計算機アーキテクチャ　80－14
（１９９０．１．２６）

ダイナミックRAM動的リフレッシュ法の応用

松井　祥悟
神奈川大学理学部情報科学科

ダイナミックRAMはリフレッシュが必要である．リフレッシュのオーバーヘッドを小さくすることが，システム全体の処理速度の向上につながる．

ダイナミックRAM動的リフレッシュ法（DDR法）は，従来法の1/nの周期で動作する．このリフレッシュ法は，従来法における不要なリフレッシュ動作を検出し，省略することで，リフレッシュ動作の回数を減らす．リフレッシュ動作の回数はCPU等からのダイナミックRAMの使用状況により動的に決まる．最適条件下の動作時ではリフレッシュ動作は行なわれない．最悪の条件でも，リフレッシュ動作の回数は従来法と同じである．

このリフレッシュ法は既存のデュアルポートRAMを用いて簡単に実装できる．また，専用素子を用いると動作アルゴリズムはさらに簡単になる．

DRAM素子のビジー率が上昇した場合の試算では，ビジー率が現在の4倍になっても，メモリ参照の頻度がある程度見込める場合には，DDR法を用いたシステムのシステムオーバーヘッドは，従来法を用いた現在のシステム程度に収まることがわかった．

Application of Dynamic Refresh Method

Shogo MATSUI

Depertment of Information and Computer Science,
Faculty of Science,Kanagawa University

2946 Tuchiya,Hiratsuka-shi,
Kanagawa 259-12,Japan

Dynamic random access memories (RAM) are widely used in computer systems because of its large capacity and low prices. Dynamic RAM must be refreshed periodically to avoid data loss. Refreshing dynamic RAMs causes the computer system to extend the processing time. Effective refresh method is required.

We propose DYNAMIC REFRESH METHOD for refreshing dynamic RAMs in this paper that is based on fact that the rows of the memory cell array are refreshed at normal access by CPU.

The method operates periodic at 1/n intervals of the ordinary method. The number of rows refreshd in one cycle vary dynamically, that is few when a normal access occurs efficiently and under the worst condition that is the same as tha ordinary method's.

1．まえがき

ダイナミックRAM（以下DRAM）のメモリセルは有限時間しかデータの保持ができない．従って，周期的にメモリをリフレッシュする必要がある．DRAMのリフレッシュはメモリセルアレイの行をアクセスすることにより行う．この動作は，CPU等がそのDRAMに対して通常の読み書き動作（通常アクセス動作）を行っていないときに行わなければならない．

従来のリフレッシュ法は一定の周期で動作し，1周期の動作でメモリセルアレイのすべての行に対してリフレッシュを行う．一般に，この動作は，CPU等の動作を止めて行う．従って，この動作に要する時間はオーバーヘッドとなり，システム全体の処理時間を延長する．

CPU等の動作を停止せずにリフレッシュ動作を行うように実装時に回路を工夫することで，従来のリフレッシュ法のオーバーヘッドは小さくなる[2]．

一方，リフレッシュ法そのものを改良し，リフレッシュ動作の回数を少なくするという考え方がある．これは，「DRAMに対する通常アクセス動作において，その時使用したメモリセルアレイ内の行に対してリフレッシュが行われる」という事実に基づく．従来のリフレッシュ法は，この通常アクセス時のリフレッシュ作用を考慮しておらず，不要な行に対するリフレッシュ動作が含まれている．このリフレッシュ不要な行を検出し，必要な行に対してだけリフレッシュを行えば，リフレッシュによるオーバーヘッドは小さくなる．

リフレッシュ動作をこのように制御するためには，リフレッシュを行う機構はメモリセルアレイのすべての行について，(1)その行に対して最後に行われたリフレッシュ動作または通常アクセス動作が行われた時刻を記憶し，(2)その時刻からの経過時間によりリフレッシュ動作を起動しなければならない．

(1),(2)の操作は，個別の行に対しては簡単な手法で実現できる．しかし，この操作はすべての行に対して並列に行わなければならない．これを個別の行の手法を単純に並列化した方法で実現しようとすると，複数のタスクを制御することになり，複雑なものとなる．

これを簡単に実現するための方法についての提案はあるが[1]，上記(1),(2)を完全に行うものではなく，効率は非常に悪い．

本論文ではDRAMのリフレッシュ周期の1/nの周期で動作する動的リフレッシュ法を提案した．この方法では，(1),(2)の時間的な操作を周期的動作によって実現している．これにより，アルゴリズムは簡潔になり，簡単な手法で実装できる．

2．動的ダイナミックRAMリフレッシュ法

本論文では，DRAM固有の定数であるリフレッシュ周期（refresh interval）を*Tri*，リフレッシュサイクル（refresh cycles）を*Nrc*と表す．また，次の用語を用いる．通常アクセス動作とはCPU等によるDRAMに対する通常の読み書き動作を言う．リフレッシュ動作とはメモリ制御装置などによるDRAMに対するリフレッシュ動作を言う．ロウアクセス動作とは，通常アクセス動作またはリフレッシュ動作を言う．

次の方法を動的ダイナミックRAMリフレッシュ法（DDR法）と呼ぶ．

DDR法の定義

- メモリセルアレイの1つの行に対するリフレッシュ動作を行うかどうかの検査（これをリフレッシュ検査と呼ぶ）は*Tdi*ごとに行う．ただし，$Tdi \leqq Tri / Ndv$，*Ndv*は2以上の整数．
- ある1つの行*j*に対する*i*回目のリフレッシュ検査の時刻をt_iとすると，次のリフレッシュ実施条件を満たしている場合にだけリフレッシュ動作を行う．

  [リフレッシュ実施条件]

  $t_{i+Ndv+1} \leqq t < t_i$を満たす時刻*t*において，行*j*に対するロウアクセス動作が起こらない．

DDR法の*Tdi*を動作周期と呼ぶ．また，*Ndv*を分割数と呼ぶ．DDR法は分割数*Ndv*によって動作効率が変わる．以下，$Ndv = n$（$n \geqq 2$）の場合のDDR法を*n*型DDR法と呼ぶ．

$Ndv = 1$の場合（DDR法には含まれない）は，[リフレッシュ実施条件]が常に真となり，従来のリフレッシュ法の動作となる．

$Ndv = 2$の動作例を図1に示す．これは，メモリセルアレイの1つの行のリフレッシュ状況を時間軸上に示したものである．$t_0, t_1, \ldots$はリフレッシュ検査を行う時刻を示す．

t_0においてリフレッシュ動作（refresh1）を行い，t_0からt_2までの間は通常アクセス動作（図1の▲）が行われないとする（図1のa）．refresh1というロウアクセス動作がt_0で行われているため，t_1のリフレッシュ検査では[リフレッシュ実施条件]を満たさず，リフレッシュ動作は行わない．t_2のリフレッシュ検査では，これを満たし，リフレッシュ動作を行う．

このリフレッシュ動作の間隔（$t_2 - t_0$）は従来のリフレッシュ法の動作間隔と等しい．以後，同様に通常アクセス動作が全く行われない場合は，従来のリフレッシュ法と同じ動作を行うことになる．

$t_2 + t'$（$t' \leqq Tdi$）において通常アクセス動作が行われても，次のリフレッシュ動作はt_4で行われる．一方，$t_4 + t''$（ただし，$Tdi \leqq t'' < 2Tdi$）において通常アクセス動作が行われた場合，t_5, t_6では[リフレッシュ実施条件]を満たさない．従って，t_6ではリフレッシュ動作は

行われず（従来法では，t_6でリフレッシュが行われる），t_7において行われる．これは，見かけ上，refresh4を1サイクルだけ遅らせたことになる．

図1のbは，通常アクセス動作が効果的に行われている場合である．このように，連続して各サイクル1回以上の通常アクセス動作が起こるとリフレッシュ動作は起こらない．

$Ndv=n$（$n\geqq 2$）の動作例を図2に示す．n型DDR法においても，通常アクセス動作が全く行われない場合は，従来法と同じ動作を行う（図2.a，refresh2）．

また，$t_n\leqq t<t_{n+1}$なるtにおいて通常アクセス動作が行われても，次のリフレッシュ動作はt_{2n}で行われる．$t_{2n+k}\leqq t'<t_{2n+k+1}$となる$t'$（ただし，$1\leqq k<n$）において通常アクセス動作が行われた場合，$t_{3n+k}$においてリフレッシュ動作が行われる．これは，refresh4をkサイクル遅らせたことになる．

図1のbのように，n-1以下のサイクルごとに通常アクセス動作が行われると，リフレッシュ動作は行われない．

DDR法の分割数Ndvはリフレッシュ動作の精度を決める要因となる．$Tdi\leqq Tri/Ndv$を満たして周期Tdiが大きくなると，通常アクセス動作が行われてからリフレッシュ動作が起動される間での平均時間は大きくなり，Triに近づく．

従って，$Tdi=Tri/Ndv$とし，$Ndv\rightarrow\infty$とした場合がDDR法の理想的な状態である．また，この状態はDRAMに対して理想的なリフレッシュが行われている状態である．

3．効率

DRAMの行アドレスを要素として持つ集合を考える．行アドレス全体の集合Xは，

$X=\{x \mid 0\leqq x<Nrc, x$は整数$\}$

となる．

DDR法は一つの行に対して周期Tdiで動作する．すべての行に対しても動作周期はTdiである．この周期Tdiを1サイクルとし，R_iをi回目のサイクルのリフレッシュ動作で使用される行アドレスの集合，K_iをi回目のサイクルの通常アクセス動作で使用される行アドレスの集合とすると，行全体に対するリフレッシュ動作は次式となる．

$$R_i=X-\{(\bigcup_{j=1}^{Ndv-1}R_{i-j})\cup(\bigcup_{j=1}^{Ndv-1}K_{i-j})\}. \qquad (1)$$

この式(1)を，要素の数の関係式に変換することで，通常アクセス動作の回数とリフレッシュ回数の関係式が得られる．r_iをi回目のサイクルのリフレッシュ動作の回数とし，集合Aの要素の数を$N(A)$と表すと，式(1)は次のようになる．

$$r_i=Nrc-\sum_{j=1}^{Ndv-1}r_{i-j}-N(\bigcup_{j=1}^{Ndv-1}K_{i-j})+\sum_{j=1}^{Ndv-1}N(R_{i-j}\cap(\bigcup_{m=1}^{j}K_{i-m})). \qquad (2)$$

式(2)におけるK_iは通常アクセス動作の動作状況により決まる．次の仮定の下で，1サイクルの通常アクセス動作で使用する行アドレスの数kとリフレッシュ回数の平均rとの関係を求める．

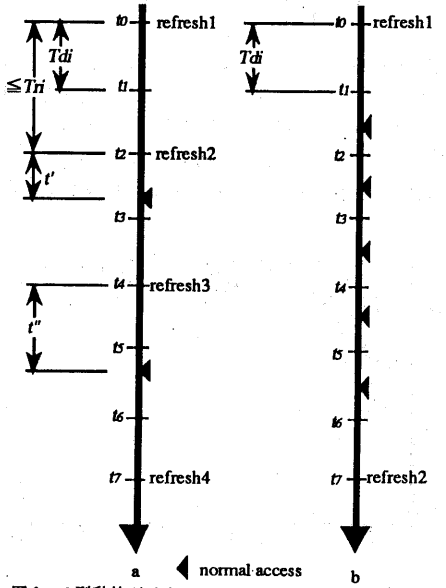

図1　2型動的ダイナミックRAMリフレッシュ法の動作

Fig.1 The action of Type 2 Dynamic Refresh Method.

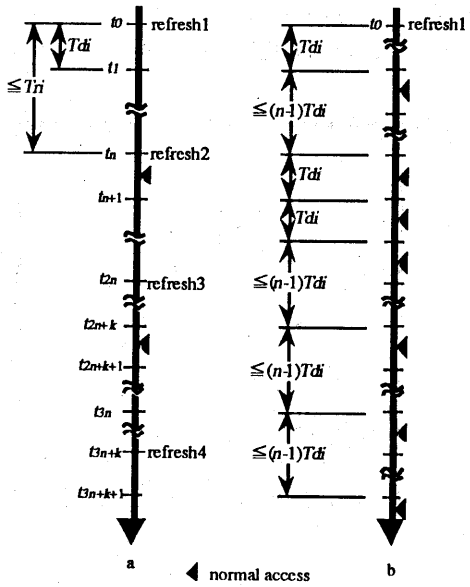

図2　n型動的ダイナミックRAMリフレッシュ法の動作

Fig.2 The action of Type n Dynamic Refresh Method.

・通常アクセス動作は定常状態であり，Tdiの間にａ回行われる．また，Nrc個ある行アドレスのうち，１回の通常アクセス動作でそれぞれの行アドレスが選択される確率は等しく，$1/Nrc$である．

式(2)より，rは次のようになる．

$$r=\frac{k\cdot Nrc(Nrc-k)^{Ndv-1}}{Nrc^{Ndv}-(Nrc-k)^{Ndv}} \quad (3)$$

式(3)はDDR法の１サイクル(Tdi)を基準にしており，これはNdvによって違う値をとる．Ndvとrの関係を観察するために，１リフレッシュサイクル(Tri)の通常アクセス動作の回数a'と１リフレッシュサイクル(Tri)の平均リフレッシュ回数r'の関係を求める．ここでは，$Tdi=Tri/Ndv$として計算する．

$$r'=Ndv\cdot Nrc(Nrc-1)^{a'-a'/Ndv}\times\frac{Nrc^{a'/Ndv}-(Nrc-1)^{a'/Ndv}}{Nrc^{a'}-(Nrc-1)^{a'}} \quad (4)$$

式(3)のkとrの関係を図３のグラフに示す．これはNdv=2,3,4,5,6,8の場合を同一グラフ上に示したものである．

$k=0$は通常アクセス動作が全くない場合である．この場合には，rはNrc/Ndvとなり，従来のリフレッシュ法のリフレッシュ回数と一致する．

$k=Nrc$は通常アクセス動作によってすべての行が使用されていることを示す．この場合，$r=0$となり，リフレッシュ動作が行われないことを示す．

また，kが大きいほどrは小さくなっている．

式(4)のa'とr'の関係を図４のグラフに示す．

これにより，次のことが言える．

・通常アクセス動作の絶対数a'が大きいほどリフレッシュ回数r'は減少する．a'が大きくなるほど，減少率は小さくなる．

・通常アクセス動作の回数が同じでも分割数（Ndv）が大きいほどリフレッシュ回数は少ない．$Ndv=2$の場合がこのリフレッシュ法の最低効率の動作である．$Ndv=\infty$の場合がこのリフレッシュ法の最高効率の動作である．これは，DRAMに最低限必要なリフレッシュ回数であり，従って，理想的なリフレッシュ動作である．

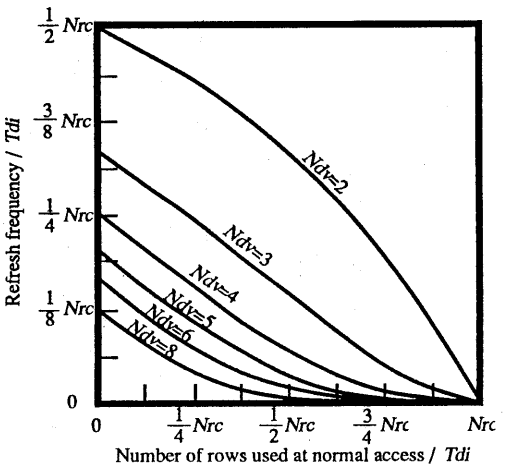

図３．行アドレス使用数とリフレッシュ回数

Fig.3 Number of rows used at normal access and

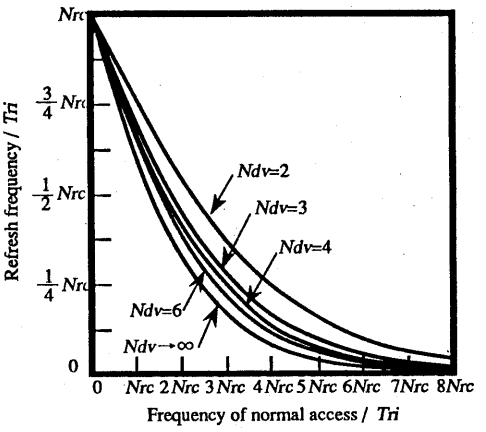

図4　通常アクセス動作とリフレッシュ回数

Fig.4 The frequency of normal access and refresh frequency

分割数（Ndv）と効率の関係を明らかにするために，$Ndv\rightarrow\infty$の場合のリフレッシュ動作の回数r'_{min}を基準に，改善率Iを次のように決める．

$$I=\frac{Nrc-r'}{Nrc-r'_{min}}$$

Nrcは従来のリフレッシュ法のTri間のリフレッシュ動作の回数である．従って，このIは，理想的なリフレッシュ動作のリフレッシュ減少回数に対する，各Ndvの動作のリフレッシュ減少回数の割合である．

図５に，通常アクセス動作の回数a'と改善率Iの関係を示す．

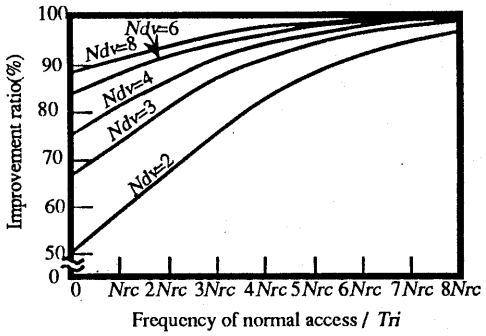

図５　通常アクセス回数と改善率

Fig.5 The frequency of normal access and improvement ratio

通常アクセス動作の回数aが大きくなると改善率Iも大きくなっている．Ndvの違いによるIの格差は，aが大きくなるにつれて小さくなる．

Ndvを固定した場合，改善率Iは$a\to 0$で最低となり，$I\to 1\text{-}1/Ndv$となる．これは，n型DDR法が，理想的なリフレッシュ動作の$1\text{-}1/n$以上の効率で動作することを示している．

4．実装法

DDR法は，過去のサイクルのロウアクセス動作を調べ，それによりリフレッシュ動作を制御する．この操作はすべてのロウアドレスについて個別に行わなければならない．

DDR法を実装するためには，次の3つの機構が必要である．

[ロウアクセス監視機構]

ロウアクセス監視機構は，DRAMに対するロウアクセス動作の発生を監視し，その時の行のアドレスをロウアドレス記憶機構へ通知する．一般にDRAMのロウアクセス動作は，行のアドレスをアドレス端子に与え，RAS信号をアクティブにすることによって行う．従って，ロウアクセス監視機構はRAS信号線を監視し，RAS信号がアクティブになったときに，そのアドレスをロウアドレス記憶機構に通知する．

[ロウアドレス記憶機構]

ロウアドレス記憶機構は，ロウアクセス監視機構の通知により，行ごとに過去のサイクルのロウアクセス動作の発生の有無を記憶する．　DDR法では，過去のNdv-1回のサイクルにおけるロウアクセス動作の発生状況が必要である．従って，実装するDDR法のNdvの値により，必要な記憶容量は異なる．また，ロウアドレス記憶機構はリフレッシュ制御機構からも参照されるため，二つの参照を調停する機能（調停機能）が必要である．

[リフレッシュ制御機構]

リフレッシュ制御機構は，一つの行につき，Tdiの周期でロウアドレス記憶機構を参照し，リフレッシュを行うかどうかを判定する．従って，この動作はTdiの周期の間にNrc回行わなければならない．

既存のデュアルポートRAMを使用したDDR法の実装例を図6に示す．このデュアルポートRAMのアドレス端子にはDRAMの行アドレスの信号線が直接つながれている．この例では，DRAMの行iに対するロウアクセス動作が起こると，デュアルポートRAMのアドレスiの内容が1となる．また，このデュアルポートRAMはロウアドレス記憶機構の調停機能部分を兼ねている．ロウアドレス記憶機構のその他の機能は，リフレッシュ制御機構内のローカルなメモリー上で実現する．

この実装例のリフレッシュ制御機構の1サイクルの動作アルゴリズムを図7に示す．このアルゴリズムでは，デュアルポートRAMは配列dual，ロウアドレス記憶機構は2次元配列localとして表現される．また，通常アクセス動作，リフレッシュ動作は，ともに手続きaccessとして表現される．

Ndvが2と3の場合には，図7のアルゴリズムは簡略化できる．特に，Ndvが2の場合には，配列localが不

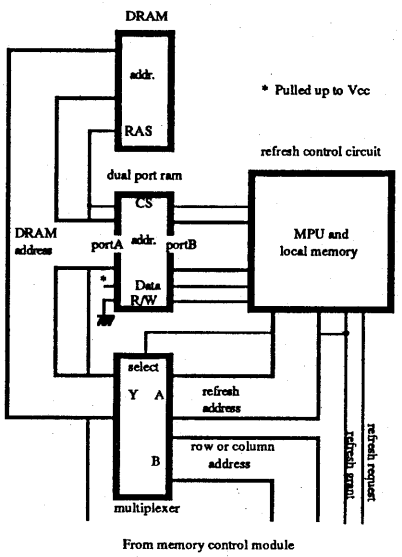

図6　動的ダイナミックRAMリフレッシュ法の実装例
Fig.6 An implementation for Dynamic Refresh Method

```
procedure refresh_one_cycle
    var i, j : integer ;
        res : boolean ;
    begin
    for i := 0 to Nrc-1 do
        begin
        { checking i_th row to be refreshed }
        local[i,0] := dual[i] ;
        { clearing flag for next cycle }
        dual[i] := FALSE ;
        res := FALSE ;
        for j := 1 to Ndv-1 do
            res := res or local[i,j-1] ;
        { refresh i_th row if needed }
        if res = FALSE then access(i) ;
        { copying array data for next cycle }
        for j := 1 to Ndv-1 do
            local[i,Ndv-j] := local[i,Ndv-j-1]
        end
    end

procedure access(i)
    begin
        << access i_th row >> ;
        dual[i] := TRUE
    end
```

図7　図6の構成のリフレッシュアルゴリズム
Fig.7 The algorithm for the configuration of Fig.6

要となり，リフレッシュ制御回路内のローカルメモリは不要となる．このリフレッシュ制御回路は制御用1チップマイクロプロセッサ（MPU）で構成することもできる．

この図6の実装例のデュアルポートRAMの部分を，シフトレジスタやマルチバイブレータをメモリセルとして構成した，あらかじめ与えられた時間を経過すると自動的に内容をリセットするようなデュアルポートRAMに換えると動作アルゴリズムは簡単になる．

メモリセルとしてシングルショットを用いたデュアルポートRAMの例を図8に示す．また，動作アルゴリズムを図9に示す．この場合，アルゴリズムはNdvに依存しないが，図4のシングルショットの時定数は，$(1-1/Ndv)Tri$ にあらかじめセットしておく必要がある．

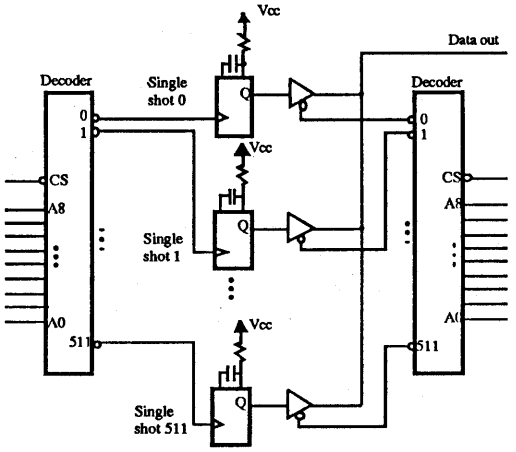

図8　タイマー機能を持ったデュアルポートRAM
Fig.8 Dual port RAM with single shot cells

```
procedure refresh_one_cycle
    var i : integer;
    begin
    for i := 0 to Nrc-1 do
        begin
        { checking i_th row to be refreshed }
        if dual[i]= FALSE then access(i);
        end
    end

procedure access(i)
    begin
        << access i_th row >>;
        dual[i] := TRUE
    end
```

図9　図8構成のリフレッシュアルゴリズム
Fig.9 The algorithm for the configration of Fig.8

5．応用法

一般に，DRAMを用いた記憶装置をリフレッシュの制御方式の観点から分類すると，(1)一度にすべてのDRAMに対してリフレッシュを行う集中制御型，(2)DRAMモジュール単位でリフレッシュを行う分散制御型に分類できる．

(1)の方法は，リフレッシュ動作を行うためにDMAを用いてCPUを定期的に止める．したがって，リフレッシュによるシステムオーバーヘッド（[リフレッシュのために延長された処理時間の合計]／[全処理時間]）は大きいが，制御回路が簡単になり小規模システムでよく使用される．

(2)の方法は，CPU等のモジュールへのアクセスとそのモジュール内のリフレッシュ動作が競合した場合にだけCPUの動作を止める．したがって，システムオーバーヘッドは小さくなるが，モジュールごとに制御回路が必要であり，大規模システムで用いられる．

DDR法を前提としたシステムを設計する場合，(1)(2)の方法で共通して配慮すべき事項は，DRAMに与える行アドレスと制御信号（RAS,CAS）である．

DDR法では，3．式(3)でわかるように，1サイクルの読み書き動作で使用する行アドレスの種類kが多いほどリフレッシュの回数が減り，効率はよくなる．従って，実際にメモリ装置を設計する際には通常の読み書き操作のアドレスを考慮し，行アドレスとして与えるアドレスビットをkが大きくなるように決定しなければならない．

CPUの通常のメモリ参照において，プログラムの参照は頻度が高く，参照アドレスが連続的に変化する場合が多い．従って，このアドレス変化が行アドレスに反映するようにアドレスビットを決定する．

実際には，CPUアドレスの下位ビットがDRAMの行アドレスとなるようにする．

上のような場合，DRAMに対する1回のアクセスで各行が選択される確率は等しいと考えてよい．この場合，DRAMに対する通常アクセス動作回数aが大きいほど行アドレスの種類kが大きくなる．aの上限はCPUの性能で決定されるが，各DRAMに加わる実質的なa'はメモリの構成法で変わる．

実質的なa'を大きくするためには，CPUの1回の読み書き動作で，なるべく多くのDRAMに対してロウアクセス動作が起こるようにする．この時，読み書きの対象となるDRAMに対しては，通常の動作を行い，それ以外のDRAMに対しては行アドレスだけを与える動作（RASオンリリフレッシュ）を行うようにする．また，これらのDRAMに与える行アドレスは共通のものにする．

DDR法を(1)の集中制御型に応用する場合は，すべてのメモリサイクルにおいて，すべてのDRAMに対して

ロウアクセス動作を行うように構成する．このようにすると，DRAMに対する読み書き時だけでなく，ROMやスタティクRAMに対する読み書き時やDMAによる読み書き時にもロウアクセス動作が起こる．また，リフレッシュも単なるDMAとして取り扱うことができ，簡単になる．

このような場合，DRAMに加わる実質的なa'はCPUの性能で決定されるa'と同じ値である．メモリ参照間隔の平均が，1 μS（CPU1），2 μS（CPU2），4 μS（CPU3），8 μS（CPU4）という4つのタイプのCPUについて，DDR法を用いた場合のリフレッシュ周期間の平均リフレッシュ回数を表1にまとめる．使用メモリ素子は1MビットDRAM（標準的な1MビットDRAMのリフレッシュサイクルNrcは512，リフレッシュ周期Triは8 mSである）とする．

表1　CPUタイプ別のリフレッシュ回数

CPU タイプ	参照回数 ／8mS (a')	平均リフレッシュ回数／8mS (r')				
		従来法	2型DDR	3型DDR	4型DDR	8型DDR
CPU1	8000	512	.41	.045	.016	.004
CPU2	4000	512	20.1	7.8	5.0	2.7
CPU3	2000	512	127.0	84.2	69.4	52.8
CPU4	1000	512	280.0	232.7	212.9	187.0

表1をシステムオーバーヘッドに換算したものを表2に示す．

これらの表のCPUタイプを現在のMPUの水準で考えると，32ビットMPUはCPU1～CPU2,16ビットMPUはCPU2～CPU3,8ビットMPUはCPU3～CPU4に相当する．

表2　CPUタイプ別のシステムオーバーヘッド

CPU タイプ	システムオーバーヘッド（%）				
	従来法	2型DDR	3型DDR	4型DDR	8型DDR
CPU1	1.6	1.3e-3	1.4e-4	0.5e-4	1.3e-8
CPU2	1.6	0.063	0.024	0.016	0.009
CPU3	1.6	0.40	0.26	0.22	0.17
CPU4	1.6	0.88	0.73	0.67	0.58

DDR法を(2)の分散制御型に応用する場合も，上と同様に，モジュールに対するアクセスがあった場合はそのモジュール内のすべてのDRAMにロウアクセス動作が起こるようにメモリモジュールを構成する．この方法では，1回の通常アクセスに要する時間を$Tacc$と表すと，a'はDRAMモジュールの使用率$Ruse$に依存し，次のようになる．

$$a' = \frac{Tri \cdot Ruse}{Tacc} .$$

通常アクセス動作とリフレッシュ動作の競合の割合もDRAM使用率に依存する．1回のリフレッシュ動作に要する時間を$Tras$と表すと，システムのオーバーヘッド$Osys$は次式となる．

$$Osys = \frac{Ruse \cdot Tras \cdot r'}{Tri} .$$

以上2式と3．式(4)より，$Ruse$と$Osys$の関係が得られる．このグラフを図10に示す．
ただし，$Tacc$を500nS,$Tras$を250nSとした．

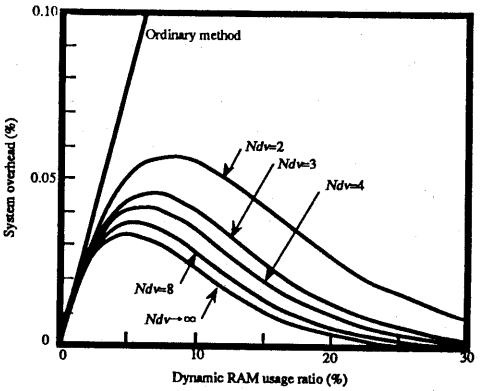

図10　DRAM使用率とシステムオーバーヘッド
Fig.10 DynamicRAM usage ratio and System overhead

DRAMのビジー率が上昇した場合，システムオーバーヘッドも上昇する．DRAMのビジー率が2～8倍に上昇した場合のDDR法のシステムオーバーヘッドについて試算すると，集中制御型では図11，分散制御型では図12となる．（どちらも2型DDR法，条件は図10と同じ）．比較のために，ビジー率が上昇しない場合の従来法のリフレッシュによるシステムオーバーヘッドも示した．

6．むすび

本論文では，ダイナミックRAMのリフレッシュ法として，動的ダイナミックRAMリフレッシュ法（DDR法）を提案し，効率，実装法，応用法について検討した．

DDR法は，従来のリフレッシュ法の1/Ndvの周期で動作する．これにより，DRAMに対する通常の読み書

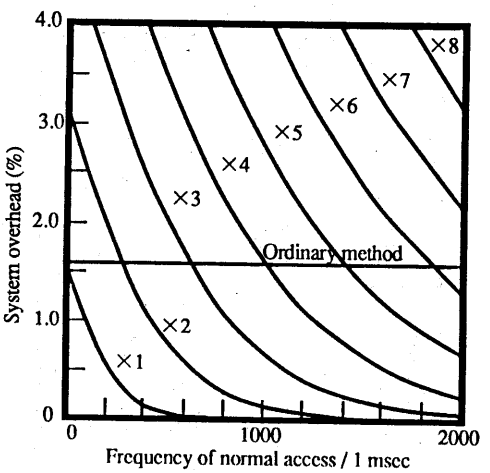

図11　通常アクセス動作とシステムオーバーヘッド
Fig.11 The frequency of normal access and System overhead

き動作の監視とリフレッシュ動作の制御が簡単に行える．DDR法は分割数*Ndv*の違いにより様々なタイプが存在し，従来法から理想的なリフレッシュ法までを表現することができる．

DDR法の単位時間あたりのリフレッシュ動作の回数は，CPU等からの通常アクセス動作の状態により動的に決まる．CPU等の動作が定常状態である場合のリフレッシュ動作の解析により次のことがわかった．

(1)単位時間あたりのリフレッシュ動作の回数は，DRAMに対する通常アクセス動作の回数が0の時に最大となり，この時のリフレッシュ回数は従来のリフレッシュ法と同じである．読み書き動作が多くなるほど，リフレッシュ回数は減少する．

(2)分割数が大きくなるほどリフレッシュ回数は減少する．この分割数を無限に大きくした状態は，リフレッシュ動作の理想的な状態である．

(3) n 型DDR法は理想的なリフレッシュ動作の1-1/n以上の効率で動作する．

また，DDR法はデュアルポートRAMを用いると簡単に実装できる．専用のデュアルポートRAMを用いると，リフレッシュ制御のアルゴリズムが簡単になる．

DDR法を実際のシステムに応用する場合には，リフレッシュ効率を最大にするための工夫が必要である．既存の32ビットや16ビットのマイクロプロセッサを用いた小規模のシステムに応用する場合の試算では，通常処理中の1リフレッシュサイクルあたりのリフレッシュ動作の回数の平均は1未満となる．また，大規模システムでは，リフレッシュによるシステムオーバーヘッドが0.1%以下となる．

DDR法は一定時間内のDRAM使用回数が上昇すると効率が上がる．DRAM素子の大容量化が進みリフレッシュのオーバーヘッド（ビジー率）が4倍程度に大きくなっても，一定時間内のDRAM使用回数が十分多いと，現在のビジー率のDRAMを従来法でリフレッシュする場合のシステムオーバーヘッドより小さくなることがわかった．

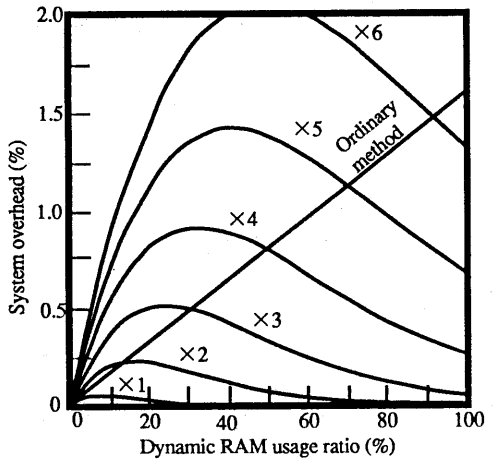

図12 DRAM使用率とシステムオーバーヘッド
Fig.12 DRAM usage ratio and System overhead

最後に，残された課題は，実際のシステムに応用しCPU等の動作状況とリフレッシュ回数の関係を実験的に求めることである．

謝辞 本研究を進めるうえでご指導をいただいた慶應義塾大学理工学部中西正和教授に感謝します．

文献

(1)G.Langdon:"Dynamic random-access memory refresh methodandmeans",IBMTech.Disckosure Bull.,22,4,pp.1636 (Sept. 1978).

(2)T.Dvorak:"Transparent refresh of dynamic random-access memories",IBM Tech. Disclosure Bull.,23,7B,pp3201-3202 (Dec.1980).

(3)松井祥悟:"動的ダイナミックRAMリフレッシュ法", 電子情報通信学会論文誌D-1,J72-D-1,3,pp.175-181 (Mar.1989).

付録

1．式(2)の導出

式(1)より，次の2式も成り立つ．

$$R_i \cap \left(\bigcup_{j=1}^{Ndv-1} R_{i-j} \right) = \phi \ ,$$

$$R_i \cap \left(\bigcup_{j=1}^{Ndv-1} K_{i-j} \right) = \phi \ .$$

これより，式(1)を要素の数の関係式に変換すると式(2)が得られる．

2．式(3)の導出

仮定に基づき，式(2)の第3項，第4項の平均を求めると次式が得られる．

$$r = Nrc - \sum_{j=1}^{Ndv-1} r \cdot \frac{Nrc^{Ndv-1} - (Nrc-k)^{Ndv-1}}{Nrc^{Ndv-2}} + \sum_{j=1}^{Ndv-1} r \frac{Nrc^{j} - (Nrc-k)^{j}}{Nrc^{j}} \ .$$

これを，rについて解くと式(3)が得られる．

3．式(4)の導出

仮定により，a（1サイクルの通常アクセス動作の回数）とkの関係は次式になる．

$$k = \frac{Nrc^{a} - (Nrc-1)^{a}}{Nrc^{a-1}} \ .$$

また，aとa'，rとr'は次の関係にある．

$$a = a'/Ndv,\ r = r'/Ndv \ .$$

この3式と式(3)より式(4)が得られる．