解　説

Illiac IV におけるメイン・メモリの構成*

村　岡　洋　一**

1. Illiac IV の構成[1)]

Illiac IV は図-1 にその構成の概略を示すように，いわゆるアレイ型の並列処理計算機である．命令を解釈する1つのコントロール・ユニット (CU) のもとに4個の演算装置 (PE) があり，CU の制御によって全 PE が同一の演算（例えば加算）を同期して行なう．各 PE には2,048語のメモリ (PEM) が付いている．便宜上 PE_i の PEM の j 番地を PEM(i, j) と書く．PE は主なレジスタとしてアキュムレータ (ACC) とインデックス・レジスタ (IX) を持つ．Illiac IV の命令は基本的には1アドレス方式で，ACC の内容とメモリの内容の演算である．従って命令

OP a(IX)

は，PE_i においては

ACC_i **OP** PEM$(i, a+IX_i)\rightarrow ACC_i$

という演算になる．ACC と IX の添字 i は，これが PE_i のものであることを示す．

Illiac IV はその構成から，特に行列演算を基本とする数値計算に適している．例えば64個の要素を持つ2つのベクトルAとBが図-1のようにメモリに格納されていればAとBの加算は，

load 100
add 101
store 102

の3ステップでできる．

PE 間では，図-2のように PE_i が $PE_{i\pm1}$ または $PE_{i\pm8}$ と直接データの授受を行なえる．これを routing と呼ぶ．従って例えば PE_8 が PE_{17} にデータを送るには，$PE_8\rightarrow PE_{16}\rightarrow PE_{17}$ という2段階が必要である．Routing は全 PE が同期して，一時に全てが同一の動作（例えば $PE_i\rightarrow PE_{i+1}$）しか行なえない．

CU からは全 PEM があたかも1つのメモリのように見える．すなわち PEM(i, j) は CU からは，$64j+i-1$ 番地に見える．CU が直接 PEM にアクセスするのは，命令のフェッチや全 PE で共通に使うデータ等である．後者は CU から全 PE に与えられる．これを broadcast と呼ぶ．

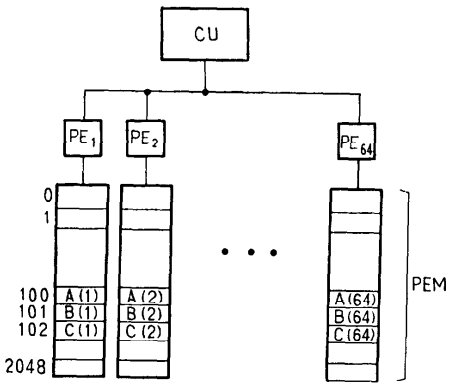

図-1　Illiac IV の構成

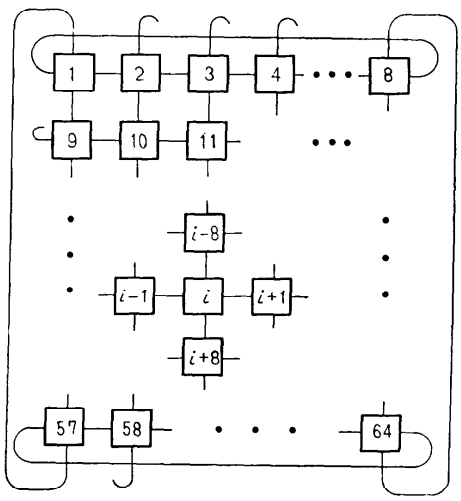

図-2　PE 間の接続

* The Configuration of Main Memory for Illiac IV by Yoichi MURAOKA (N. T. T. Yokosuka Electrical Communication Laboratory)

** 日本電信電話公社　横須賀電気通信研究所データ通信研究部

Illiac IV は科学技術計算を主とする特殊計算機である．従ってその設計思想も科学技術計算に偏っているのも当然である．メモリについていえば，汎用計算機のメモリがプログラムやファイルの格納場所として特に意識されるのに比べて，Illiac IV では計算データの格納場所という意識が強い．すなわち汎用計算機では複数の利用者のプログラムの多重処理の効率良い実現が，その設計思想の根幹である．これに対して Illiac IV の場合には複数プログラムの多重処理も勿論可能であるが，むしろ1つのプログラムの効率良く，かつ高速な処理が大きな目的であり，このためのいろいろ工夫がなされている．本解説では，まず PEM を中心とした Illiac IV のメモリ構成を第2節で説明し，次にメモリ・マネジメントを中心としたソフトウエア上の工夫を，第3節で紹介する．最後に第4節で Illiac IV のメモリ構成の問題点と今後の課題をとり上げる．

2. メモリの構成

Illiac IV の PEM はフェアチャイルド社で製造された IC メモリである．Illiac IV の設計・製造が進められていたのが今から約10年以上前であったので，IC メモリ採用の決定は当時としては画期的なことであった．

メモリ・チップは256ビットを収容した DIP 形ケースに入った，バイポーラ IC メモリである．Illiac IV の1語は64ビットなので，1PEM 当り 128k ビットすなわち512チップ必要となる．

CU, PE および PEM の関係を**図-3**に示した．同図中の MLU はメモリ・ロジック・ユニットで，PEM へのアクセスの制御を司どる．

PEM と外部世界は，**図-4**のように DISK を介して接続されている．外部からのデータは一旦この DISK に入れられ，ここから PEM へ出し入れされる．DISK の容量は 10^9 ビットで，PEM 全体の容量である 8×10^6 ビットの約120倍である．DISK のデータ転送率は 10^9 ビット/秒であり，PEM のデータ・アクセス率は 8×10^9 ビット/秒である．従って DISK から PEM へのデータ転送には，PEM の memory cycle stealing がされる．

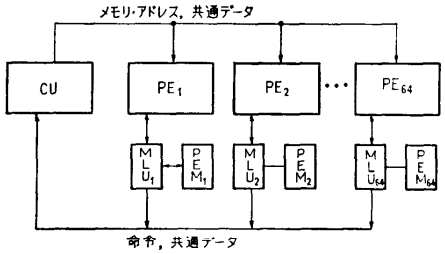

図-3　メモリ構成

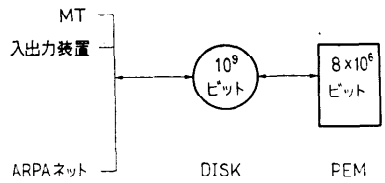

図-4　DISK と PEM

DISK の回転時間は 40 ms なので，平均回転待ち時間は 20 ms となる．仮りに PEM を 1k 語ずつの2つの部分に分け，1方のデータを計算しているとき他方のデータを DISK と入出力するという交代バッファリング的な使用をしたとする．さらに1語の計算に5命令必要で，1命令の実行時間が 2 μs としよう．すると 1k 語の計算には 2 μs×5×1k，すなわち 10 ms かかることになる．これに対して DISK の平均回転待ち時間は 20 ms なので，このままでは Illiac IV は常に DISK 入出力待ちとなり idle になる恐れがある．この問題を解決するために，ハードウェアおよびソフトウェア上の工夫がなされた．ソフトウェアの工夫については次節で説明することとし，以下ではハードウェアの工夫の1つである DISK キューについて述べる．

DISK は，1度に 24 個までの入出力要求を保持できるキューを持つ．キュー中の全ての入出力要求の DISK アドレスは，読み書きヘッドの下の DISK のアドレスと常に比較されており，もしどれかが合致するとその合致した入出力要求が処理される．これは汎用計算機のページング・ドラム等で使われている，shortest—access—time—first アルゴリズム[2] と同等であり，DISK の回転待ち時間減少に有効である．

3. メモリ・マネジメント

Illiac IV を効率良く使いこなすためには次の2項を考慮しなければならない．

（1）　PE idle の防止

（2）　入出力時間と処理時間のオーバラップ

これらの問題と，メモリ・マネジメントの関係を次に説明する．

3.1 PE idle の防止

Illiac IV のように，各 PE に対応して1つの PEM

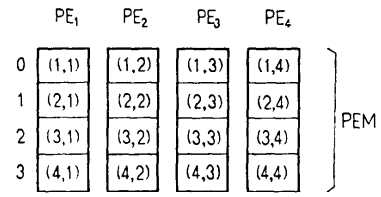

図-5 Straight Storage Mapping

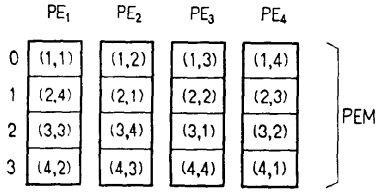

図-6 Skewed Storage Mapping

が用意されている特殊な構造では，データの要素をいかに PEM に割りつけるかという storage, mapping が重要な問題となる．以下に4行4列の2次元行列を例にとり，4PE システムを使って説明する[3]．

最も簡単な storage mapping 法は，図-5のように行列の行方向を PEM のアドレスに，列方向を PEM の番号に対応させる方式であり，これを straight storage mapping と呼ぶ．この方式では1つの列は完全に1つの PEM に含まれる．i 番目の行は，各 PE のインデックス値を i に設定すれば同時に全要素にアクセスできるが，j 番目の列の全要素には1つの PE しかアクセスできない．従って列要素全てに対する同時アクセスが有利な column pivoting 処理や ADI には，不適当である．

この欠点を補うのが，図-6に示す skewed storage mapping である．この方式では各行の先頭が順次1PEづつずらして，PEM に格納される．従って i 番目の行は，各 PE のインデックス値を i に設定すれば同時に全要素にアクセスできる．j 番目の列にアクセスするには，k 番目の PE のインデックス値を $(k-j)_{\mathrm{mod}N}$ にすれば良い．N は全 PE 数である．

列の数が PE の数 (64) より大きい行列は，図-7 (a)のように 64×64 の正方副行列に分割して，PEM に格納する．もし列の数が 64 の整数倍でないと，同図の副行列 $A_{1,2}$ のように正方でないものができる．これを例えば straight storage mapping 法で PEM に格納すると，図-7 (b) のように PEM に空きができてしまう．Skewed strage mapping 法では，行と列の区別なく全要素に1度にアクセスできるから，図-7

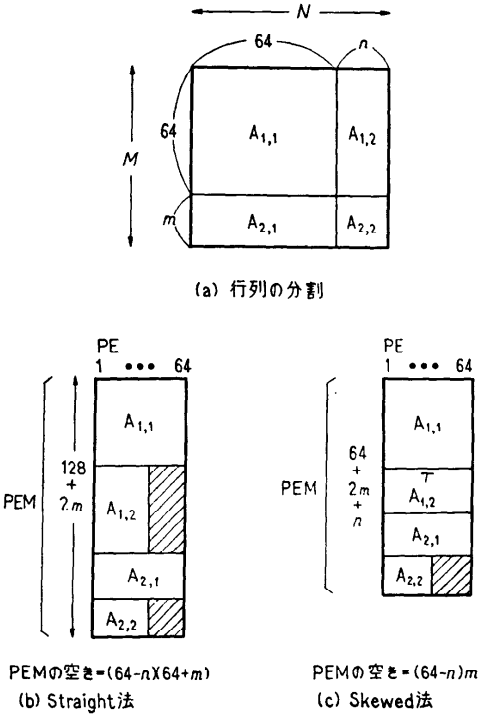

図-7 大規模行列の分割と格納

(c)のように副行列 $A_{1,2}$ の転移行列 $A_{1,2}^T$ を格納することによって，PEM の無駄を防ぐことができる．

3.2 入出力時間と処理時間のオーバラップ

1節で述べたように，処理によっては DISK の平均回転待ち時間よりも，Illiac IV の処理時間が短かくなってしまう．従って特に考慮しないと，Illiac IV が DISK 入出力待ちになってしまう．これを解決するソフトウェアの手段として，DISK 上のデータ・マッピングまたは入出力スケジューリングがある．今対象とする問題のデータ（例えば行列）が大きすぎて，PEM に1度に格納できないとする．このデータを適当な大きさのサブ・データ $A_1, A_2, \cdots, A_n$ に分割し，サブ・データ単位で処理をする．サブ・データ1個の処理に P ms 必要とする．今サブ・データ A_i を DISK から読み込んで，この処理を開始したとしよう．Illiac IV の処理を完全に DISK 入出力とオーバラップさせるには，次の2つの方法のいずれかをとればよい．

(1) サブ・データ A_{i+1} が必要となる 40 ms (DISK 回転時間) 前に，A_{i+1} に対する入出力要求を出しておく．

(2) DISK 上に，サブ・データ A_i と A_{i+1} の間隔を回転時間で計った距離が P ms 以下となるように，A_i と A_{i+1} を配置する．A_i の処理を開始すると同時に A_{i+1} への入出力要求を出す．

P の値は，問題の性格（例えば行列演算）が明確であれば推測可能である．特に Illiac IV の場合は，用途が限られるのでこれが容易である．(1)の方法は，P の値が 40 ms より小さいと実現が困難となる．これに比べて，DISK 上のデータ配置をスケジュールする(2)の方法は，DISK へのデータのローディング処理を適当に行なうことによって，比較的容易に実現できる[4]．一般の多重処理計算機では，プログラムのタイミングの制御が困難であるが，同一処理を大きなデータ・ベースに適用することが主目的の Illiac IV では，入出力処理と Illiac IV 処理の同期制御は重要である．

4. 問題点と今後の課題

Illiac IV のメモリ構成の問題の1つは，PE 間接続にある．1ステップで，全 PE が同期した均一なしかも一定距離の routing しかできないため，例えば**図-8**のように数列の和処理を行ないたいときには，中間結果の PE 間の routing によるオーバヘッドが無視できなくなる．PEM と PE 間にクロスバ・スイッチをおけば，同一時間に任意の PE が任意の PEM に1ステップでアクセスできる*．しかしクロスバ・スイッチは一般に経済的に実現するのは非常に困難である．そのための妥協の1例が Illiac IV の routing 接続であった．プロセッサとメモリ間のより良い接続法の開発は，今後とも並列処理計算機設計の成功の鍵といえる[5]．

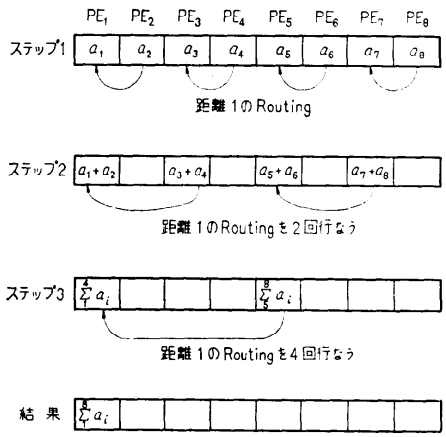

図-8 Routing 処理のオーバーヘッド

* ただし同一 PEM への複数 PE からのアクセス競合がある．

参 考 文 献

1) G. H. Barnes, et al.: The Illiac IV Computer, IEEE TC, C-17, No. 8, pp. 746～757 (1968).
2) J. Abate and H. Dubner: Optimizing the Performance of a Drum-Like Storage, IEEE TC, C-18, No. 11, pp. 992～996 (1969).
3) D. J. Kuck: Illiac IV Software and Application Programming, IEEE TC, C-17, No. 8, pp. 758～769 (1968).
4) D. E. Gold: Application of Some Switching Network Results to Dynamic Allocation of Memories in Hierarchy, Compcon 72(1972).
5) D. J. Kuck & Y. Muraoka: Fast Computers from Slow Parts, Compcon 72 (1972).

(昭和49年12月23日受付)
(昭和50年2月6日再受付)