（1） 仮数部を p だけ左(右)にシフトする。

（2） 指数部の値を p だけ減じる(加える)。

仮数部が純小数(整数)でない一般の浮動小数点数も同様に，シフト操作によって正規化することができる。

（c） 指数の数表現

コンピュータ内部での浮動小数点数表現2進数の指数(部)は固定小数点数と同じように，① 符号-絶対値表現；② 1の補数表現；③ 2の補数表現；のいずれかで表す。

ここで，図3.10に例を示すように，浮動小数点数表現の指数部に**バイアス**(bias，げた履き)**値**という整数定数を加えて，指数部には正整数だけを指定するようにしてしまう。これを**バイアス表現**あるいは**げた履き表現**という。バイアス値としてはそのコンピュータ内部で表現可能な最小指数(負数)の絶対値を選ぶ。これによって，指数(部)はゼロ以上の正整数で表現でき，符号ビットや補数表現は不要となる。コンピュータから演算結果を取り出すときなどの必要時に指数部からバイアス値を減じて実際値に戻せばよい。

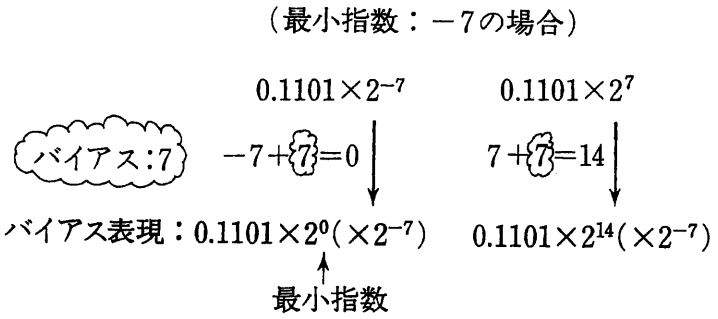

図 3.10 バイアス表現の例

指数(部)をバイアス表現(バイアス値は B)する場合，仮数 m，指数 e で2進数浮動小数点数表現した実数 R の10進数値は，

$$R=m\times 2^{e-B} \tag{3.27}$$

である。たとえば，指数部が8ビットで，それで表す最小指数が −127 ならば，式(3.27)において $B=127$ で，

$$R=m\times 2^{e-127} \tag{3.28}$$

となる。

（d） 浮動小数点数の範囲と精度

式(3.25)で表す正規化した浮動小数点数表現 r 進数の仮数 m

の範囲は，m を符号ビットなしの正の純小数(正規化による)で表現する場合

$$r^{-1} \leqq m < 1 \tag{3.29}$$

である。特に，式(3.26)と図3.7で表す正規化した浮動小数点数表現2進数の場合には，

$$0.5 \leqq m < 1 \tag{3.30}$$

である。r 進数，2進数いずれの場合も，仮数 m の範囲は仮数部の長さ p にかかわらず一定である。

浮動小数点数表現の仮数は“有効数字(有効桁，有効ビット)”を表している。2進数の場合，p ビットの仮数部では，2^p (概数，厳密には仮数の数表現で異なる)個の仮数が表現できる。仮数の範囲は式(3.29)や(3.30)で示したように一定であるから，仮数部の長さは一定の範囲にある仮数の個数，すなわち，それで表す浮動小数点数そのものの精度を決める。(b)で述べた“正規化”とは，“仮数部の有効数字(のビット数)すなわち精度を最も高く(最大に)する”ことでもある。

一方，浮動小数点数表現した実数 R そのものの範囲はその指数 e で決まる。指数部の長さを q ビット(図3.7参照)とし，e を符号ビットなしの正整数で表現する場合，e の範囲は $0 \leqq e \leqq 2^q - 1$ となる。このとき，実数 R の範囲は，式(3.30)によって，

$$0.5 \leqq R < 2^{2^q - 1} \tag{3.31}$$

となる。

仮数部の長さ p が24ビット，指数部の長さ q が8ビットの浮動小数点数表現で表す実数 R の範囲は，

$$0.5 \leqq R < 2^{255} \tag{3.32}$$

である。符号ビットなしで32ビット(1ワード)すべてを整数部とする固定小数点数で表現する正整数 N の範囲

$$0 \leqq N < 2^{32} \tag{3.33}$$

と比べると格段に広い(大きい)。

浮動小数点数表現の精度は仮数部の長さによって決まる。**単精度**とは仮数部と指数部とを合わせて1ワード(普通は32ビット)の浮動小数点数表現をいう。2ワード以上の場合は**多倍精度**(2ワードの場合は単に**倍精度**)という。

浮動小数点数表現においては，指数部の長さによって決まる範囲と仮数部の長さによって決まる精度との間にトレードオフ関係がある。すなわち，指数部を長く(短く，以下カッコ内に対応)すれば範囲が広く(狭く)なるが，一方で，仮数部が短く(長く)なって精度が低く(高く)なる。

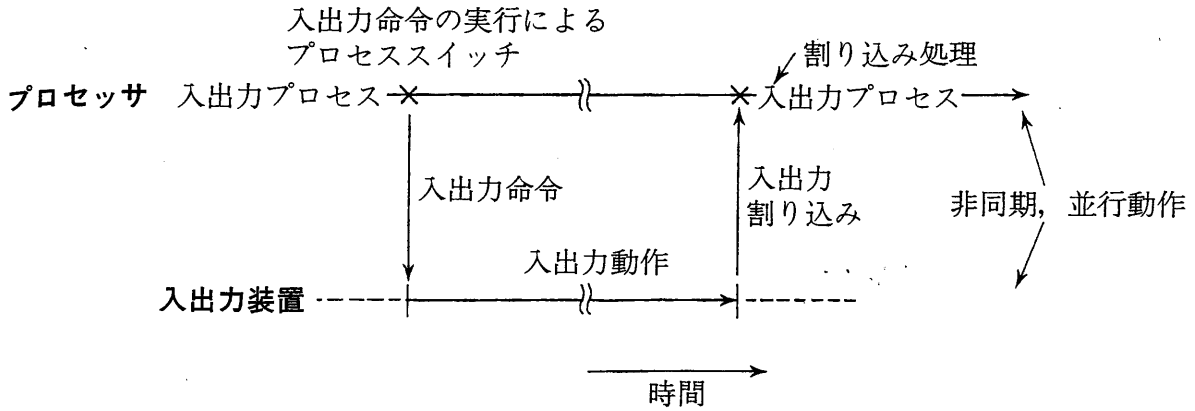

図 5.26 入出力割り込み

うマシン命令を実行すれば，そのタイミングで同期はとれる。これとは逆に，プロセッサからの入出力動作指令によって（プロセッサとは非同期に）動作している入出力装置がプロセッサに自分の状態を知らせるための手段が**入出力割り込み**である。

図 5.26 に示すように，入出力割り込みは入出力動作の完了や異常などの入出力装置の状態をプロセッサ側に通和するためにある。これによってプロセッサやメインメモリと入出力装置とが独立に並行して動作可能となる。

入出力装置を制御する（入出力動作を行う）プロセスを“入出力プロセス”という。プロセッサで実行する入出力プロセスを管理・制御する OS と入出力装置の動作（入出力動作）との関係を時間経過にしたがって説明すると次のようになる。

まず，プロセッサが入出力命令を実行すると，OS は，① 入出力命令を実行した入出力プロセス（＊）を実行中状態から実行待ち状態にする；② プロセススイッチによって別のプロセスを実行中にする（切り替える）；を行う。一方，入出力装置は，③ プロセッサからの入出力指令によって入出力動作を開始する；④ 入出力動作を完了するとプロセッサに対して入出力割り込みを起こしてそれを通知する；を行う。プロセッサ(OS)は，⑤ 入出力割り込みを受け付けて，その割り込み処理を始める；⑥ 入出力割り込み（という事象）を待っていた入出力プロセス（＊）を実行待ち状態から実行可能状態にする（ウェイクアップする）；を行う。③ と ④ の間の入出力動作中には，プロセッサは入出力装置とは非同期に（独立して）別のプロセスを実行できる。

入出力割り込みは“入出力装置が非同期に並行動作するプロセッサに同期をとってもらう”機能の実現である。入出力割り込みを用いた入出力制御機能の詳細については 8.2 節で述べる。

$$\bar{\bar{N}}=\bar{N}+1 \tag{3.20}$$

となる。ただし，式(3.20)では，$\bar{N}$ と $\bar{\bar{N}}$ は10進数値で表し，それぞれの符号ビット(MSb)もそれぞれの数値部と連結・一体化して考える。式(3.20)より，2進数表現したある整数 N の2の補数 $\bar{N}$ (2進数表現)は“N の1の補数 $\bar{\bar{N}}$ に $+1$ (1を加算)する”操作で得ることができる。

2進数表現した整数 N の2の補数 $\bar{N}$ (2進数表現)を求める手順(前述の(1)～(4))を式(3.20)にしたがって書き直すと，次のようになる。

(1) N の各ビットを反転($0 \Leftrightarrow 1$)して，まず，1の補数 $\bar{\bar{N}}$ を得る。

(2) $\bar{\bar{N}}+1$ ($\bar{\bar{N}}$ に1を加算する)によって2の補数 $\bar{N}$ を得る。

2の補数(表現)には，1の補数(表現)と比較して，① 加減算を行うときに補正は不要である(6.1.1項(f)で詳述)；という長所と，② いったん1の補数を得て，それに $+1$ の加算操作を行って2の補数を得るので，2の補数化機構では加算器が必要となる(6.1.2項(f)参照)；という短所がある。現代のコンピュータでは，②の短所よりも①の長所を重視して，“固定小数点数による負数の数表現は2の補数(表現)で行う”のが普通である。

(e) 固定小数点数の範囲と精度

n ビットの整数部(符号ビットを含む)と m ビットの小数部をもつ固定小数点数表現(2進数表現)では，補数表現を使うと，次の式(3.21)と(3.22)の範囲にある(すべてではなく有限個の)実数 R を表現できる。

(1) 1の補数表現による実数 R の範囲は

$$2^{-m}-2^{n-1} \leqq R \leqq 2^{n-1}-2^{-m} \tag{3.21}$$

となる。式(3.21)の範囲に，1の補数による固定小数点数表現で表せる2進実数 R は $(2^{n+m}-1)$ 個ある。なお，1の補数表現では“$+0$”と“-0”の表現が異なるが，数学的な違いはない。

(2) 2の補数表現による実数 R の範囲は

$$-2^{n-1} \leqq R \leqq 2^{n-1}-2^{-m} \tag{3.22}$$

となる。式(3.22)の範囲に，2の補数による固定小数点数表現で表せる2進実数 R は 2^{n+m} 個ある。

(1)(2)いずれの固定小数点数表現の場合でも，“精度は 2^{-m} で一定”である。

また，n ビットの2進整数の固定小数点数表現(整数表現)では，補数表現を使うと，次の式(3.23)と(3.24)の範囲にある(すべてのかつ有限個の)整数 N を表現できる。

(1) 1の補数表現による整数 N の範囲は

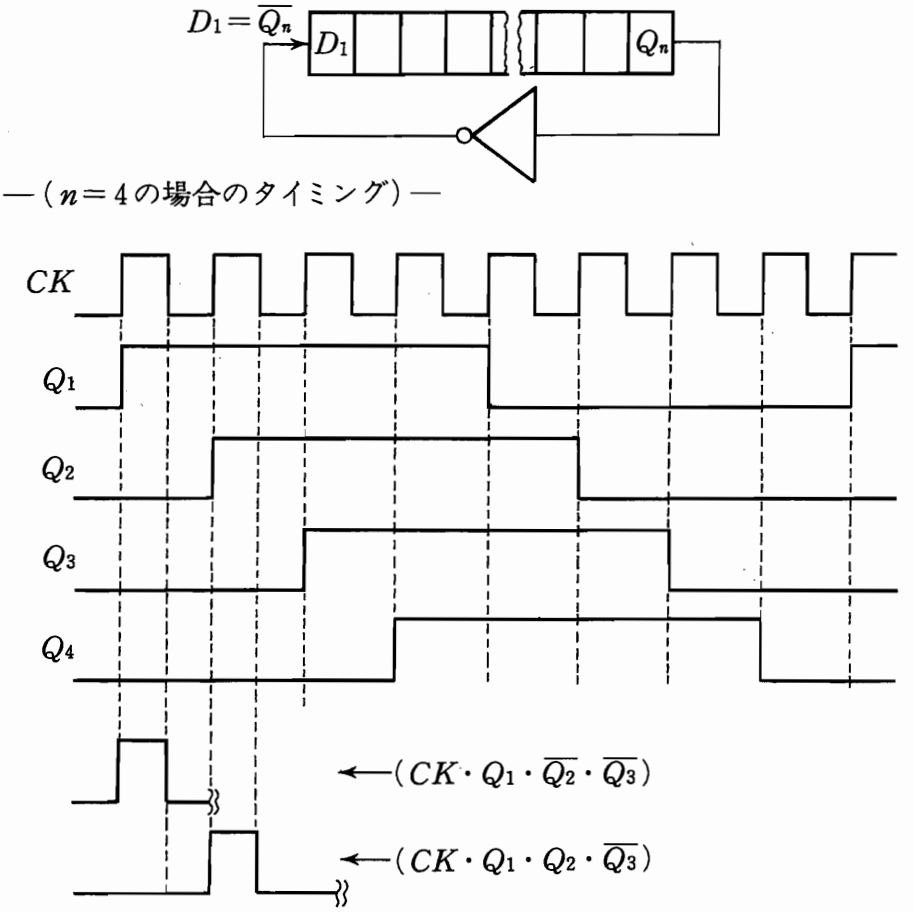

図 4.20　nビットジョンソンカウンタの構成と動作タイミング（n=4 の場合）

4.2.4　順序回路による基本ハードウェア機構の構成例

実際にコンピュータのハードウェア機構を構成する基本順序回路を紹介してみよう。

（a）　タイミング生成回路

コンピュータ（のハードウェア）の動作を制御（同期）するマスタ（原）クロックパルスから位相のずれたタイミングパルスを生成する回路を**タイミング生成回路**という。**タイミングパルス**はクロック周波数を分周した周波数をもつが，そのパルス幅はマスタクロックと同じである。

基本的なタイミングパルスは，前の 4.2.3 項 (d) で述べたジョンソンカウンタなどを利用して次のような手順で作ることができる（図 4.21 参照）。

（1）　周波数 f，デューティ比（“1”を示すパルスの時間と“0”を示すそれとの比）1：1のマスタクロックパルスを$2n$分周し，周波数$\frac{f}{2n}$，デューティ比1：1のパルスを作る。

（2）　このデューティ比を1：$(4n-1)$にし，マスタクロックのパルス幅と

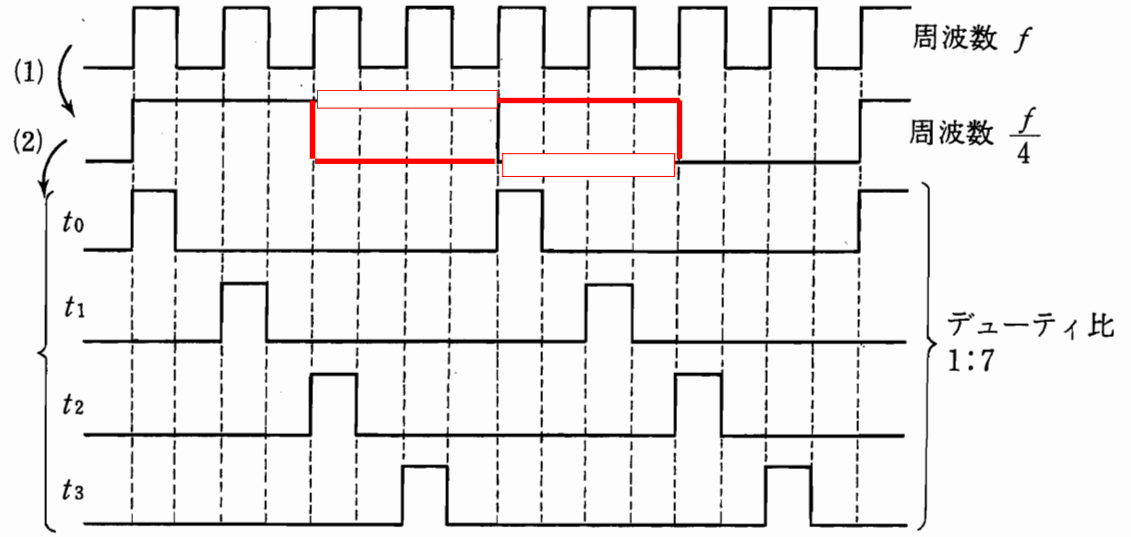

図 4.21 タイミングパルスの生成手順 ($n=2$)

同じにする。

マスタクロックから作るタイミングパルスはコンピュータの制御機構 (特に同期式制御機構，5.1.3 項 (c) 参照) のタイミング合わせ (同期動作) に使う。

周波数が f のマスタクロックの周期 (cycle；1 クロックサイクル) は $\frac{1}{f}$ 秒であり，これを**マシンサイクル** (machine cycle) という。マシンサイクルはコンピュータの動作を同期させるために使う最小単位時間 (間隔) である。

(b) パルスエッジ検出回路

クロックに同期していないパルスのエッジ (edge；立ち上がりと立ち下がり) を検出する回路を**パルスエッジ検出回路**という。非同期信号の同期化などのコンピュータの基本ハードウェア機構 (特に制御機構) の構成には不可欠である。

パルスエッジ検出回路は，図 4.22 に示すように，2 個の D フリップフロップを使用して構成できる。

(c) アービタ

“あるハードウェア機構や装置へ複数のアクセスが同時に生じる”ことを“アクセス競合”という。たとえば，メインメモリへプロセッサと入出力装置とが同時にアクセスする場合や，プロセッサ内の ALU とレジスタがバスを同時に使用する場合など，ハードウェア利用の多重化を図っている現代のコンピュータではアクセス競合が発生する回数は非常に多い。このアクセス競合をある戦略で調停し解決する制御機構を**アービタ** (arbiter，調停機構) という。アービタは順序回路の一種であり，非同期に生じる信号を順序付けする。