計算機アーキテクチャ 110−20
設 計 自 動 化 73−20
（1995. 1. 20）

21世紀に向けた新しい汎用機能部品 *PPRAM* — 並列計算モデルの検討 —

村上和彰　岩下茂信　吉井 卓†

九州大学 大学院総合理工学研究科 情報システム学専攻
†: 九州大学 工学部 情報工学科
〒816 福岡県春日市春日公園 6-1

E-mail: {murakami, iwashita, yoshii}@is.kyushu-u.ac.jp

順調に成長を続けるトランジスタ集積度を技術的背景として，(i) 汎用マイクロプロセッサ，(ii) 汎用メモリ，および，(iii) 粗粒度機能メモリの3面性を備えた新しい汎用機能部品 *PPRAM*（*Parallel Processing Random Access Memory*, *Practical Parallel Random Access Machine*）を提案している．*PPRAM* は，(i) 大容量の汎用メモリ（SRAM あるいは DRAM），(ii) 1個以上の汎用プロセッサ，および，(iii) 外部インタフェースを1チップに集積したものである．その方向性は，今日の高性能マイクロプロセッサが進みつつある方向と一致してはいるものの，その目指す製品形態は大きく異なる．本稿では，*PPRAM* の応用形態の1つである「並列 *PPRAM* 構成」に必要な並列計算モデルを検討している．基本的な並列計算モデルである PRAM，そして，より現実的な並列計算モデルである Phase PRAM，BSP，LogP を取り上げ，その *PPRAM* への適用を考察している．

Parallel Computation Models for *PPRAM*

Kazuaki MURAKAMI　Shigenobu IWASHITA
Takashi YOSHII

Department of Information Systems
Interdisciplinary Graduate School of Engineering Sciences
Kyushu University
6-1 Kasuga-koen, Kasuga-shi, Fukuoka 816 Japan

E-mail: {murakami, iwashita, yoshii}@is.kyushu-u.ac.jp

This paper proposes a novel concept of LSI products, called *PPRAM*(*Parallel Processing Random Access Memory*, *Practical Parallel Random Access Machine*), which provides the trinity of conventional LSI products: (i) microprocessors, (ii) DRAM and SRAM, and (iii) coarse-grain functional memory. The *PPRAM*is defined as an LSI which incorporates (i) DRAM or SRAM, (ii) one or more processors, and (iii) external interface logic into a single chip. This paper discusses parallel computation models for *concurrent PPRAMorganization*. Several existing models, such as PRAM, Phase PRAM, BSP, and LogP, are investigated.

1 はじめに – *PPRAM* とは –

我々は,「今日のマイクロプロセッサおよびメモリの高集積化の行き着く先の1つの『製品形態』」として,

1. **汎用マイクロプロセッサ**（シングルプロセッサあるいはマルチプロセッサ）,
2. **汎用メモリ**（SRAM ないし DRAM）, および,
3. **粗粒度機能メモリ**

の3面性を備えた新しい汎用機能部品 *PPRAM*（*Parallel Processing Random Access Memory*, *Practical Parallel Random Access Machine*）を提案している [3].

PPRAM とは,

- 大容量の汎用メモリ（SRAM あるいは DRAM）,
- 1個以上の汎用プロセッサ, および,
- 外部インタフェース

を1チップ（あるいは, MCM (multiple-chip module) 等の1モジュール）に集積したものと定義する. 図1にその論理レベルのブロック図を示す.

上記の定義は *PPRAM* として満たすべき最低限の条件を規定したものであり, 理想的には,

- プロセッサ数に等しいポート数を備えた大容量のマルチポート汎用メモリ（SRAM あるいは DRAM）, および,
- 出来るだけ簡素な汎用プロセッサを2個以上かつ出来るだけ多数個

を現在の RAM 程度の大きさおよびピン数のパッケージに封止した製品形態が望ましい. 図2にそのイメージを示す. *PPRAM* という名称は, この

- *Parallel Processing Random Access Memory*（並列処理する RAM）: 複数個のプロセッサが大容量メモリを共有して並列処理を行なう
- *Practical Parallel Random Access Machine*（実用的な PRAM）: さらに, その共有メモリをマルチポート・メモリとすることで,「理想の並列マシン」と言われる **PRAM** (*Parallel Random Access Machine*) を小規模ながら1チップ上で実現する

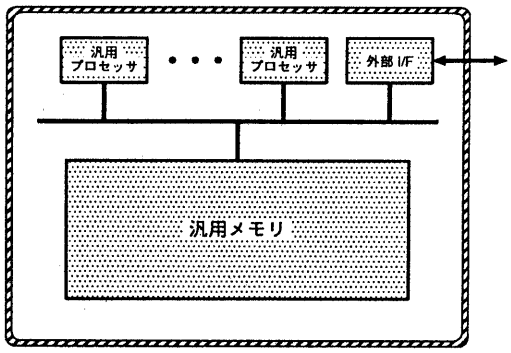

図 1: *PPRAM* の論理ブロック図

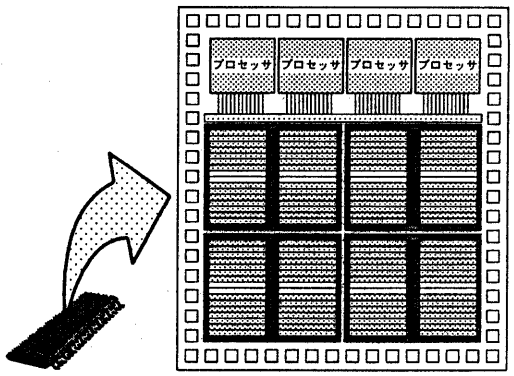

図 2: 理想とする *PPRAM* の製品形態

という理想的な *PPRAM* が有する性質から来ている.

PPRAM の応用は, 以下の通り多岐にわたって可能である [3].

- *PPRAM-and-Associates* 構成
  - マイクロプロセッサ・チップの置換
  - 汎用メモリ・チップの置換
  - 粗粒度機能メモリの追加
  - 完全な仮想記憶の実現
- *All-is-PPRAM* 構成
  - *All-in-One-PPRAM* 構成
  - 並列 *PPRAM* 構成

本稿では, 上記の各種応用のうち**並列 *PPRAM* 構成**を取り上げ, 本構成における並列計算モデルを検討する. 並列 *PPRAM* 構成とは, 図3に示すよ

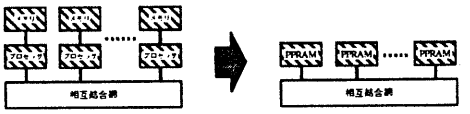

図 3: 並列 *PPRAM* 構成

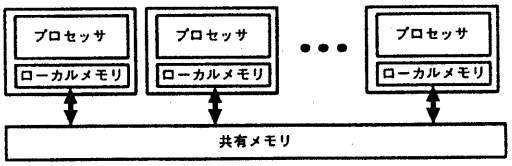

図 4: PRAM

うに，複数の *PPRAM* を相互結合網を介して結合し，並列マシンを構成したものである．

まず，2章で最も基本的な並列計算モデルである PRAM モデルについて述べ，3章でより現実の並列マシンに近いモデルについて述べる．そして，4章で，*PPRAM* に適した並列計算モデルを考察する．

2 並列計算モデル PRAM

2.1 PRAM

PRAM (*Parallel Random Access Machine*) は，並列アルゴリズムを設計する上での並列計算モデルとして良く用いられているモデルである [2].

PRAM は図 4に示すように，複数のプロセッサおよび共有メモリから構成される．各プロセッサは，局所メモリを持った RAM (Random Access Machine) である．各プロセッサは，与えられたプログラムに従って，以下のような命令を実行する．

- 読出し：共有メモリから局所メモリへデータを読み出す．
- 書込み：局所メモリから共有メモリへデータを書き込む．
- 計算：RAM と同様に，局所メモリ内のオペランドを用いて計算を行ない，結果も局所メモリに格納する．

プログラムは 1 番目の命令から順に実行され，各命令は 1 単位時間で実行される．さらに，各プロセッサの命令実行は同期しているものとする．すなわち，いま i 番目の命令を実行しているとすると，全プロセッサが当該命令の実行を終了しない限り，どのプロセッサも $i+1$ 番目の命令の実行に移れない．各プロセッサが同じプログラムを実行していても，異なるデータの下で命令を実行することができる．また，各プロセッサのプログラム・カウンタは異なることがあり得る．

一般に複数のプロセッサが共有メモリに対して読み書きができるので，2 個以上のプロセッサが同時に共有メモリの同一ロケーションに対してアクセスすることがある．このとき，まず読出しを行なったあと，書込みを行なうものとする．さらに，読出しおよび書込みを同時に行なえるプロセッサの数に関して，以下のように分類される．

- **EREW** (*Exclusive Read, Exclusive Write*) PRAM：2 個以上のプロセッサが共有メモリの同一ロケーションに対して同時に読み出したり，書き込んだりするのを許さない．
- **CREW** (*Concurrent Read, Exclusive Write*) PRAM：読出しに関しては，2 個以上のプロセッサが共有メモリの同一ロケーションに対して同時に読み出しするのを許す．書込みに関しては，2 個以上のプロセッサが共有メモリの同一ロケーションに対して同時に書き込むのを許さない．
- **CRCW** (*Concurrent Read, Concurrent Write*) PRAM：2 個以上のプロセッサが共有メモリの同一ロケーションに対して同時に読み出したり，書き込んだりするのを許す．このとき，書込みに関してプロセッサ間で競合が生じるが，これには以下のような解消法が考えられている．
  - PRIORITY 解消法：同時に書き込もうとしているプロセッサのうち，最もプロセッサ番号の小さいものの書込みが行なわれ，他のプロセッサの書込みは行なわれない．
  - ARBITRARY 解消法：同時に書き込もうとしているプロセッサのうち，任意の 1 プロセッサの書込みが行なわれ，他のプロセッサの書込みは行なわれない．
  - COMMON 解消法：同時に書き込もうとしているプロセッサがすべて同じ値を書き込む場合のみ，書込みが行なわれる．

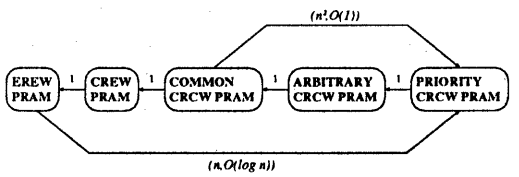

図 5: PRAM 間の関係

EREW PRAM のプログラムはそのまま CREW PRAM 上で同じ計算を行ない，また CREW PRAM のプログラムもそのまま CRCW PRAM 上で同じ計算を行なう．CRCW PRAM に関しては，COMMON CRCW PRAM のプログラムはそのまま ARBITRARY CRCW PRAM でも実行可能である．また，ARBITRARY CRCW PRAM 上で計算できるなら，PRIORITY CRCW PRAM 上でも同じである．図 5に PRAM 間の関係をまとめる．ここで，$A \leftarrow B$は「Bが A を模倣できる」ことを，$(n^2, O(1))$ および $(n, O(\log n))$ は (プロセッサ数, 時間) を，そして 1 は自明な模倣をそれぞれ表す．

2.2 PRAM の問題点

PRAM モデルは非常に単純かつ強力な並列計算モデルであり，共有メモリによりプロセッサ間の通信や同期の詳細を隠蔽しているためアルゴリズムの設計が容易である．しかしながら，あまりにも理想的な並列計算モデルであるがゆえ，実在する並列マシンとのギャップが大きい [4]．そのため，PRAM 上で開発されたアルゴリズムを実際の並列マシン上で実行するには，アルゴリズムそのものを当該マシンに合うように変更するか，あるいは，PRAM モデルを当該マシン上でシミュレーションする必要がある．

PRAM モデルと実在する並列マシンとの間のギャップには，以下のものがある [7]．

- **非同期性** (asynchrony)：PRAM モデルでは，すべてのプロセッサは完全に同期して動作するものとしている．しかし，実在する並列マシンでは，各プロセッサは互いに非同期に動作する．よって，プロセッサ間の同期を取るには，明示的な同期操作が必要となる．

- **相互結合網のトポロジー**：PRAM モデルでは，プロセッサと共有メモリとの間の相互結合網は共有メモリをハブとしたスター網となっている．したがって，相互結合網上で衝突が起こることもなく，また，経路選択も考慮する必要がない．しかし，実在する並列マシンの相互結合網ではスター網以外のトポロジーを採用したものが多く，経路選択を考慮する必要がある．また，相互結合網上で衝突も起こり得るので，データ移動の際に出来るだけ衝突を回避するよう考慮する必要もある．

- **レイテンシ**：PRAM モデルでは，メモリ・アクセス・レイテンシは一定かつ 1 単位時間であるとしている．しかし，実在する並列マシンでは，上述の通りスター網以外の相互結合網を介してメモリ・アクセスを行なうので，メモリ・アクセス・レイテンシが必ずしも一定であるとは限らない．また，階層メモリないし分散メモリの場合，やはりメモリ・アクセス・レイテンシは一定とはならない．

- **メモリ競合**：現在の半導体メモリ技術では，PRAM モデルで想定しているような理想的なマルチポート・メモリを実現するのは難しい．よって，同時読出しや同時書込みは原理的に不可能である．

3 現実的な並列計算モデル

2.2節で述べたように，PRAM モデルと実在する並列マシンとの間には大きなギャップが存在する．このギャップへの対応策としては，以下のようなアプローチが考えられる．

1. PRAM アルゴリズムを実在する並列マシン向きに書き直す．
2. PRAM モデルを実在する並列マシンでシミュレーションする [8]．
3. より現実的な（つまり，実在する並列マシンに対してギャップの小さい）並列計算モデルを策定し，その上で（出来るだけ PRAM アルゴリズムに負けないような）アルゴリズムを開発する [1]．

ここでは，第 3 のアプローチに着目する．PRAM モデルよりもより現実的な並列計算モデルとしては，既にいくつかのものが提案されている [7]．その中から，

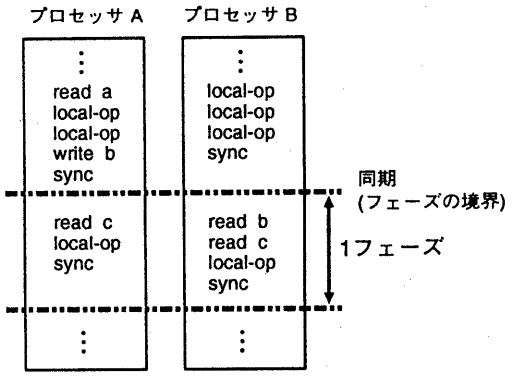

図 6: Phase PRAM のフェーズ

- Phase PRAM
- BSP
- LogP

の 3 つのモデルについて検討する（表 1参照）.

3.1 Phase PRAM

Phase PRAM は，Gibbons が 1989 年に提案した並列計算モデルである [6]. Phase PRAM は PRAM 同様，複数のプロセッサおよび共有メモリから構成され，各プロセッサは局所メモリを有する. PRAM と異なるのは，各プロセッサの命令実行が同期しない非同期 PRAM (asynchronous PRAM) であるという点である.

各プロセッサは，それぞれのローカル・クロックの 1 クロックにつき 1 命令を発行できる. 各命令はある有限時間で終了する. 命令には以下のものがある.

- 読出し：共有メモリから局所メモリへデータを読み出す.
- 書込み：局所メモリから共有メモリへデータを書き込む.
- 計算：RAM と同様に，局所メモリ内のオペランドを用いて計算を行ない，結果も局所メモリに格納する.
- 同期

同期命令以外は，PRAM の命令と同じである. Phase PRAM では，全プロセッサが同期に参加する. 各プロセッサは同期点に達すると，他のすべてのプロセッサが当該同期点に達するまで待つ. すべてのプロセッサが同期点に達したら，各プロセッサは各々のプログラムを再開する. 各プロセッサのプログラムは図 6に示すように，一連のフェーズ (*phase*) から成る. フェーズは同期命令で区切られ，フェーズ内では各プロセッサは独立に動作する. あるフェーズ内の全プロセッサの全命令が終了しないと，どのプロセッサも次のフェーズの命令を発行することは出来ない.

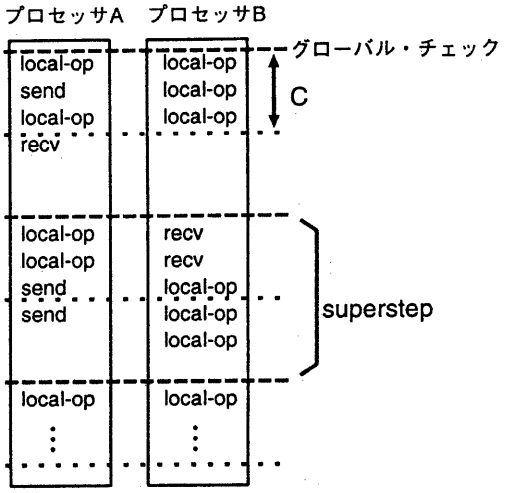

図 7: BSP のスーパステップ

各プロセッサは共有メモリの読み書きを非同期に行なえるが，他のプロセッサが書き込んだロケーションからの読出しは，その間に同期がない限り行なえない.

Phase PRAM では，以下のものをパラメータ化している.

- P：プロセッサ数
- L：通信レイテンシ（文献 [6] では d と表記）
- B：同期コスト

3.2 BSP

BSP (*Bulk-Synchronous Parallel*) は，Valiant が 1990 年に提案した並列計算モデルである [10]. BSP は，各々が局所メモリを有する複数のプロセッサ，相互結合網，および，バリア同期機構から構成される. 各プロセッサの命令実行は非同期に行なわ

表 1: 並列計算モデルの比較

並列計算モデル	PRAM	Phase PRAM	BSP	LogP
構成	• プロセッサ +局所メモリ • 共有メモリ	• プロセッサ +局所メモリ • 共有メモリ • バリア同期機構	• プロセッサ +局所メモリ • 相互結合網 • バリア同期機構	• プロセッサ +局所メモリ • 相互結合網
命令実行	同期	非同期	非同期	非同期
プロセッサ間通信	共有メモリ	共有メモリ	メッセージ交換	メッセージ交換
パラメータ	P:プロセッサ数	P:プロセッサ数 L:通信レイテンシ B:同期コスト	P:プロセッサ数 S:計算速度 C:チェック間隔 T:計算対通信速度比	P:プロセッサ数 L:通信レイテンシ O:通信オーバーヘッド G:通信開始可能間隔

れ，プロセッサ間通信は1対1のメッセージ交換で行われる．

BSP では，以下のものをパラメータ化している．

- P：プロセッサ数
- S：プロセッサの計算速度（文献 [10] では s と表記）
- C：チェック間隔（文献 [10] では L と表記）
- T：計算対通信速度比（文献 [10] では gと表記），すなわち，(1秒間に全プロセッサで実行される計算の数)/(1秒間に相互結合網を流れるメッセージの数)

各プロセッサのプログラムは図 7に示すように，一連のスーパステップ (*superstep*) から成る．スーパステップは，Phase PRAM のフェーズに相当する．1個のスーパステップは，

- 当該スーパステップの開始時点で局所メモリ内の存在していたオペランドを用いた複数の計算
- 複数のメッセージ送受信

から成り，スーパステップ間はバリア同期で区切られる．周期 C毎に，全プロセッサが今のスーパステップを終了しているか否かチェックする．全プロセッサが当該スーパステップを終了していれば次のスーパステップへと進む．

3.3 LogP

LogP (*Latency, overhead, gap, Processors*) は，1993 年に Culler らが提案した並列計算モデルである [5]．LogP は BSP 同様，各々が局所メモリを有する複数のプロセッサおよび相互結合網から構成される．各プロセッサの命令実行は非同期に行なわれ，プロセッサ間通信は1対1のメッセージ交換で行われる．BSP と異なるのは，スーパステップに相当する概念が存在しない，したがってバリア同期機構を持たない，という点である．

LogP では，以下のものをパラメータ化している．

- P：プロセッサ数
- L：通信レイテンシの上界
- O：通信オーバーヘッド（文献 [5] では o と表記）
- G：通信開始可能間隔（文献 [5] では gと表記）

4 *PPRAM*に適した並列計算モデル

並列 *PPRAM* 構成は，図 8に示すようなクラスタ構造を採る．クラスタ内とクラスタ間とでは，以下のような相違がある．

- クラスタ内：複数のプロセッサが1個のメモリを共有し，プロセッサ間通信はこの共有メモリを介して行なう．メモリ・バンド巾はクラスタ間に比べて，かなり大きくすることが出来る．また，プロセッサ数に見合っただけのポート数を備えたマルチポート・メモリにすることも可能で，この場合（キャッシュ・メモリを設けなければ）メモリ・アクセス・レイテンシを一定にすることも可能である．ただ

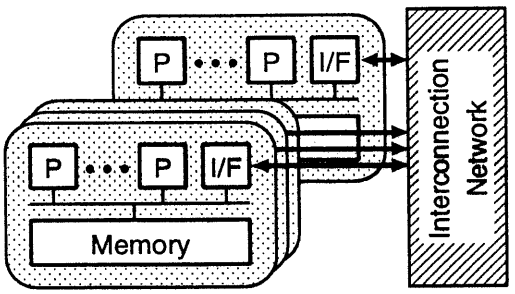

図 8: 並列 *PPRAM* 構成のクラスタ構造

し，メモリの容量対面積比を考えると，メモリ自身はシングルポートにしておき，各プロセッサにキャッシュを設ける方が現実的かも知れない．この場合，メモリ・アクセス・レイテンシは一定とならない．いずれにせよ，オンチップ・マルチプロセッサであっても，各プロセッサの命令実行を同期させるのは困難である．

- クラスタ間：クラスタ間通信モデルは，共有メモリ・モデルないしメッセージ交換モデルのいずれでも可．いずれにせよ，クラスタ間通信バンド巾はクラスタ内メモリ・バンド巾に比べてかなり小さくなり，同時に，クラスタ間通信レイテンシはクラスタ内メモリ・アクセス・レイテンシに比べてかなり大きくなる．

以上から，*PPRAM*用の並列計算モデルとしては，

- クラスタ内は，Phase PRAM のような asynchronous PRAM モデル
- クラスタ間は，
  - クラスタ間通信モデルが共有メモリ・モデルなら，クラスタ内と同じ asynchronous PRAM モデル（ただし，そのパラメータ値は，クラスタ内のパラメータ値とは自ずと異なる）
  - クラスタ間通信モデルがメッセージ交換モデルなら，BSP ないし LogP に類似した並列計算モデル

といった2階層の並列計算モデルが適していると考える．我々は，これを**クラスタ化PRAM** (*clustered PRAM*) と呼び，現在その開発を進めている．

また，クラスタ間通信モデルがメッセージ交換モデルでもプログラミング・インタフェースとしては共有メモリ・モデルをランタイム・システムにより提供することが可能である．すなわち，「共有メモリ・インタフェース on メッセージ交換モデル」，一般に**共有仮想メモリ** (*shared virtual memory*) [9] と呼ばれているものである．この技術を用いれば，

- クラスタ内通信モデルは共有メモリ・モデル
- クラスタ間通信モデルはメッセージ交換モデル

と異なっていても，並列計算モデルとしてはクラスタの内外を問わず共通のモデル（たとえば，Phase PRAM のような asynchronous PRAM モデル）を提供することが可能である．

5 おわりに

以上，*PPRAM*用の並列計算モデルを検討した．現在，我々は並列計算モデルに関して，

- クラスタ化 PRAM モデルの開発
- 「共有メモリ・インタフェース on メッセージ交換モデル」の各種実装方式の評価

を進めている．

さらに，上記以外に，

- DRAM や SRAM といった大容量メモリとプロセッサとを同一チップ上に混載する上での問題点の洗い出し
- *PPRAM*（および，その並列計算モデル）に適したプロセッサ・アーキテクチャの検討
- 汎用プロセッサ以外の専用プロセッサの組込みに関する検討
- 外部インタフェースの論理的仕様（これは，並列計算モデルに依る），電気的仕様，および，機械的仕様の策定ならびに標準化

といった作業を今後行なっていく予定である．

謝辞

日頃から御討論頂く，九州大学 大学院総合理工学研究科 安浦寛人 教授，岩井原瑞穂 助手，および，安浦研究室の諸氏に感謝致します．

貴重なご意見を頂いた (株) 富士通研究所 和田英一 博士, 木村康則 博士, (株) 東芝 増田英司 氏, および, 電子技術総合研究所 関口智嗣 氏に感謝致します.

参考文献

[1] 岩間一雄, "PRAM 上の並列アルゴリズム," 情報処理, vol.33, no.9, pp.1033–1041, 1992 年 9 月.

[2] 宮野 悟, 並列アルゴリズム, 第 2 章, pp.39–67, 近代科学社, 1993 年.

[3] 村上和彰, 吉井 卓, 岩下茂信, "21 世紀に向けた新しい汎用機能部品 *PPRAM* の提案," 情処研報, ARC–108–8, 1994 年 10 月.

[4] 安浦寛人, "並列計算機と並列計算モデル," 情報処理, vol.33, no.9, pp.1024–1032, 1992 年 9 月.

[5] Culler, D. et al., "LogP: Towards a Realistic Model of Parallel Computation," *Proc. 4th Symp. Principles & Practice of Parallel Programming (PPOPP)*, pp.1–12, May 1993.

[6] Gibbons, P. B., "A More Practical PRAM Model," *Proc. 1st Symp. Parallel Algorithms and Architectures (SPAA)*, pp.158–168, June 1989.

[7] Goodrich, M. T., "Models of Computation," *ACM SIGACT News*, vol.24, no.4, pp.16–21, Dec. 1993.

[8] Harris, T. J., "A Survey of PRAM Simulation Techniques," *ACM Computing Surveys*, vol.26, no.2, pp.186–206, June 1994.

[9] Li, K. and Hudak, P., "Memory Coherence in Shared Virtual Memory Systems," *ACM Trans. Computer Systems*, vol.7, no.4, pp.321–359, Nov. 1989.

[10] Valiant, L. G., "A Bridging Model for Parallel Computation," *CACM*, vol.33, no.8, pp.103–111, Aug. 1990.