マルチメディアインタフェース

IEEE1394 ハイパフォーマンスシリアルバス

加藤 淳二[†]

1. ま え が き

1995年9月、デジタルカムコーダにIEEE 1394 High Performance Serial Bus(以下1394)準拠のデジタルインタフェースが搭載され発売された。1394は、もともとPCと周辺機器を接続するSmall Computer System Interface(SCSI)に代わるインタフェースとしてApple Computer, Inc.で設計され、その後IEEEで標準化された。最近では、Microsoft Corporationが次世代のPCの仕様として1394のサポートを表明している。1394の主な特徴は、

- ・データ転送速度が高速である。(現在100, 200, 400 Mb/sをサポート^{*1}, 混在可能)
- ・シリアル通信であるため、コネクタが小型でケーブルは細い。最長4.5mまで。
- ・活線挿抜(ホットプラグ)が可能で、機器を識別するためのアドレス(ノードID)は自動設定される。
- ・物理的接続はツリー構造をとるため、接続が容易である。
- ・1つの1394バスには最大63台の機器が接続できる。また、バスブリッジを用いることにより1023のバスを接続することが可能である。最大ノード数は $63 \times 1023 = 64449$ 台となる。
- ・リアルタイムデータ転送に適した、アイソクロナスデータ転送モードを持つ。

本稿では、このように将来広く応用が期待されている、1394規格について解説する。

図1 トポロジー

2. 物理層の概要

2.1 トポロジー

1394のトポロジーを図1に示す。各機器をループのないツリー状に接続することで、最大63台の機器を1つのバスに接続することができる。機器間は最長16ホップ^{*2}まで接続できる。各機器のポートは、受信したデータ信号を連続的に他のポートへリピートし、データ信号はバス全体に伝搬する。論理的にはバスとして機能することになる。また、バスブリッジを用いて異なるバス間での通信も可能である。

2.2 ケーブル

図2(a)に、1394ケーブルの構造を示す。2組の差動信号線TPA, TPBと、電源ペアVG(グラウンド), VP(電源)で構成される。TPA, TPBの2本の信号線で、バスの動的なコンフィグレーション、アービトレーション、データ信号の伝搬を行う。2つの機器間において、それぞれTPAとTPBが接続する(図2(b))。TPAには常時バイアスをかけており、TPBではそれを検出することにより、アクティブなケーブルの接続の有無の判定している。VGとVP

[†] ソニー株式会社 アーキテクチャ研究所 ネットワークラボラトリー

“Digital Interface; (end) IEEE1394 High Performance Serial Bus” by Junji Kato (Network Laboratory, Architecture Laboratories, Sony Corporation, Atsugi)

^{*1} 正確には 98.304 ± 100 ppm, 196.608 ± 100 ppm, 393.216 ± 100 ppm Mb/sである。最小のレートをbase-rateと呼ぶ。

^{*2} ホップとは、2つの機器間の物理的な接続数をいう。図1において、MDとPrinter間のホップ数は4となる。

図 2(a) ケーブルの構造

図 2(b) ケーブルの結線

図 3 DS コーディング方式

は、最大 60 W (40 V, 1.5 A) の電力の供給に用いられる。

2.3 物理層の信号

TPA, TPB の 2 本の信号線は、“1”、“0”、“Z” の 3 値を取る。ここで、“Z” は接続されたポートのどちらにも駆動されていないハイインピーダンス状態を意味する。データ信号転送時は、送信ノードがこの 2 つの信号線を“1”または“0”に駆動し、それを受信したノードは他の接続のあるポートに信号をリピートするので、常にバス全体に伝搬する。また、TPA, TPB の 2 本の信号線を DS コーディング方式 (図 3) を用いて、データ転送を行う。TPA からはストローブと呼ばれる信号を、TPB からはデータを送信する。シリアルデータはこのデータとストローブの 2 つの信号線で 1 ビットずつ転送される。ストローブ信号は、データが連続して変化しないときに反転する。受信側ではデータとストローブの排他的論理和をとることによりクロックを得ることができる。そして、このクロックの変化点でデータを読取る。

1394 は半二重双方向通信であり、ある一時点においてはバスにデータを送信できるノードは 1 つだけで

ある。しかしバス上には複数のノードが存在するので、データの送信が行われる前には、必ずバスの使用権を取得するアービトレーションが必要になる。アービトレーションは、“1”、“0”、“Z” の 3 値を用い、TPA, TPB を各々に双方向に駆動して、その優勢な値を受信して行う。この時の TPA, TPB 各々の値の組合せによって、アービトレーション時のデータ送信リクエストや許可などの意味を持つ。このアービトレーション信号は、データ信号と違い無条件にバス全体にリピートされない。

2.4 バスイニシャライゼーション

(1) バス初期化

各ポートは、接続相手の TPA が出力するバイアスの有無を検出して、ノードが接続されたか取り外されたかを判定する。ポートの接続状態の変化を検出したノードは、他の接続のあるポートに対して一定時間バスリセット信号を送信する。これを受信したノードは、さらに接続のある他のポートにバスリセット信号を送信する。この動作を繰り返し、最終的にバスに接続された全ノードにバスリセット信号が伝わる。バスリセット信号を受信したノードは、それ以前のトポロジー情報や、各自のノード ID をクリアする。その後、各ノードは自らがブランチ (隣接する複数のノードに接続されている状態) かリーフ (隣接するノードは 1 つだけ) か、孤立した状態かを認識する (図 4 参照)。

(2) ツリーアイデンティファイ (ツリー認識)

ツリーアイデンティファイでは、各ノードの親子関係付けを行い、ツリー構造を作り上げそのルートノードの決定が行われる。各ノードは、自分の接続のあるポートに「親」(ルートに近いノードに接続されているポート) か、「子」(ルートから遠いノードに接続されているポート) のラベル付けを行う。バス初期化後、各ノードは自分の接続のあるポートのうち、「親」「子」のラベル付けがすんでいないポートが 1 つだけになった場合は (リーフはすでにこの状態である)、そのポートの相手ノードに対して、parent_notify 信号を送信する。これを受けた相手のノードはこのノードの親となる。子ノードのポートには「親」が、親ノードのポ

図 4 バスイニシャライゼーション

図 6 セルフアイデンティファイ

図 5 ツリーアイデンティファイ

ートには「子」がラベル付けられる．この動作をバスに接続されたノードで行い最終的にすべてのポートに「子」のラベル付けがされた唯一のルートノードが決定する．ルートは以後アービターとして機能する（図 5 参照）．

(3) セルフアイデンティファイ（自己認識）

セルフアイデンティファイでは、ノードの識別に必要なノード ID 付けを行う．バス上の全ノードは各親ノードからの指示により順に、セルフ ID パケットをバスに送信する．各親ノードは自分のすべての子ノードがセルフ ID パケットを送信し終わると、自分が送信する．ルートは最後にセルフ ID パケットを送信する．セルフ ID パケットはバス上の全ノードに伝搬し、各ノードはセルフ ID パケットを受信した回数をカウントしており、自分がセルフ ID パケットを送信するまでに受信した回数が各ノードのノード ID となる．最終的にバス上の全ノードがそれぞれセルフアイデンティファイを終了すると、それぞれ重複のないノード ID が割り付けられる(図 6 参照)．

セルフ ID パケットのフォーマットを図 7 に示す．ノード ID のほかにも、ギャップカウント（後述）

transmitted first													
10	phy_ID	0	1	gap_cnt	sp	del	c	pwr	p0	p1	p2	i	m
2	6	1	1	6	2	2	1	3	2	2	2	1	1
logical inverse of first quadlet													
transmitted last													

セルフIDパケット # 0

transmitted first													
10	phy_ID	1	n	rsv	pa	pd	pc	pd	pe	pf	pg	ph	r
2	6	1	3	2	2	2	2	2	2	2	2	2	1
logical inverse of first quadlet													
transmitted last													

セルフIDパケット #1, #2 & #3

	n	pa	pb	pc	pd	pe	pf	pg	ph
packet# 1	0	p3	p4	p5	p6	p7	p8	p9	p10
packet# 2	1	p11	p12	p13	p14	p15	p16	p17	p18
packet# 3	2	p19	p20	p21	p22	p23	p24	p25	p26

図 7 セルフ ID パケットフォーマット

や、各ポートの親子状態（図中の p 0～p 26）、後述するアイソクロナスリソースマネージャ対応かどうか、などの情報が設定され送信される．

2.5 ノーマルアービトレーション

バス上のすべてのノードがデータの送信をする必要がないときバスはアイドル状態になる（2つのツイストペア線が“Z”の状態）．ノードがデータ送信のためアービトレーションを行えるのは、このアイドル状態が一定期間継続した後でなくてはならない．このアイドル状態の期間をギャップと呼ぶ．あるノードがデータ送信を行っている間は、バスはアイドル状態にならないため、他のノードはアービトレーションを行うことはできない．この送信が終了し、ギャップを検出できたらアービトレーションを行うことができる．

図 8 に示すように、ノード # 0 とノード # 2 とが同時にアービトレーションを行う場合を例にとって動作を説明する．

図 8(a) ノーマルアービトレーション(1)～(3)

図 8(b) ノーマルアービトレーション(4), (5)

図 9 IEEE 1212 アーキテクチャ

- (1) ノード#0はギャップを検出すると、その親であるノード#4に送信要求信号を出力する。また、ノード#0の動作とほぼ同時にノード#2もギャップを検出するので、その親であるノード#3に送信要求信号を出力する(図8(a)の(1)参照)。
- (2) 子ノード#2から送信要求信号を受信したノード#3はさらに親であるノード#4が存在するので、それに対して送信要求信号を出力する。また、別の子ノード#1に対しては、禁止信号を送信する。禁止信号を受けたノード#1は、データ受信状態になる(図8(a)の(2)参照)。
- (3) ルートであるノード#4は別々のポートに送信要求信号を受信することになるが、先に信号が到着したノード#0側のポートに送信許可の信号を出力し、その他のポートには禁止信号を出力する。送信禁止を受けたノード#3は送信要求信号を取り下げ、データ受信状態になる。

ノード#0は送信許可信号を検出し、バスの使用権を獲得する(図8(a)の(3)参照)。

- (4) アービトレーションに勝ったノード#0は、送信開始信号を出力し、それを受信したノード#4は送信許可信号の出力を止める(図8(b)の(4)参照)。
- (5) この時点で、データが伝搬する方向が決定する^{*3}(図8(b)の(5)参照)。ノード#0は送信開始信号に続きDSコーディングに従ってデータパケットを送信する。最後に送信終了信号を出力して、送信を終える。

3. アドレッシング(IEEE1212)

1394ではIEEE 1212で規定される64ビット幅のアドレス空間を用いる(図9参照)。

上位10ビット(最大1023)のbus IDとそれに続く6

^{*3} 実際には禁止信号というのは定義されおらず、送信開始信号を出力する。

図 10 アシンクロナス通信

ビット(最大 63)のノード ID で機器を識別する。さらに残りの 48 ビット(256 TByte)はノード内のアドレス空間として割り当てられる。また、特別に bus ID の 1023 はそのノードが接続されているバス (Local bus)を示す。さらにノード ID の 63 はその指定したバスに接続された全機器(Broadcast)を示す。48 ビット幅のアドレス空間が割り当てられているノード内の空間は、上位 20 ビットのレジスタスペースと下位 28 ビットのレジスタアドレスに分けられる。この空間は、初期メモリー空間(レジスタスペース 0 から [F.FFFDh] まで)、プライベート空間 (レジスタスペース [FFFFEh]), 初期レジスタ空間 (レジスタスペース [FFFFFFh], レジスタアドレス 0 から [7 FFh] まで)、初期ユニット空間 (レジスタスペース [FFF.FFh], レジスタアドレス [800 h] から [FFFFFF.Fh] まで) の 4 つの空間に大別される。このうち初期レジスタ空間には、Control and Status Register (CSR)と、Configuration ROM が標準化される。

4. アシンクロナス通信

4.1 アシンクロナストランザクション

1394 でデータパケットの転送方法のひとつとして、アシンクロナス通信 (非同期通信) がある(図 10 参照)。これは片方向のデータパケット転送である。送信側はデータパケット転送先アドレスをパケットヘッダに明記し、バスに送信する。データパケットはバス上のすべてのノードまで伝搬する。パケットヘッダに明記された転送先アドレスに対応するノードは、その

データパケットを受信し、受信結果(Acknowledge, 以下 ack)を返送する。この一連の転送プロセスをアシンクロナスサブアクションと呼ぶ。アシンクロナスサブアクションを開始するためには、サブアクションギャップ*4と呼ばれる一定期間、バスがアイドルにならなくていけない。また、受信側がデータパケットを受け取って、ack を返送する間もバスはアイドルとなり、この間隔を ack ギャップ*5と呼ぶ。ack ギャップはサブアクションギャップに比べて充分短いため、他のサブアクションが開始されることはない。

1394 では、このアシンクロナス通信を用いてトランザクションを定めており、アプリケーションに 1394 アドレス空間のリード、ライト、ロック*6のサービスを提供している。各トランザクションはリクエストサブアクションとレスポンスサブアクションで構成することができる。例えば図 11(b)にリードトランザクションを示す。リクエストパケットには読出すアドレス、データ長、リードリクエストのパケットであることを示すトランザクションコード等が明記され、リードレスポンスパケットには、その処理の結果を示すレスポンスコードや、読出しに成功した場合は、そのデータが格納される。同様に、ライト、ロックの場合を図 11(a), (c)に示す。図中の ack_com-

*4 $(27 + \text{gap_count} \times 16) / \text{base_rate} \leq \text{subaction_gap} \leq (29 + \text{gap_count} \times 16) / \text{base_rate}$

*5 $0.04 \mu\text{s} \leq \text{ack_gap} \leq 0.05 \mu\text{s}$

*6 排他的アクセスを行うトランザクション。比較と交換 (compare_swap), フェッチと追加 (fetch_add) 等のサブコマンドがある。

図 11 分割トランザクション

plete はノードが受信を成功し、その後レスポンスパケットを送信しないことを意味する。ack_pending はノードが受信を成功し、その後レスポンスパケットが送信されることを示す*7。

これらのようにリクエスト、レスポンスが2つのサブアクションで構成されるトランザクションを分割トランザクションと呼ぶ。1394 では、さらにバスを有効に用いる方法として、ライトトランザクションに適用できる統合トランザクションや、リードまたはロックトランザクションに適用できる連結トランザクションも使ってもよい。この一例を図 12 に示す。

4.2 フェアアービトレーション

後述するアイソクロナス通信は帯域幅とチャンネルを取得していれば、毎サイクル送信できることが保証されるが、アシンクロナス通信の場合は、アービトレ

図 12 統合トランザクション、結合トランザクション

ーションに複数のノードが参加した場合は、アービトレーションの原理上、ルートに近いノードが必ず勝つ*8。アシンクロナス通信では、アービトレーションに参加したノードに平等にバス使用权を提供するために、フェアアービトレーションプロトコルを適用している。

通常、アシンクロナスパケットを送信する場合はサブアクションギャップでアービトレーションに参加できることを 4.1 節のアシンクロナス通信で述べた。フェアアービトレーションプロトコルでは、一度アシンクロナスパケットを送信したノードはサブアクションギャップよりさらに長い間隔を持つアービトレーションリセットギャップの間は、アービトレーション不可能状態になる。この状態はアービトレーションリセットギャップを検出すると解除され、すぐにアービトレーションに参加することが可能となる。アービトレ

図 13 フェアアービトレーションの概念

*7 その他に受信失敗を示す ack_busy, ack_data_error, ack_type_error などがある。

*8 サイクルマスタはこの性質を利用している。

図 14 サイクルスタートパケット

図 15 アイソクロナス通信

ションに参加するノードがこのように振る舞うことによって、アシンクロナス通信の平等なアクセスが実現される。この概念を図 13 に示す。

5. アイソクロナス通信

1394 でデータを転送するもうひとつの方法はアイソクロナス通信である。アイソクロナス通信は、バス上に 1 台存在するサイクルマスタが一定間隔 ($8 \text{ kHz} \pm 100 \text{ ppm}$, $125 \mu\text{s} \pm 12.5 \text{ ns}$) で送信するサイクルスタートパケット (図 14 参照) に同期して行われる。

アイソクロナス通信の送信側のノードはサイクルスタートパケットを受信するとアイソクロナスギャップ*9 を待ってアービトレーションを開始し、アイソクロナスパケットを送信する。他にも送信ノードがいる場合は、引き続きアイソクロナスギャップで、アービトレーションを開始しアイソクロナスパケットを送信する (図 15 参照)。

アイソクロナスギャップは、サブアクションギャップより充分短い期間になっているため、この間にアシンクロナス通信を行いたいノードがいても、サブアクションギャップが検出できないため送信できない。つまり、毎サイクル、アイソクロナスパケットを送信するノードに優先権が与えられることになる。

また、アイソクロナス通信はアシンクロナス通信のように、データ転送先のアドレスを指定することではなく、バスにブロードキャストされる。アイソクロナスパケットには 0 から 63 までのチャンネル番号が割り振られ、ノードは必要なチャンネル番号のアイソクロナスパケットを受信すればよい。

サイクルマスタが送信するサイクルスタートパケットは、アービトレーションで最高の優先アクセスを割り当てなくてはならない。このためルートがサイクル

*9 $0.04 \mu\text{s} \leq \text{isochronous_gap} \leq 0.05 \mu\text{s}$

図 16 サイクルの構造

マスタとなる。

その時点にアイソクロナス通信に割り当てることができる帯域幅は、バス上に 1 台存在するアイソクロナスリソースマネージャの BANDWIDTH_AVAILABLE レジスタに表される。また、使用できるチャンネルはアイソクロナスリソースマネージャの CHANNELS_AVAILABLE レジスタに表される。アイソクロナスパケットの送信開始は、その通信で使用する帯域幅と、チャンネルをアイソクロナスリソースマネージャのこれらのレジスタに設定してから許可される。もし、必要なチャンネルがすでに使用されているか、または帯域幅が足りない場合は、アイソクロナスパケットの送信はできない。各レジスタに関しては 6.2 節のアイソクロナスリソース管理で説明する。

アイソクロナスリソースマネージャの決定は、セルフアイデンティファイの時に行われる。2.4 節(3)セルフアイデンティファイで述べたように、各ノードが順次送信するセルフ ID パケットには、そのノードがアイソクロナスリソースマネージャ対応可否の情報が示されており、そのノードのうち最大のノード ID のノードがアイソクロナスリソースマネージャとなる。セルフ ID パケットはバス上の全ノードに行き渡るので、すべてのノードがアイソクロナスリソースマネージャのノード ID を知ることができる。

6. シリアルバス管理

シリアルバス管理はノード制御、アイソクロナスリソース管理、バス管理の 3 つの機能に分類できる。アイソクロナス通信およびアシンクロナス通信を行うノードは、常にノード制御が必要である。また、アイソクロナス通信を行うバスでは、バス上に 1 台アイソクロナスリソース管理を行うノードが必要である（アイソクロナスリソースマネージャ）。さらに、バストポロジ、バススピード、電力管理を行うには、バス管理

を行うノードがバス上に 1 台必要となる（バスマネージャ）。

6.1 ノード制御

IEEE 1212 で定められる CSR の実装と、その内容をノードの動作に反映する。つまり、バス上の他の機器からアシンクロナス通信で読み書きできる CSR の実体をここで持つ。また、書込まれた場合には、その値にしたがってノードの動作を制御させる。例えば、SPLIT_TIMEOUT レジスタへの書込みは、そのノードが分割トランザクションで用いるタイムアウトの値を設定することができる。

同様に、ノード固有の情報を格納する、Configuration ROM の実装も行う。

6.2 アイソクロナスリソース管理

アイソクロナス通信を行うバスには、必ずバス上に 1 台アイソクロナスリソースマネージャが存在する。アイソクロナスリソースマネージャはバス上のノードからアクセスできる 3 種類のレジスタを提供する。

BANDWIDTH_AVAILABLE レジスタはアイソクロナス通信で使用する帯域幅の残量を示す 4 Byte のレジスタである。リードとロックトランザクションをサポートしている。アイソクロナス通信を開始しようとするノードはこのレジスタの値から使用帯域幅を減算して設定し、通信を終えたら加算する。このように各ノードが振る舞うことにより、アイソクロナス通信に割り当てられる帯域幅の制限をしている。

CHANNELS_AVAILABLE レジスタはアイソクロナス通信に割り当てられる帯域幅の制限をしている。

CHANNELS_AVAILABLE レジスタは、アイソクロナス通信の各チャンネルの割り当て状態を示す計 8 Byte のレジスタである。1 チャンネルの割り当て状態を 1 ビットで表し、値“0”はすでに割り当てられていることを示し、値“1”は利用可能であることを

reserved 19	bw_remaining 13
----------------	--------------------

(a) BANDWIDTH_AVAILABLE レジスタ
destination offset : FFFF F000 0220h

channels_available_hi 32
channels_available_lo 32

(b) CHANNELS_AVAILABLE レジスタ
destination offset : FFFF F000 0224h

reserved 26	bus_mgr_id 6
----------------	-----------------

(c) BUS_MANAGER_ID レジスタ
destination offset : FFFF F000 021Ch

図 17 アイソクロナスリソース管理レジスタ

示す。アイソクロナス通信を開始しようとする場合は、使用するチャンネルの対応するビットが1であることを確認し、“0”に書き換えてから通信を行わなくてはならない。

BUS_MANAGER_ID レジスタはバスマネージャの決定に用いる 4 Byte のレジスタである。初期値[3 Fh]はバスマネージャが存在しないことを意味する。バスマネージャになろうとするノードは、バスリセット後一定時間内にこのレジスタの値が[3 Fh]であることを確認し、自分のノード ID を設定することによりバスマネージャの役割を行うことができる。

これらのレジスタフォーマットを図 17 に示す。値の変更はロックトランザクションの Compare_Swap を用いる。

6.3 バス管理

バス管理は、基本的にはバス上に 1 台存在するバスマネージャが行う。サイクルマスタのアクティブ化、ギャップの最適化、電力管理、SPEED_MAP レジスタ、TOPOLOGY_MAP レジスタの実施を行う。ここでは、ギャップの最適化について簡単に述べる。1394 で通信されるデータパケットは ack ギャップ、アイソクロナスギャップ、サブアクションギャップ、アービトレーションリセットギャップの 4 種類のギャップで区切られている。これらのギャップのタイミングは、パラメータギャップカウントの値によって決定される。各ギャップでは、データパケットの送信開始はできないため、ギャップを長く設定すると通信の効率が悪くなる。例えば、ギャップカウントの初期値

“63” の場合、サブアクションギャップは約 10 μ s となり、これは 100 Mb/s で 1 kByte 程度のデータを送信できる帯域幅に相当する。ギャップカウントは小さくした方がバスの利用効率が上がるが、小さくしすぎると信号のケーブルや中継箇所での伝搬遅延があるため、アービトレーションリセットギャップより（合計伝搬遅延+サブアクションギャップ）の方が長くなり、フェアアービトレーションが正しく動作しなくなる。バスマネージャはバストポロジの最大ホップ数等から適切なギャップカウントを計算し、設定することができる。また、バスマネージャが存在しない場合は、アイソクロナスリソースマネージャがこの処理を行ってもよい。

7. む す び

1394 は単なる高速なシリアルバスというだけでなく、柔軟なトポロジや活線挿抜に対応していることや、アイソクロナス通信などの多くの特徴を持つため、PC や情報家電機器を接続する次世代のインタフェースとして期待されている。また、1394 の新しい動きとして、従来の 1394 の拡張、高速化(1.6 Gb/s)、ロングケーブル、バスブリッジ等が IEEE で検討されている。本稿では、全仕様の詳細を説明することはできなかったが、これらの 1394 の特徴的な機能に重点を絞って解説した。読者の参考になれば幸いである。

〔参 考 文 献〕

- 1) IEEE1394 Standard for High Performance Serial Bus, The Institute of Electrical and Electronics Engineers, INC 30 (Aug. 1996)
- 2) 松原 敦, 倉田雅弘, 斎藤貴之: “次世代パソコンの夢と現実 第3部新インターフェースへの対応”, 日経バイト, 1997 年 6 月号, pp. 158-163
- 3) 北山洋幸, 藤本卓也, 影山敏広, 鈴木昌利: “エレクトロニクス 1997 年 1 月号付録, 次世代シリアルインターフェース IEEE 1394 がわかる本, オーム社
- 4) 嶋 久登, 濱田一郎, 藤森隆洋, 小川和幸: “デジタル VTR 用デジタルインターフェース”, 信学技報, MR 95-36 (Oct. 1995)

かとう じゅんじ
加藤 淳二 1993 年、三重大学電子工学科修士課程修了。同年、ソニー(株)に入社。デジタル・カムコーダ等のデジタルインタフェースの開発に従事。