リアルタイムシステムで PCI Express を適用するために必要な高精度同期技術について

中谷 好博，西村 康裕

オムロンが開発するマシンオートメーションコントローラの NJ シリーズや NX シリーズでは、内部の CPU（Central Processing Unit）と構成するデバイス間が PCIe（PCI Express）を用いて接続されている。PCIe の特長として DMA（Direct Memory Access）を用いて CPU を介さずに直接メモリアクセスを実行し、データ転送を実行することが可能である。マシンオートメーションコントローラといったリアルタイムシステムに PCIe を導入するためには、メモリリソースが競合しないように DMA 転送タイミングを制御する必要がある。

従来は 1 つの CPU による集中制御であったため、DMA 転送タイミングの制御が容易であり、メモリリソースが競合しないように制御することができていた。近年のリアルタイムシステムでは多機能でありつつ高速・高精度な制御性能が求められている。これらを両立するために複数の PCIe デバイスが CPU との間で DMA 転送する分散処理アーキテクチャが必要となる。リアルタイム処理に影響を与えないようにするためには、各 PCIe デバイスが制御周期と高精度に同期したタイミングでデータ転送を行う必要がある。

本稿では PCIe デバイスが CPU とのリソース競合を回避する技術とその技術を実現するために必要不可欠であるハードウェアによる高精度時刻同期技術を提案する。それにより 39 ナノ秒以下という時刻同期精度を実現することができ、複数の PCIe デバイスが CPU と DMA 転送する際にリアルタイム処理への影響を抑えられる。

High-accuracy Synchronization Technology for PCI Express in Real-Time Systems

NAKATANI Yoshihiro and NISHIMURA Yasuhiro

In the NJ and NX series of OMRON's machine automation controllers, the internal CPU (Central Processing Unit) and the internal devices are connected using PCIe (PCI Express). One of the features of PCIe is that it allows direct memory access (DMA) to be executed data transfer without going through the CPU. To introduce PCIe into real-time systems such as machine automation controllers, it is necessary to control the DMA transfer timing to avoid memory resource conflicts.

Traditionally, centralized control by a single CPU made it easy to control DMA transfer timing and avoid memory resource conflicts. However, modern real-time systems require multifunctionality as well as high-speed and high-precision control cycles. To achieve both, a distributed processing architecture is needed where multiple PCIe devices perform DMA transfers with the CPU. To prevent impacts on real-time processing, each PCIe device must transfer data at timings synchronized with the control cycle and with high precision.

This paper proposes a resource conflict avoidance technology for PCIe devices with the CPU and a high-precision time synchronization technology using hardware, which is essential to realize this technology. Synchronization accuracy can be achieved less than 39 ns. This allows multiple PCIe devices to perform DMA transfers with the CPU while minimizing the impact on real-time processing.

Contact : *NAKATANI Yoshihiro* yoshihiro.nakatani@omron.com

1. まえがき

オムロンが開発しているマシンオートメーションコントローラ[1]のNJシリーズやNXシリーズ（以下、コントローラ）は高速高精度な制御を実現するリアルタイムシステムである。コントローラは複雑な制御処理を高速に実行するために高性能な汎用CPU（Central Processing Unit、以下汎用CPU）を用いてリアルタイム制御処理を実行している。汎用CPUを用いることで、世の中の半導体技術の進化に追従してその処理性能の向上を図ることができている。しかし、汎用CPUはリアルタイムシステムに専用設計されているわけではないため、定周期かつ低遅延で制御を実行するために必要なタイマ回路がほとんど実装されていない。低遅延で処理を実行するためにはタイマ割込み発生の都度どの処理を実行するのかを判断するのではなく、処理に対応したタイマ割込みを発生させることが望ましいため、複数のタイマ回路が必要となる。汎用CPUを使用するだけでは、リアルタイムシステムを制御するために必要なタイミングを精確に複数発生させることが難しいといった課題がある。

この課題を解決するために、コントローラでは独自のタイマデバイス（以下、タイマデバイス）を搭載している。タイマデバイスには複数のタイマ回路が実装されており、リアルタイムシステムを制御するために必要なタイミングで精確な割り込みを複数発生させることができる。その割り込みに応じて汎用CPUが処理を実施することにより、定周期かつ低遅延な処理を実現している。

汎用CPUは一般的にPCI Express[2]（Peripheral Component Interconnect Express以下、PCIe）という広帯域な汎用バスインターフェースが外部デバイスと接続する標準バスとなっている。コントローラにおいても汎用CPUとタイマデバイスとの接続にはPCIeを使用している。

また、コントローラではタイマデバイス以外に汎用CPUからフィールドバスやローカルバスへアクセスするインターフェース変換デバイスとの接続にもPCIeを使用している。汎用CPUとPCIeデバイス間のデータ転送においては、汎用CPUがPCIeデバイスに対してReadまたはWriteを指示するプログラム転送によって汎用CPUが持つメモリとPCIeデバイスとの間でデータの転送が実施される。しかし、データ転送のためには汎用CPUが指示する必要があり、転送オーバーヘッドが大きい上に汎用CPUの演算リソースが消費されるといった課題がある。

その課題の解決のために一般的にPCIeの通信ではDMA（Direct Memory Access）転送という汎用CPUの演算処理部分を経由せずにデバイス間で直接的にメモリ領域のデータにアクセスできる機能を使用する。DMA転送を活用することによって、汎用CPUの演算処理に負荷をかけずに汎用CPUが持つメモリとPCIeデバイス間でデータを高速に転送することができる。

近年ではコントローラにはリアルタイム性が必要な制御処理以外にも他デバイスとの通信など非リアルタイムの機能追加の要望がある。それらの非リアルタイム機能をコントローラに搭載することにより、汎用CPUの演算処理の負荷が上昇している。リアルタイム処理と非リアルタイム処理が1つのCPUに共存することでリソース競合が発生し、制御処理のリアルタイム性能へ及ぼす影響が顕著になりつつある。そうした中、コントローラを始めとするリアルタイムシステムにおいては、非リアルタイム処理を制御処理用CPU（以下、制御用CPU）とは別のCPU（以下、非リアルタイム用CPU）で制御処理以外の処理を実行する分散アーキテクチャが考えられている。分散アーキテクチャの一形態として、非リアルタイム用CPUと制御用CPUがPCIeによって接続され、DMAを使用したデータ交換を実現するアーキテクチャがある。

非リアルタイム用CPUから制御用CPUへのデータ転送は制御周期とは非同期で行うが、PCIeの標準機能にはデータ転送タイミングを制御する仕組みがない。データ転送タイミングによっては制御用CPUが重要な制御処理実行中にデータ転送されることでメモリアクセスに関するリソース競合が発生し、制御処理に影響する懸念が生じている。データ転送による制御処理への影響を最小化するためには、制御処理の重要な処理期間を避けてデータ転送を行えば良い。そのためには制御処理とDMA転送のタイミングを高精度に同期して制御する必要がある。

この機能を実現するため本稿では、複数のPCIeデバイス間において精確な時刻同期を実現する技術と、DMA転送タイミングを制御する技術の組み合わせによるデータ転送方法を提案する。

2. リアルタイムシステムにおけるPCIeの利用

2.1 コントローラの従来構成

リアルタイムシステムを実現する上で必要となるインターフェース変換機能とタイマ機能を持ったデバイス（以下、リアルタイムサポートデバイス）とコントローラの汎用CPUで構成される従来のコントローラシステムの構成図を図1に示す。汎用CPUとリアルタイムサポートデバイスはPCIeを介して接続されており、汎用CPU側がPCIeのRP（Root Port）、リアルタイムサポートデバイス側がPCIeのEP（Endpoint）として接続されている。

図1に示すようにリアルタイムサポートデバイスはその内部にタイマデバイスを有している。タイマデバイス内にはFRC（Free Run Counter）と呼ぶカウンタを持っている。FRCは時刻情報としてナノ秒単位で常に高精度にカウントアップしている。タイマデバイスは事前に設定した値にFRCが達すると、PCIeが持つ機能の一つであるMSI（Message Signal Interrupt）を用いて汎用CPUに対して割り

込みを通知する。汎用 CPU は割り込み通知を受けることによって、割り込みを起点にして制御処理を実行することができる。

タイマデバイスは高精度な FRC を元に割り込みを一定周期で発生させることができるため、汎用 CPU は定周期に制御処理を実行することができる。

さらにリアルタイムサポートデバイス内には、図 1 に示すように外部 IF やコントローラのローカルバスへのデータ転送デバイスとして DMAC（DMA Controller）を有している。汎用 CPU が PCIe を介して DMAC に DMA 転送の指示を与えることで外部 IF である Ethernet やローカルバスへのデータ転送を制御に影響のないタイミングで実行することができる。

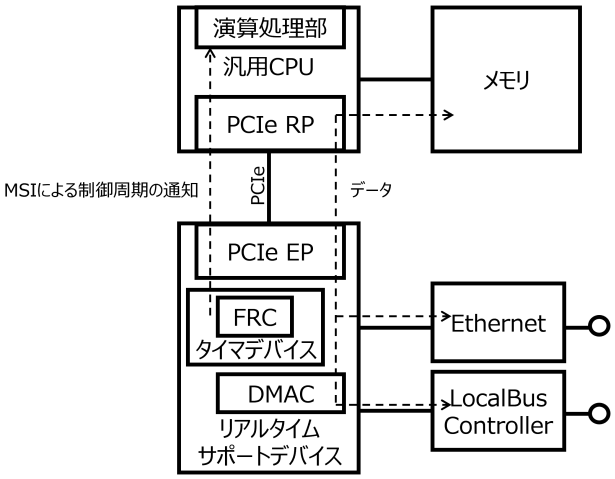

図 1　従来のコントローラの PCIe 接続構成

2.2 コントローラの分散処理システムへの進化

近年ではコントローラには制御以外に通信など他の機能の搭載も要求されてきている。これらの要求に応えるため、コントローラには制御以外の機能の搭載が進められている。しかし、追加される機能には高い演算処理性能が求められる場合が多く、追加機能を搭載することにより制御処理に悪影響が発生するといった課題が発生してきている。

制御性能と追加機能を両立するために、コントローラを始めとしたリアルタイムシステムでは、追加機能を非リアルタイム用 CPU へオフロードする分散処理システムへの進化が考えられる。図 2 に分散処理を実施するためのリアルタイムシステムにおける汎用 CPU 間の接続例の図を示す。非リアルタイム用 CPU と制御用 CPU は、DMAC を内蔵した CPU 間転送デバイスを介して PCIe で接続されている。1 つの制御用 CPU 対して非リアルタイム用 CPU は複数台接続することが可能である。非リアルタイム用 CPU を複数台接続することで追加する機能の規模に応じてシステムを拡張することができる。非リアルタイム用 CPU は CPU 間転送デバイスの DMAC を使用して制御用 CPU とデータ交換を行う。これにより、それぞれの CPU の演算処理部を介さずに必要なデータ転送を行うこととなり、演算処理への負荷を低減した状態でのデータ転送が可能となっている。

図 2 に示す構成においては、非リアルタイム用 CPU が持つメモリと制御用 CPU が持つメモリとの間で DMA による転送が実施される。DMAC によるデータ転送の実施タイミングは DMAC を起動するタイミングによって決まる。非リアルタイム用 CPU は制御用 CPU がリアルタイム処理をしているタイミングかどうかに関わらず DMAC を起動するため、データ転送は制御処理と非同期に行われることになる。リアルタイム処理中に制御用 CPU は頻繁にメモリにアクセスをしており、リアルタイム性を維持するためにはリアルタイム処理中にメモリアクセスを阻害しないことが重要になる。非リアルタイム用 CPU からの DMA によるメモリアクセスが制御用 CPU 内部におけるリアルタイム処理中に実施された場合、制御用 CPU 内部でリソース競合が発生し、リアルタイム処理が規定時間内に終了しない可能性が発生する。さらに制御用 CPU に複数の非リアルタイム用 CPU が接続されるシステムにおいては、DMA によるデータ転送量も増加して制御用 CPU のリアルタイム性能に及ぼす影響はより増大することになる。

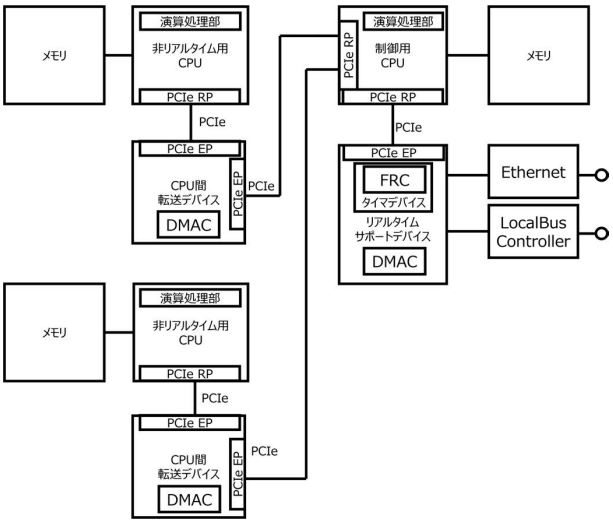

図 2　分散型リアルタイムシステムの CPU 接続図

2.3 PCIe RP 間接続のための DMAC 構成における課題

DMA 転送するための指示は制御用 CPU 及び非リアルタイム用 CPU ともに FW（Firmware）で実現している。DMA 転送に必要な転送先のメモリアドレスを入手するためには FW が転送先とハンドシェークを行って、これを取得する必要がある。FW による DMA 転送制御を簡略化するためには転送先アドレスを把握しなくても制御用 CPU と非リアルタイム用 CPU 間で DMA による転送が実行できるよう

にすることが望ましい。これを実現する CPU 間転送デバイスの DMAC 部分として、図 3 に示すような TX 側 DMAC と RX 側 DMAC を備えた構成[3] が考えられる。各 CPU がそれぞれ RX 側 DMAC を制御するため、各 CPU が管理するメモリアドレスを RX 側 DMAC に設定することができる。

リアルタイムシステムでは、制御処理を実行する際には処理に必要なデータが全て揃っている必要があるため、送信したデータが確実に相手側で受信されていることが重要である。そのため各 CPU はそれぞれの RX 側 DMAC を起動し、受信可能状態にしておく。これにより、データ送信時において CPU 間転送デバイス内に送信データを滞留させることなくメモリ領域に送信することができ、送信した直後に確実に相手側で受信することができる。

データを転送する際はそれぞれの CPU がそれぞれの TX 側 DMAC を起動する。TX 側 DMAC はそれぞれの CPU のメモリからデータを取得し、RX 側 DMAC へ渡し、相手先のメモリへデータを転送する。これにより送信する際のタイミングはそれぞれの CPU が決めることができる。制御用 CPU はリアルタイム性に影響のある処理をしている期間は、DMAC を起動しないことによってデータ送信を抑制することが可能である。一方で受信については、DMAC を常に受信可能状態にしておく必要があるため、受信するデータの転送を止めることはできない。制御用 CPU へデータが転送されてくるタイミングは非リアルタイム用 CPU 側 DMAC の送信タイミングに依存する。その結果、DMA 転送タイミングによって制御用 CPU のリアルタイム性能に影響を与えるといった課題が発生する。

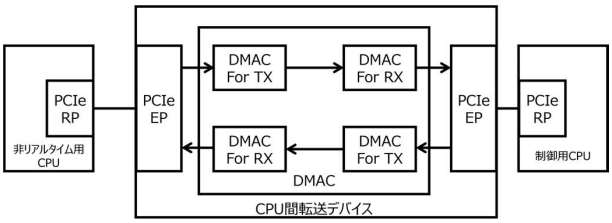

図 3　CPU 間転送デバイスの DMAC 構成

3. 提案する手法と新たに発生する課題

先出の DMA 転送タイミング課題の解決には、各 PCIe に接続される DMA が制御用 CPU のリアルタイム処理期間を避けて転送する必要がある。これを実現するために、制御用 CPU と高精度に時刻同期するタイマ機能と、このタイマにより転送タイミングを自動で制御する DMAC によるデータ転送手法を提案する。

3.1 DMAC の転送タイミング制御

リアルタイム処理への影響の最小化を目的として、非リアルタイム用 CPU から制御用 CPU へのデータ転送の中断／再開を自動で行う DMA 転送マスク機能[4] を提案する。

図 4 に DMA 転送マスク機能のタイミングチャートを示す。転送マスク機能は CPU 間転送デバイスが有する。マスクタイミングをリアルタイムサポートデバイスから HW による信号によって配信する方法もあるが、この方法では CPU 間転送デバイスが増える毎にマスク信号を追加する必要がある。HW の変更無く柔軟に複数の CPU 間転送デバイスに転送マスク機能を追加するために、CPU 間転送デバイスでマスクタイミングを生成する方式とする。転送マスクが ON の状態では CPU 間転送デバイスは非リアルタイム用 CPU から制御用 CPU に対して DMA 転送を一時停止させる。一時停止は CPU 間転送デバイスの RX 側 DMAC に対して、次の PCIe パケットの送出を停止させることで実現する。転送マスクが OFF となった後、DMA 転送を再開させる。PCIe はデータの塊をパケットとして転送しており、パケットの途中で転送を停止できない。一時停止時に即座に転送を停止し、リアルタイム処理への影響を最小限に抑えるために、PCIe のパケットサイズを最小の 128 Byte とする。それによってパケット数が増えるために転送効率が悪化し転送帯域は低下するが、転送マスクが ON になってから DMA 転送が一時停止するまでに時間を最小化することができ、リアルタイム処理の定周期性を確保しやすくなる。

転送マスクは CPU 間転送デバイスごとに複数設定することができる。制御用 CPU に複数の非リアルタイム用 CPU が接続されていても、DMA 転送タイミングを個別に設定することが可能なため、メモリ競合を低減させることが可能となる。

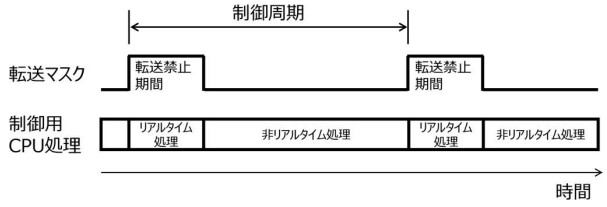

図 4　DMA 転送マスク機能

3.2 PCIe ノード間の高精度時刻同期と新たに発生する課題

前述した DMA 転送マスク機能は制御用 CPU 上の FW がマスクの開始時刻と終了時刻を CPU 間転送デバイスに設定することによって、CPU 間転送デバイス内の FRC がその設定値と一致したタイミングで DMA 転送の一時停止／再開を制御するものである。この場合、制御用 CPU は制御サイクルごとに新たなマスク設定値を CPU 間転送デバイスに設定する必要があり、FW の負荷となる。この課題を解決するため、CPU 間転送デバイスが DMA 転送マスク機能の設定値を周期的に更新する構成とする。すなわち、コントローラは定周期で動作しているため、制御用 CPU 上の FW から CPU 間転送デバイスに対して、周期と転送禁止

期間の初期設定を行うことで DMA 転送マスク機能は制御周期と同期されたタイミングで有効に機能する。初期設定以降は FW による周期的にマスク設定すると言った制御は不要となる。この DMA 転送マスク機能を制御周期に同期して機能させるためには、図5に示すようにCPU間転送デバイスにタイマデバイスを持たせ、リアルタイムサポートデバイスのタイマデバイスと CPU 間転送デバイスのタイマデバイスのタイマ値が高精度に同期する必要がある。コントローラと外部機器間の制御で要求される同期精度は1マイクロ秒以下である。この同期精度を達成するためにコントローラの設計としては制御処理の揺らぎをその 10% 以下である 100 ナノ秒以下に抑えることを目標とする。

この目標を実現するためには、タイマデバイス間の高精度な時刻同期が不可欠である。DMA 転送を一時停止させたいタイミングによっては、最大で 128 Byte の転送が行われた後に DMA 転送が実際に停止される。すなわち、転送が停止するまでの最大時間は PCIe Gen1 X1 で 128 Byte を転送する時間に等しい 61 ナノ秒となる。機器間の同期精度の目標値が 100 ナノ秒以下であることから、DMA の一時停止にかかる最大時間である 61 ナノ秒を考慮し、リアルタイムサポートデバイスのタイマデバイスと CPU 間転送デバイスのタイマデバイスのタイマ値が 39 ナノ秒以下という高精度で時刻同期することをタイマデバイス間の同期目標とする。各タイマデバイスの高精度な時刻同期は、タイマの基準となる FRC が高精度に値を同期することで実現できる。

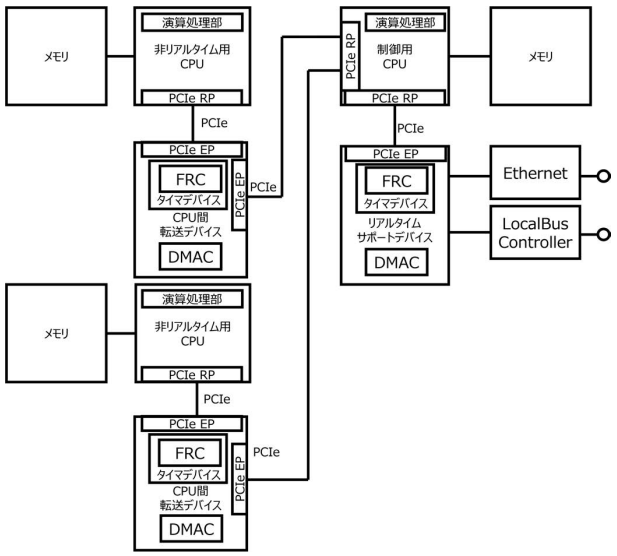

図 5　CPU 間転送デバイスにタイマデバイスを搭載した構成図

CPU 間転送デバイスとリアルタイムサポートデバイスではそれぞれ独立した基準クロックを使用するため、各基準クロックの偏差によってそれぞれの FRC が徐々にずれていく現象が発生する。基準クロックに使用されるクロック発振器は比較的精度の良いもので1ミリ秒につき最大で±25 ナノ秒程度の偏差を持っている。これらの発振器を図5に示すCPU間転送デバイスにタイマデバイスを搭載した構成の基準クロックとして採用した場合、CPU 間転送デバイス側のクロックとの精度は、1ミリ秒につき 2 倍の最大±50 ナノ秒ずれる可能性がある。各 CPU 間転送デバイスの FRC 値を高精度に同期するためには、定期的に CPU 間転送デバイスの FRC の値を、基準となるリアルタイムサポートデバイスの FRC の値に補正をする処理が必要である。従来において採用していた時刻同期の実装方法では FW が各 FRC の時刻情報を取得し、対象の FRC を定期的に補正していた。高精度に FRC 値を同期するためには補正周期を短くする必要があるが、これは制御用 CPU の演算リソースの消費増大につながり制御性能に影響が出てしまう。

一例として、FW で時刻同期することを想定して 8 ミリ秒に 1 回補正を行ったケースを考えると、その 8 ミリ秒の間に最大±400 ナノ秒のずれが発生する可能性がある。前述した通り必要な同期精度は 39 ナノ秒以下のため、±400 ナノ秒の時刻情報のずれは大きな課題となる。その解決策として高精度時刻同期技術を提案する。

4. 高精度時刻同期課題解決手法の提案

4.1 HW による時刻同期

前述の DMA 転送マスク機能を高精度に機能させるためには、PCIe デバイス同士が同じ時刻情報を持って、正確に同期する必要がある。従来においてデバイス間の時刻同期にはいくつかの手法が取られている。その手法の 1 つを図6を用いて説明する。プライマリとなる FRC を持ったデバイスからトリガ信号を出力する。プライマリ／レプリカ双方の FRC はトリガ信号の受信時の値を保存する。それぞれの保存した値を比較し、その差分をレプリカの FRC に対して補正をかけることで時刻を同期させている。

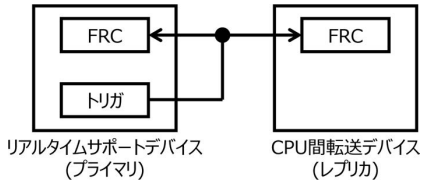

図 6　従来の時刻同期手法

従来は FW によって、プライマリに対してトリガ発行を指示する。その後FWがプライマリ／レプリカのFRCの値を取得し、FW が取得した FRC の値の差分を元にレプリカの FRC 補正していた。この方法は定期的に高頻度に行う必要があり、FW の負荷が増大する傾向がある。FW の負荷を軽減し、より高速に時刻補正を行うために HW（Hard-

ware）によって処理を実現[5]することを提案する。

図 7 に HW による時刻補正の構成図を示す。プライマリはトリガ発行に応答して FRC の値を取得し、その値を PCIe とは独立した 1 本の専用線を用いた非同期シリアル伝送にてレプリカに直接転送できる構成とした。レプリカが複数存在している状態においても、プライマリからの FRC データ送信は複数のレプリカに対してマルチドロップ接続によってブロードキャストされる。

図 8 に時刻補正のタイミング図を示す。最初にプライマリはトリガを発行する。プライマリ／レプリカはそれぞれ、トリガ受信時の FRC 値を保存する。次にプライマリはその保存した FRC 値をレプリカに伝送する。レプリカはプライマリからの FRC 値と自身が保存している FRC 値とを比較し、その差分をレプリカの FRC に対して補正処理を実行する。これらにより、FW を使用せず HW のみの実装で時刻同期を実現することができる。

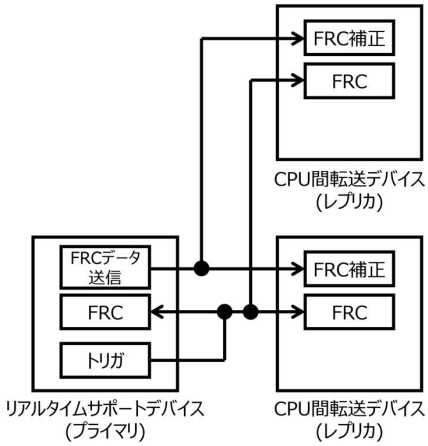

図 7　HW による時刻補正の構成図

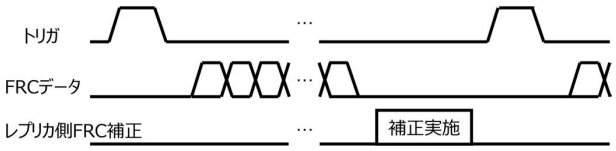

図 8　HW による時刻補正のタイミング図

5. 本提案手法の効果

前章で提案した高速な時刻同期を行うために、本章では HW を用いた時刻同期の実機確認を行い、その結果を図 9、図 10 に示す。

時刻同期実施後、プライマリとレプリカの時刻のずれ幅の大きさは補正処理によっても変わるが、ずれ幅に対して最も影響する要素が補正周期である。そこで、実機確認では補正周期を 8 ミリ秒と 250 マイクロ秒で比較した。

本実機確認で採用した補正処理について説明する。時刻同期におけるレプリカ側での補正処理の実装は様々ある。もっとも単純にはレプリカ側が認識したプライマリ側の FRC 値をそのままレプリカ側にコピーするなどの方法があるが、それでは FRC 値が大きく変化する場合があり、FRC を使用しているタイマへの影響が出てしまう。そこで、レプリカ側の FRC 値を少しずつ早める、または遅くするなどの補正処理を行うのが一般的である。今回の補正処理では、補正周期が 8 ミリ秒の場合は 70 マイクロ秒毎に 1 ナノ秒の補正、補正周期が 250 マイクロ秒の場合は 1 マイクロ秒毎に 1 ナノ秒の補正を実施することとし、最大 100 ナノ秒の補正を行える回路とした。

5.1 HW を用いた時刻同期手法の実機確認

前章で提案した HW を用いた時刻同期を実施した結果を図 9 に示す。図 9 では図 2 において、リアルタイムサポートデバイス側と CPU 間転送デバイス側の FRC の一部のビットをオシロスコープにて観測した波形である。リアルタイムサポートデバイス側の波形の立ち上がりをトリガにし CPU 側転送デバイスの波形を観測したものである。図 9 では HW にて 8 ミリ秒に一度補正処理を実行しており、補正元である制御用 CPU 側の FRC と比較して、−65.2 ナノ秒～−24.8 ナノ秒（センター：−45.0 ナノ秒）の精度となっている。FRC 値の補正する際は徐々に FRC 値を増減していく手法を採用しているため、トリガ発生のタイミングから FRC 値の補正が完了するまでには時間がかかる。そのため、FRC 値の補正が完了した段階においては、トリガ発生時に発生していた FRC 値のずれ量の補正が完了しているが、補正完了までにかかる時間の間に FRC ずれが発生しており、そのずれが 24.8 ナノ秒という形で見えていると考える。また、FRC の補正が完了してから次の補正が開始されるまでの 8 ミリ秒の間に 40.4 ナノ秒の FRC ずれが発生したと考えられる。

本手法によって、HW で問題なく時刻同期が実現できることがわかった。8 ミリ秒で時刻同期する場合、理論上最大 400 ナノ秒の偏差が出る可能性があったが、今回実験に用いた発振器の精度、温度等の条件が比較的良く、時刻のずれ幅の大きさについては約 65 ナノ秒が観測されている。

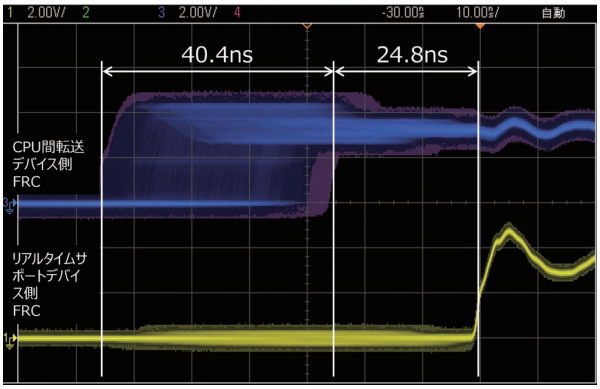

図 9　8 ミリ秒周期で時刻補正時の FRC 出力

5.2 HW による時刻補正周期短縮化の効果

HW で高精度な時刻同期を実現するため、高頻度に時刻補正を実施した結果を図 10 に示す。HW で補正するようにしたため、250 マイクロ秒という高速な補正周期を実現することができた。その同期精度は－6.5 ナノ秒～＋4.5 ナノ秒（センター：－1.0 ナノ秒）となっており、8 ミリ秒間隔の補正よりも大幅に同期精度が向上している。

今回実験に使用したタイマデバイス内部のクロック周期は 8 ナノ秒であり、トリガ信号は 8 ナノ秒のクロックでラッチされる。トリガ信号とクロックとは非同期であるため、トリガ信号をラッチするタイミングによって±8 ナノ秒の時刻同期誤差は発生する。そのため、図 10 においてリアルタイムサポートデバイスよりも CPU 間転送デバイスの方が最大 +4.5 ナノ秒進んだ状態が観測されたものと考えられる。

補正周期を短くすることで、FRC 値の同期精度を大幅に改善することが実機評価でも確認でき、目標である 39 ナノ秒以下である 11 ナノ秒の同期精度を実現することができた。

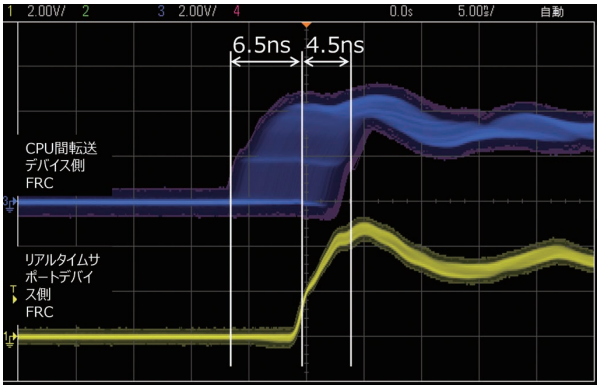

図 10　250 マイクロ秒周期で時刻補正時の FRC 出力

この HW による時刻同期技術によって、リアルタイムサポートデバイスと CPU 間転送デバイスの FRC 値が高精度に同期することができるため、DMA 転送マスク機能を高精度に機能させることができる。DMA 転送マスクは CPU 間転送デバイスごとに設定することができるため、複数の CPU 間転送デバイスからの DMA 転送においてもメモリ競合なく DMA 転送することができ、リアルタイムシステムへの影響を最小限に抑えることができる。

6. むすび

本稿ではリアルタイムシステムに PCIe を適用するために必要な技術について論説した。コントローラの進化の一形態として分散型リアルタイムシステム構成を示した。分散型リアルタイムシステムでは複数の PCIe デバイスからの DMA 転送によってリアルタイム性能への影響が発生するといった課題に対し、DMA 転送マスク機能によるリアルタイム処理中の DMA 転送を抑制する仕組みを提案した。DMA 転送マスク機能を効果的に機能させるためにはリアルタイムサポートデバイスと CPU 間転送デバイスの各タイマデバイスのタイマ値を同期させて DMA 転送タイミングを高精度に同期させるといった新たな課題が発生した。これに対して HW による時刻同期技術を新たに提案し、高頻度に時刻同期を実施することにより、時刻同期の目標値を達成することができた。

今後においても、リアルタイムシステムに PCIe デバイスが活用されていくことが予想される。特に Time Sensitive Network（TSN）など、コントローラは外部機器とのリアルタイム性を維持した接続が求められていくと考える。コントローラにとって時刻同期はリアルタイムシステムとして重要な要素となり、コントローラ内の時刻同期精度が外部機器との同期精度とも密接に関係している。本稿で提案した時刻同期技術は今後、外部機器と高精度に時刻同期するために重要な技術となる。

参考文献

1）オムロン株式会社. *マシンオートメーションコントローラ NJ/NX シリーズ*, 第 AH版.（2025）. Accessed: Feb. 17, 2025.［Online］. Available: https://www.fa.omron.co.jp/data_pdf/cat/nj_nx_sbca-100_14_2.pdf
2）*PCI-SIG*, PCI Express Base Specification Revision 2.1, 2009.
3）オムロン株式会社, “転送装置、情報処理装置、および、データ転送方式,” 特許第 7326863号, Aug. 7, 2023.
4）オムロン株式会社, “情報処理装置,” 特許第 7259537号, Apr. 10, 2023.
5）オムロン株式会社, “制御装置,” 特開 2022-119703, Aug. 17, 2022.

執筆者紹介

中谷 好博　NAKATANI Yoshihiro
インダストリアルオートメーションビジネス
カンパニー
商品事業本部 センサ事業部
第 2 開発部
専門：電子工学

西村 康裕　NISHIMURA Yasuhiro
インダストリアルオートメーションビジネス
カンパニー
商品事業本部 コントローラ事業部
ハードウェア開発部
専門：情報工学
技術士（情報工学部門）