４Ｈ−１０　ＯＳＣＡＲ上でのセルラ・ニューラル・ネットワーク・シミュレーションの並列処理手法

吉岡　明広　　林　俊成　　笠原　博徳　　成田　誠之助　　Leon. O. Chua*

早稲田大学理工学部電気工学科　　*University of California, Berkeley

１．はじめに

セルラ・ニューラル・ネットワーク[4][5]（以下ＣＮＮと略記）は１９８８年にLeon O. Chua等のグループにより提案された情報処理システムであり、ニューラルネットワークモデルの１つであるHopfield型のネットワークとCellularAutomataを融合したモデルである。

ＣＮＮの大きな特徴にテンプレートと呼ばれるパラメータ群の設定を換えることにより、全く新しい処理システムを作り出すことができることがあり、それは、多種の処理に容易に適用できるという利点となっている。そして、その有効性はシミュレーションにより数々の画像処理の分野において確認されている。

また、このシステムはハードウェア化による高速化の可能性が極めて高い。だがしかし一旦ハードウェア化されると自由度に欠けてしまうため、高速なシミュレーションが必要とされている。

そこで、筆者等が以前から提案している、最少数のプロセッサで最少の処理時間を可能とするスタティック・マルチプロセッサ・スケジューリング・アルゴリズム[1][2]を適用し、汎用目的マルチプロセッサ・システム・ＯＳＣＡＲ[3]上での、セルラー・ニューラル・ネットワーク・シミュレーションを行った。本稿ではその並列処理手法について述べる。

２．ＯＳＣＡＲのアーキテクチャ[3]

ＯＳＣＡＲ(*Optimally SCeduled Advanced multiprocessoR*)のプロセッサクラスタは、１６台のプロセッサエレメント(PE)、３つのローカル共有メモリ(CM)、及びローカルコントロールプロセッサ(LCP) を３本バスで接続した平等型共有メモリマルチプロセッサ・システムである。各PEは、浮動小数点演算を含めた全てのインストラクションを１クロックで実行する３２ビットRISCライクプロセッサであり、６４個の汎用レジスタ、PE間転送用２ポートメモリ（分散共有メモリ）、２５６Kwローカルデータメモリ、２バンクの６４Kwインストラクションメモリ、及びスタックメモリ等を持っている。

３．ＣＮＮ

ＣＮＮは、たくさんのセルと呼ばれる基本回路ユニットからなる、大規模非線形アナログ回路である。図１に回路の例を示す。一つ一つのセルは規則正しく配置されており、隣合うセルとのみ結合していて、互いに作用し合う。また、それ以外の隣合わないセル同士の間にも、間接的な相互作用がある。図２にＭｘＮセルを持つ２次元ＣＮＮの結合例を示す。

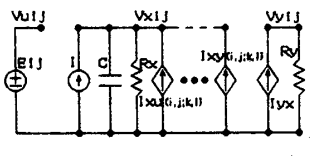

図１　ＣＮＮの基本回路例

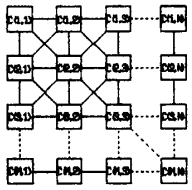

図２　MxNのCNNの例

ここで、ＣＮＮの状態方程式、出力方程式を式１，２に示す。

状態方程式：

$$C\frac{dv_{xij}(t)}{dt}=\frac{1}{Rx}v_{xij}(t)+\sum_{C(k,l)\in Nr(i,j)}A(i,j;k,l)v_{xij}(t)+\sum_{C(k,l)\in Nr(i,j)}B(i,j;k,l)v_{uij}(t)+I \quad (1)$$

出力方程式：

$$v_{yij}(t)=\frac{1}{2}\left(|v_{xij}(t)+1|-|v_{xij}(t)-1|\right),\ 1\le i\le M;1\le j\le N \quad (2)$$

$v_{xij}, v_{yij}, v_{uij}$ はそれぞれ各ユニットの状態変数、出力変数およびバイアスであり、Ｉは全ユニット共通のバイアスである。M,N はセルのサイズであり、また、Nrはテンプレートで制限される結合をc(i,j)を中心にしてまとめて表現したものであり、ｒ近傍といい、式３のようにに定義される。

$$Nr(i,j)=\{C(k,l)\mid \max\{|k-i|,|l-j|\}\le r,1\le M;1\le l\le N\} \quad (3)$$

Ａ及びＢはテンプレートで、それぞれフィードバック・テンプレート、コントロール・テンプレートと呼ばれる。(1)でのシグマはつまりテンプレートによって結合されたセル同士の作用の総量を表す。

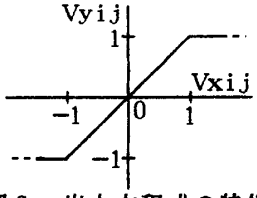

図２　出力方程式の特性

出力方程式は図３に示すようにv_{xij}の絶対値が１以上の時は１か−１の値を取り、それ以下の場合はその値自身を取る。

通常、ＣＮＮは初期状態をv_xにセットし、v_u 及びＩを固定し、そしてその後そのダイナミクスに従って収束させる。即ちＣＮＮの最終的な出力は収束した後に各ユニット出力v_yとして得られる。

４．並列処理手法

本手法では、タスク（プロセッサへの基本割当単位）生成部、タスク・グラフ生成部、タスク・スケジューリング部、及びマシンコード生成部から構成される。ホストコンピュータ上で生成されたコードは、ローカル共有メモリーにダウンロードされ、並列に実行される。

３．１　タスク生成

タスク生成では、処理すべき計算全体をタスクに分割する。この際、タスク・グラニュラリティの決定が重要なポイントとなる。グラニュラリティが細かすぎると、タスク間同期やデータ転送などのオーバーヘッドが増大し、逆に大きすぎると、並列性が引き出されない。本シミュレーションでは常微分方程式求解法の１つである Euler法により、ＣＮＮの状態方程式 (1)を解き、そして、その結果を出力方程式 (2)に代入し計算している。この際、タスクサイズの小さい方から加減乗除等の算術演算、演算式の計算、セルの計算を単位とする分割が考えられる。本手法はどのタスク・グラニュラ

A Parallel Processing Scheme for the Cellular Neural Network Simulation on OSCAR
Akihiro YOSHIOKA, Chunchen LIN, Hironori KASAHARA, Seinosuke NARITA, L. O. CHUA*
WASEDA University, *U. C. Berkeley

リティについても同様に適用できるが、この問題に関しては、複数インストラクションレベルのグラニュラリティを採用している。その際、１セルへのテンプレートに関する演算、即ち式(1) におけるシグマ部を一つのタスクとする事などにより、同時処理可能なタスクを最大にするように決定した。図４にタスクの例を示す。

```
-  c 7  v 17
/  t -1  c 13
+  t -1  c 44
:= v 1  t -1
*  c 14  v 33
*  c 15  v 34
*  c 17  v 37
*  c 18  v 38
+  v 1  t -4  t -3  t -2  t -1
:= v 1  t -1
```

図３　タスクの例

３．２　タスク・グラフ生成

生成されたタスク間には、タスク間のフロー依存や出力依存といった依存関係によるに実行順序関係の制約である先行制約が生じる。この先行制約は、タスク・グラフという無サイクル有向グラフで表現できる。この際、各タスクの１プロセッサ上での処理時間の推定値も算出する。

今回のＣＮＮのシミュレーションにおいては１近傍のテンプレートで１６ｘ１６の対象セル数において１０４８個のノードを持ったタスクグラフが生成された。また、例として、４ｘ４の対象セル数のＣＮＮのタスクグラフを図５に示す。図４はデータ依存関係はとても複雑、かつ多くなっている一方、同一レベルに配置されたタスクが非常に多くなっている。１６ｘ１６の場合は同様なタスクグラフが１６倍の幅を持った様になる。

３．３　タスク・スケジューリング

生成されたタスク集合のプロセッサ上への最適割当、実行順序決定問題は、実行終了時間最少マルチプロセッサ・スケジューリング問題に帰着されるが、本手法では筆者等が開発したスタティク・マルチプロセッサ・スケジューリング・アルゴリズムCP/DT/MISF[2] を用いて最適スケジューリングを生成する。

３．４　マシンコード生成

次にタスク・スケジューリングの結果に基づき、使用するマルチプロセッサ・システム中の各プロセッサ上で実行されるマシンコードを生成する。各プロセッサ用のマシンコードは、そのプロセッサに割り当てられたタスク用のマシンコードを実行順序にしたがって並べたものに、タスク間同期を取るマシンコードを挿入した形となっている。本手法ではこのマシンコード生成の際に、同期オーバーヘッドの最少化、各プロセッサ内のレジスタ利用の最適化を行う。

４．　ＯＳＣＡＲ上での性能評価

本手法を用い、ＯＳＣＡＲの１プロセッサ・クラスタ[3] 上で、ＣＮＮシミュレーションの並列処理を行った結果について述べる。対象となったのは１近傍のテンプレートで１６ｘ１６のＣＮＮである。処理時間は実測値でＰＥが１台の時１５４μｓであったのに対し、２台では７９μｓ、４台では４１μｓ、８台においても２５μｓという結果が得られ、効率の良い並列処理が行われることが確かめられた。図６にＰＥ１台と各ＰＥ台数の実行時間の比のグラフを示す。実線はＯＳＣＡＲでの実処理時間であり、波線はＯＳＣＡＲのアーキテクチャを前提としたＯＳＣＡＲのシミュレーションにより得られた予測処理時間である。図より実測値と予測値がほぼ一致していることからＯＳＣＡＲ上でほぼシミュレーションされたのと同様にマシンコードが実行されていると考えられる。これより、マシンコード生成時にスタティク・スケジューリング結果を用いてクロックレベルの厳密な最適化を行うのが有効であると確認できる。

また、たとえデータ依存関係が非常に複雑であり、先行制約条件が複雑であっても十分な同一レベルのタスクの数が存在すれば、ＯＳＣＡＲ上のブロードキャスト転送モードを有効に使用することにより、効率の良い並列処理が行えることが確認された。

５．　結び

本稿では、スタティック・マルチプロセッサ・スケジューリング・アルゴリズムを用いたＣＮＮのシミュレーションの並列処理手法を提案し、それを用いるとＯＳＣＡＲ上で非常に効率の良い並列処理が実現できることを示した。

参考文献

(1)H.Kasahara and S.Narita, "Practical multiprocessor scheduling algrithms for efficient parallel processing" ,IEEE Trans. Comput., pp.1023-1029, Nov. 1984

(2)H. Kasahara, H. Honda and S. Narita, "Parallel processing of near fine grain tasks using static scheduling on OSCAR", Supercomputing Nov. 1990

(3)笠原，成田，橋本，"ＯＳＣＡＲのアーキテクチャ"，電子情報通信学会論文誌Ｄ，Vol.J71-D No.8 pp1440-1445,1988年8月

(4)L.Chua and L. Yang, "Cellular Neural Networks: Theory", IEEE Trans. Circuit., pp1257-1272, Oct. 1988

(5)L.Chua and L. Yang, "Cellular Neural Networks: Applications", IEEE Trans. Circuit., pp1273-1290, Oct. 1988

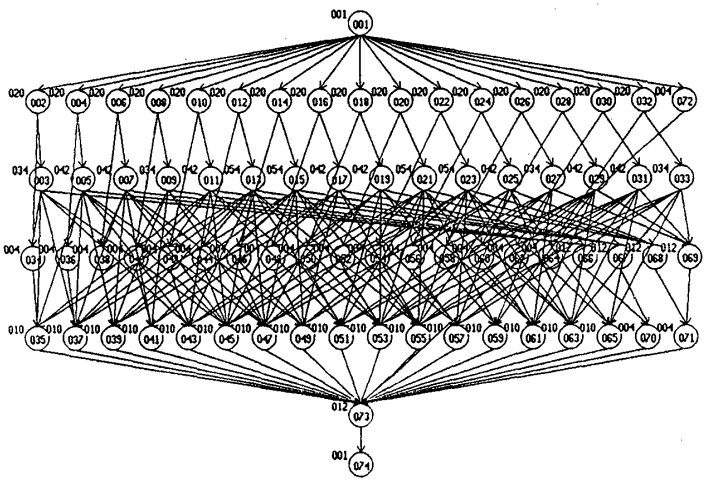

図４　４ｘ４セルのＣＮＮシミュレーションの
タスク・グラフ

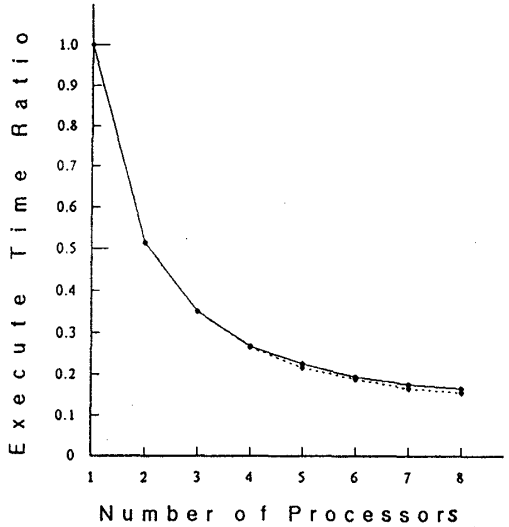

図５　プロセッサ数と並列効果