計算機アーキテクチャ 91−14
（1991. 11. 22）

IEEE標準バスFuturebusのバスアービタの性能評価

山本 欧　鳥居 淳　天野英晴

慶應義塾大学 理工学部

IEEEの標準バスであるFuturebusのバスアービトレーション・プロトコルについて、理論解析と実測データの両面から性能評価を行なう。理論解析では、バスアービトレーション・プロトコルをマルコフ・チェーンによってモデル化し、アービトレーションに伴う時間的ロスについて評価した。実測データによる解析では、Futurebusの実装された並列計算機から、ハードウェアモニタによってデータを収集し、理論解析の結果と比較しながらアービトレーションの時間的ロスや公平性等について評価した。その結果アービトレーションの時間的ロスは、プロセッサ数が小さい場合を除き問題にならない程度であるが、Futurebusのフェアネス機構は公平性の観点からも、性能に対する影響についても多少の問題があることが明らかになった。

A performance evaluation of the arbitor of IEEE standard backplane bus: Futurebus

Ou Yamamoto　Sunao Torii　Hideharu Amano

Faculty of Science and Technology, Keio University

The performance of IEEE standard backplane bus, Futurebus (IEEE986.1), is evaluated both by a theoritical model and measurement of the actual parallel machine.

The theoritical model is based on the Marcov-chain. Using this model, the behavior of the arbitor is analyzed and discussed. The overhead of the arbitration, efficiency of the fairness release mechanism, and parking ratio are measured using a multiprocessor testbed, ATTEMPT-0, with some application programs. These results of the measurement are compared with the results by the theoritical model.

Through the evaluation, it appears that the performance of the arbitor is sufficient exception in a system with small numbers of processors, but the fairness release mechanism has some problems.

1 はじめに

Futurebus は、国際標準として IEEE から提案され、1987 年に IEEE P.896 として承認されたシステム・バス規格である [1]。Futurebus は、マルチプロセッサ・システムでの使用が前提とされている高性能なバスであり、我々が開発した並列計算機テストベッド ATTEMPT-0 のバックプレーン・バスに採用されている。本報告では、システム・バスの性能に対し大きな比重を占める、バスアービトレーション・プロトコルに焦点を当て、このプロトコルの性能評価を理論解析と実測データの両面から行なう。理論解析では、バスアービトレーション・アルゴリズムをマルコフ・チェーンによってモデル化し、アービトレーションに伴う時間的ロスについて評価する。実測データによる解析では、ハードウェアモニタによって ATTEMPT-0 からデータを収集し、理論解析の結果と比較しながらアービトレーションの時間的ロスや公平性等について評価する。

2 Futurebus のアービトレーションプロトコル

Futurebus では、図 1 に示すようなオープンコレクタを用いたアービトレーション回路がバスマスタ候補毎に存在する。

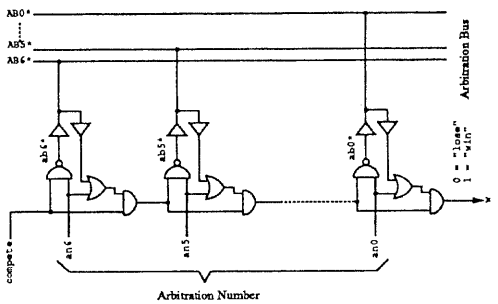

図 1: Futurebus のアービトレーション回路

各バスマスタ候補には、システム内でユニークな 7 ビットのアービトレーション番号 ($an0 \sim an6$) が割り当てられている。このうち $an6$ は、後に述べるように飢餓状態回避等の制御において使用される特別なビットであり、$an0$ はパリティビットとして用いられるため、ユニークなアービトレーション番号として意味を持つのは $an1 \sim an5$ の 5 ビットである。したがってバスマスタ候補の数は最大 32 となるが、Futurebus では電気的制約からバスマスタの数は最大 20 に制限している。バスマスタ候補は、バス要求を検出すると信号線 compete をアクティブにしてアービトレーションを開始する。この信号によって、アービトレーション番号がアービトレーション・バス上に出力される。この際、アービトレーション番号はアクティブロウ・レベルに変換されて出力される。Futurebus では、各バスマスタがバスライン上に出力する値を小文字で表わし、その結果としてバスライン上に現れる値を大文字で表わす。また、アクティブロウ・レベルであることを示すために信号名の右上に記号 (*) をつける。したがって、アービトレーション番号 $an0 \sim an6$ は、アクティブロウ・レベル $ab0^* \sim ab6^*$ に変換されて出力される。そして、各バスマスタの出力した $ab0^* \sim ab6^*$ をワイアード・オアした値が、アービトレーション・バス上に現われる値 $AB0^* \sim AB6^*$ となる。アービトレーション番号は上位桁 $ab6^*$ から順に比較され、一定のセトリング時間の後、最も高いアービトレーション番号を出力したバスマスタ候補の信号線 w がアクティブになり、アービトレーションに勝ったことを示す。

上に述べたプロトコルのみでは、アービトレーション番号の低いバスマスタ候補にバスマスタになる機会が与えられない状態（飢餓状態）が生じる可能性がある。Furutebus では、フェアネスクラスというクラスをバスマスタ候補に対して用意することによって飢餓状態を回避している。これはフェアネス機構と呼ばれる。バスマスタ候補は、アービトレーション番号の最上位桁 ($an6$) が 0 のとき、かつそのときのみフェアネスクラスに属する。これに対し、$an6$ が 1 である全てのバスマスタ候補の属するクラスを、プライオリティクラスという。フェアネスクラスに属するバスマスタ候補には、次の制約が課せられる：

(1) 一度バスマスタになったことのあるバスマスタ候補は、続けてアービトレーションに参加することを禁止される。(これを再アービトレーションの禁止と呼ぶ。) 再びアービトレーションに参加するには、この禁止を解除する必要がある。

(2) 再アービトレーションの禁止を解除できるのは、自分以外にアービトレーションを開始するバスマスタ候補が存在していないときか、存在していてもそれらが全てフェアネスクラスに属し、かつ既にバスマスタになったことがある場合のみである。

(1),(2) の制約によって、フェアネスクラス内での飢餓状態が回避される。

Futurebus では、アービトレーションに関するシーケンスとして次の 2 つがある：

1. アービトレーションサイクル：アービトレーションによってバスマスタを決定するシーケンス
2. フェアネス解放サイクル：フェアネスクラスのバスマスタ候補が、再アービトレーションの禁止を解除するシーケンス

Futurebus ではこれら 2 つのシーケンスを制御するため、シーケンスをオペレーションと呼ばれる動作に分割している。オペレーションは全部で 6 種類あり、アービトレーションサイクルではオペレーション 1 からオペレーション 6 までが行なわれ、

フェアネス解放サイクルではオペレーション1からオペレーション3までが行なわれる。

各オペレーションにおいて、バスマスタ候補は次の7つの状態間を遷移する。

- current master (CM):現バスマスタ
- free bystander (FB):目下バスマスタになる必要がなく、かつ再アービトレーションを禁止されていない状態
- inhibited bystander (IB):再アービトレーションを禁止されているが、目下バスマスタになる必要のない状態。
- competitor (C):再アービトレーションを禁止されておらず、かつバスマスタになろうとしている状態。
- withholder (W):再アービトレーションを禁止されており、かつバスマスタになろうとしている状態。
- master elect (ME):アービトレーションで勝利し、current master がバス使用権を譲ってくれるのを待っている状態。
- recompeting master (RM):バスロックを解除するために、再度アービトレーションを開始した current master。

図2に、バスマスタ候補の状態遷移を示す。

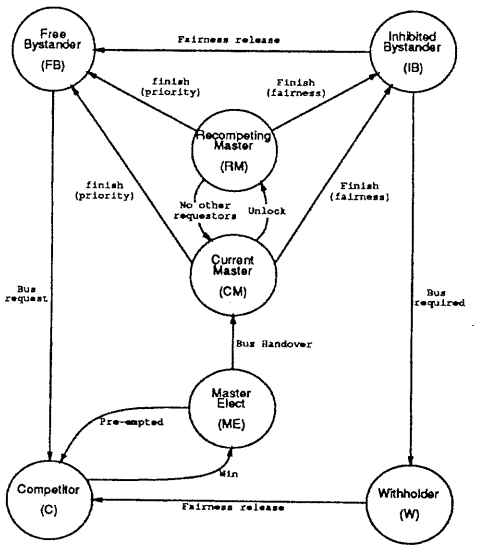

図 2: バスマスタ候補の状態遷移

3 Futurebus バスアービトレーション・プロトコルのマルコフ解析

本節では、前節で述べた Futurebus のアービトレーション・プロトコルの理論解析について述べる。

3.1 Futurebus のアービトレーション・プロトコルのモデル化

Futurebus を用いたシステムでは、バスマスタ候補は2節で述べた7つの状態をとる。しかし、マルコフ解析に要する計算量を考えると、バスマスタ候補のとる状態数は少ない方がよい。そこで、状態数を減らすために次の3つのことを仮定する：

(1) バス・ロックの機能は使わない。(ここでは複数のシステム・バスを持つシステムを想定しない。)

(2) バスマスタの先取り、および緊急メッセージの放送は、ないものとする。(これらが発生する確率はきわめて小さいと考えられるから。)

(3) バスマスタ候補は、すべてフェアネスクラスに属するものとする。(フェアネス機構の評価に重点を置くため。)

仮定(1)により、状態RMは省略することができる。また、仮定(2)により、状態MEとなったモジュールは現バスマスタの後に必ずバスマスタになることができるので、状態MEは省略することができる。さらに、仮定(3)より、状態CMにあるモジュールは、バスマスタから退いた後は必ずIBに遷移するので、CMからFBへの遷移は省略できる。

以上の仮定に従い作成したマルコフ解析モデルを図3に示す。

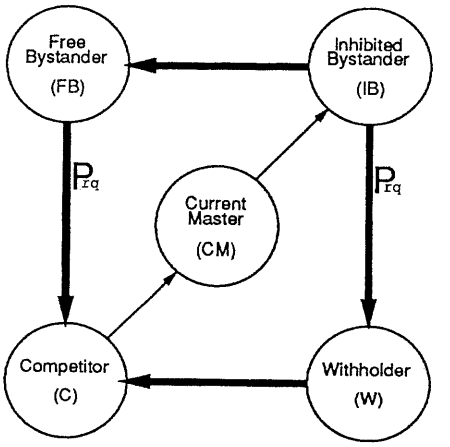

図 3: マルコフ解析のモデル

バスマスタ候補の数はNとし、各状態にあるバスマスタ候補の個数を次のように定める。

- 状態FBにあるバスマスタ候補の数： fb
- 状態Cにあるバスマスタ候補の数： c
- 状態CMにあるバスマスタ候補の数： cm
- 状態IBにあるバスマスタ候補の数： ib

- 状態 W にあるバスマスタ候補の数： w
- $fb+c+cm+ib+w=N$

全バスマスタ候補は、単位時間毎に状態が遷移する。これは、図3のモデル中をバスマスタ候補が移動することに相当する。バスマスタ候補は、各ノードに単位時間留まる。また、太線の矢印は、バスマスタ候補が単位時間中に任意数同時に通過できることを示し、細線の矢印はバスマスタ候補が1つだけ単位時間中に通過できるもことを示す。以下、この単位時間の刻みをステップと呼ぶ。

いま、状態FBにある1つのバスマスタ候補は、次のステップにおいて、確率 P_{rq} で状態Cに遷移するものとする。すなわち、P_{rq} は、バスマスタ候補がバス使用権の要求を出す確率である。また、状態IBにある1つのバスマスタ候補も、次のステップにおいて、確率 P_{rq} で状態Wに遷移するものとする。

バスマスタの数は2以上になることはないので、状態CMにあるバスマスタ候補の数 cm は1または0である。また、状態Cにあるバスマスタ候補の数 c が0でない場合、次のステップにおいて、1つのバスマスタ候補が必ず状態CMに遷移する。このとき、状態CMにあったバスマスタ候補は、後からきたバスマスタ候補に押し出される形で状態IBに遷移する。

状態Wにあるバスマスタ候補は、$c=0$ の場合にのみ、次のステップにおいて、すべてCに遷移する（これはフェアネス解放サイクルを表す）。c が0でない場合はそのままWに留まる。 状態IBにあるバスマスタ候補は、$c=0$ かつ $w\neq 0$ であった場合にのみ、次のステップですべてFBに遷移し（フェアネス解放サイクル）、それ以外の場合はそのままIBに留まる。

3.2 マルコフ解析

各バスマスタ候補の状態を個々に考えると、3.1節で述べたような5つがあるが、バスマスタ候補の各状態間に相互作用があるため、システム全体の状態をなんらかの方法で特徴づける必要がある。そこで、5つの状態にあるバスマスタ候補の数の組 (fb,c,cm,ib,w) によって全体の状態を表すことにする。

ところで、$fb+c+cm+ib+w=N$ の関係が成り立つことから、fb, c, cm, ib, w のうちどれか4つが決まればシステム全体の状態が決まる。ここでは fb, c, dm, ib の4つの組 (fb, c, cm, ib) でシステムの状態を表すことにする。

バスマスタ候補の数をNとすると、状態数は

$cm=0$ のとき：

$$\begin{pmatrix} (N+1)+(4-1)-1 \\ (4-1) \end{pmatrix}$$

$cm=1$ のとき：

$$\begin{pmatrix} ((N-1)+1)+(4-1)-1 \\ (4-1) \end{pmatrix}$$

従って全部で：

$$\frac{1}{6}(N+1)(N+2)(2N+3)$$

となる。

次に、2つの状態 I： $(fb,c,cm,ib)=(fb_i,c_i,cm_i,ib_i)$ と J： $(fb,c,cm,ib)=(fb_j,c_j,cm_j,ib_j)$ 間の遷移確率行列を考える。このとき、状態 I が状態 J に遷移する確率 P_{IJ} は以下のように求めることができる。：

いま、あるステップにおいて、システムが状態 I を取ったとする。そして次のステップで x 個のバスマスタ候補が状態FBから状態Cに遷移し、y 個のバスマスタ候補が状態IBから状態Wに遷移したとする。このとき次の式が成立する。

(1) $c_i>0$ のとき（バスマスタの交代が行なわれる）：

$$\begin{cases} fb_j=fb_i-x \\ c_j=c_i-1+x \\ m_j=1 \\ ib_j=ib_i-y+cm_i \\ w_j=w_i+y \end{cases}$$

(2) $c_i=0$ かつ $w_i>0$ のとき（フェアネス解放サイクルが行なわれる）：

$$\begin{cases} fb_j=fb_i-x+ib_i \\ c_j=x+w_i \\ cm_j=cm_i \\ ib_j=0 \\ w_j=0 \end{cases}$$

(3) $c_i=0$ かつ $w_i=0$ のとき（FBからC、およびIBからWへの遷移のみが行なわれる）：

$$\begin{cases} fb_j=fb_i-x \\ c_j=x \\ cm_j=cm_i \\ ib_j=ib_i-y \\ w_j=y \end{cases}$$

(1),(2),(3) の各場合について x,y を計算し、次式によって状態 I から状態 J への遷移確率 P_{IJ} を計算する。：

(1),(3) の場合、 $x\geq 0$, $y\geq 0$, $fb_i\geq x$, $ib_i\geq y$ が成立する状態に対しては、

$$P_{IJ}=\begin{pmatrix} fb_i \\ x \end{pmatrix}\times(P_{rq})^x\times(1-P_{rq})^{fb_i-x}$$
$$\times\begin{pmatrix} ib_i \\ y \end{pmatrix}\times(P_{rq})^y\times(1-P_{rq})^{ib_i-y}$$

(2) の場合、$x\geq 0$, $y\geq 0$, $fb_i\geq x$, $ib_i\geq y$が成立する場合に対しては、

$$P_{IJ}=\begin{pmatrix} fb_i \\ x \end{pmatrix}\times(P_{rq})^x\times(1-P_{rq})^{fb_i-x}$$

(1),(2),(3) の各場合で、上記以外の場合に対しては、

$$P_{IJ} = 0$$

遷移確率行列 S は、この P_{IJ} を要素とする正方行列となる。初期状態は、全バスマスタ候補が FB にある状態を I_0 : $(fb, c, cm, ib, w) = (N, 0, 0, 0, 0)$ とすると、状態 I_0 に対応する要素が 1 である状態確率横ベクトル

$$s_0 = (0, 0, ...0, 1, 0, ...0)$$

で表される。この S および s_0 を用いて、次の漸化式から k ステップ目の状態確率横ベクトル s_k を求めることができる。

$$\begin{cases} s_1 = s_0 \times S \\ s_{k+1} = s_k \times S \quad k = 1, 2, \cdots \end{cases}$$

3.3 マルコフ解析による性能評価

ここでは、前節で説明した解析モデルを使って、Futurebus のアービトレーションプロトコルの性能評価を行なう。ここでは評価項目として、アービトレーションサイクルとフェアネス解放サイクルの発生頻度、および状態 W にあるバスマスタ候補の数の期待値を選んだ。フェアネス解放サイクルは飢餓状態を回避するために必要である一方、時間的ロスを生じるため、これがどの程度の頻度で起こるか、また幾つのバスマスタ候補がフェアネス解放を待っているかを知ることが、アービトレーションプロトコルを評価する上で重要となるからである。

いま、状態確率横ベクトル s_k を考える。このベクトルの要素で、状態 (fb, c, cm, ib, w) に対応するものを $P_{fb,c,cm,ib,w}$ で表すとすると、次のステップでフェアネス解放サイクルが起こる確率 P_{fr} は、$c = 0$ かつ $w > 0$ である状態の $P_{fb,c,cm,ib,w}$ の和

$$P_{fr} = \sum_{c=0 \ \wedge \ w>0} P_{fb,c,cm,ib,w}$$

で表される。同様に、アービトレーションサイクルの起こる確率 P_{arb} は、$c > 0$ である状態の $P_{fb,c,cm,ib,w}$ の和

$$P_{arb} = \sum_{c>0} P_{fb,c,cm,ib,w}$$

となる。また、状態 W にあるバスマスタ候補の数の期待値 Ew は、

$$Ew = \sum w \times P_{fb,c,cm,ib,w}$$

で表される。

P_{rq} をパラメータとしたときの、P_{fr} のステップ k に対する変化を図 4 に示す。また、P_{arb}, Ew についての同様の変化を、それぞれ図 5、図 6 に示す。

どの図においても、プロセッサ数 N は 10 として計算してある。まず、図 4、図 5 について見ると、次のことがわかる：

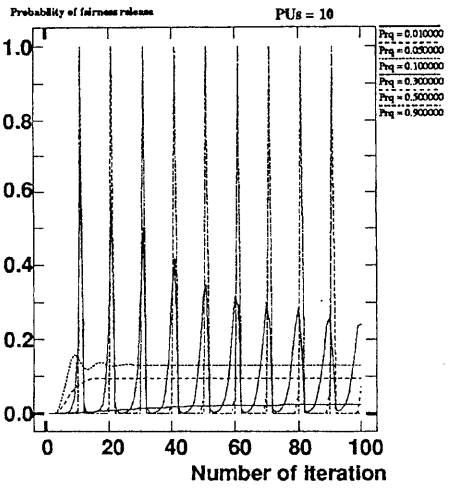

図 4: フェアネス解放サイクルの起こる確率

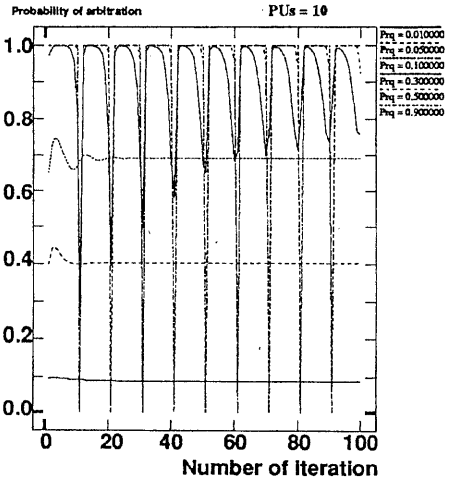

図 5: アービトレーションサイクルの起こる確率

- P_{rq} が小さい場合、P_{fr}, P_{arb} は速やかに一定値に収束するが、P_{rq} が大きくなるにつれて振動を始め、$P_{rq} = 0.90$ では定常的に振動するようになる。
- 振動の周期はほぼ 10 ステップである。定常振動時には最大値は 1、最小値は 0 となり、10 ステップ毎に必ずフェアネス解放サイクルが起こる。
- $P_{rq} = 0.01$, 0.05, 0.10 のそれぞれの場合の収束値について、アービトレーションサイクルに対するフェアネス解放

サイクルの割合 R_{fr} ($= P_{fr}/P_{arb}$) を計算すると、

- $P_{rq} = 0.01$ のとき： $R_{fr} = 0.32$
- $P_{rq} = 0.05$ のとき： $R_{fr} = 0.25$
- $P_{rq} = 0.10$ のとき： $R_{fr} = 0.20$

となり、フェアネス解放サイクルは 20% ～ 30% の割合で起こっている。これは、定常振動時の割合よりも大きな値である。

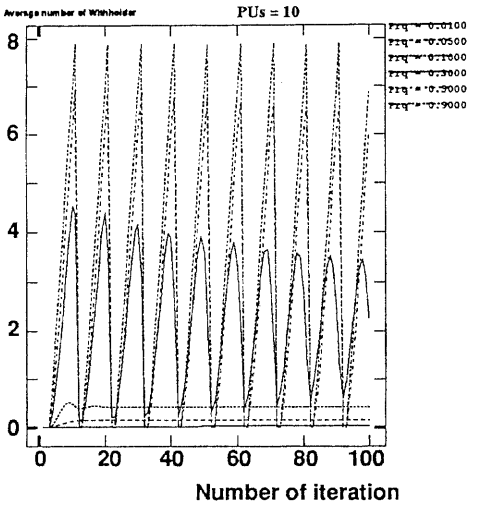

図 6: フェアネス解放サイクルを待つバスマスタ候補の平均値

次に、図6について見ても、やはり P_{rq} の増加と共に振動を始める傾向が見られる。振動の周期はほぼ10 ステップで、定常振動時の最大、最小値はそれぞれ 8、0 である。

以上の結果より次のことがわかる：

- P_{rq} が小さい場合は、フェアネス解放サイクルが比較的高い割合で定常的に発生し、アービトレーションの時間的ロスは大きい。
- P_{rq} が大きくなると、フェアネス解放サイクルは P_{rq} の小さいときより小さい割合で周期的に発生するようになり、アービトレーションの時間的ロスは小さくなる。

4 実測環境

4.1 並列計算機 ATTEMPT-0

実測評価には我々が開発した並列計算機テストベット ATTEMPT-0 [2] を用いた。ATTEMPT-0 は Futurebus を用いたバス結合型マルチプロセッサで、各プロセッサ (68030+68882) はローカルメモリと、ライトスルーキャッシュを持つ。プロセッサ間の同期はシンクロナイザ [3] と呼ばれる一種のブロードキャストメモリによって行なわれる。表1にシンクロナイザ及びキャッシュアクセスに必要な時間を示す。

アクセスの種類	時間	転送 byte	バスアクセス
Sync. Read	200 ns	4 byte	なし
Sync. Write	650 ns	4 byte	あり
Sync. Fetch&Dec	750 ns	4 byte	あり
Cache Read hit	150 ns	4 byte	なし
Cache Read miss	18200 ns	64 byte	あり
Cache Write	750 ns	4 byte	あり

表1 アクセス時間

ATTEMPT-0 では、Futurebus のアービタは PLA による同期型順序回路で構成している。アービトレーションにに要する時間は 880ns であるが、フェアネス解放を伴う場合には、1150ns 要する。また、ATTEMPT-0 ではハードウェアイベントモニタによって Futurebus やキャッシュ等に関する種々のデータを測定することが可能である。

4.2 評価用プログラム

以下のプログラムを ATTEMPT-0 上で実行し、アービタの実際の振舞いを測定した。

GAUSS ガウスの消去法により連立一次方程式の解を求めるプログラム

CG 共役勾配法により連立一次方程式の解を求めるプログラム

LOGIQUE 並列論理シミュレータ
Chandy-Misra の問い合わせを用いた分散時刻管理に基づく並列論理シミュレータ [4]。

今回の実装では GAUSS、CG は共有データを全てシンクロナイザ、LOGIQUE は頻繁にアクセスされる共有データをシンクロナイザに配置した。

5 評価

5.1 解析モデルと実測評価との比較

前節で述べた解析モデルは、次の点で実際のシステムとは異なっている。：解析モデルでは、全てのバスマスタ候補が一定の時間の刻み（ステップ）に同期して状態遷移を行なう。特に、バスマスタ候補がバスマスタ状態（CM）に留まる時間は、フェアネス解放サイクルの場合を除いて1ステップ時間のみである。これに対し、実際のシステムでは、2章で述べたオペレーションに同期して状態遷移が行なわれ、オペレーションに要する時間は実行に依存し一定でない。また、バスマスタ候補が CM に

留まる時間も一定でない。したがって、解析モデルによる評価結果が、実測評価にどれだけ近いかを調べる必要がある。

表2は、ATTEMPT上で実測した P_{rq} の値、およびアービトレーションサイクルに対するフェアネスリリースサイクルの割合 R_{fr} を、アプリケーション毎にまとめたものである。これより、実際のアプリケーションでは、P_{rq} の値は0.01前後であることがわかる。したがって、マルコフ・チェーンによる解析結果より、比較的高い割合でフェアネス解放サイクルが発生していると予想される。そこで、R_{fr} に注目すると、12%～24%の範囲であり、ほぼ予想通りの値になってる。

アプリケーション	P_{rq}	R_{fr}
GAUSS(8PU)	0.0180	0.164
GAUSS(10PU)	0.0136	0.128
CG(8PU)	0.0038	0.150
LOGIQUE(8PU)	0.0095	0.248

表2 P_{rq}, R_{fr}の実測値

以上の点から解析モデルは実際の系の挙動を予測するために有用であることが確かめられた。

5.2 アービトレーションが実行時間に与える影響

表2に示されるように、今回のアプリケーションでは P_{rq} が、バスアービタが過負荷で振動状態になる程大きくはない。これはひとつには、今回のアプリケーションがいずれも、コードとローカル変数をローカルメモリに置いたため、バスに対する負荷が軽くなっているためである。プロセッサ数が増大するとともにバスに対する要求は増大するが、バスの待ち時間自体の増大と、同期待ち時間の増大により、通常 P_{rq} は一定値以上は大きくならない。

次にアービトレーション自体が実行時間に与える影響を考える。アービトレーションはバス上のデータ交換とオーバラップされるため、実行時間に対する直接の影響を調べることは難しい。ここでは、目安として、アービトレーション自体の実行時間が計算時間に占める割合を調べたところ、最もバスに対する負荷の大きいGAUSSで4.5%(2PU)、1.4%(10PU)であった。すなわち、プロセッサ数の小さい範囲でアービトレーション時間の影響は大きくなっている。これはプロセッサ数が少ないと順番にバスをアクセスするため、フェアネスリリースが起きる頻度が高いことが一因になっている。

5.3 フェアネス機構の公平性

フェアネス機構はプロセッサの飢餓状態は避けることができるが、各プロセッサがバスを獲得できる機会は、アービトレーション番号が大きい方が多い。すなわち弱公平性のみを保証する方法である。そこで、この不公平がどの程度実行時間に影響を与えるかを調べた。図7にアービタに対する優先順位が最強(Strong)と最弱(Weak)のプロセッサのバス待ち時間の相違を示した。アプリケーションは最もバスに対する負荷の大きいGAUSSである。

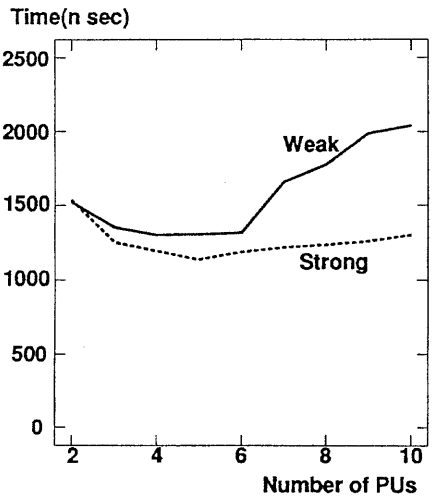

図7: アービトレーションの公平性

この結果によると、10プロセッサの場合最弱のプロセッサは最強のプロセッサに対して約2倍待たされることがわかる。このアプリケーションにおいてはバスの混雑がさほど大きくないため、待ち時間の増大は全体の計算時間の約1%にすぎないが、バスの混雑がひどくなるほど大きくなる。以上の点で、フェアネス機構は公平性の維持の点でもやや問題があることがわかる。

5.4 パーキング率

あるプロセッサがバスを使用し、他のプロセッサがバスの使用権の要求を出す前に再びそのプロセッサがバスを使用する場合は、マスタ権は移動する必要がない。このような場合にアービトレーションを省略するために、Futurebusではマスタは転送終了後もマスタ権を仮に保持しておくことができる。この機構をFuturebusではパーキングと呼び、パーキングによりアービトレーションを省略できた割合を、ここではパーキング率と呼ぶ。ATTEMPT-0上で3つのアプリケーションを実行した場合のパーキング率を図8に示す。パーキング率はプロセッサがどの程度連続にバスを要求するかに依存するため、アプリケーションによって大幅に異なる。図8中のCGとGAUSSは行列計算であるため、各プロセッサは平均的に頻繁にバスに要求を出す。このため、プロセッサ数にあまり依存せず10%～20%程度であまり大きくない。プロセッサ数が増加するにつれパーキング率はやや上がる傾向にあるが、これはプロセッサの受け

持つ行列のサイズが小さくなるにつれ、アルゴリズム上の待ちが大きくなり、バス要求の頻度が小さくなることによる。

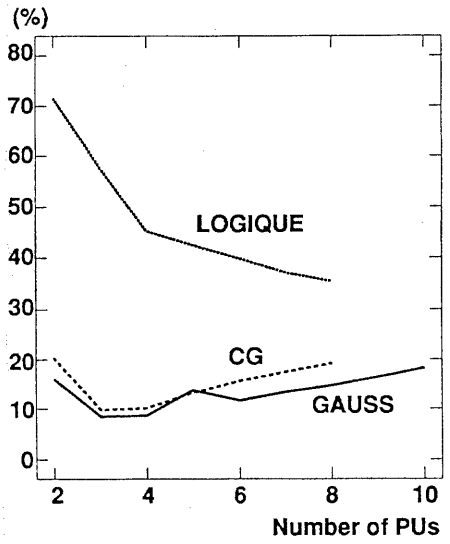

図 8: バス・バーキング率

並列論理シミュレータ LOGIQUE は、プロセッサが連続して領域をアクセスするため、プロセッサ数が小さい状態ではバーキング率は 70%にも達している。ただし、プロセッサ数が大きくなると低下し、8 プロセッサでは 40%程度になる。

以上の結果により、行列計算のようなアプリケーションでもバーキング率は 10%以上を維持し、問題によっては 70%にも上る。バーキングの効果は、アービトレーションの時間的損失が効率に影響するようなプロセッサの数が小さい場合に大きく、バーキング機構は備える価値が十分あることがわかる。

6 むすび

IEEE 標準バスである Futurebus のアービトレーションプロトコルについて、理論解析と実測値の両面から性能評価を行なった。その結果、飢餓状態回避のためのフェアネス機構は、アービトレーションの頻度が小さい場合に時間的なロスを生じること、および公平性の点で多少の問題があることがわかった。一方、バーキング機構については、実際のアプリケーションにおいてその有用性が確かめられた。

7 謝辞

本報告にあたり、数々の御指導を頂いた慶應義塾大学計算機科学専攻天野研究室の皆様に感謝致します。

参考文献

[1] *"IEEE Standard Backplane Bus Specification for Multiprocessor Architectures: Futurebus"*, IEEE Jun., 1988

[2] 鳥居 淳, 天野英晴, "並列計算機テストベッド ATTEMPT の交信機構の評価", 並列処理シンポジウム JSPP '91 論文集, pp.205-212, May 1991

[3] H.Amano, T.Terasawa, and T.Kudoh *"Cache with Synchronization Mechanism"*, Proc. of 11th IFIP, pp. 1001-1006, 1989

[4] 工藤, 木村, 寺沢, 天野, "共有メモリを想定した並列論理シミュレータ" 信学技報 CPSY91-23, Jun. 1991.