Vol 56, No 1—January 2022

ANALOG DEVICES
想像を超える可能性を
AHEAD OF WHAT'S POSSIBLE™

AnalogDialogue

ウェアラブル機器に不可欠なDMA、ペリフェラルに対するデータ転送を低消費電力で実現

著者：Brandon Hurst、ハードウェア／組み込みファームウェア・エンジニア

概要

本稿では、組み込みシステムのプログラミングにおけるDMA（Direct Memory Access）のユース・ケースを紹介します。また、DMAがもたらすメリットとデメリットについても解説します。DMAは、CPUをより効率的に動作させるために利用されます。その基本的な役割は、CPUを介すことなく、ペリフェラルやメモリ・モジュールなどとの間でデータ転送を行えるようにすることです。本稿では、その仕組みについて簡単に説明します。その上で、DMAバスにアクセスするための3種のアーキテクチャを取り上げ、それぞれがもたらすメリットについて解説します。

DMAの概要

あらゆる組み込みシステムには、共通するいくつかのタスクがあります。その1つが外部入力の管理です。そのための処理は、CPUに対して多くの無駄な演算負荷をかける可能性があります。結果として、アクティブ・パワー・モードで動作する時間が長くなったり、応答時間が長くかかったりといったことが起こり得ます。組み込みシステムにおいては、消費電力を最適化し、イベントに対する迅速な応答を維持しつつ、大量のデータを連続的に転送できるよう管理を行う必要があります。多くの場合、それに向けた最良のソリューションはDMA機能を備えるマイクロコントローラとなるでしょう。

通常、システム・アプリケーションでは数多くのペリフェラルが使用されます。そうしたアプリケーションでは、CPU（マイクロプロセッサやマイクロコントローラ）がボトルネックになるケースが少なくありません。例えば、絶えずデータを送信するA/Dコンバータを管理する場合には、非常に頻繁に割り込みが生じることになります。そうすると、CPUは他のタスクを完了することが難しくなるかもしれません。DMAは、このような課題を解決するために使用されるデータの転送手段です。DMAを利用することにより、大量／高速なトランザクション（データのやり取りを伴う一連の処理）に対するCPUの関与の度合いを最小限に抑えることができます。DMAに使用するコントローラは、メモリやペリフェラルを対象としたデータ転送だけをサポートするコプロセッサだと考えることができます。DMAコントローラを利用すれば、CPUによって負荷の大きいペリフェラルを適切に管理したり、CPUを他のタスクに専念させたりすることが可能になります。また、バックグラウンドでトランザクション処理が行われている間、CPUをスリープ状態に移行させて消費電力を削減するといったことも行えます。例えば、Arm®アーキテクチャの場合、DMA用のモジュールはLP2（スリープ）モードでもLP3（実行）モードでも動作することが可能です。これは、ウェアラブルなセンサー・ハブやスマートウォッチなど、バッテリ寿命の長さが重要なアプリケーションにとって非常に望ましい機能です。

DMAがもたらすメリット、デメリット

DMAは、多くのデジタル・システムにとって有用なものです。例えば、バスにおける大量のトラフィックを管理するためにDMAが必要になるケースがあります。あるいは、ネットワーク・カードやグラフィックス・カードでもDMAが大きな役割を果たすことがあります。初期のIBM互換PCでもDMAは使われていました。しかし、DMAを設計に盛り込む場合には、いくつかのトレードオフについて検討する必要があります。DMAを採用することで、表1、表2に示すようなメリット、デメリットが生じるからです。

表1. DMAがもたらすメリット

DMAがもたらすメリット	
CPU時間	DMAは、プロセッサの実行時間と割り込みの必要性を最小限に抑えます。そのため、トランザクションに必要なCPU時間が短縮されます。
消費電力	DMAを利用したデータ転送の最中にCPUをスリープにできる場合には、消費電力を最小限に抑えられる可能性があります。
並列処理	システム・バスのアーキテクチャによっては、ペリフェラルに対するトランザクションの実行中に、プロセッサによって他の処理を実行できる可能性があります。

表2. DMAがもたらすデメリット

DMAがもたらすデメリット	
コスト	DMAを利用するにはDMAコントローラが必要です。それによってシステムのコストが増える可能性があります。
複雑さ	DMAを使えば割り込みの頻度を下げられるかもしれません。しかし、アプリケーション・ファームウェアのサイズと複雑さは増大する可能性があります。
プラットフォームへの依存性	DMAコントローラの内部アーキテクチャはメーカー間で異なります。また、同一メーカーの製品の間でも異なる可能性があります。そうしたネイティブなバス・アクセス機構に依存して動作が異なるかもしれません。
キャッシュのインコヒーレンシ	DMAによるトランザクションでは、メモリ階層におけるキャッシュ層へのデータの書き込みによって論理エラーが生じる可能性があります。この問題は、キャッシュ・コヒーレントなシステム・アーキテクチャを採用するか、DMAによる処理の完了時にキャッシュ・ストレージを無効にすることによって解消できます。

バスへのアクセスとCPUサイクル

DMAコントローラは、消費電力の低減や組み込みシステムの高速化という面で多大な効果を発揮する可能性があります。ただ、その実装については厳格な標準化が図られているわけではありません。CPUから内部バスへの同時アクセスを防ぐための機構は、何種類も存在します。バスのアクセス機構で実現したいことは、同じメモリ・アドレスに対する同時アクセスを回避することです。そのようなアクセスが発生すると、キャッシュのインコヒーレンシや論理エラーが発生するおそれがあるからです。通常、DMAコントローラは各種のアクセス機構のうちいずれかに構成（コンフィギュレーション）されます。各アクセス機構によって、必要なハードウェアやファームウェアによる制御内容が異なるからです。多くのDMAコントローラで採用されているアクセス機構としては、バースト、サイクルスチール、トランスペアレントという3つの方式が挙げられます。

バースト方式では、大量のデータのバースト転送が不定期に発生します。それに対応し、DMAコントローラは送信先のバッファに格納可能な範囲でできるだけ多くのデータを転送します（図1）。DMAコントローラは、CPUの処理をごく短時間停止し、大量のデータの転送を行った後に、バスの権限をCPUに返します。このような制御が、データの転送が完了するまで繰り返されます。一般に、バースト方式は最も高速なDMA方式だと考えられています。

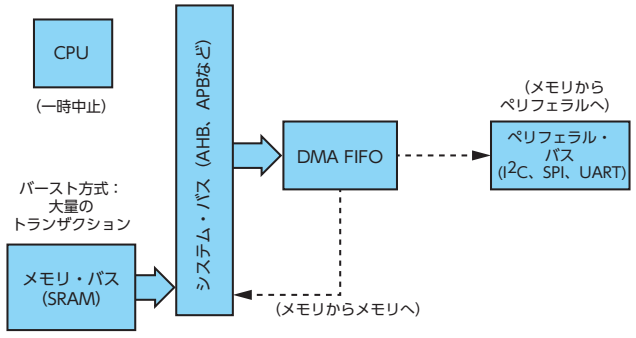

図1. バースト方式のDMA

一方、**サイクルスチール方式**（シングルバイト転送方式）では、CPUからキューを取得し、命令が実行される合間をぬってDMAの処理が行われます。2つのCPUサイクルの間に1つの処理を挿入することで、CPU時間を実質的にスチールする（盗む）ということです。サイクルスチール方式では、1度に1つのDMA処理しか実行できません。そのため、一般的にバースト方式よりも低速だと言えます。

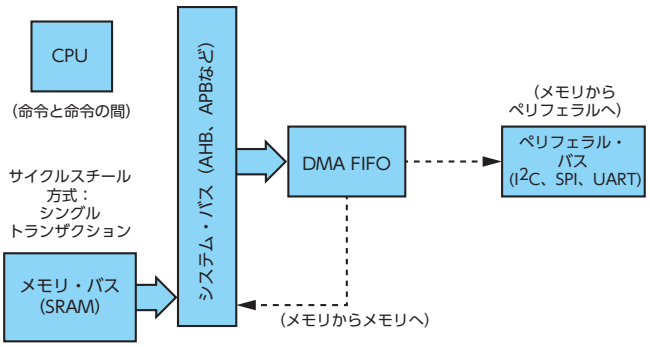

図2. サイクルスチール方式のDMA。
2つのCPUサイクルの間にDMA処理が実行されます。

トランスペアレント方式においても、1度に実行できるDMA処理は1つです（図3）。更に、この方式では、対象とするデータやアドレス・バスにアクセスする命令をCPUが実行するのを待たなければなりません。それに伴い、トランスペアレント方式では、このアクセス制限についての検証を実施するためのロジックが追加で必要になります。また、この方式は3種のDMA方式の中で最も低速になります。トランスペアレント方式は、メモリ・バスへのアクセスを必要としない処理が他にも存在するアプリケーションにおいて有用である可能性があります。この方式の長所は、プロセッサが完全に動作を停止する必要がなく、CPUのスロットリングが不要であることです。

表3は、3つの方式の長所と短所をまとめたものです。

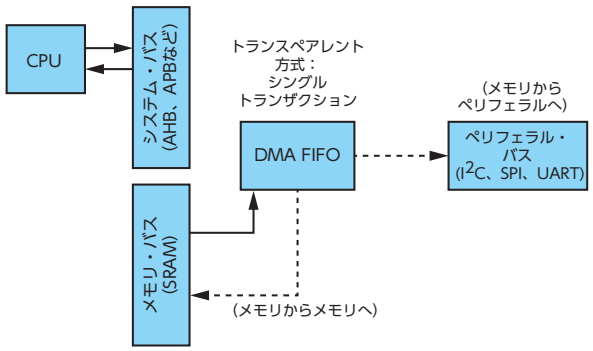

図3. トランスペアレント方式のDMA。
CPUがデータやアドレス・バスにアクセスしない
タスクを実行している間にDMA処理を行います。

表3. DMAの各方式の長所と短所

DMAの方式	長所	短所
バースト方式	最も高速な DMA方式	CPUのアイドル時間が 比較的長い
サイクル スチール方式	CPUが長い期間、 連続的にアイドル状態に なることはない	バースト方式よりも 低速
トランス ペアレント方式	CPUのスロットリング が不要	最も低速な DMA方式

バースト方式の実例

ここでは、バースト方式のDMAコントローラ（以下、DMAC）を採用したマイクロコントローラの例として「MAX32660」を紹介します。図4において、上部のパスは、AHB（Advanced High Performance Bus）とDMACのロジックの間のデータ・フローに対応します。一方、下部のパスは両者の間の制御フロー／ステータス・フローに対応しています。DMACは、構成方法によっては、AHBとメモリ／ペリフェラル・モジュールの間のバッファ・インターフェースとして動作させることも可能です。DMACのロジックは、DMA用のバッファと各ペリフェラルの間に位置します。そして、各ペリフェラルに固有のペリフェラル・バスをトランザクションの実行時に個別に管理します。1つのDMA処理によって、最大32バイトのデータを一度に転送することが可能です。但し、転送元／転送先のバッファがそれだけのデータを格納できる状態になければなりません。バッファは最大で16MBのデータを格納できます。内部メモリに対する転送に加えて、I^2C、SPI（Serial Peripheral Interface）、I^2S、UART（Universal Asynchronous Receiver/Transmitter）を介したデータの送受信に対応できるよう構成することが可能です。DMA制御を実現するためのプログラミングは、プロトコルに応じて少し異なる可能性があります。ただ、いずれにせよ、ペリフェラルのトランザクションはDMACによって排他的に管理されます。アービタ・モジュールは、4つのDMAチャンネルとCPUの間のアクセス制御を担います。それにより、優先度に基づいてアクセスの要求が許可されます。

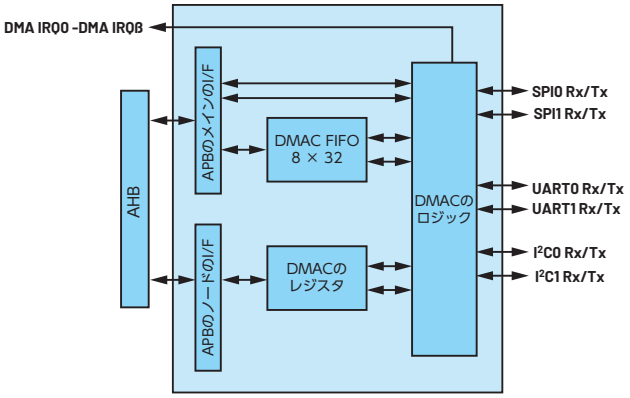

図4. MAX32660が備える
DMACのアーキテクチャ

低消費電力の機器におけるDMAの重要性

最新の組み込みシステムの例としては、大量のセンサーを管理する機能を担うものが挙げられます。そうしたシステムには、高いスループット、高い効率、少ない消費電力で動作することが求められます。DMAは、そうしたシステムに不可欠な機能だと言えます。また、メモリ・バスとペリフェラル・バスを対象とするトランザクションに特化したコプロセッサとして動作します。

多くのアプリケーションでは、消費電力を最小限に抑えつつ、プロセッサの負荷を軽減することが求められます。そうしたアプリケーションではDMAの利用が不可欠です。例えば、健康管理機器やウェアラブル機器は、重要性の高いデータを大量に処理できるだけのスループットを備えている必要があります。その一方で、バッテリの寿命はできるだけ延伸しなければなりません。アナログ・デバイセズは、MAX32660や「MAX32670」など、ウェアラブル機器に適した低消費電力のマイクロコントローラを提供しています。それらの製品では、DMAのアーキテクチャとして高速なバースト方式を採用しています。また、アナログ・デバイセズは、「MAX32666」などのDARWIN Armマイクロコントローラを、Bluetooth® 5に対応するウェアラブル機器やIoT（Internet of Things）アプリケーション向けに提供しています。それらの製品は、バースト方式に対応する8チャンネルのDMACを2個搭載しています。それらにより、イベント・ベースのトランザクションをサポートします。また、セキュアなブートローダやTPU（Trust Protection Unit）など、最高レベルのセキュリティを実現するためのハードウェアを搭載しています。そのため、ECDSA（Elliptic Curve Digital Signature Algorithm）、SHA-2、AES（Advanced Encryption Standard）に対応する暗号化を高速に実現できます。DMAは、初期のIBM互換PCやネットワーク・カードで使われていました。現在、そのDMA機能は、セキュアで低消費電力のウェアラブル機器やIoT機器といった最新のデジタル・システムに欠かせないものとなっています。

著者について

Brandon Hurst（brandon.hurst@analog.com）は、Maxim Integrated（現在はアナログ・デバイセズの一部門）のハードウェア／組み込みファームウェア・エンジニアです。入社は2021年1月で、トレーニング／技術サービス（TTS）グループに所属しています。カリフォルニア工科州立大学サンルイスオビスポ校で電気工学の学士号を取得。MaximのTTSチームとAppleの製品安全性エンジニアリング・チームでインターンシップを経験しました。

AD5601-0-1/22