ハードウェア化を考慮したマルチメディア指向 ATMスイッチセルスケジューリングアルゴリズム

鈴木 健一[†], 大庭 信之[‡], 小林 広明[†], 中村 維男[†]

東北大学大学院情報科学研究科[†], 日本アイ・ビー・エム東京基礎研究所[‡]

Abstract

本報告は、高速ATMスイッチングシステム "StarCore" を提案する。"StarCore" は、ハードウェアだけで構成される高速なスイッチと、新しいセルスケジューリングアルゴリズムを特徴とする。このアルゴリズムは、仮想回線に割り当てられたバンド幅と複数の優先クラスの存在を考慮に入れて、スケジューリングを行なう。シミュレーションにより、このアルゴリズムの有効性が証明された。

1　はじめに

マルチメディアのネットワークに用いるスイッチは、音声、映像、データなどの異なった伝送レートをもつ媒体を統合して扱わなければならず、また、高スループット、低レイテンシを要求される。この目的に最有力視されているのがATMスイッチである。ATMでは、スイッチを通過するセルを処理する際にソフトウェアが介在する時間的余裕がもはやないので、全ての処理はハードウェアで行なわれなければならない。しかも、各種の媒体が、それぞれ異なったバンド幅、損失率を要求するので、ATMスイッチはそれらを的確に処理してやることが必要である。

本報告では、ATMスイッチング機構である "StarCore" を提案し、そこで用いられるセルスケジューリングアルゴリズムを示す。このアルゴリズムは、完全にハードウェアで実現されるので、高速なセルスイッチングが可能となる。"StarCore" では基本となるスイッチ機構としてクロスバースイッチを使用する。クロスバースイッチは、internally non-blocking であることと、レイテンシが小さいという利点がある[1]。本報告ではクロスバースイッチだけを扱うが、ここで提案するセルスケジューリングは、他の形態のATMスイッチにも応用が可能である。

"StarCore" は、次のような特徴を持つ。

- "StarCore" のセルスケジューリングアルゴリズムでは、複数のクラスのセルを扱う。これにより、上位クラスのセルは下位のクラスのセルに対して、絶対的な優先度を持つことになる。
- 出力での衝突は、アービタにより、ラウンドロビンの形で処理される。その際、アービタは、各仮想伝送路に割り振られたバンド幅と優先クラスを考慮に入れる。

A Multimedia-Oriented ATM Switch Cell Scheduling Algorithm and its Hardware Implementation

Ken-ichi Suzuki[†], Nobuyuki Ooba[‡], Hiroaki Kobayashi[†], Tadao Nakamura[†]

[†]Graduate School of Information Science, Tohoku University

[‡]IBM Research, Tokyo Research Laboratory, IBM Japan LTD.

- 通常の重み付きラウンドロビン法[2]は、複雑な制御ロジックを必要とするので、高速なアービトレーションには適さない。そこで、本報告では、簡単な論理ゲートだけで構成できる高速な重み付きラウンドロビンアービタ用ハードウェアを提案する。この方法によるアービトレーション時間は、$O(\log_k M)$である。ここで、MはATMスイッチの入力ポート数であり、kはarbiterに使用されるORゲートの入力数である。最近のVLSIチップでは、16個以上の入力を持つORゲートも使用可能になってきており、このarbitrationハードウェアを256以上の入力ポートを持ったATMスイッチにも適用することも可能であると考えられる。

2 "StarCore"のセルスケジューリングアルゴリズム

2.1 クロスバースイッチによるセルスケジューリング

Fig. 1は、$M \times N$のクロスバー形式のATMスイッチのブロック図である。このスイッチは、$M \times N$の交点のそれぞれにクロスポイントスイッチ$SW(i,j)$ $(1 \leq i \leq M, 1 \leq j \leq N)$を持つ。各クロスポイントスイッチは、1個ずつのデータスイッチと、出力ポート番号$O(j)$ $(1 \leq j \leq N)$を識別するデコーダを持っている。さらに、同じ列にあるクロスポイントスイッチは、アービタ$ARB(j)$を

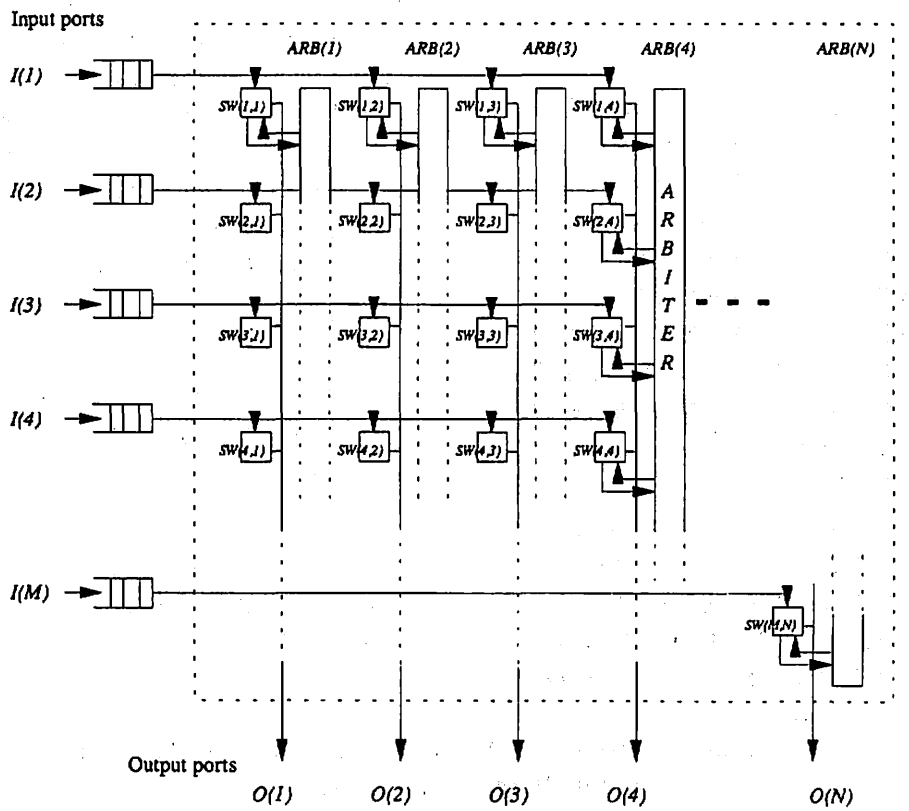

Fig. 1 :クロスバー形式のATMスイッチ

共有している。このアービタは、出力ポート $O(j)$ についての衝突を処理するためのものである。入力ポート i に出力ポート j へのセルが入って来た場合、Fig. 2. に示したように、クロスポイントスイッチ $SW(i,j)$ がそれを検出し、アービタ $ARB(j)$ に要求 $R(i,j)$ を出す。$ARB(j)$ が一度に複数のクロスポイントスイッチからの要求を受け取った場合、すなわち $O(j)$ についての衝突が発生している場合には、$ARB(j)$ がその解決を行なう。

全てのクロスポイントスイッチ $SW(i,j)$ $(1 \leq i \leq M, 1 \leq j \leq N)$ とアービタ $ARB(j)$ $(1 \leq j \leq N)$ は、並列に動作できる。

アービタは、クロスポイントスイッチからの要求を受け取り、いずれか一つのクロスポイントスイッチに応答を返さなければならない。衝突が起きている場合には、複数のクロスポイントスイッチから一つを選択するわけであるが、これを行なうのがスケジューリングアルゴリズムである。

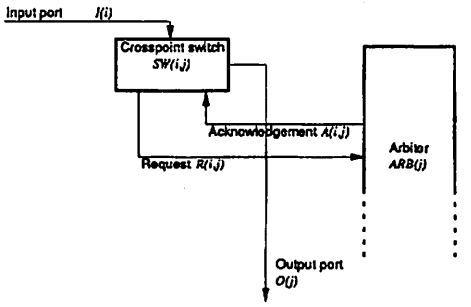

Fig. 2: クロスポイントスイッチ

2.2 重み付きラウンドロビン法によるセルスケジューリング

送信側は、実際にデータを送る前に、ネットワーク制御センター (NCC) に対して、所用のバンド幅の仮想回線を要求するものとする。その要求を受けたNCC は、送信側と受信側の間に仮想回線を確立する。このようにして確立された仮想回線を通して、所定のセルフォーマットで、データを送ることができる。

この仮想回線の確立は、ATM スイッチにおいては、クロスポイントスイッチでのセル転送スケジューリングとして反映される。多くのバンド幅を割り当てられた仮想回線について、それだけ多くのセルが ATM スイッチを通過できるようにスケジューリングされればよいわけである。本報告で提案するアルゴリズムは、重み付きラウンドロビン法において、より多くの" 重み" を割り当てることで、これを実現している。

セルのスケジューリングは以下のようにして行なわれる。ATM スイッチを M 入力 N 出力とする。その他のパラメータを次のように定める。

$v_{i\to j}^{k}$: 入力ポート i から入って出力ポート j に向かう仮想回線 k。

$b(v_{i\to j}^{k})$: 仮想回線 k のバンド幅。

入力ポート i から出力ポート j への仮想回線のバンド幅の和は、次のように与えられる。

$$b(V_{i\to j}) = \sum_{k} b(v_{i\to j}^{k}) \qquad (1)$$

本アルゴリズムの最も基本的な考え方は、$\frac{b(V_{i\to j})}{\sum_{i} b(V_{i\to j})}$ に比例した weight を使って、重み付きラウンドロビンスケジューリングを行なう、というものである。

まず、サイクル t において、クロスポイントスイッチ $SW(i,j)$ に $P_{SW(i,j)}^{t}$ という優先度コードを与える。各サイクルにおいて、それぞれのクロスポイントスイッチには異なった優先度コードが与えられる。すなわち、任意の t について、

$$P_{SW(i,j)}^{t} \neq P_{SW(i',j)}^{t} \text{ if } i \neq i' \qquad (2)$$

となる。優先度コードは、通常のラウンドロビンスケジューリングと同様に、いくつかのサイクル毎に繰り返される。この繰り返されるサイクルの集まりのことをフェイズと呼ぶ。

あるサイクルにおいて、出力ポートで衝突が発生した場合、より大きな優先度コードを持つクロスポイントスイッチが実際にセルを出力できる。従って、優先度コードの最大値 $\hat{P}_{SW(i,j)}^{t}$ が特に重要

な意味を持つ。というのは、最大の優先度コードを持つクロスポイントスイッチが要求を出した場合には、必ずアービトレーションを獲得できるからである。各入出力ポート間に割り当てられたバンド幅の和が$b(V_{i\to j})$の場合、最大の優先度コードをその回線に1フェイズあたりα回割り当てることにする。ここで、αは、

$$\alpha = \frac{b(V_{i\to j})}{\sum_i b(V_{i\to j})} \times \lambda \qquad (3)$$

によって与えられ、λはフェイズに含まれるサイクルの数である。

複数クラスのスケジューリングの場合、上位クラスの仮想回線には、下位クラスの回線よりも常に大きな優先度コードを割り当てる。クラスがC段階あるものとすると、Fig. 3に示したように、セル分配器が到着したセルをクラス別のFIFOキューに振り分ける。セルはスイッチに入る前にセルマージャにより集められるれる。アービタによって、そのサイクルに最大の優先度コードを持つセルが選択され、それがスイッチに送られる。

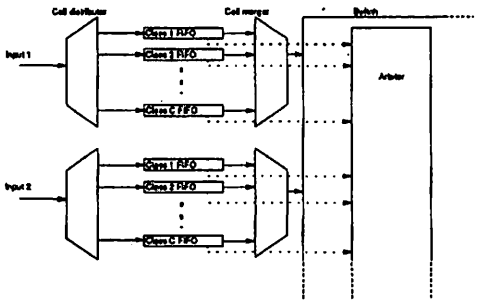

Fig. 3: 複数クラスの場合のスイッチ

例として、2段階のクラスの3×3クロスバースイッチを考える。入力ポートのそれぞれには、上位クラスと下位クラスの二本のキューが用意される。各出力ポートの持つ最大バンド幅を16とする。各出力ポートについて独立にアービトレーションが行なわれるので、ここでは、とくに出力ポート1に注目する。各入力ポートに割り当てられたバンド幅をTable 1に示す。"1H"と"1L"は、それぞれ、入力ポート1の上位クラスと下位クラスを表す。

	Port1		Port2		Port3	
	1H	1L	2H	2L	3H	3L
Allocated bandwidth	4	2	2	1	2	1

Table 1: バンド幅の割り当ての例

Phase	Cycle	Port number 1H	1L	2H	2L	3H	3L
	1	6	3	5	2	4	1
	2	4	1	6	3	4	2
	3	6	3	5	2	4	1
	4	5	2	4	1	6	3
	5	6	3	5	2	4	1
	6	4	1	6	3	5	2
	7	6	3	5	2	4	1
	8	5	2	4	1	6	3
	9	6	3	5	2	4	1
	10	4	1	6	3	5	2
	11	6	3	5	2	4	1
	12	5	2	4	1	6	3
	13	6	3	5	2	4	1
	14	4	1	6	3	5	2
	15	6	3	5	2	4	1
	16	5	2	4	1	6	3

Table 2: 優先度コードの割り当ての例

セルスケジューリングは、1から6までの優先度コードをサイクル毎に各クロスポイントスイッチに割り当てることで実現する。Table 2. は、クロスポイントスイッチへの優先度コードの割り当ての例である。サイクル1からサイクル16までの間に、1Hには最大の優先度コード6(表では網掛けしてある)が8回割り当てられていることは重要である。同様に、2Hと3Hには、最大の優先度コード6(これも網掛けしてある)が4回ずつ割り当てられている。また、上位クラスである1H、2H、3Hには、常に、下位クラスである4H、5H、6Hよりも大きな優先度コードが割り当てられる。この重み付きラウンドロビン法の基本とな

る考えは、割り当てられたバンド幅に応じて、最大の優先度コードを与えるというものである。優先度コードの割り当て法については、付録を参照されたい。

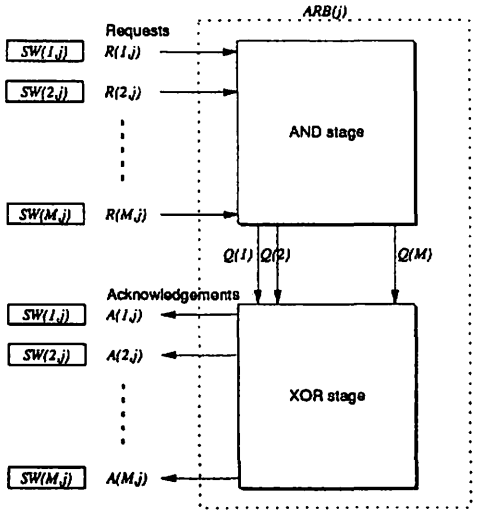

Fig. 4: ANDブロックとXORブロックからなるarbiter

3 アービタ用ハードウェア

ここでは、簡単のため、単一クラスのスケジューリングのアービタ用のハードウェアについて述べるにとどめる。しかし、これをFig. 3で示したような複数クラスのキューの場合に拡張することは、容易である。

アービタは、Fig. 4に示すように、ANDブロックとXORブロックの二つのブロックからなる。Fig. 5は、ANDブロックの内部構造を示す。これは、M^2個の優先度レジスタとM^2個のANDゲートからなる。優先度レジスタは、それぞれが1ビットのフリップフロップである。同じ列にある"and"ゲートの出力は互いにORを取られ、内部信号$Q(y)$ $(1 \le y \le M)$となる。$Q(y)$は、XORブロックの入力となる。

$P(i,y)$ $(1 \le i,y \le M)$は、優先度レジスタであり、ANDゲート$AND(i,y)$に附属している。$AND(i,y)$の出力は、要求$R(i,j)$と優先度レジスタ$P(i,y)$の論理積である。すなわち、

$$AND(i,y) = R(i,j) \bullet P(i,y) \qquad (4)$$

である。ここで、$\bullet$は、論理積を表す。

優先度レジスタ$P(i,y)$は、各クロスポイントスイッチに割り当てられている優先度コードを変形して得られる2進数を持つ。すなわち、優先度コードの値がnの場合、$P(i,y)$はn個の1を含む。例えば、優先度コードが4ならば、$\{P(i,y)\ (1 \le y \le 8)\} = \{0,0,0,0,1,1,1,1\}$である。

内部信号$Q(y)$は、y番目の列にあるANDゲートの出力のORを取ることで得られる。すなわち、$\bigcup$をOR演算子として、

$$Q(y) = \bigcup_{i=1}^{M} AND(i,y) \qquad (5)$$

と表される。演算子$\bigcup$に相当するORステージの数は、$\log_k M$である。ここで、kは、アービタで使われるORゲートの入力の数である。

XORブロックは、Fig. 6のように、"exclusive-or"ゲートからなる。"and"ゲートの代わりに"exclusive"ゲートが各交点に置かれる以外は、ANDブロックと同様の構成である。

XORブロックの出力は、応答信号$A(i,j)$ $(1 \le i \le M, 1 \le j \le N)$である。これは、次のように表される。

$$\begin{aligned} A(i,j) &= \overline{\bigcup_{y=1}^{M} XOR(i,y)} \\ &= \overline{\bigcup_{y=1}^{M} Q(y) \oplus P(i,y)} \end{aligned} \qquad (6)$$

ここで、$\oplus$はexclusive-or演算子である。XORブロックもANDブロックと同じ優先度レジスタを使用する。

以上から、アービトレーション時間は次の式で与えられる。

$$Delay(AND) + Delay(XOR) + 2 \times \log_k M \times Delay(OR)$$

最近のVLSI技術では、ここで扱っているような簡単なゲートを10nsecオーダーの遅延時間で実現でき、また、16入力のORゲートも可能である。ゲートの遅延を0.5nsecとすると、256×256のスイッチのarbitration時間は、式(7)により、3nsecにすぎないことが分かる。

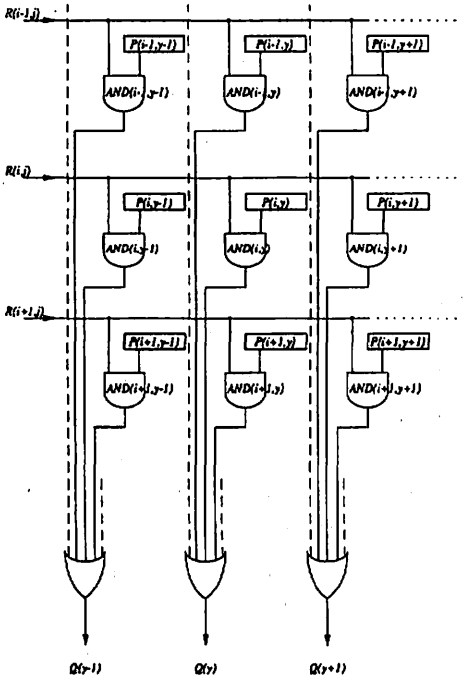

Fig. 5: ANDブロック

4 シミュレーション

このアルゴリズムの有効性を確認するために、バンド幅を考慮しないラウンドロビンと、ここで提案した重み付きラウンドロビンを、シミュレーションにより、比較した。入力ポートには、マルコフ分布に従ってセルが到着するものとし、それぞれのアルゴリズムに従って処理を行なった。

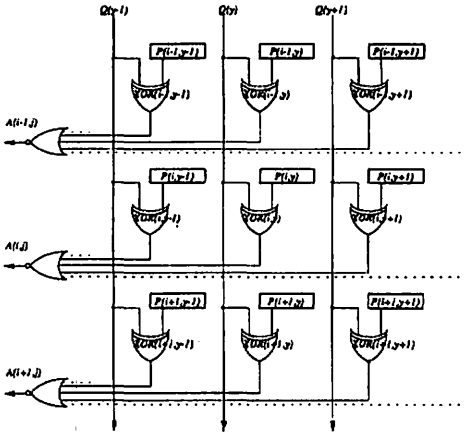

Fig. 6: XORブロック

Fig. 7は、$M = 6$の単一クラスの場合の結果である。横軸はセルの到着率、縦軸は、セルの損失率である。ここでは、入力ポート1に他の6個のポートの1/4の平均到着間隔でセルが到着するものとし、スイッチの各キューのバッファサイズをパラメタとしている。いずれのバッファサイズの場合でも、本アルゴリズムの方が低い損失率となっており、セルの到着率が上がって衝突が増えるほど有利であることも読み取れる。

Fig. 8とFig. 9は、$M = 3$で2つのクラスのセルが到着する場合である。クラス毎にキューが用意されるので、キューの数は先の場合と同様に8本である。上位クラス、下位クラスともに、入力ポート1に他の3個の入力ポートの1/4の平均到着間隔でセルが到着するものとし、また、上位クラスと下位クラスのセルの平均到着間隔は等しくした。先と同じように、バッファサイズをパラメタとして示している。Fig. 8が上位クラス、Fig. 9が下位クラスについての結果である。いずれの場合も、本アルゴリズムの方が低い損失率を得ている。Fig. 8の上位クラスの結果で、8以上の大きなバッファサイズの場合にセルがほとんど損失しないのは、到着率を上位クラスと下位クラスのセル

到着数の合計から求めているためである。Fig. 9から分かるように、セルの損失しやすい下位クラスにおいて、本アルゴリズムは損失率を大幅に低下させることができ、非常に有効である。

以上から、本アルゴリズムは、到着率が高い場合や下位クラスのセルを扱う場合のような、衝突の多発する、所謂シビアな条件下で大きな効果を発揮することが示された。

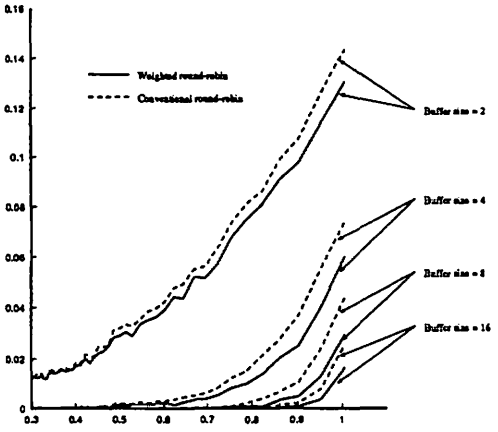

Fig. 7: 単一クラスの場合のセル損失率

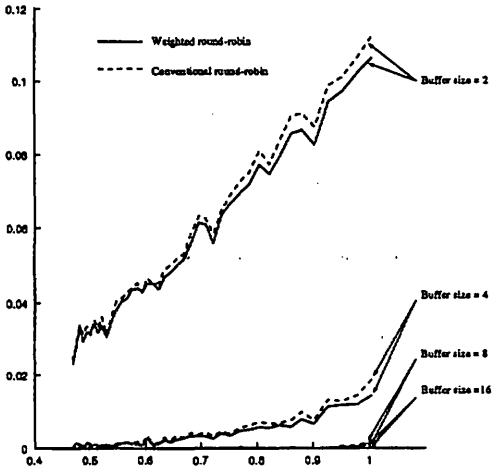

Fig. 8: 複数クラスの場合のセル損失率 (上位クラス)

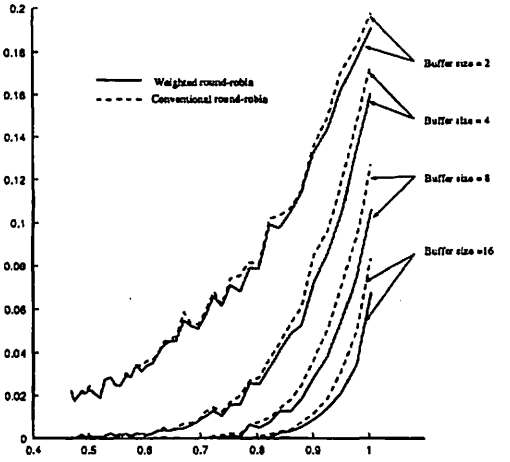

Fig. 9: 複数クラスの場合のセル損失率 (下位クラス)

5 まとめ

高速なセルスイッチングにおいては、簡単で高速なハードウェアによるスイッチングが要求される。本報告は、これを実現するハードウェアとスケジューリングアルゴリズムを提案した。このアルゴリズムは、仮想回線に割り当てられたバンド幅を考慮に入れた重み付きラウンドロビン法に基いている。シミュレーションにより、本アルゴリズムが低いセル損失率を実現できることを示した。

参考文献

[1] H. Ahmadi and W. E. Denzel, "A Survey of Modern High-Performance Switching Techniques," *IEEE J. Select. Areas Commun.*, Vol. 7, No. 7, September 1989, pp. 1091-1103

[2] M. Katevenis, S. Sidiropoulos, and C. Courcoubetis, "Weighted Round-Robin Cell Multiplexing in a General-Purpose ATM Switch Chip," *IEEE J. Select. Areas Commun.*, Vol. 9, No. 8, October 1991, pp. 1265-1279

付録

優先度コードの割り当て手順

1フェイズがλサイクルからなるものとし、各サイクルを A_k $(1 \leq k \leq \lambda)$ で表す。本文中で述べたように、本アルゴリズムでは、優先度コードの最大値が重要な意味を持ち、クロスポイントスイッチ SWj $(1 \leq j \leq M)$ に割り当てられた最大優先度コードの数を Dj $(\sum_{j=1}^{M} \leq \lambda)$ とする。

まず、SWjを連続的に割り当てるとすると、Fig. A-1のようになる。これは、λが16、$D1$が4、$D2$が4、$D3$が8の場合のものである。Fig. A-1の中央の列は、そのサイクルにどのクロスポイントスイッチが最大優先度コードを持つかを示している。$SW1$はサイクル $A1 \sim A4$ で最大優先度を持ち、また、$SW2$、$SW3$はそれぞれ、$A5 \sim A8$、$A9 \sim A16$ で最大優先度を持つことになる。しかし、このように連続的に割り当てた場合、セルが到着するサイクルによってスイッチを通過できる可能性が大きく異なると考えられるので、好ましくない。

Arbitration cycle	SW#	D
A1	1	
A2	1	D1 = 4
A3	1	
A4	1	
A5	2	
A6	2	D2 = 4
A7	2	
A8	2	
A9	3	
A10	3	
A11	3	
A12	3	D3 = 8
A13	3	
A14	3	
A15	3	
A16	3	

Fig. A-1

そこで、Fig. A-2に示したような木を使って$SW\#$をシャッフルすることを考える。この木には、各ノードに1個ずつ、トグルスイッチがある。そして、各ノードをトークンが通過するたびに、トグルスイッチの方向が(反対側に)切り替わるようになっている。葉は、左から順に、$A1$, $A2$,,, のサイクルに相当している。初期状態では、全トグルスイッチが左側にセットしてあるので、最初のトークンは木の1番左側の葉に向かう。この最初のトークンが通過した後では、"*" を付したトグルスイッチが右側に切り替わっていることになる。

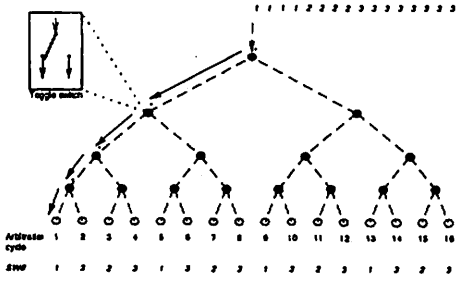

Fig. A-2

この木を使ってFig. A-1の$SW\#$をシャッフルしたのが、Fig. A-3である。

Arbitration cycle	SW#
A1	1
A2	3
A3	2
A4	3
A5	1
A6	3
A7	2
A8	3
A9	1
A10	3
A11	2
A12	3
A13	1
A14	3
A15	2
A16	3

Fig. A-3

アービトレーションの例

次に、本報告で提案したアービトレーションハードウェアの動作例を、4 入力のスイッチの場合について、Fig. A-4 に示した。このサイクルでは、スイッチ 1 が、'1111' で表される最高の優先度を持ち、逆に、スイッチ 4 が '0001' という最低の優先度を持っている。

AND ブロックの機能は、内部信号 Q を発生することである。Q は、現在のサイクルにおける最高の優先度を表す信号である。例えば、今のサイクルに全てのスイッチが要求を発しているものとすると、AND ブロックの第 1、第 2、第 3、第 4 の行の出力は、それぞれ、'1111', '0111', '0011', '0001' となるので、それらの OR を取ることで得られる Q は '1111' となる。言い換えると、低い優先度を持つ 2, 3, 4 のスイッチは、最高の優先度を持つスイッチ 1 によってマスクされてしまっているわけである。別な場合として、スイッチ 2 と 4 だけが要求を発しており、1 と 3 は発していないケースを考える。この場合、内部信号 Q は、'0111' (= '0000' OR '0111' OR '0000' OR '0001') になる。スイッチ 4 の要求が、スイッチ 2 によってマスクされているわけである。

一方、XOR ブロックの機能は、そのサイクルに、どのスイッチが最高優先度コードを発したかを特定することである。すなわち、XOR ブロックは、内部信号 Q と優先度コードを比較することで、要求を出したスイッチの内どれが最高優先度を持つものかを見付けることができる。先ほどの前者の例 (全てのスイッチが要求を出した場合) を再び考えると、内部信号の '1111' は、スイッチ 1 の優先度コード '1111' に一致するので、スイッチ 1 が応答信号を受け取ることになる。後の例 (スイッチ 2 と 4 だけが要求を出す場合) には、内部信号 '0111' がスイッチ 2 の優先度コードに一致するので、スイッチ 2 が応答信号を受け取るわけである。

本文中で述べたように、優先度コードはサイクル毎に遷移していくので、公平なアービトレーションを行なうことができる。

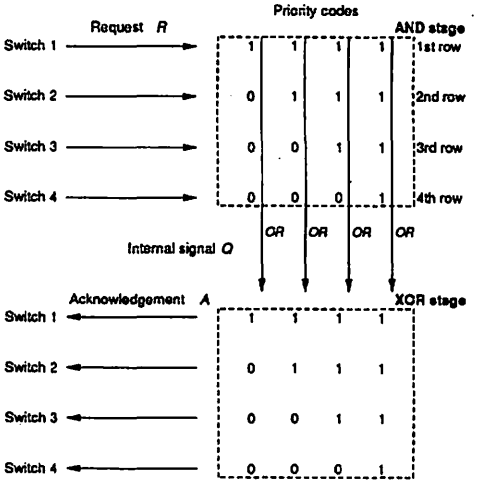

Fig. A-4