Figure 1.3.1.1](#) 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、[Fig. 1.3.1.4](#) のようにスイッチに置き換えれば理解しやすいかもしれませんが。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るので、PMOS のソースとゲートの間に電流が流れられ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

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 になるのと同じことで，出力電圧は 0 V になります。

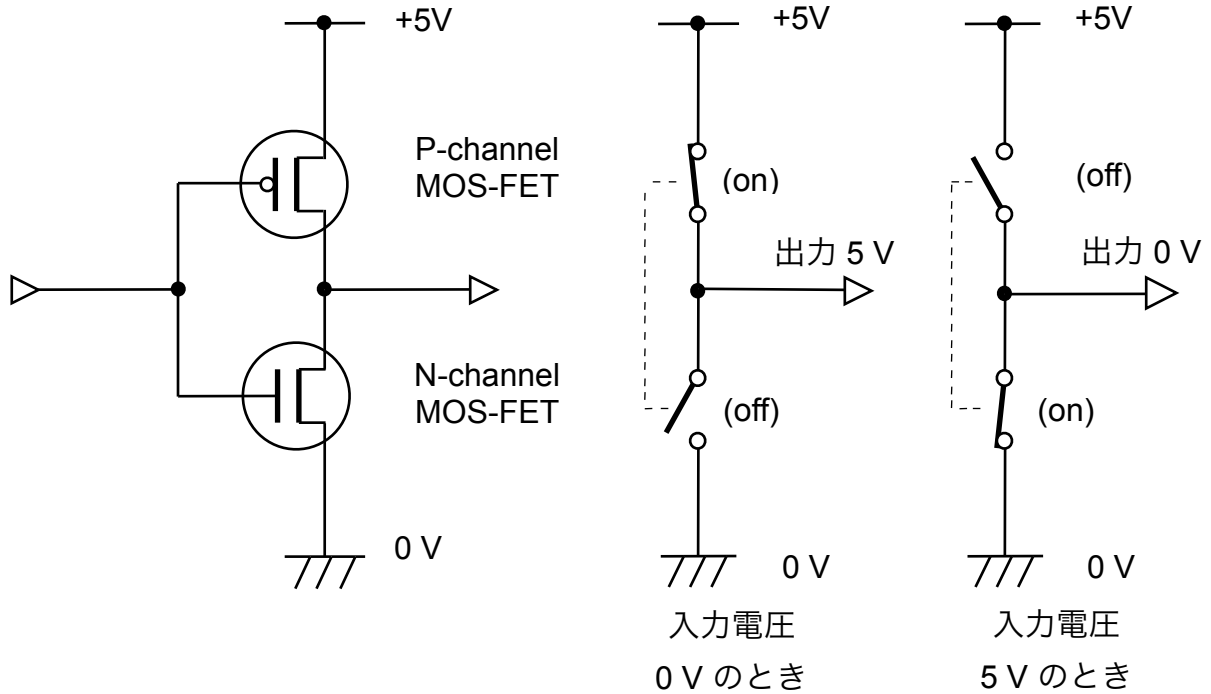


Fig. 1.3.1.4 NOT 回路の MOS-FET をスイッチに置き換えて考える

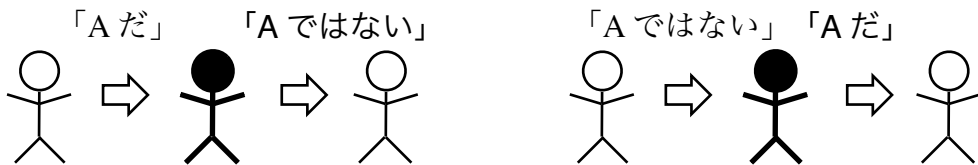


Fig. 1.3.1.5 NOT 回路のはたらき。つねに聞いたことと反対のことを言う人と同じ。

NOT 回路のはたらきは，[Fig. 1.3.1.5](#) のように，「つねに聞いたことと反対のことを言う人」と同じことと思っても良いでしょう。

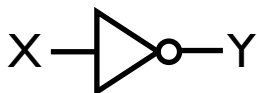


Fig. 1.3.1.6 NOT 素子 (ゲート) の MIL 記号

一つの NOT 回路ごとに [Fig. 1.3.1.1](#) のような回路を描くと回路図が複雑になりすぎるので、実際の論理回路図では、NOT 回路はひとまとまりとして、[Fig. 1.3.1.6](#) のような記号であらわされます。これは米国軍が利用した MIL (ミル) (Military) 記号と呼ばれるもので、現在でも一般的に使われています。

現実のデジタル電子回路の NOT 素子は電源の +5 V と 0 V の端子に導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 に接続する導線は省略します。

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。[Figure 1.3.2.1](#) 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。

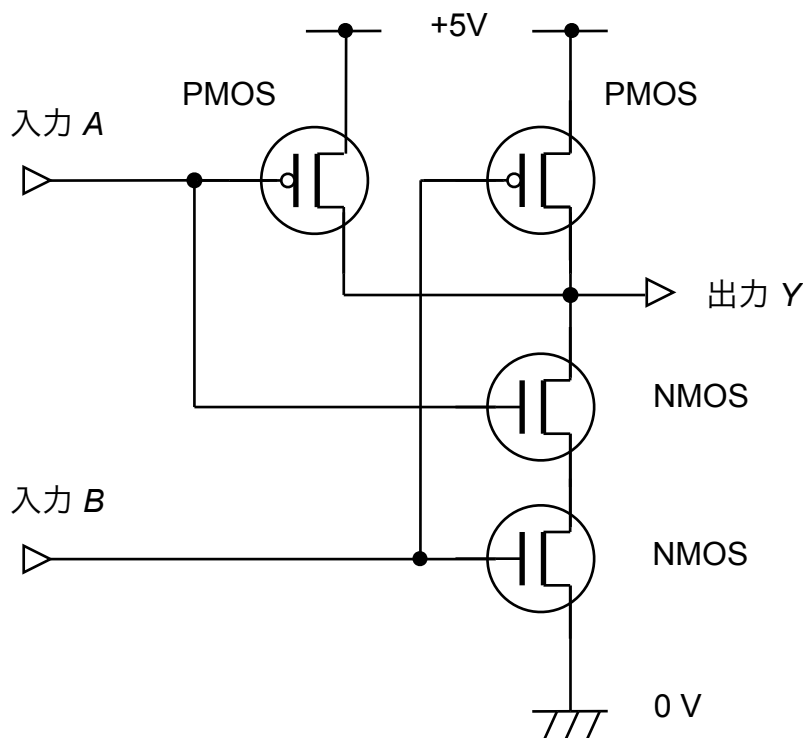


Fig. 1.3.2.1 MOS-FET による NAND 回路

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、0 V 端子とは絶縁し

た状態になり、やはり出力は5Vです。入力が両方とも5Vのときだけ出力が0Vになります。0V→0, 5V→1として、[Table 1.3.2.1](#)に示すような関係があります。

Table 1.3.2.1 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

NAND回路のMOSをスイッチに置き換えれば[Fig. 1.3.2.2](#)のようになります。

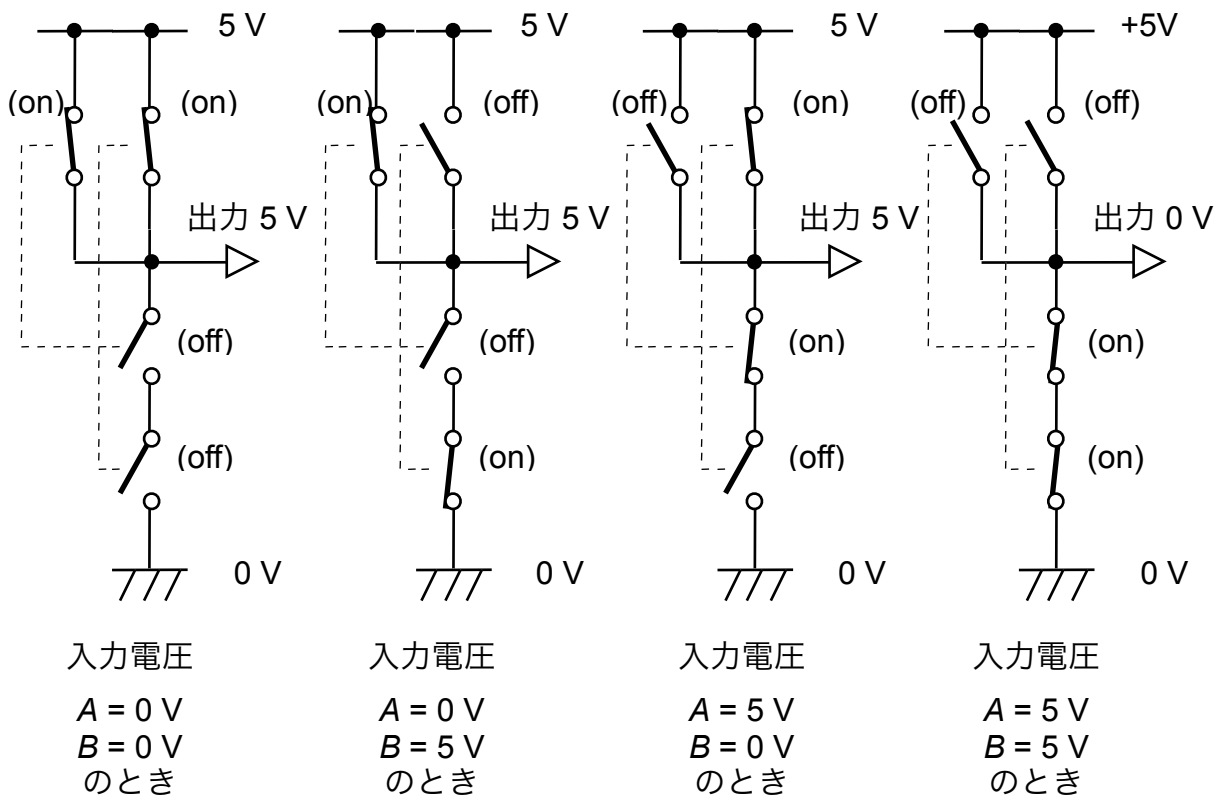


Fig. 1.3.2.2 NAND回路のMOS-FETをスイッチにおきかえて考える

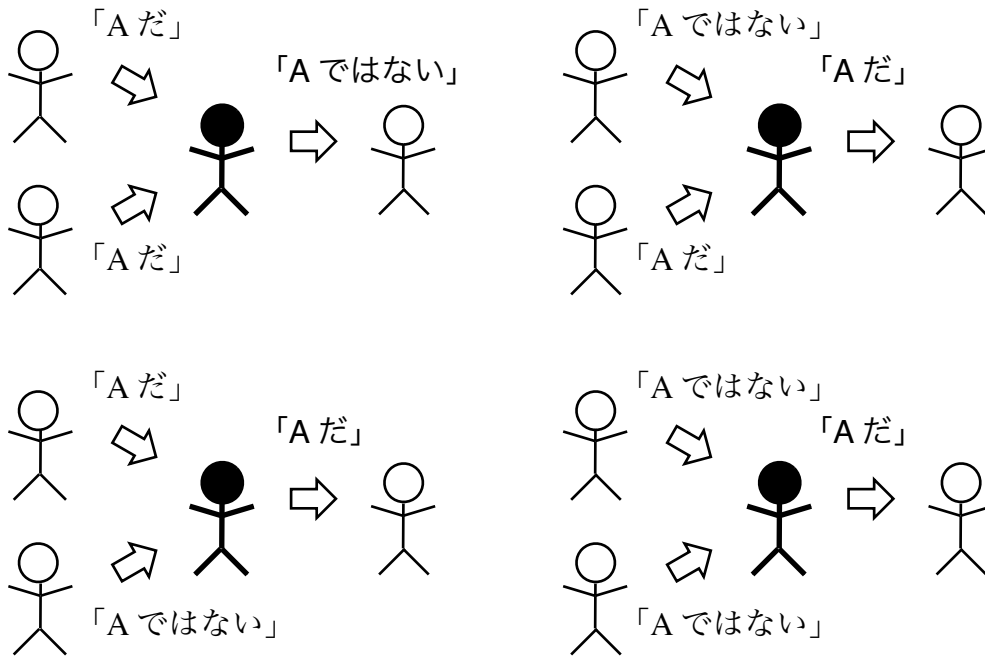


Fig. 1.3.2.3 NAND 回路のはたらき

論理回路で NAND 演算を実現するための素子 (NAND ゲート) は、[Fig. 1.3.2.4](#) のような MIL 記号であらわされます。

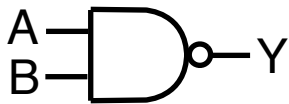


Fig. 1.3.2.4 NAND ゲートの MIL 記号

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 ([Fig. 1.3.3.1](#), [Fig. 1.3.3.2](#))。

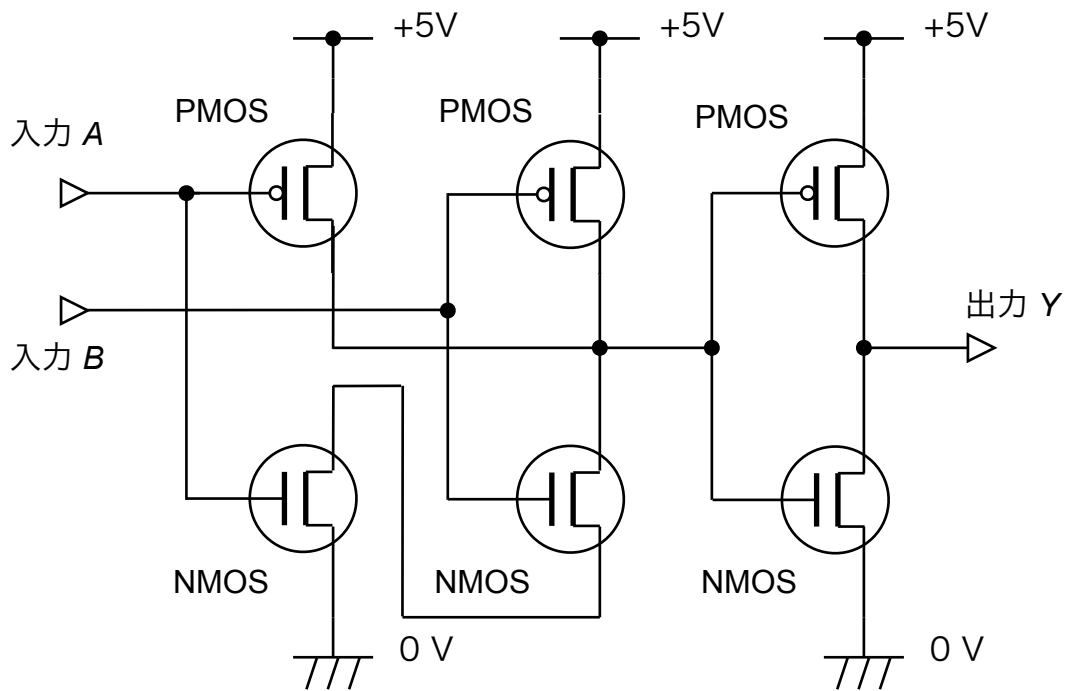


Fig. 1.3.3.1 MOS-FET による AND 回路

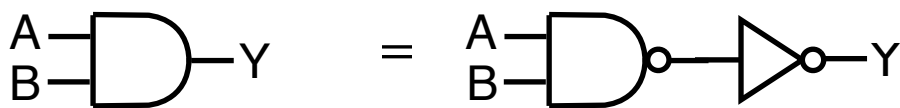


Fig. 1.3.3.2 AND ゲートの MIL 記号 (左)。NAND ゲートと NOT ゲートの組み合わせで実現される (右)。

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 ([Fig. 1.3.4.1](#))。また NOR 回路を NOT 回路と接続すれば OR (オア) 回路になります ([Fig. 1.3.4.2](#))。NOR ゲートと OR ゲートの MIL 記号を[Fig. 1.3.4.3](#) に示します。

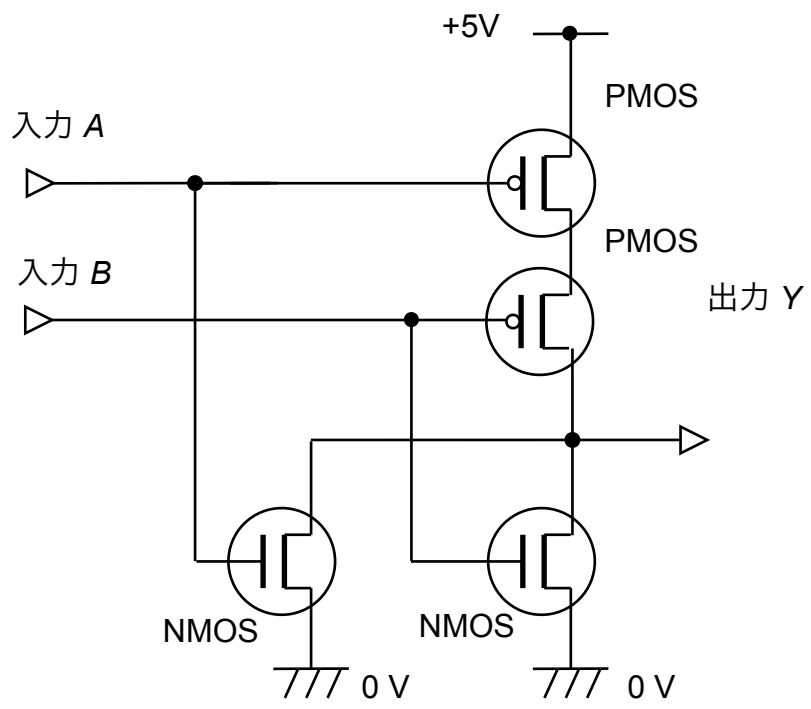


Fig. 1.3.4.1 MOS-FET による NOR 回路

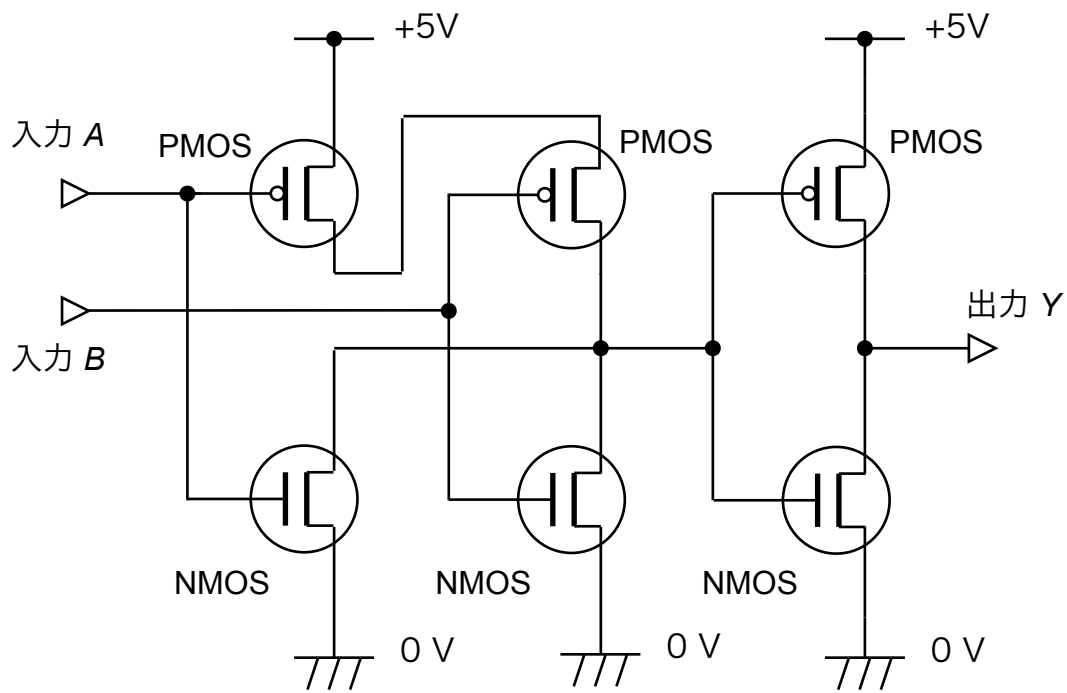


Fig. 1.3.4.2 MOS-FET による OR 回路



Fig. 1.3.4.3 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：

$$Y = (A \vee B) \wedge \overline{A \wedge B} = (A \wedge \overline{B}) \vee (\overline{A} \wedge B) \quad (1.3.5.1)$$

は排他的論理和, exclusive OR (エクスクルーシブ・オア), XOR (エックス・オア; ゴア), EOR (イー・オア) と呼ばれ, 意外に良く使われます。そのうちの一例は第3章で取り上げる「算術演算」です。

式 (1.3.5.1) のような論理式の通りに NOT, AND, OR 素子を組み合わせても XOR 素子を作れるのですが, MOS-FET を組み合わせるなら Fig. 1.3.5.1 に示すように 3 組の CMOS で排他的論理和回路が作れます (補足 1.3.5.A)。XOR ゲートの MIL 記号を Fig. 1.3.5.2 に示します。

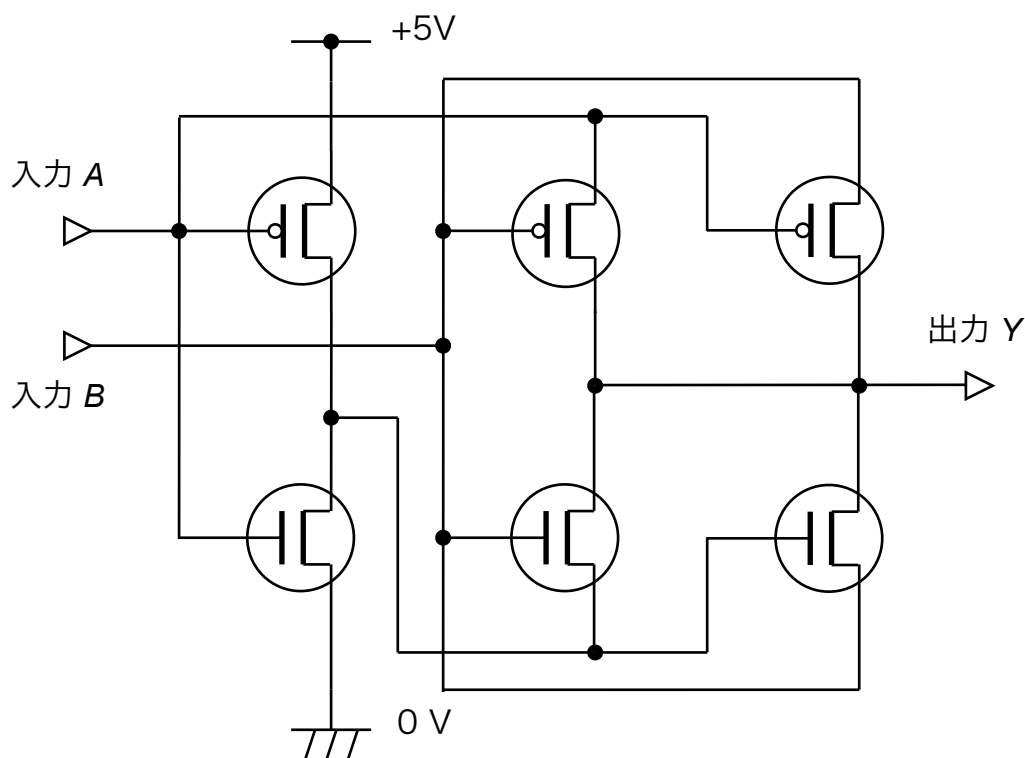


Fig. 1.3.5.1 MOS-FET による XOR 回路

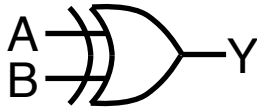


Fig. 1.3.5.2 XOR ゲートの MIL 記号

補足

(補足 1.A) コンピュータ (computer) (↔)

英語では計算をするための機械（計算機）を表す単語として calculator の語と computer の語がある。例外もあるが、一般的には“...er”と綴られるのは「...をする人」の意味で，“...or”と綴られるのは「...をするための道具」または「...をするために雇われる人」の意味である場合が多い。

米国で原子爆弾を開発するためのマンハッタン計画 (Manhattan Project) では、爆縮 (bomb's implosion) によって連鎖的な核分裂反応を起こす臨界量を超えさせるためにどうすれば良いか、そして連鎖的な核分裂反応により放出されるエネルギーを正確に予想することなどのために、大規模な計算が必要になった。1943 年からマンハッタン計画に加わったコンピュータ技術者スタンリー・ファンケル Stanley Fankel は、米国ニューメキシコ州のロスアラモス研究所 (Los Alamos National Laboratory; LANL) で計算チームのチーフをしていた時期に、機械式タイプライタなどの製造会社であった IBM 社に複数の機械式加算機 (adder) や乗算機 (multiplier) などを作らせ、一つの計算機のオペレータから次の計算機のオペレータにカードを送って並列に計算を進めるシステムを考案し、一人のオペレータが複数の計算機を扱うのと比べて、飛躍的に計算速度を向上させたと言われる。しかしファンケルはコンピュータを本来の仕事とは無関係の遊び (パンチ・カード穿孔機 (tabulator) を使って逆正接函数表を自動的に生成させるにはどうしたら良いか考えることなど) をしてばかりで現場への指示をしなくなり、作業効率が極端に低下することになった。同じ時期にロスアラモス研究所でマンハッタン計画の計算部門に加わっていた米国の理論物理学者リチャード・ファインマン Richard P. Feynman は、自伝的な著書 “Surely You’re Joking, Mr. Feynman!” の中で「ファンケルはコンピュータの発明者であり、はじめのコンピュータ病 (computer disease) 患者であった」と記している (Feynman, [1985](#))。

過去には computer の日本語訳語として「コンピューター」と表記される場合が多かった。例えば令和 4 年 (2022 年) 12 月 19 日づけの文部科学省初等中等教育局から都道府県委員会担当者などにあてた事務連絡では「コンピュータ」の表記が用いられたが (<https://www.mext.go.jp/content/000207291.pdf>)、朝日新聞社はそれを「コンピューター」の表記を用いて報じた (<https://www.asahi.com/articles/ASQDL3RG4QDKULZU005.html>)。 (↔)

(補足 1.1.A) 論理代数・論理演算の使われ方 (↔)

コンピュータで使われる「論理」と、日本語や英語などの「自然言語」による「論理」とは似ている面がある。

たとえば、「今日は休講だ」ということを A という記号で表して、それが「正しい」ことを「 A は値 1 を取る」と言い換え、「正しくない」ことを「 A は値 0 を取る」と言い換えることができる。

今日が休講なら $A = 1$, 休講でなければ $A = 0$ と表現することもできる。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, 「今日は休講でない」ということが \bar{A} と表され, もし今日が休講なら $A = 1$ であり $\bar{A} = 0$, 休講でなければ $A = 0$ であり $\bar{A} = 1$ であることは [Table 1.1](#) の 1 番目の「NOT 演算の規則」から確かめることができる。

「来週は試験だ」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 $A \wedge B$ と表され, 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が正しい」ということが $A \vee B$ と表されることになる。

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ないが, 実際に使う論理演算は, (論理と言う名前はついているが) 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ない。

必要なことは, 普通の ($+$, $-$, \times , \div などの) 計算の規則ではなく, [Table 1.1](#) として示した「論理演算の規則」に従うということだけであろう。混乱しそうになれば, 論理演算の規則に戻って考えるようにすれば良いと思われる。 (\leftarrow)

(補足 1.2.A) ド・モルガンの法則に意味を持たせること (\leftarrow)

ド・モルガンの法則に特別に深い意味のようなものを持たせる必要はないが, 意味を持たせたい場合があれば, たとえば次のような記述を想像してみても良いであろう。

「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」

「タワケ」を記号 A で表し, 「アホ」を記号 B で表せば, 上の記述は $Q = \overline{A \wedge B}$ と表すことができる。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」 ($Q = A \vee B$) と同じことを言おうとしている。「A ならば B である」 ($A \rightarrow B$) 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」 ($\bar{A} \vee B$) と等価だということも理解しにくいかもしれないが, 上の記述が「Q 君はタワケでなければアホである」 $\bar{A} \rightarrow B$ と言い換えられることに対応する。

どうして当たり前のことにわざわざ「ド・モルガンの法則」と名前を付けて呼ばれることが多いのだろうか?

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便利で役に立ち, 頻繁に用いられるからと思われる。現実の論理演算は電子回路を使って実現される。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, 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ることのできる場合がある。 (\leftarrow)

(補足 1.3.5.A) XOR 回路のしくみ (↔)

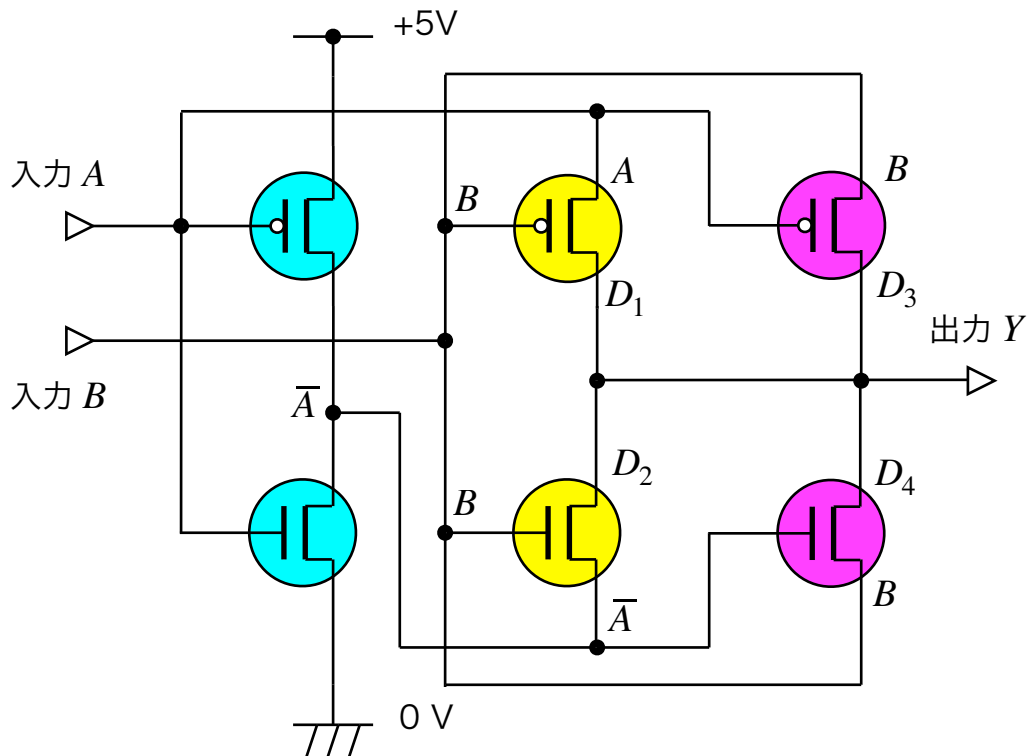


Fig. 1.3.5.A.1 XOR 回路のしくみを説明する図。3組の CMOS を水色，黄色，ピンクに色分けして示す。

[Figure 1.3.5.A.1](#) に XOR 回路で使われる 3 組の CMOS を水色，黄色，ピンクに色分けして示す。

[Figure 1.3.5.A.1](#) 中，水色の CMOS は NOT 回路（[1-3-1 節](#)）であり，入力 A はこの NOT 回路の入力として接続される。この NOT 回路の出力は \bar{A} と表される。

残りの 2 組の CMOS（黄色とピンク）の出力が並列に接続されたものが XOR 回路全体の出力 Y である。この部分の CMOS の働きを確かめるために PMOS（上段）と NMOS（下段）とに分けて考える。

[Figure 1.3.5.A.1](#) の中で，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，PMOS のソースは A に，NMOS のソースは \bar{A} に接続されている。PMOS はゲート電圧 (B) がソース電圧 (A) より低い場合 ($B < A$) のときだけソース (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 $D_1 = A = 1$ と確定する。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 は確定しない。このことを [表 1.3.5.A.1](#) の D_1 の値として示す。一方，NMOS はゲート電圧 (B) がソース電圧 (\bar{A}) より高い場合 ($\bar{A} < B$) だけソース (\bar{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 $D_2 = \bar{A} = 0$ と確定し，それ以外の場合 D_2 は不定である。このことを [表 1.3.5.A.1](#) の D_2 の値として示す。

[Figure 1.3.5.A.1](#) 中，ピンクの PMOS（上）のゲートは A に，NMOS（下）のゲートは \bar{A} に接続され，ソースはいずれも入力 B に接続されている。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 は不定である。このことを [Table 1.3.5.A.1](#) では D_3 の値として示す。

最後に、ピンクのNMOSはゲート電圧(\bar{A})がソース電圧(B)より高い場合($B < \bar{A}$)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 は不定である。このことを [Table 1.3.5.A.1](#) の D_4 の値として示す。

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るが、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る。

Table 1.3.5.A.1 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	\bar{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

[↔](#)

参考文献

Feynman, R., P. (1985). “*Surely You’re Joking, Mr. Feynman!*”: *Adventures of a Curious Character*, W. W. Norton [ISBN 0-393-01921-7]; Feynman, R. P. (1997). “*Surely You’re Joking, Mr. Feynman!*”: *Adventures of a Curious Character (paperback)*, W. W. Norton [ISBN 0-393-31604-1]; リチャード P. ファインマン(2000) (大貫昌子訳) 「ご冗談でしょうファインマンさん (上)」岩波現代文庫 社会 5, [ISBN-13: 978-4006030056]; リチャード P. ファインマン(2000) (大貫昌子訳) 「ご冗談でしょうファインマンさん (下)」岩波現代文庫 社会 6, [ISBN-13: 978-4006030063] [↔](#)