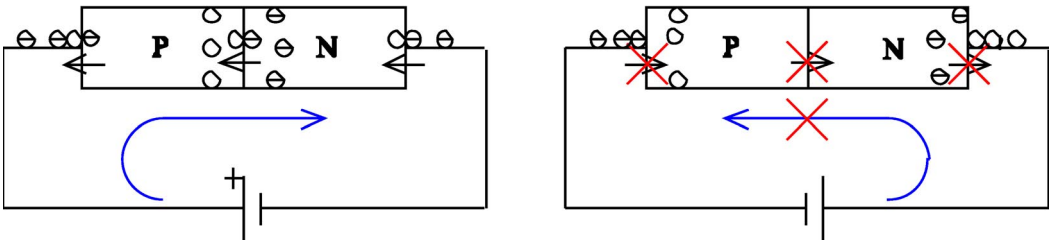


図 4.8: ダイオードの仕組み

トランジスタ　下図のように、トランジスタは PNP 接合 (NPN 接合) することで電流の増幅作用を行います。トランジスタにはエミッタ・コレクタ・ベースがあり、PNP 接合の場合にはベースに電流を流すことによってコレクタからエミッタへ増幅された電流が流れます。

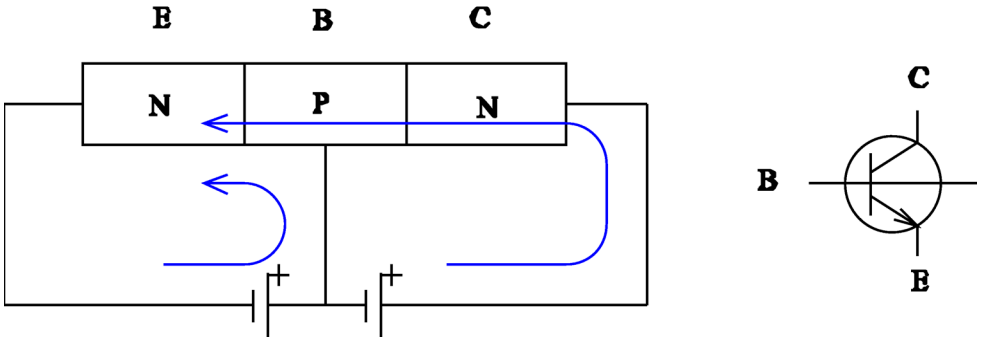


図 4.9: トランジスタと回路記号

4.2 論理素子

論理素子は論理演算を回路で実現したものです。現代のコンピュータは半導体によって実現されていますが、半導体が発明されるまでは**電磁リレー**や**真空管**を使用していました。そのような背景から、構成される論理素子の違いによってコンピュータの世代が特徴づけされていて、第 1 世代は真空管、第 2 世代は個別の半導体トランジスタ、第 3 世代は半導体の集積回路 (IC:Integrated Circuit)、第 4 世代は集積度の高い大規模集積回路 (LSI:Large Scale Integrated circuit) に分類されます。現在は第 4 世代になります。

基本的な論理演算である AND,OR,NOT の論理素子は次のようにように表されます。また、1 つの論理関数で最小万能演算系となる否定積は AND と NOT の論理素子をくっつけて以下のように表されます。

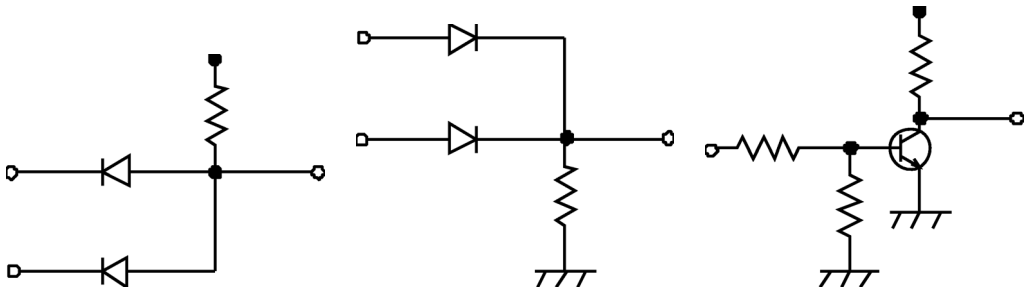


図 4.10: 論理素子

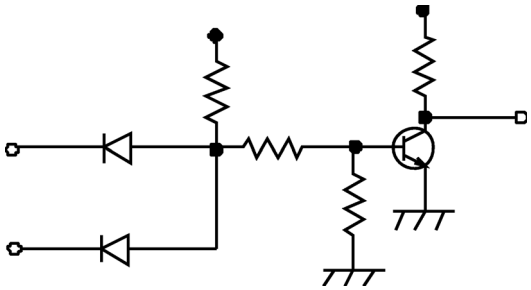


図 4.11: NAND 回路

論理積回路の動作について以下の図を参考にして解説します。ただし、A または B が 1 のとき、A または B に電圧 V がかかるものとします。$A = 1$ かつ $B = 1$ のときは、左下の図のように電圧 V を回路にかけると A または B に電流を流そうとします。しかし、A と B のどちらも電圧 V がかかっており、残っている X に電流を流します。すなわち、$X = 1$ の出力を得ます。それ以外の時は、例えば右下の図のように $A = 0$ のとき、A に電圧がかかっていないので、電圧 V によって流れる電流は A に流れてしまい、X に電流が流れないので $X = 0$ となります 。

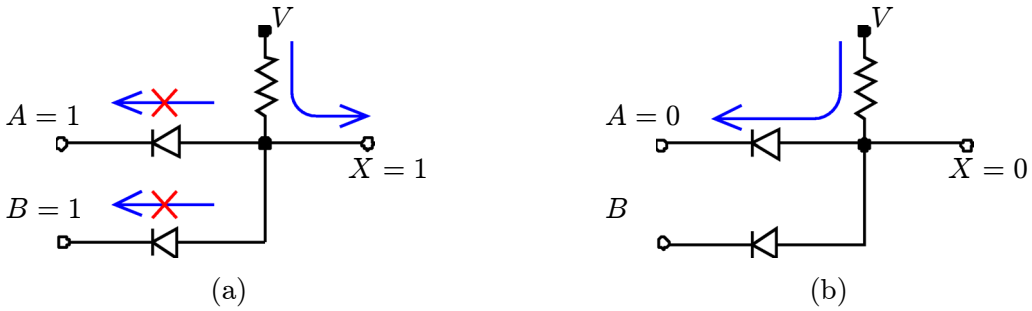


図 4.12: AND 回路の動作

否定回路の動作については $A = 1$ のときはトランジスタのベースに電流が流れるので、図の電圧 V によって流れる電流はアースに流れてしまい、$X = 0$ となります。また、$A = 0$ のときはトランジスタのベースに電流が流れないので、図の電圧 V によって流れる電流をアースに流すことができないので X に流します。すなわち、$X = 1$ となります。

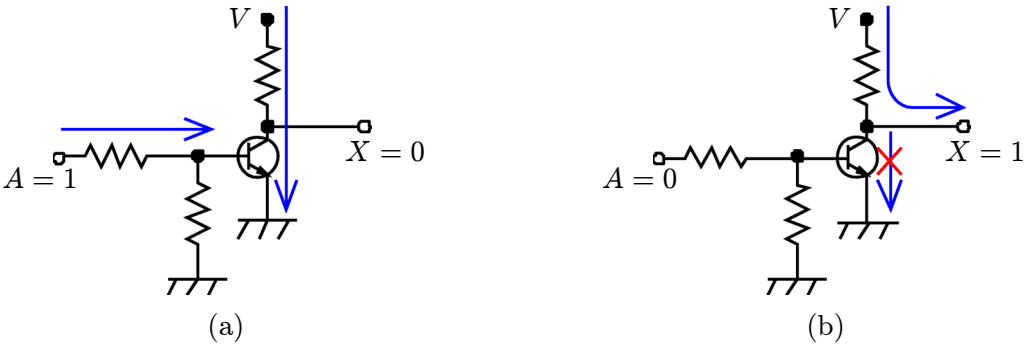


図 4.13: NOT 回路の動作

ここで、論理素子を簡略化して書き表すための **MIL 記号**を以下に紹介しておきます。以後、この記号を使って話を進めていきます。もともと、MIL 記号はアメリカ合衆国の軍隊で使われていたものですが、コンピュータは砲弾の弾道を計算するために開発されたという歴史的背景によって、現在でもこの記号が使われます。下図の記号では、左側が入力で、右側が出力です。

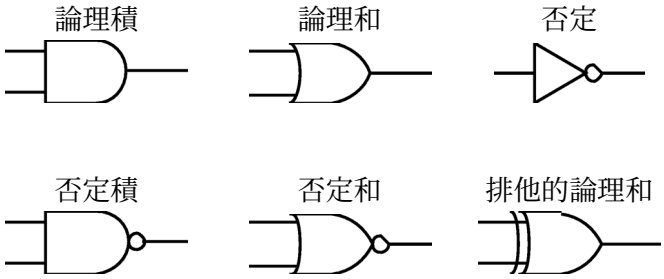
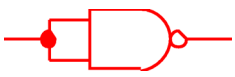


図 4.14: MIL 記号

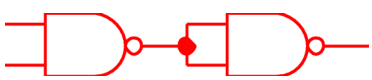
例題 1 MIL 記号の NAND 回路のみを使って NOT 回路を作りなさい。

解答例 3.3 節の例題 1 より $\overline{A} = \overline{A \cdot A} = A \mid A$ であるから



例題 2 MIL 記号の NAND 回路のみを使って AND 回路を作りなさい。

解答例 3.3 節の例題 1 より $A \cdot B = (A \mid B) \mid (A \mid B)$ であるから



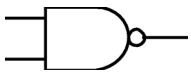
問題 1　MIL 記号の NAND 回路のみを使って OR 回路を作りなさい。

問題 2　MIL 記号の NAND 回路のみを使って NOR 回路を作りなさい。

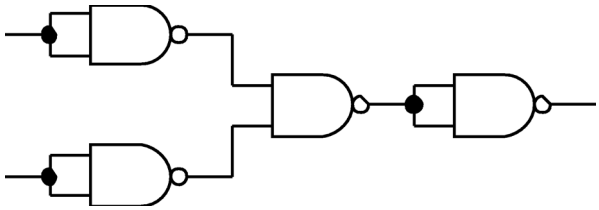
問題 3　MIL 記号の NAND 回路のみを使って XOR 回路を作りなさい。

問題 4　次の論理回路で表される論理関数を求め、否定和 ($\downarrow$) のみで表しなさい。

(1)



(2)



問題 5　MIL 記号の NAND 回路のみを使って論理式 $(A \cdot B + C \cdot D) \cdot \overline{C}$ を実現する回路を作りなさい。

4.3　加算回路

2 章で述べたように、コンピュータの算術演算の基本は加算とシフト演算です。シフト演算の回路を構成するのは簡単なので省略します。ここで重要なのは加算回路で、論理素子を使って構成されます。

最初に、1 桁の加算[1] $S = A + B$ を考えてみましょう。$A = 1$ かつ $B = 1$ のとき繰り上がりが起きることに注意します。繰り上がりの有無を変数 C とし、A と B の組み合わせを全て求めると表 4.1 になります 。

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

表 4.1: 1 桁の加算

$$\Longleftrightarrow \qquad \begin{aligned} S &= A \oplus B \\ C &= A \cdot B \end{aligned}$$

従って、S と C の関係式が求まるので、図 4.15(a) のような回路によって 1 桁の加算回路を構成することができます。ただし、任意の桁の加算を行うときには下の桁からの繰り上がりを考慮する必要があり、この回路で実現されるのは繰り上がりだけです。そのため、**半加算器**(half adder, HA) と呼ばれ、図 4.15(b) のようにブロック図に置き換えて表します。

[1] 論理和ではないことに注意しなさい。

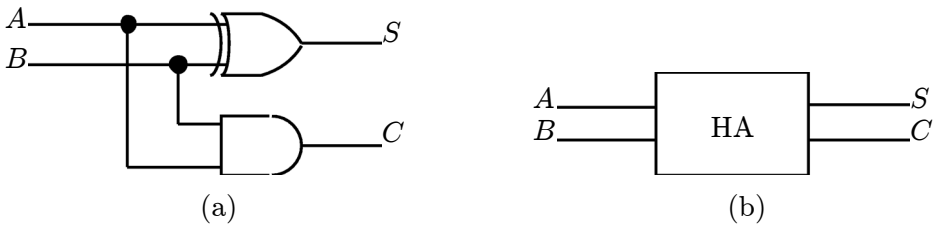


図 4.15: 1 桁の加算回路

全加算器(full adder, FA) は 2 つの半加算器を使って作ることができます。半加算器と同じように考えて、i 桁目の全加算器を考えてみましょう。i 桁目の A_i, B_i と $i-1$ 桁目の繰り上がり C_{i-1} の全ての組み合わせから S_i と C_i の値は表 4.2 となります 。

C_{i-1}	A_i	B_i	S_i	C_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$\Longleftrightarrow \quad S_i = (A \oplus B) \oplus C_{i-1}$$

$$C_i = (A \oplus B) \cdot C_{i-1} + A_i \cdot B_i$$

表 4.2: i 桁目の全加算

従って、S_i と C_i の関係式が求まるので、図 4.16(a) のような回路によって i 桁目の全加算器を構成することができます。また、図 4.16(b) のようにブロック図に置き換えて表します。

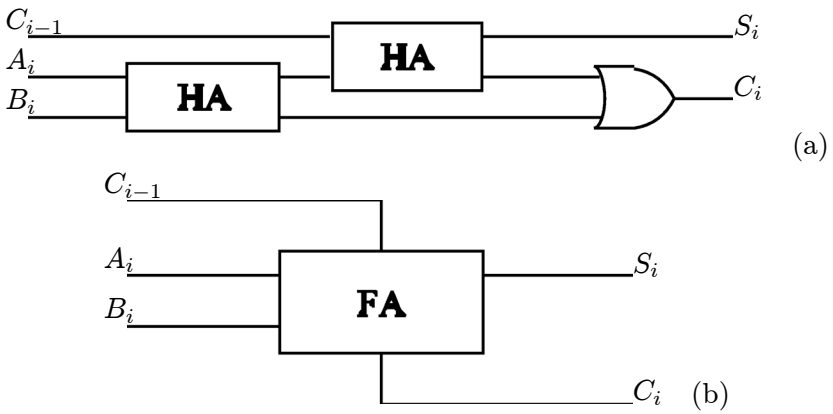


図 4.16: i 桁目の全加算器

全加算器を使って 4 ビットの加算器を作ると図 4.17 のようになります。

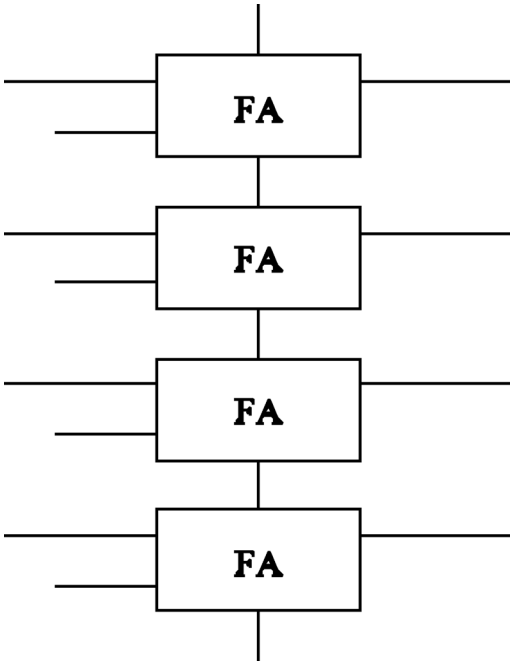


図 4.17: 4 ビットの全加算器

4.4　フリップフロップ

コンピュータを構成する上で、演算回路の他にレジスタやプログラムカウンタなどの記憶回路が必要となります。実際、**フリップフロップ**(flip flop) によって実現されていて、図 4.18 のような回路で実現されます。

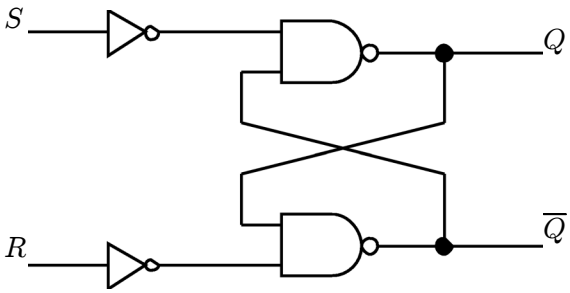


図 4.18: フリップフロップ

図 4.18 の回路は、$S=1$ かつ $R=0$ のとき $Q=1, \overline{Q}=0$ の安定状態となります (この回路は対称なので $S=0$ かつ $R=1$ のとき $Q=0, \overline{Q}=1$ の安定状態になります)。また、$S=1$ かつ $R=0$ の状態から S の値を $1 \to 0 \to 1 \to \cdots$ に変化させても $Q, \overline{Q}$ の値は変化しないことがわかります。逆に、$S=1, R=0$ または $S=0, R=0$ の状態から $S=0, R=1$ の状態にすると $Q=0, \overline{Q}=1$ の状態になり値が変化します。以上をまとめると、表 4.3 のように S と R の状態によって $Q, \overline{Q}$ の値が変化します。ただし、$S=1, R=1$ のときは状態が不安定になるので使いません 。

S	R	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	0	1
1	0	1	0

表 4.3: R, S による $Q, \overline{Q}$ の状態

このように、フリップフロップ回路は S と R の値によって Q の値を保持や変更が可能になります。

実際のコンピュータでは**クロック**(clock) と同期を取って値の変更を行います (図 4.19 参照)。なお、クロック C は周期的に 1 と 0 の状態を交互に作り出します 。

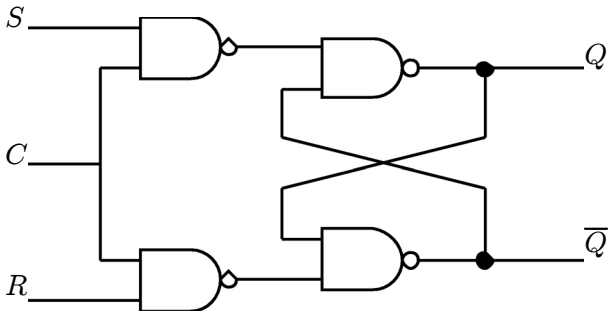


図 4.19: クロックを用いたフリップフロップ

初期状態を $S=1, R=0$ として、時間的な流れによる変化を見ていきましょう。

クロックだけで Q の値を変えない場合 $(S=0, R=0)$

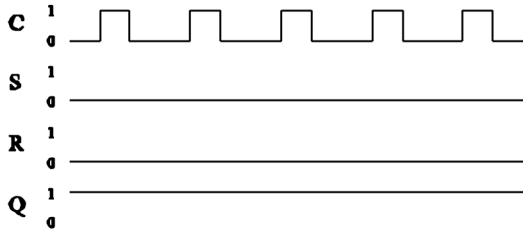


図 4.20: $S=0, R=0$

Q の値を 0 にセットする場合 $(S = 0, R = 1)$

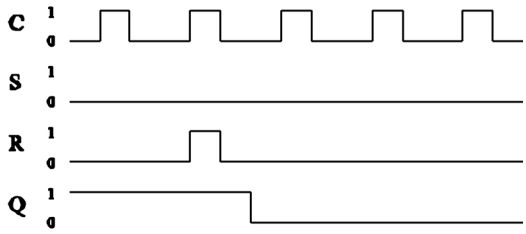


図 4.21: $S = 0, R = 1$

Q の値を 1 に再セットする場合 $(S = 1, R = 0)$

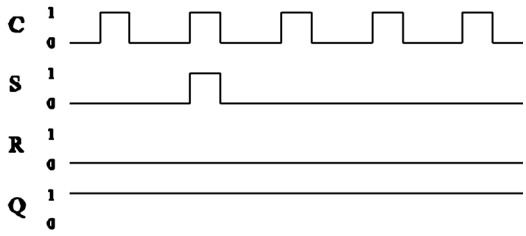


図 4.22: $S = 1, R = 0$