パラレル･ワールド法

- - 相関器メモリネック回避法

連続メモリ領域に対するアクセスでコストを低減する。

近田義広･国立天文台

CHIKADA.Yoshihiro@nao.ac.jp

（相関器 WS　２００４）

以下は、相関器の FFT 部分を念頭に書きましたが、一般に、計算装置において、「data flow graph を書いたとき、データの流れがこんがらかっていて、整理しにくい故に、実効メモリ・バンド幅が減少してしまう場合」
に適用できる手法だと思っています。

DRAM にせよ SRAM にせよメモリアドレス上で連続した領域に対する読み書きの方が、連続しない領域に対するそれよりも、高速であることが多い。従って、アドレスが連続するように algorithm を組み立てれば、より安く作れる可能性がある。
しかし、FFT の場合、バタフライの入力出力は、各段毎に異なったパターンとなるため、たとえある段のバタフライの二つの出力がアドレスで連続するように割り当てられたとしても、次の段のバタフライの二つの入力のアドレスは連続でなくなってしまう。(下図参照)

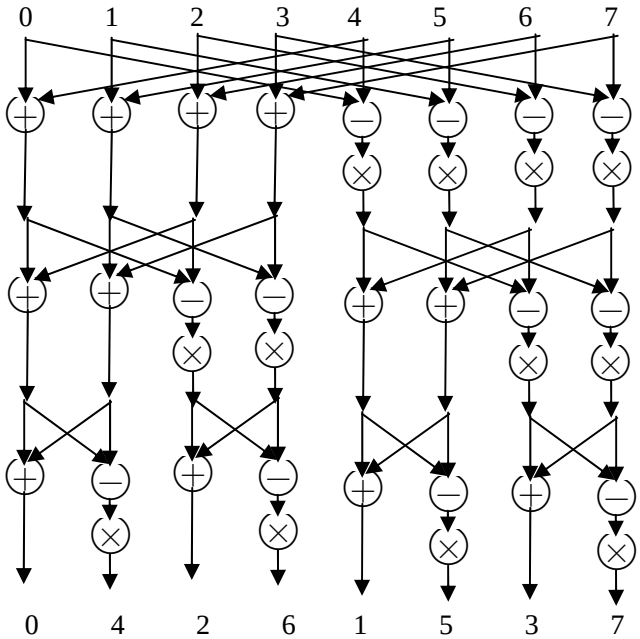

Fig. Data flow in an 8 point FFT.

【パラレル・ワールド法】

同型のデータフロウを持つ n 個の大きさ N のデータ配列 A[i,j] (i=1,…,n, j=1,…,N)を演算する、例えば N 点 FFT をｎ素子アンテナについて計算する場合について考える。あるいは時間的に隣り合うｎ個の N 点 FFT segment を計算する場合でも同様である。

普通は、素子毎あるいは segment 毎の計算を優先させて内側のループに置き、

```
For i=1 to n do
        …
        B=A[i,*]
        For j=1 to N do
                …
                For k=1 to K do
                        j[k]=g(j,k)
                        J[k]=G(j,k)
                enddo
                (B[J[1]],…, B[J[K]])=f(B[j[1]],…, B[j[K]])
                …
        enddo
        …
enddo
```

のように計算するが、これを do ループの内外を逆転させ、かつ i（素子番号や segment 番号）だけが異なるデータが連続的にメモリに割り付けられるようにして計算する。

```
(* i をアドレスの下位ビットに割り付けて下の計算をするように *)
(* 必要なら A[i,j]の転置を事前に行っておく *)
For j=1 to N do
        …
        For k=1 to K do
                j[k]=g(j,k)
                J[k]=G(j,k)
        enddo
        For i=1 to n do
                …
                (A[i,J[1]],…, A[i,J[K]])=f(A[i,j[1]],…, A[i,j[K]])
                …
        enddo
        …
enddo
```

このようにすると、最小ループの中の計算は、連続アドレスへの n 回のアクセスになり、

アクセスが高速化あるいは低コスト化される。一方、短所として、最小ループの中に n×N の配列である A[i,j]が持ち込まれるので、メモリの必要量が増える。また A[i,j]の転置のためにコストがかかる場合もある。しかし、アクセスの連続化によって SRAM を安価な DRAM で置き換えられるなら、短所は補っておつりが来るであろう。

また、i を FFT segment 番号とすれば、X 部のかけ算器直後のメモリ・アクセスをレジスタ・アクセスに置き換える”Short-Term Buffer”法との相性がよいです。また、i を素子アンテナ番号とした場合は、相関相手が近いアドレスにある事になるので、メリットがあるかも知れません。

【部分的パラレル・ワールド法】

全体が、パラレル・ワールドでなくても、ある部分を取り出してパラレル・ワールドに分割できる場合がある。FFT の場合、下左図の赤点線枠の部分同士がパラレル・ワールドになっている。また、データフロウの書き方を変えれば、下右図のようにも見ることができる。

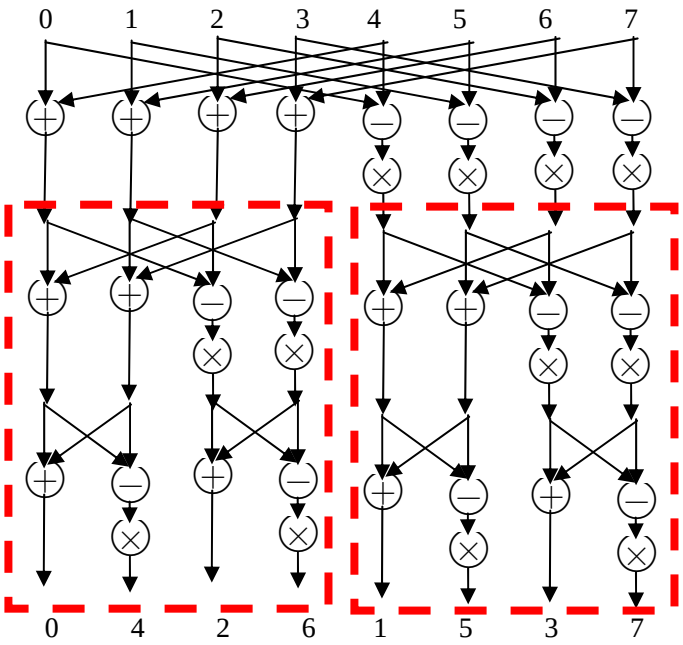

Fig. “Parallel Worlds” in the data flow in an 8 point FFT.

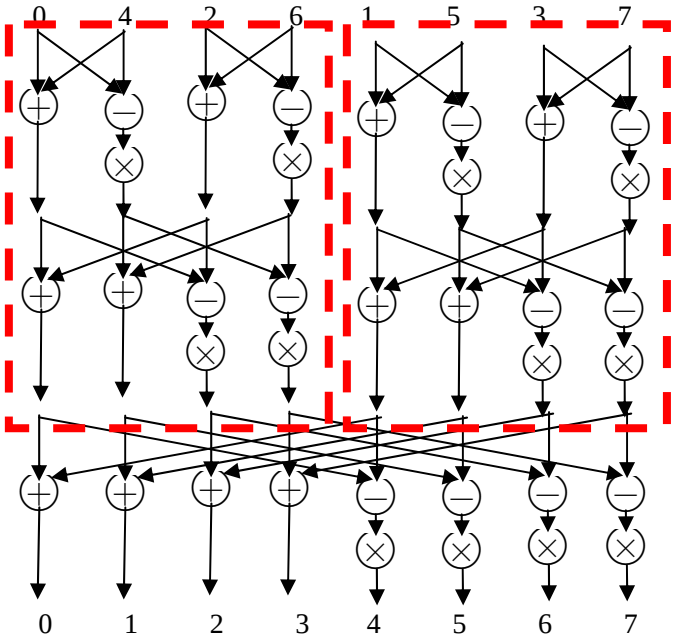

Fig. Another “Parallel Worlds” in the data flow in an 8 point FFT.

さらに下図のようにも見ることができる。

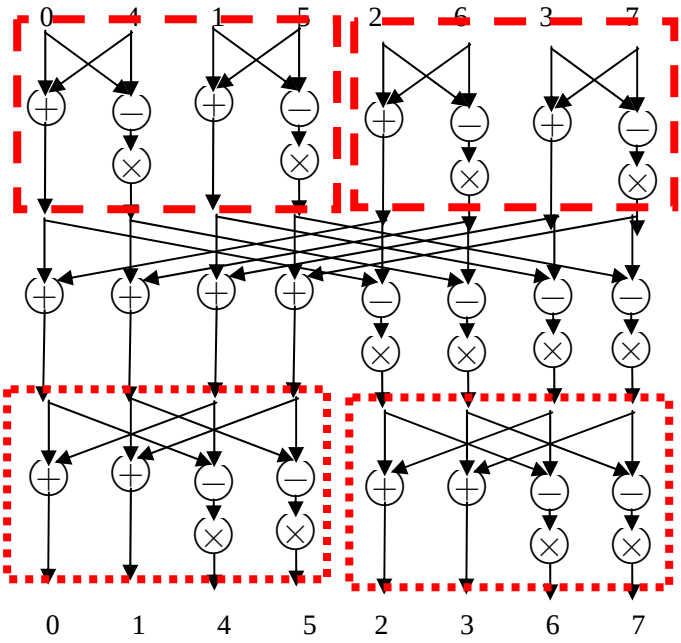

Fig. Yet another “Parallel Worlds” in the data flow in an 8 point FFT.

FFTでなくても部分的にパラレル・ワールドに分けることができる場合がある。例えば、FX のX 部は周波数方向のパラレル・ワールドに分けることができる。

全体の中でパラレル・ワールドにできなかった部分の計算は、同時に読み書きしていたデータ同士（言い換えればキャッシュ内のデータ同士）の計算なので、そこではメモリ・アクセスのためのコストは元々最小である。

従ってこの部分的パラレル・ワールド法を使えば、メモリ・アクセス・コストを最小にでき、かつ前述のパラレル・ワールド法のように、メモリの大きさを増やさないで済む。

【部分的パラレル・ワールド法の実装】

実装は様々あり得るが、演算ユニットに double buffer のキャッシュ・メモリを付けて、主メモリとキャッシュ・メモリとのやりとりは、主メモリ上での連続したアドレスに対する pre-fetch など burst access で実現するのが一つである。キャッシュ・メモリの大きさの下限 CMmin は、burst access するメモリ量（＝パラレル・ワールド度 P）×演算ユニットの入出力幅 Nio（radix-2 FFT なら２）×double buffer の２である。すなわち：

$$CMmin = 2\ P\ Nio$$

である。