物理屋のための電子回路論 第 12 回

勝本信吾
東京大学理学部・理学系研究科 (物性研究所)

2016 年 1 月 13 日

6.4.2 パルス変調

パルス変調は, 搬送波としてパルス列を用いる方式で, やはり振幅変調 (PAM), 位置変調 (PPM), 幅変調 (PWM) 等がある.

パルス振幅変調

PAM 方式の場合，各パルスにどのように信号を乗せているかは別として，これらを何らかの方法により取得した場合，パルス列は時間に対する離散信号と見ることができるから，受信側では前節の標本化信号が得られることになる．従って，復調のためには，信号帯域に対してパルス間隔が標本化定理の条件を満たしていることが必要である．前節では「十分」しか示していないが，間隔が広く，フーリエ成分に重なりが生じた場合は，単純にカットして復調した場合，重なりによってひずみ (aliasing noise，折り返し雑音) が生じることが明らかである．

各パルスが間隔に比して十分狭く，デルタ関数 (インパルス) 列とみなせる場合は，PAM は前節の標本化定理そのものである．インパルス標本化，理想標本化などと呼ぶ．間隔 τ のインパルス列キャリアは $c(t) = \delta_\tau(t)$ であり，そのフーリエ級数展開は (6.61) である．また，式 (6.65) が入力信号と変調信号との関係を示している．更に，(6.62) のフーリエ変換が周波数スペクトルを表している．すでに前節で示したことであるが，一応入力信号，変調信号，変調信号スペクトルの例を示すと図 6.22 のようになる．

このようなデルタ関数的キャリアの PAM 信号 (理想 PAM と呼ぶことがある) の復調は，従って前節で述べた図 6.22(c) のスペクトルから最低周波数バンドだけ抜き出すフィルターにかければ良い[*1]ことがわかる．このフィルターを伝達関数 $\Xi(i\omega)$ で表すと，これは (6.63) そのもの，すなわち $\Xi(i\omega) = P_{\pi/\tau}(\omega)$ である．

PAM 方式では，現実の信号の物理的制限から，パルス幅 $\tau_{\rm p}$ が周期 τ に比べて無視できず，方形波として扱わなければならないことも多い．この場合，ヘヴィサイド関数 $H(x)$ あるいは，(6.63) で定義される窓関数を使ってキャリ

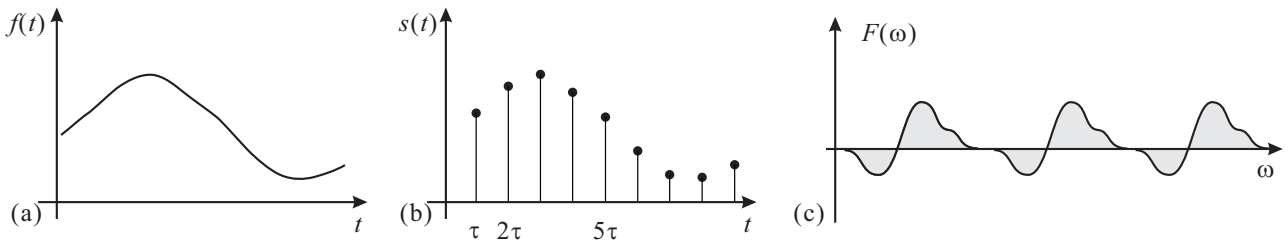


図 6.22　PAM 変調．(a) 原信号．(b) PAM 変調信号．(c) PAM のフーリエ変換スペクトル例．模式図

[*1] 図 6.22(c) はパワースペクトルであり位相情報は失われているが，当然フィルターは各周波数に対して位相を保持するものでなければならない.

ア $c(t)$ を

$$c(t) = \frac{1}{\tau_{\rm p}} \sum_{n=-\infty}^{\infty} \left[H\left(t - n\tau + \frac{\tau_{\rm p}}{2}\right) - H\left(t - n\tau - \frac{\tau_{\rm p}}{2}\right) \right] = \frac{1}{\tau_{\rm p}} \sum_{n=-\infty}^{\infty} P_{\tau_{\rm p}/2}(t - n\tau) \tag{6.66}$$

と書くと，窓関数のフーリエ変換である sinc 関数 (6.64) を用いて，$c(t)$ のフーリエ級数展開

$$c(t) = \frac{1}{\tau_{\rm p}} \sum_{n=-\infty}^{\infty} \mathrm{sinc}\left(n\frac{\tau_{\rm p}}{\tau}\right) \exp\left(2n\pi i \frac{t}{\tau}\right) \tag{6.67}$$

が得られる．入力 $f(t)$ に対する変調信号 $s(t)$ は $s(t) = f(t)c(t)$ であるから，その周波数スペクトル (フーリエ変換)$S(i\omega)$ は $\mathscr{F}\{f(t)\} = F(i\omega)$ と書いて

$$S(i\omega) = \frac{1}{\tau_{\rm p}} \sum_{n=-\infty}^{\infty} \mathrm{sinc}\left(n\frac{\tau_{\rm p}}{\tau}\right) F[i(\omega - 2n\pi/\tau)] \tag{6.68}$$

である．

6.4.3 離散フーリエ変換

PAM の復調の場合などは特に問題となるが，実際にサンプリングによって手元に集められるデータは必ず有限時間のものである．このような有限時間サンプリングデータに対する処方箋として，特にフーリエ変換系の場合は周期関数化が有効な方法である．以下これを考え，離散フーリエ変換を導入する．

時刻 $[0, \zeta)$ の間だけゼロでない値を取る信号 $f(t)$ があり，そのフーリエ変換 $F(\omega)$ が $(-\pi/\tau, \pi/\tau)$ の区間だけでゼロでない値を取るとする．後の便宜のため，ζ，τ を

$$N = \frac{\zeta}{\tau} \tag{6.69}$$

が正整数になるようにとっておく (f，F の定義を満たすような，ζ，τ が適当に大きく取れることは明らかである)．f から周期 ζ の関数を次のように作り出し，$\breve{f}$，そのフーリエ変換を $\breve{F}$ と置く．

$$\breve{f}(t) = f(t) * \delta_\zeta(t) = \sum_{n=-\infty}^{\infty} f(t - n\zeta), \quad \breve{F}(\omega) = \sum_{n=-\infty}^{\infty} F\left(\omega + n\frac{2\pi}{\zeta}\right). \tag{6.70}$$

周期関数 $\breve{f}(t)$ は次のようにフーリエ級数展開できる．

$$\breve{f}(t) = \frac{1}{\zeta} \sum_{n=-\infty}^{\infty} F\left(n\frac{2\pi}{\zeta}\right) \exp\left(2n\pi i \frac{t}{\zeta}\right). \tag{6.71}$$

指数 n を $n = l + mN$ (l，m は整数) と書いて n の和を $\displaystyle\sum_{l=0}^{N-1} \sum_{m=-\infty}^{\infty}$ とする．t を離散的な値 $t = j\tau$ (j は整数) に取ると，

$$\begin{aligned} \breve{f}(j\tau) &= \frac{1}{\zeta} \sum_{l=0}^{N-1} \sum_{m=-\infty}^{\infty} F\left[(l + mN)\frac{2\pi}{\zeta}\right] \exp\left[(l + mN)2\pi i \frac{j\tau}{\zeta}\right] \\ &= \frac{1}{N\tau} \sum_{l=0}^{N-1} \sum_{m=-\infty}^{\infty} F\left(\frac{2\pi l}{\zeta} + m\frac{2\pi}{\tau}\right) \exp\left(2\pi i \frac{lj}{N}\right) = \frac{1}{N\tau} \sum_{l=0}^{N-1} \breve{F}\left(l\frac{2\pi}{\zeta}\right) \exp\left(2\pi i \frac{lj}{N}\right) \end{aligned} \tag{6.72}$$

と書ける．回転因子を

$$W_N \equiv \exp\left(-i\frac{2\pi}{N}\right) \tag{6.73}$$

とし，$\eta \equiv 2\pi/\zeta$ と書くと，

$$\breve{f}(j\tau) = \frac{1}{N\tau} \sum_{l=0}^{N-1} \breve{F}(l\eta) W_N^{-lj} \tag{6.74}$$

と形式的に書くことができる．なお，W_N の次の性質は明らかであろう．

$$\forall n, m \in \mathbb{Z} \quad W_N^{n+mN} = W_N^n, \tag{6.75a}$$

$$\frac{1}{N}\sum_{n=0}^{N-1} W_N^{nm} = \begin{cases} 1 & \text{for} \quad m = 0, \\ 0 & \text{for} \quad m \neq 0. \end{cases} \tag{6.75b}$$

(6.74) の両辺に τW_N^{mj} をかけ，j について $0, \cdots, N-1$ で和を取ると，特に (6.75b) の性質を使って次が得られる．

$$\tau \sum_{j=0}^{N-1} \breve{f}(j\tau) W_N^{mj} = \sum_{j=0}^{N-1} \left[\frac{1}{N} \sum_{l=0}^{N-1} \breve{F}(l\eta) W_N^{(m-l)j} \right] = \breve{F}(m\eta). \tag{6.76}$$

そこで，$n, k \in \mathbb{Z}$ に対して

$$f_n \equiv \breve{f}(n\tau), \quad F_k \equiv \frac{1}{\tau}\breve{F}(k\eta) \tag{6.77}$$

と定義すると，(6.74)，(6.76) は次の形に書けることになる．

$$F_k = \sum_{n=0}^{N-1} f_n W_N^{kn}, \tag{6.78a}$$

$$f_n = \frac{1}{N}\sum_{k=0}^{N-1} F_k W_N^{-kn}. \tag{6.78b}$$

(6.78a) が**離散フーリエ変換** (discrete Fourier transform, DFT)，(6.78b) がその逆変換を表している．以上からわかるように，DFT は等間隔離散化 (標本化) された周期関数に対して適用されるフーリエ変換と見ることができ，フーリエ変換に対して成立する一般的な性質の多くを保持している．

(6.78) はまた，$\boldsymbol{F} = {}^t\{F_i\}$，$\boldsymbol{W} = \{W_N^{ij}\}$，$\boldsymbol{f} = {}^t\{f_i\}$ と置くことで，

$$\boldsymbol{F} = \boldsymbol{W}\boldsymbol{f}, \quad \boldsymbol{f} = \frac{1}{N}\boldsymbol{W}^*\boldsymbol{F} \tag{6.79}$$

と書くことができる．これより，${}^t\boldsymbol{W}^*\boldsymbol{W} = N\boldsymbol{I}_N$ (N 次元単位行列) であるから，$\boldsymbol{W}/\sqrt{N}$ がユニタリ行列になっていることがわかる．このように，離散フーリエ変換において積分変換が有限次元線形代数に変換できたのは，無論，フーリエ変換の線形性が最も重要な因子であり，それが離散化されることで可能になったことであるが，(6.69) のように逆空間スケールの比を整数 N にとって無限回の繰り返しを有限次数の空間に集積したことが重要なトリックであったことには注意しておこう．

現実に実験で得られるデータは離散データでありかつ有限区間での測定であるから等間隔でないものは適当な補間で等間隔とし，また，仮想的に測定区間で周期的な関数であるとすることで離散フーリエ変換が可能になる．現実に離散フーリエ変換をデータから実行する極めて速いアルゴリズムが**高速フーリエ変換** (fast Fourer transform, FFT) であり，物理の実験家で FFT のお世話にならない人はまずいないという程ポピュラーな解析法である．いくつかのアルゴリズムがある内の 1 つを付録 J に示している．

6.4.4 z 変換

離散信号に対する離散フーリエ変換に対し，同じく離散信号に対するラプラス変換を **z 変換** (z-transform) と呼んでいる．ただし，有限時間信号に対する周期化の手法は使用できない．$t \geq 0$ での離散時間 (標本化) 信号

$$\tilde{f}_\tau(t) = \sum_{n=0}^{\infty} f(n\tau)\delta(t - n\tau) \tag{6.80}$$

の (ラプラス変換を考えるため n の和を 0 以上 ($t \geq 0$) とした) ラプラス変換は

$$\mathscr{L}\{\tilde{f}_\tau(t)\} = \mathscr{L}\left\{\sum_{n=0}^{\infty} f(n\tau)\delta(t - n\tau)\right\} = \sum_{n=0}^{\infty} f(n\tau)\mathscr{L}\{\delta(t - n\tau)\} = \sum_{n=0}^{\infty} f(n\tau)\exp(-sn\tau) \tag{6.81}$$

f_n	$F(z)$	収束領域
$\delta(n)$	1	全平面
1	$\dfrac{1}{1-z^{-1}}$	$\lvert z\rvert > 1$
n	$\dfrac{z^{-1}}{(1-z^{-1})^2}$	$\lvert z\rvert > 1$
n^k	$\left(-z\dfrac{d}{dz}\right)^k \dfrac{1}{1-z^{-1}}$	$\lvert z\rvert > 1$
a^n	$\dfrac{1}{1-az^{-1}}$	$\lvert z\rvert > \lvert a\rvert$
$\sin(n\omega\tau)$	$\dfrac{\sin(\omega\tau)z^{-1}}{1-2\cos(\omega\tau)z^{-1}+z^{-2}}$	$\lvert z\rvert > 1$
$e^{-n\alpha\tau}\cos(n\omega\tau)$	$\dfrac{1-e^{-\alpha\tau}\cos(\omega\tau)z^{-1}}{1-2e^{-\alpha\tau}\cos(\omega\tau)z^{-1}+e^{-2\alpha\tau}z^{-2}}$	$\lvert z\rvert > e^{-\alpha\tau}$

表 6.1　片側 z 変換の例と収束領域.

で与えられる.

ここで,

$$z = \exp(s\tau), \tag{6.82}$$

更に，$f_n = f(n\tau)$，$F(z) = \mathscr{L}\{\tilde{f}_\tau(t)\}$ と置くと，(6.81) は

$$F(z) = \sum_{n=0}^{\infty} f_n z^{-n} \tag{6.83}$$

と書くことができる．この級数を**片側 z 変換** (one sided z-transform) と言い，$F(z) = \mathscr{Z}[f]$ などと表す．(6.83) の級数が収束する領域はある正の実数 r_0 を使って一般に

$$|z| > r_0 \tag{6.84}$$

の形に表すことができる．表 6.1 にいくつかの例を示した.

z 変換の逆変換は，Bromwich 複素積分法を用いると,

$$f_n = \frac{1}{2\pi i}\oint_c F(z) z^{n-1} dz \tag{6.85}$$

と書かれる.

性質名	信号	z 変換
線形性	$af_n + bg_n$	$aF(z) + bG(z)$
相似則	$f_{\alpha n}$	$F(z^{1/\alpha})$
時間移動	f_{n+k}	$z^k\left[F(z) - \sum_{l=0}^{k-1} f(l) z^l\right]$
時間移動 II	f_{n-k}	$z^{-k}F(z)$
スケール変換	$e^{\mp\alpha n} f_n$	$F(e^{\pm\alpha} z)$
スケール変換 II	$a^n x_n$	$F(a^{-1}z)$
インデックスとの積	nf_n	$-z\dfrac{d}{dz}F(z)$
微分	$n^k f_n$	$\left(-z\dfrac{d}{dz}\right)^n F(z)$
積分	$\dfrac{f_n}{n+a}$	$z^a \int_z^\infty \xi^{-a+1} F(\xi) d\xi$
畳み込み	$f_n * g_n$	$F(z)\cdot G(z)$
積	$f_n \cdot g_n$	$\dfrac{1}{2\pi i}\oint_c F(\xi) G\left(\dfrac{z}{\xi}\right)\xi^{-1} d\xi$

表 6.2　片側 z 変換の主要な性質. $\mathscr{Z}[f_n] = F(z)$，$\mathscr{Z}[g_n] = G(z)$ としている.

(6.83) に対して，$n < 0$ でも f_n が有限値を持っている場合，**両側 z 変換** (bilateral z-transform) の必要が出てくる．デジタルフィルターなどでは両側 z 変換を考えることもそれ程稀ではない[*2]．

$$F(z) = \sum_{n=-\infty}^{\infty} f_n z^{-n}. \tag{6.86}$$

これは，ローラン展開と呼ばれるものであり，収束領域は一般に $r_1, r_2 \in \boldsymbol{R}$，$0 < r_1 < r_2$ を使って

$$r_1 < |z| < r_2 \tag{6.87}$$

のように表される．

z 変換は次に示す伝達関数を通してディジタル信号処理システムの記述に，また，ラプラス変換と類似の性質を持つことから，ラプラス変換が演算子法を通して常微分方程式の解法に使用されたのと同様，差分方程式を解くのに使用されたりする．主要な性質について表 6.2 にまとめている．

6.4.5 離散化信号と伝達関数

離散化信号入力に対する線形システムの応答を考える．離散化信号を $\tilde{f}_\tau(t) = \sum_{k=-\infty}^{\infty} f(t)\delta(t - k\tau)$ のように書き，$t = n\tau$ $n \in \mathbb{Z}$ とする．$f_n = f(n\tau)$ とし，系の各 δ 関数に対する応答を $h_k = \mathscr{R}\{\delta(k\tau)\}$ と書くと，$\tilde{f}_\tau(n\tau)$ に対する応答 g_n は，(離散) 畳み込みにより，

$$g_n = \mathscr{R}\{\tilde{f}_\tau(n\tau)\} = \mathscr{R}\left\{\sum_{k'=-\infty}^{\infty} f(k'\tau)\delta[(n-k')\tau)\right\} = \sum_{k'=-\infty}^{\infty} f_{k'} h_{n-k'} = \sum_{k=-\infty}^{\infty} h_k f_{n-k} \tag{6.88}$$

と表される．この時，g_n の z 変換は

$$G(z) = \mathscr{Z}[g_n] = \mathscr{Z}\left[\sum_{k=-\infty}^{\infty} h_k f_{n-k}\right] = \sum_{n=-\infty}^{\infty}\left(\sum_{k=-\infty}^{\infty} h_k f_{n-k}\right) z^{-n} = \sum_{k=-\infty}^{\infty} h_k \sum_{n=-\infty}^{\infty} f_{n-k} z^{-n} = \sum_{k=-\infty}^{\infty} h_k z^{-k} F(z)$$

と計算される．ここで，インパルス応答列に対する z 変換を

$$H(z) = \mathscr{Z}[h_n] = \sum_{k=-\infty}^{\infty} h_k z^{-k}$$

と書くと，

$$G(z) = H(z)F(z) \tag{6.89}$$

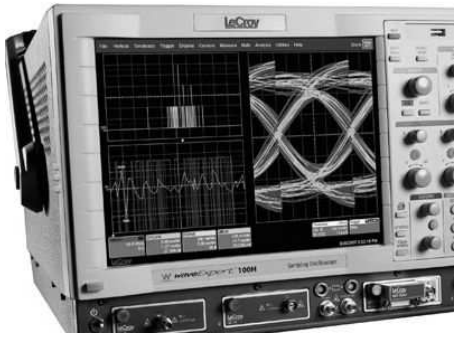
と，(2.11) 式と形式的に同じになる．そこで，この $H(z)$ をやはり，離散信号に対する伝達関数と呼ぶ．

[*2] 通常のラプラス変換でも無論，$t < 0$ の領域も考える両側ラプラス変換は存在する．

第 7 章 ディジタル信号とディジタル回路

7.1 ディジタル信号序論

前章で時間に対して離散化された信号を扱ったが，信号レベルも離散化したものが**ディジタル信号** (digitized signal) である．信号として意味を持つ最小の単位である 2 つのレベルを使用することが最も多く，この信号単位を**ビット** (bit) と称する．ディジタル回路とは，このディジタル信号を扱うことを前提とした電子回路である．ディジタル信号と言っても物理的には電磁気を用いた信号であることに変わりはなく，それを扱う回路も素子レベルまでブレークダウンすればこれまでに見てきた回路素子と全く違いはない．しかし，前章終わりで見たように，離散化によって信号の現実的な扱いには大きな差が生じる．



更にディジタル-アナログ回路の差を明確にしたのが集積回路による回路のカプセル化であり，次節から紹介する線形/非線形の区別も概念上あまり意味をなさない程非線形な論理ゲート IC と，フィードバックを使って線形領域で運用することを主な使用環境とするリニア IC，その中間に位置する目的限定型の IC などのアナログ IC とでは入力に対する応答が全く異なる．ディジタル-アナログでは全く異なる物理系を形成するようになり，この 2 つの「世界」の間をつなぐアナログ-ディジタルインターフェイス回路のためにも専用 IC が作られるようになった．ところが，これまでも度々述べてきたようにディジタル回路が極めて早い応答を求められるようになり，同期のためのクロック信号周波数が飛躍的に高くなった結果，回路配線を伝送線路，信号を導体に沿う電磁波の一種として扱わなければならなくなった．Gbit/s (すなわち，1ns に 1 ビット以上の情報を伝送する) 伝送信号になると，右の写真のオシロスコープ画面波形のように，波形自身は離散化されていないアナログ波形として捉えなければならなくなり，エンジニアは両者渾然とした「電子回路」を相手とする必要が生じている．

これまで，信号としては時間 t に依存する 1 つの物理量 $f(t)$ として扱ってきたが，1 伝送路で 2 値に離散化された信号を送るとすると，同時に n ビットの論理値を伝送するためには n 本の**並列** (パラレル) 伝送路が必要になる．そこで，一般にディジタル信号は多次元化して $q_i(t)$ のようにして扱う．$\{q_i\}$ で 1 つの論理値を表すので，これを入力 $f(t)$ と思えば信号論としては 1 次元で扱うことになる．

ディジタル回路の機能は，ディジタル信号の伝送，演算，蓄積の 3 つにまとめられる．情報の形態は大きく違うものの，物理的な回路としては，アナログ，ディジタルでの区別はなく，特に伝送路は何の違いもない．一方，回路の動作モードは，線形領域で使用することが多かったアナログ回路に比べ，ディジタル回路ではほとんどが非線形領域での動作となる．

物理実験においては，パルスを扱う際の機器間の動作，タイミング調整のために自前のディジタル回路を構成する必要が生じる可能性がある．場合によっては PC による処理が追いつかず，大掛かりなディジタル回路を自作せざるを得ないこともあるが，このような場合は最近では FPGA を用いるのが普通である．が，これを行うためにも個別論理ゲートを用いたディジタル回路の知識は不可欠である．

7.2 論理ゲート

1 ビット信号の 2 つのレベルに対しては，

1. 0，1 に対応させて 2 進数で表す，
2. 偽，真 (false F, true T) と対応させて論理値として扱う，

という処理が (もちろんこれらは同値) 行われる．この論理値に対して「演算」を行う回路は，ブール代数の論理演算と 1 対 1 の対応があり，その動作は入力に対して出力 (演算結果) を表す真理値表を用いて規定される．これを論理ゲート回路と称している．

論理ゲート回路は，その出力が入力データだけで決定される**組み合わせ回路** (combinational logic) と，そのゲートの直前の「状態」にも依存する**順序回路** (sequential logic) とに別れる．今，論理ゲート回路の入力が n ビット，出力が l ビットあり，下の表のように，時刻 $t_1, \cdots, t_m$ に対して入力が $\{f_{ij}\}$ で表されるように変化したとする．入力が一定の間は変化直後の信号遅延時間を除いて出力は一定で変化しない (因果律)．時刻 t_j の入力変化に対して．出力がどのように変化するかを表形式で示したものが**真理値表** (truth table) である．また，入出力の変化を時間に対する矩形波形で表したものが**タイミングチャート** (timing chart) である．

		input					output			
t		t_1	t_2	$\cdots$	t_m		t_1	t_2	$\cdots$	t_m
Ch.	1	0	1	$\cdots$	f_{1m}	1	1	1	$\cdots$	q_{1m}
	2	1	0	$\cdots$	f_{2m}	2	0	1	$\cdots$	q_{2m}
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	n	0	1	$\cdots$	f_{nm}	l	0	1	$\cdots$	f_{lm}

以下，ごく基本的な論理ゲートに限って簡単に紹介しておく．

7.2.1 組み合わせ回路：１入力ゲート

input	buffer	not
0	0	1
1	1	0

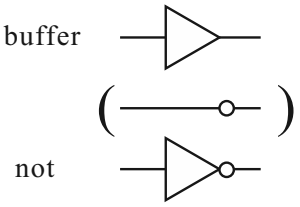


図 7.1　バッファ，及びノットゲートの真理値表と回路図．

組み合わせ回路のうち，入力が 1 つのみのゲートは，図 7.1 に示す 2 種類である．この内，中段に示した丸の記号は独立したゲートではなく，否定 (not) を表す記号であり，バッファ記号との組み合わせで not ゲートを表す．バッファは論理的には何もしていないが，現実には例えばファンアウト (1 つの出力を使って幾つものゲートを同時にドライブしたい場合) の数を取りたい場合や LED 点灯やモータードライブなど電力を必要とする回路に接続する際などに挿入することで良く使用されるゲートである．

7.2.2 組み合わせ回路：２入力ゲート

２入力で出力が１のゲートは，２つの論理値から１つの論理値を導くことに相当し，あらゆる論理演算の基本である．また，他の組み合わせ回路は，すべてこれらの更なる組み合わせから構成できる．特に重要なものは，図 7.2 に示したような，２入力に対して対称な応答をするゲートである．また，この図に示したように，ゲートの出力のところに１入力ゲートで示した白い丸を配することで「否定」を表し，否定出力ゲートとすることができる．真理値表では例として nand ゲートを示した．この nand ゲートはユニバーサルゲートであり，このゲート１つから他のすべての組み合わせゲートを構成することができる．以下，真理値表を用いて主要なフリップフロップの動作を紹介する．順序回路であるから本来はタイミングチャートも描く意味があるが，真理値表でも十分動作は規定できる．

input1	input 2	and	or	xor	nand
0	0	0	0	0	1
1	0	0	1	1	1
0	1	0	1	1	1
1	1	1	1	0	0

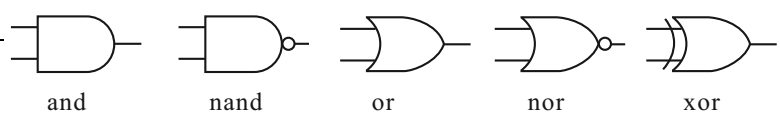


図 7.2　代表的２入力ゲートの真理値表と回路図

7.2.3 順序回路：フリップフロップ

順序回路の中で最も基本的なものがフリップフロップ (flip-flop, FF) であり，他の順序回路も FF と組み合わせ回路ゲートを用いて構成することができる．

RS フリップフロップ

リセット/セット (reset set, RS) フリップフロップは，単にラッチとも呼ばれ，データを保持 (ラッチ) する機能がある．図 7.3 のように R (reset) 入力と S (set) 入力に対し出力 Q と反出力 $\bar{Q}$ が存在する．nand および not ゲートを使って図のように表現することもでき，実際汎用ゲートをこのように配線することで RS フリップフロップとして動作させることもできる．RS 入力によって Q ($\bar{Q}$) を決めると，入力が 0 に落ちてもこの値を維持し，次の RS 入力によって出力が変更されるまでこれを維持する．RS が同時に 1 を取ると，出力は不定になってしまう．

S	R	Q	$\bar{Q}$	動作
0	0	Q	$\bar{Q}$	不変
0	1	0	1	reset
1	0	1	0	set
1	1	不定		

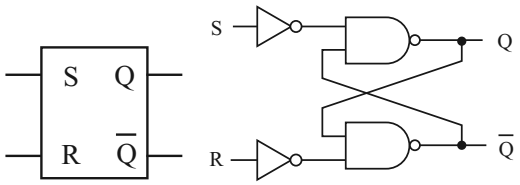


図 7.3　RS フリップフロップの真理値表，回路記号，及び，これを 2 入力論理ゲートで表した回路図.

JK フリップフロップ

JK フリップフロップは，クロック入力端子 (CLK) を持ち，クロック端子が 0→1 (あるいは 1→ 0) と変化する際に出力が変化する．その変化の仕方は，図 7.4 の真理値表のように規定されている．

J	K	Q	次の CLK 入力時の Q
0	0	0	0
0	0	1	1
0	1	–	0
1	0	–	1
1	1	0	1
1	1	1	0

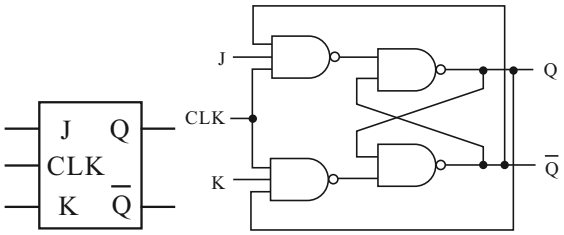
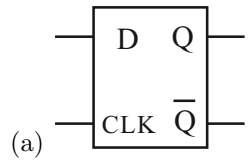


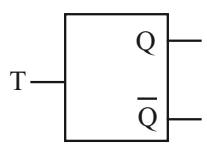
図 7.4　JK フリップフロップの真理値表，回路記号，及び，これを 2 入力論理ゲートで表した回路図.

D フリップフロップ，T フリップフロップ

D (データ) フリップフロップは，CLK 信号の立ち上がりで D 入力端子の値が出力され，以降，CLK が一旦下がって再び立ち上がるまでの間この値を保持する．T（トグル) フリップフロップは，T 入力端子の値が上下するたびに出力が反転する．これらの回路記号と真理値表を図 7.5 に示した．



D	CLK	Q
0	↑	0
1	↑	1
–	↓	Q (hold)



T	Q	Q
↓	0	0
↓	1	1
↑	0	1
↑	1	0

図 7.5　(a) D フリップフロップの回路記号と真理値表．(b) T フリップフロップの回路記号と真理値表.

JK FF による他の FF の表現

以上見てわかるように，紹介した FF は適当に配線，使用することで相互に入れ替えることが可能である．例えば，JK-FF を使って他の 3 つを表すと，図 7.6 のようになる．

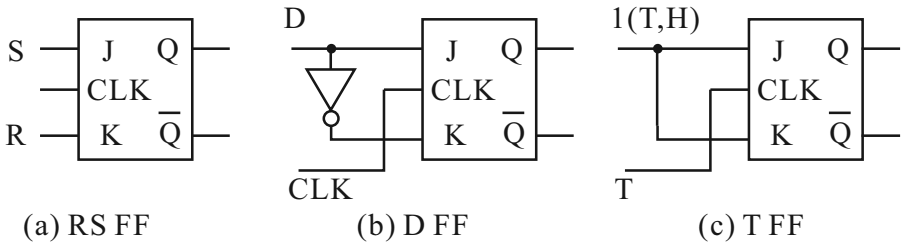


図 7.6　JK-FF を用いて，(a) RS-FF (クロック付)，(b) D-FF，(c) T-FF を構成する．(c) で 1 (T, H) とあるのは，1 の固定信号 (実際には電源の H(高電圧側)) を常時入力することを示す．

7.2.4　順序回路：カウンタ

カウンタは，入力される信号の変化の回数を出力するものである．

非同期カウンタ

図 7.7 に示したのは最も簡単なカウンタ回路であり，単に T-FF をシリーズで接続したものである．クロック入力信号が変化して 0→1 という変化が生じる度に 4 ビットの出力 2 進数が減少するダウンカウンタである．$\bar{Q}$ 側を使えば，アップカウンタにすることができる．このカウンタは，入力の変化が生じる度にその変化がゲートを 1 つづつ経て波のように広がってゆく．このことから，このような出力変化が同期していない非同期カウンタを「リップルカウンタ」とも呼ぶ．

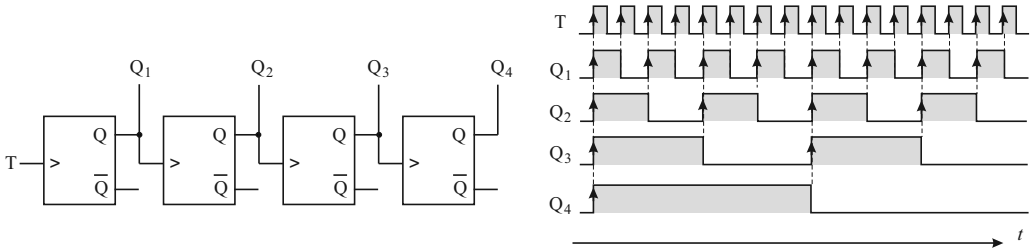


図 7.7　T-FF 4 つを用いた最も簡単な 4 ビット非同期 (リップル)2 進ダウンカウンタ．右はタイミングチャート．

同期カウンタ

リップルカウンタはリップルが広がっていく間は誤ったデータを出力していることになる．これに対して，出力を (もちろんゲート遅延時間のばらつき誤差範囲で) 同時に変化させたい場合，図 7.8 のような同期カウンタを用いる必要がある．このためには，カウント出力をデータとして予め用意し，D-FF を使ってクロック入力に同期して出力しラッチするのが簡単な方法である．

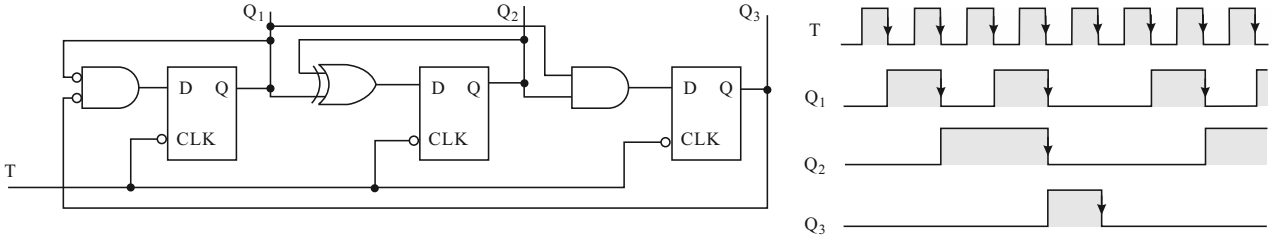
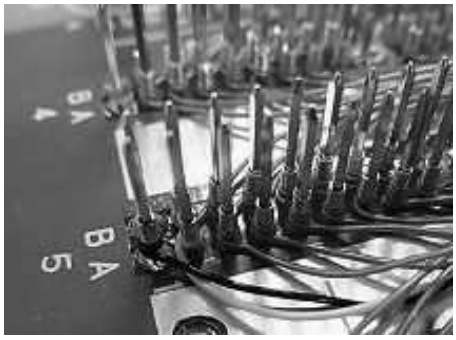


図 7.8　3 ビット同期アップカウンタ回路例とタイミングチャート．なお，CLK 信号入力に丸 (否定) が付いているのは，信号の立下りで D-FF が D 入力を Q 出力へラッチすることを示す．また，2 つの丸に and ゲートがついたゲートは，入力を a，b とすると，$\bar{a} \cap \bar{b}$ を表し，ド・モルガンの定理より $\overline{a \cup b}$ であるから，nor ゲートと同じである．

7.3 論理ゲートの実装



以上簡単な例で見たように，IC へのカプセル化によって回路図や配線は劇的に簡単になり論理 IC のピン間を回路図通りにつなげば良い，と考えられていた時期があった．このため，自作のディジタル回路では，左の写真のように長いピンを持つ IC ソケットに硬めの配線を巻き付けて結線するワイヤラッピング配線法が盛んに使用されていたこともあり，現在でもシミュレータで十分に拾いきれない問題を調べるため，テスト回路などでは使われている．しかし，すでに述べたように，高いクロック周波数での動作，また前回述べた EMC の観点などから，このような配線法はあまり行われなくなり，自作やテストも FPGA を用いることが多くなった．

以上は余談であり，本節の目的はこのように「カプセル化」されているディジタル IC 論理ゲートのアナログ的中身について簡単に見ておくことである．これは，カプセル-ブロック化した IC の様々な物理的規定がどのような原因で生じているかを理解するためである．図 7.9 に論理ゲートの回路構成法の代表である，TTL (transistor-transistor logic) と CMOS (complimentary metal-oxide-semicondcutor) による nand ゲートの構成例 (等価回路) を示した．いずれも 4 つのトランジスタまたは FET で構成されているが，その仕様のされ方はかなり異なっている．

(a) の TTL 構成では，入力にダブルエミッタトランジスタを用いている．これは挿入図にも示したように 2 つのトランジスタが並列になっているものとして動作を理解することでができる．従って，$A_{\rm in}$ または $B_{\rm in}$ が 0 (L, GND) であれば，トランジスタ Q_1 が ON になって電流が流れ，Q_2 を ON にする．Q_3 と Q_4 はプッシュプル回路を構成しており，Q_2 が ON の時は Q_3 が ON で Q_4 が OFF，Q_2 が OFF の時は Q_3 が OFF で Q_4 が ON になるように回路定数が決められている．以上から，入力が両方 H になった時のみ Q_1 が OFF になって出力が GND 側に振れ，nand 動作をすることがわかる．この回路から，TTL では入力レベルとして初段のトランジスタを ON/OFF することが求められること，出力は $V_{\rm CC}$ から GND まで振れるわけではなく，出力段のプッシュプルトランジスタによる電圧ドロップを考慮しなければならず，また，出力の負荷抵抗によって出力電圧が変化することも理解される．

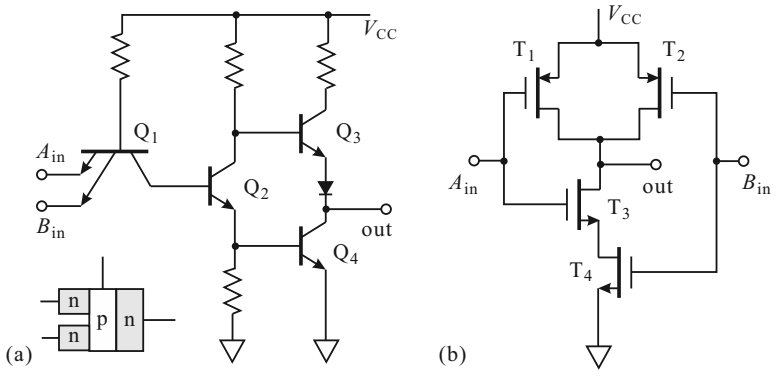


図 7.9 (a) TTL 構成による nand ゲート内部回路の等価回路例．挿入図はダブルエミッタトランジスタの層構成模式図．(b) CMOS 構成による nand ゲート内部回路の等価回路例．内向き矢印 MOS-FET は p チャネル，外向き矢印は n チャネル．(エンハンストモード)

これに対して，(b) の CMOS 構成は非常に簡潔である．complimentary という名前の通り，必ず p チャネル，n チャネル両方の MOS-FET を使用する．これは，全く同じ入力電圧に対して ON と OFF の逆の動作をするスイッチが使えることを意味し，このような入力に対して自在に $V_{\rm CC}$ 側と GND 側とをつなぎ替えることが可能である．(b) では，$A_{\rm in}$，$B_{\rm in}$ の H 信号に対して OFF になる p-MOSFET を $V_{\rm CC}$ に並列につなぎ，ON になる n-MOSFET を GND に直列接続している．このため，両方が H になった時のみ出力が GND に接続され，それ以外は $V_{\rm CC}$ に接続されて nand 動作をすることがわかる．MOS 回路は次段の入力抵抗が非常に高いのが普通であり，これに対して FET の ON 抵抗は無視でき，ON/OFF 電圧の $V_{\rm CC}$/GND からのずれは，FET の自己バイアス分で TTL に比べて大きな出力の振幅が取れることがわかる．もちろん，この素朴な回路では，電源側に接続した並列回路と接地側の直

列回路とが遷移領域で同時に ON となって筒抜けになる危険性があるなど，この周辺に様々な安全回路を設ける必要がある．

以上から，論理ゲートを動作させるための特徴的な電圧が，V_{CC} を 5V に設定した場合，図 7.10 のようになることが理解される．ただし，特に TTL の場合，適当な負荷や電源の周囲配線をすることが前提となっている．また，V_{CC} を 5V から大きく変更することは，TTL の場合，通常の半導体物質パラメタから困難であることも理解されるであろう．

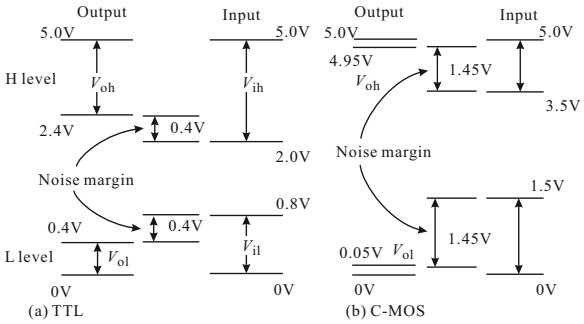


図 7.10　標準論理ゲート動作の電圧ダイアグラム．(a) TTL，(b) C-MOS

歴史的にみると，IC 化され論理ゲートして盛んに使用され始めたのは TTL が先であり，圧倒的な利用範囲を持っていたため論理信号の H，L が 5V，0V というのが標準となった．初期の頃の CMOS IC は，動作のためにむしろ高い電源電圧を必要とし，TTL 回路と接続のためにはレベル変換回路を用意したりした．また，やはり初期は CMOS の動作速度は TTL を改良した LS 型，ALS 型に及ばず，PC の周辺回路等，ほとんど TTL で組まれていた．しかし，クロック周波数が高くなり LSI の集積度が高くなるにつれて，素子間の距離をできるだけ短くすることが非常に重要になり，ゲートあたりの素子数が少なく，電流がゲートの帯電/放電時のもの程度で消費電力が極めて小さい CMOS が次第に有利となった．素子特性も単ゲート化によってむしろ改善され 5V より低い電圧でも十分動作するようになり速度も向上した．3.3V や 2.5V を H レベルとする回路も増え，最近では京などのスーパーコンピュータの CPU も CMOS となり，現在のところ論理ゲートの世界は CMOS に完全支配されてしまったように見える．

ただし，CMOS も含めて論理演算そのものが現在様々な物理的壁に当たっており，並列計算などのソフトウェア手法で全体としての行き詰まりを回避しているのが現状であるから，ハードウェア側にも何か大きなフレークスルーがある可能性はある．

7.4　論理演算の回路化と簡単化

アナログ回路が力学系など，物理系の表現と見ることができたのに対し，ディジタル演算回路は，論理式の具体的表現と見ることができる．従って，論理式が演算を代数的に行うことで簡単化されれば，回路を簡単化することになる．その代表的方法を説明する．また，順序回路は「状態」概念が入ってくるため，これに対応する論理式側の取り扱い概念が必要となる．その 1 つ (ツール) であるダイアグラムを紹介する．

7.4.1　カルノー図

A，B を論理値として論理式

$$Y = A \cdot B + A \cdot \overline{B} + \overline{A} \cdot B \tag{7.1}$$

を考える．$\cdot$ は and，$+$ は or を表す．論理値においては，$A + A = A$ であったことから，右辺に $A \cdot B$ を加えて変形すると

$$\begin{aligned} Y &= A \cdot B + A \cdot B + A \cdot \overline{B} + \overline{A} \cdot B \\ &= A \cdot (B + \overline{B}) + B \cdot (A + \overline{A}) = A + B \end{aligned} \tag{7.2}$$

となり，単なる or 演算であったことがわかる．$A \cdot B$ を加えるような論理演算は式だけでは思いつきにくく，真理値表を睨むなどの非機械的手法がむしろ有効である．これを比較的機械的に行う手法が**カルノー図** (Karnaugh map) の手法である．

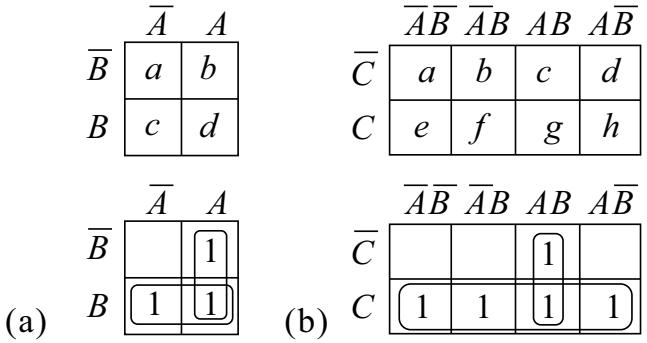


図 7.11　カルノー図の例．(a) 2 論理変数に対するカルノー図．下は or 論理演算の場合．(b) 3 論理変数に対するカルノー図．下は，$\overline{A} \cdot \overline{B} \cdot C + \overline{A} \cdot B \cdot C + A \cdot B \cdot C + A \cdot \overline{B} \cdot C + A \cdot B \cdot \overline{C}$ の場合．

カルノー図とは，ある論理式に対し図 7.11 のようにそれを構成する論理変数のすべての場合を 2 次元に並べて書いたものである．マスの中には出力が 1 (T) となる場合のところに 1 を入れるようにする．3 変数の場合，2 変数ずつをまとめて書くと図 7.11(b) のように 2 次元にしておくことができる．この時，隣のコラムを作る際に 2 変数同時に否定を取ることはせず，1 つづつ取って場合を潰すことが必要である．このように論理変数の数が増加すると次第に図が描きにくくなる．

式 (7.1) の場合をカルノー図に描くと図 7.11(a) の下の図になる．隣接する「1」を□で括ると，2 つの括りができることがわかる．すなわち，このカルノー図が表す論理式は，この 2 つの「括り」が表す論理式の or を取ったものである．

$$Y = A \cdot \overline{B} + A \cdot B + \overline{A} \cdot B + A \cdot B = A + B \tag{7.3}$$

と，当然 (7.2) の結果が得られる．図 7.11(b) の下の図は

$$Y = \overline{A} \cdot \overline{B} \cdot C + \overline{A} \cdot B \cdot C + A \cdot B \cdot C + A \cdot \overline{B} \cdot C + A \cdot B \cdot \overline{C} + A \cdot B \cdot C = A \cdot B + C \tag{7.4}$$

である．

以上見てわかるように，カルノー図による簡単化の原理は

$$A \cdot B + A \cdot \overline{B} = A \tag{7.5}$$

という簡単なものであるが，図 7.11 のように 2 次元に並べて隣接する 1 を括ることでこのようにして消せるペアを組織的に見つけられる，というところが利点である．なお，「隣接する 1」を探す際，行や列の端同士は隣接している，と見なければならないので注意が必要である．例えば，図 7.11(b) の上の図で a と d や e と h は隣接している．

7.4.2　クワイン・マクラスキー法

この簡単化を更に機械的手続きにすることを考える．このためには，最初に与えられた論理式の形は，簡単化を標準化するためにはむしろ障害になることが多く，カルノー図のように一旦真理値表で表してしまい，そこから一定の標準的手続きにより論理式に戻すことにする．

論理式 $Y = f(A_1, A_2, \cdots, A_n)$ を考える．真理値 0，1 に対して，各論理変数を与える関数 g_i を

$$g_i(0) = \overline{A}_i, \qquad g_i(1) = A_i \tag{7.6}$$

と定義する．各変数またはその否定をすべて 1 つずつ含む論理式 (項) を**標準項** (canonical term) という．うち，論理和 + を区切りとして分解される「項」の数が最も少ないものを最小項と呼ぶが，最小項は当然項数 1 個であるから，結局最小項とは，$\prod_{i=1}^{n} g_i(a_i)$ で表される．$a_i = 0$ または 1 で，$\prod$ は論理積を表す．Y を真理値表で表す時，ある 1 行の入力項は 0 または 1 の n 個の真理値列であるから，これを $\{a_i\}$ と見ると，真理値表の各行に 1 個の最小項

が対応することがわかる．そこで，$Y=1$ を与える $\{a_i\}$ をすべてリストアップして指数 j を付け，行列の形で $\{a_{ij}\}$ と表すと，Y の論理式としてこれらで指定される最小項をすべて論理加算したもの

$$Y=\sum_j\prod_{i=1}^{n}g_i(a_{ij}) \tag{7.7}$$

を考えることができる．このように最小項の論理和での論理値の表現を**主加法標準展開** (principal disjunctive canonical expansion) と呼ぶ．

主加法標準展開から (7.5) を使い冗長項を圧縮していく．この時，最小項を作るために考えた $\{a_i\}$ を並べたものを２進数として取り扱う方法を特にクワイン・マクラスキー法 (Quine-McCluskey algorithm) と呼んでいる．これら最小項を表す２進数を，その中に出てくる「1」の個数で分類する．２進数表示での (7.5) は，0 と 1 とを取るある桁以外は同じ２進数の間でこの桁を消去する式であるから，圧縮はこの分類で１つだけ異なるグループ間で行われる．消去した桁は_(アンダースコア) として並べることで第１次圧縮リストができる．再度１の個数で分類，圧縮を可能な限り繰り返す．

こうして簡単化された論理式について，各項 (「主項」と呼ぶ) を行指数とし，元の主加法標準展開に含まれる最小項を列指数とする (反対でも良い) 表 (主項図) を作る．最小項の中に何らかの形で主項が含まれている欄に印をつける (○とする)．例えば，主項が __11 であれば，0011，0111，1011，1111 の 4 つがそれに当たる．論理的には「これらの最小項を主項が包含する」ことになる．この表をまず縦に見て，○が１回だけ現れる最小項の○を◎に変更する．次に横に見て，◎がついた主項 (必須項) の行に◎になっていない○があればこれをすべて◎に変更する．最後に○しか付いていない主項が省略候補である．すべての最小項をカバーするように必要な主項があれば残し，後を省略することで簡単化が終了する．

7.4.3　状態遷移図

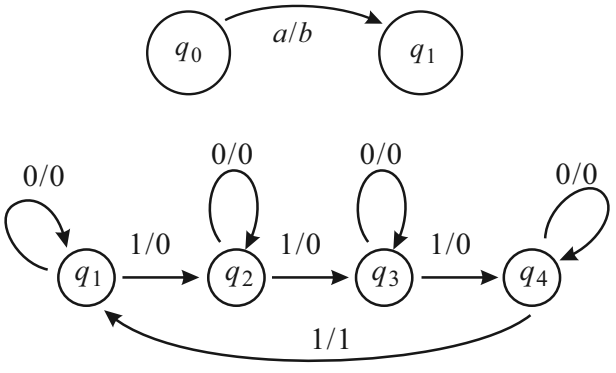


図 7.12　状態遷移図．(a) 概念図．(b) 4 進カウンタ．

順序回路を設計する際に，ダイアグラムとして助けになるのが**状態遷移図** (state transition diagram) である．これは図 7.12(a) のように，入力 a に対して回路の状態が q_0 から q_1 へと遷移し，b が出力されるということを表している．

これを使って，7.2.4 節で見たカウンタ回路の設計手順を追ってみる．簡単のため，2 ビット (4 進) カウンタとする．カウント入力が入る度に状態が変化し，初期状態から 4 つ目のカウント入力で 1 を出力して初期状態に戻るから，状態は全部で $q_1,\cdots,q_4$ の 4 つである．図 7.12(b) のようにこの 4 つの状態を並べて書き，入力 $x=0$ の場合は状態が変化せず出力も 0 なので，0/0 でループを描いて元の状態に戻り，1 に対しては状態が 1 シフトし，$q_4\to q_1$ の初期値に戻るときのみ 1 が出力されるので 1/1，それ以外は 1/0 となる，ということで，図 7.12(b) のような状態遷移図が描かれる．

2 ビットであるから T-FF を 2 個使用して実現することにする．これら２個の出力を，状態 q_n に対してそれぞれ，$Q_n^{(1)}$，$Q_n^{(2)}$ と書く．n は 4 進で回る，すなわち $4+1=1$ とする．状態遷移表をカルノー図の形にするため，$Q_{n+1}^{(1)}$ と $Q_{n+1}^{(2)}$ とに分けて描くと次のようになる．ただし，最初にビットが上がる方を (2) の T-FF としている．($Q_{n+1}^{(i)}$ を決めるための論理変数が，x，$Q_n^{(1)}$，$Q_n^{(2)}$ の３つ存在するため，図 7.11(b) を縦にした図になっている．

カルノー図であるから，□で隣接する 1 を囲い，簡単化により次のような漸化式を得る．

$$Q_{n+1}^{(1)}=\overline{x}\cdot Q_n^{(1)}+Q_n^{(1)}\cdot\overline{Q}_n^{(2)}+x\cdot\overline{Q}_n^{(1)}Q_n^{(2)}, \tag{7.8a}$$

$$Q_{n+1}^{(2)}=\overline{x}\cdot Q_n^{(2)}+x\cdot\overline{Q}_n^{(2)}. \tag{7.8b}$$

$Q_n^{(1)}$	$Q_n^{(2)}$	$Q_{n+1}^{(1)}$ x 0	1
0	0		
0	1		1
1	1	1	
1	0	1	1

$Q_n^{(1)}$	$Q_n^{(2)}$	$Q_{n+1}^{(2)}$ x 0	1
0	0		1
0	1	1	
1	1	1	
1	0		1

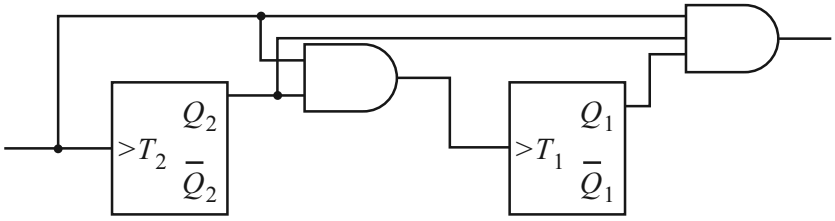


図 7.13　設計した 4 進 (2 ビット) カウンタ

また，一般に FF については，入力 x で表される論理式 α，β を用いて

$$Q_{n+1} = \alpha Q_n + \beta \overline{Q}_n \tag{7.9}$$

という，**特性方程式** (characteristic equation) と呼ばれる式が成立する．実際，図 6.5(b) の T-FF の真理値表から，$\alpha = \overline{T}$，$\beta = T$ として，(7.9) が成り立っていることがわかる．T-FF の (1)，(2) に対応して $\alpha_{1,2}$，$\beta_{1,2}$ と添字をつけると，(7.8a)，(7.8b) より，

$$\alpha_1 = \overline{x} + \overline{Q}_n^{(2)} = \overline{x \cdot Q_n^{(2)}}, \qquad \beta_1 = x \cdot Q_n^{(2)}, \tag{7.10a}$$

$$\alpha_2 = \overline{x}, \qquad \beta_2 = x \tag{7.10b}$$

である．以上から，図 7.13 の回路図がほぼ自動的に描かれる．

付録 J：高速フーリエ変換

まず，「時間間引き」アルゴリズムと呼ばれるものを紹介しよう．離散フーリエ変換 (6.78) の N を偶数に取る．また，添え字 n が表す数を偶数，奇数に分けて次のように書き直す．

$$F_k = \sum_{n=0}^{N/2-1} f_{2n} W_N^{2nk} + \sum_{n=0}^{N/2-1} f_{2n+1} W_N^{(2n+1)k} = \sum_{n=0}^{N/2-1} f_{2n} W_{N/2}^{nk} + W_N^k \sum_{n=0}^{N/2-1} f_{2n+1} W_{N/2}^{nk}. \tag{J.1}$$

2 番目の変形では，W_N の性質より，$W_N^{2nk} = W_{N/2}^{nk}$ となることを使った．$\{F_k\}$ を得るためには各 $k = 0, \cdots, N-1$ について (J.1) を計算するが，このうち，$k = N/2, \cdots, N-1$ については，$k \to k + N/2$ と置換え，$W_{N/2}^{nN/2} = 1$，$W_N^{k+N/2} = -W_N^k$ を使うと，

$$F_{k+N/2} = \sum_{n=0}^{N/2-1} f_{2n} W_{N/2}^{n(k+N/2)} + W_N^{k+N/2} \sum_{n=0}^{N/2-1} f_{2n+1} W_{N/2}^{n(k+N/2)} = \sum_{n=0}^{N/2-1} f_{2n} W_{N/2}^{nk} - W_N^k \sum_{n=0}^{N/2-1} W_{N/2}^{nk} \tag{J.2}$$

と書ける．ここで，

$$X_k = \sum_{n=0}^{N/2-1} f_{2n} W_{N/2}^{nk}, \quad Y_k = \sum_{n=0}^{N/2-1} f_{2n+1} W_{N/2}^{nk} \quad (k = 0, \cdots, N/2-1) \tag{J.3}$$

と置くと，これらはいずれも $N/2$ 点の離散フーリエ変換 (DFT) と見ることができる．元の $\{F_k\}$ はこれらを用いて

$$\left.\begin{aligned} F_k &= X_k + W_N^k Y_k, \\ F_{k+N/2} &= X_k - W_N^k Y_k, \end{aligned}\right\} \quad k = 0, \cdots, N/2-1 \tag{J.4}$$

と表される．

以下，N が 2 のべき乗として $N = 2^q$ (q は整数) という数であったとすると，この手順を繰り返せば，最終的には 1 点の DFT を N 個求め，$(N/2)\log_2 N$ 回の積を実行することに帰着する．1 点の DFT は 1 の乗算で計算不要であり，計算量は $(N/2)\log_2 N$ 回の乗算で，元の DFT の N^2 回の乗算に比べると，大幅に計算量を減らしたことになる．逆フーリエ変換も，同様に「周波数間引き」を用いることで行うことができる．