アナログ・デバイセズの DSP、プロセッサ、開発ツール用テクニカル・ノート
http://www.analog.com/jp/ee-notes、http://www.analog.com/jp/processorsにはさまざまな情報を掲載しています。

Blackfin® プロセッサ用のシステム最適化技術

著者: Kaushal Sanghai Rev 1—2007 年 7 月 10 日

はじめに

効率良いシステム・リソース配分は、組込型プラットフォームで広帯域を必要とするアプリケーション開発で不可欠です。スループット条件がシステムの制限を満たしていても、実システムで帯域幅が不足することも有り得ます。スループットが期待値を下回る重要な原因としては、外部メモリアクセスにおける遅延と非効率なシステム・リソース配分があります。組込型プロセッサの能力を最大限、利用するためには、システム・アーキテクチャと使用可能なシステム最適化技術を理解することが重要です。この EE ノートは、Blackfin®プロセッサのメモリ階層とそのシステム・アーキテクチャに対するクイック・リファレンスとして機能する他、システム・リソースの効率良い利用のための幾つかの最適化技術の使用ガイドラインを提供し、推奨最適化技術を評価するベンチマーク・スタディについても説明します。

Blackfinプロセッサのアーキテクチャ

このセクションでは、Blackfin メモリ階層とシステム・アーキテクチャについて説明します。各セクションでは、始めにリソース(メモリ、システム・バス、DMA コントローラなど)の説明を行い、続いてリソースをさらに効率良く使用するための推奨事項を示します。アーキテクチャは主に性能の点から説明するため、詳細については述べません。詳細については、該当する「ハードウェア・リファレンス・マニュアル」[1、2、3]を参照してください。

メモリ階層

このセクションでは、Blackfinプロセッサのメモリ階層(図 1)と内蔵メモリ(L1 およびL2)、外部メモリ間のトレードオフについて説明します。また、コードとデータをメモリ階層に効率良く配置して、メモリ・アクセス遅延を最小にするためのガイドラインも提供します。

L1 メモリの説明

Blackfin プロセッサでは、命令 L1 メモリ空間とデータ L1 メモリ空間が分離しています。L1 データ・メモリはさらにデータ・バンク A とデータ・バンク B に分割されています。高性能を実現するために、L1 メモリは複数の要求側エレメント(コア、DMA など)に同時アクセスを許容するシングル・ポート・サブバンクとして構成されています。また、L1 メモリではアプリケーション固有の負荷特性を利用するため SRAM とキャッシュ・メモリの設定機能も提供しています。

L1 コードおよびデータ・メモリのサブバンク機能

L1 コード・メモリは、シングル・ポート・サブバンクとして構成されているため、実質的にデュアル・ポートになっています。このため、コアおよび DMA またはシステム・バスは、同じサブバンクに対してアクセスを行わない限り、L1 コード・メモリを同時にアクセスすることができます。同様に、L1 データ・メモリもマルチポート・サブバンクを構成するため、コア・クロックの 1 サイクルで次の内容を実行するこができます:

- 32 ビット・データ・アドレス・ジェネレータ(DAG)のロードを 2 回
- パイプライン 32 ビット DAG のストアを 1 回
- 64 ビット DMA I/O を 1 回
- 64 ビット・キャッシュのフィル/ビクティム・アクセスを 1 回

L1 SRAM/キャッシュの構成

デフォルトでは、すべての L1 メモリはコアに対して直接アクセス可能な SRAM メモリになっています。SRAM メモリは命令またはデータ項目に対するシングル・サイクル・アクセスを保証しているため、キャッシュ・ミスは発生しませんが、使用可能なメモリ空間サイズの制限があります。コードとデータが L1 メモリ空間より大きいアプリケーションの場合、L1 の一部をキャッシュとして構成して、メモリ・アクセス遅延を小さくすることができます。

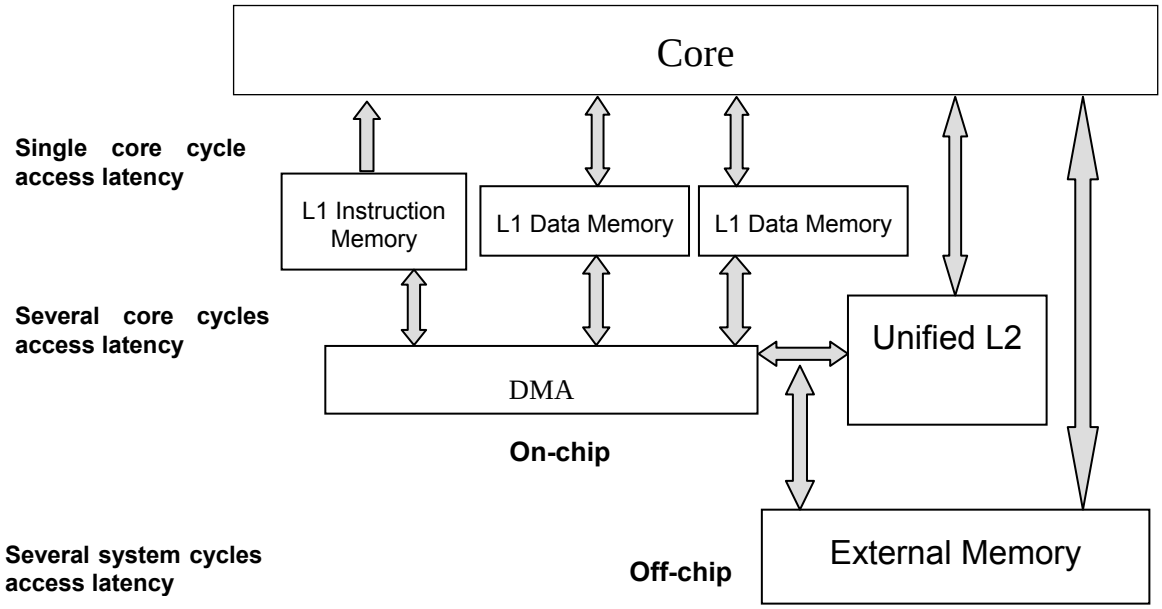

図 1 Blackfin プロセッサのメモリ階層

メモリ・キャッシュは、L2 または外部メモリに配置されたコードとデータの実行で大きな利点を提供することができます。キャッシュの性能は、アプリケーションの一時的および空間的配置の特性に依存します。キャッシュ・メモリの欠点は、キャッシュ・ミスの弊害であり、これによりメモリ・アクセス遅延が大きくなるため、外部メモリの帯域幅要求が大きくなることです。また、データのストリーミングでは、新しいデータが外部メモリへ転送されたときにキャッシュ・ラインを無効にする必要があります。キャッシュ・ラインを無効にすると犠牲が大きく、性能が大幅に低下します。

ガイドライン

マルチポート・サブバンクの利用

前述のように、コアとDMAがメモリ内の同じサブバンクにアクセスすると、メモリ競合が発生します。図 2に、データ・オブジェクトのL1 メモリへの効率的な配置によりメモリ競合を回避する方法を示します。

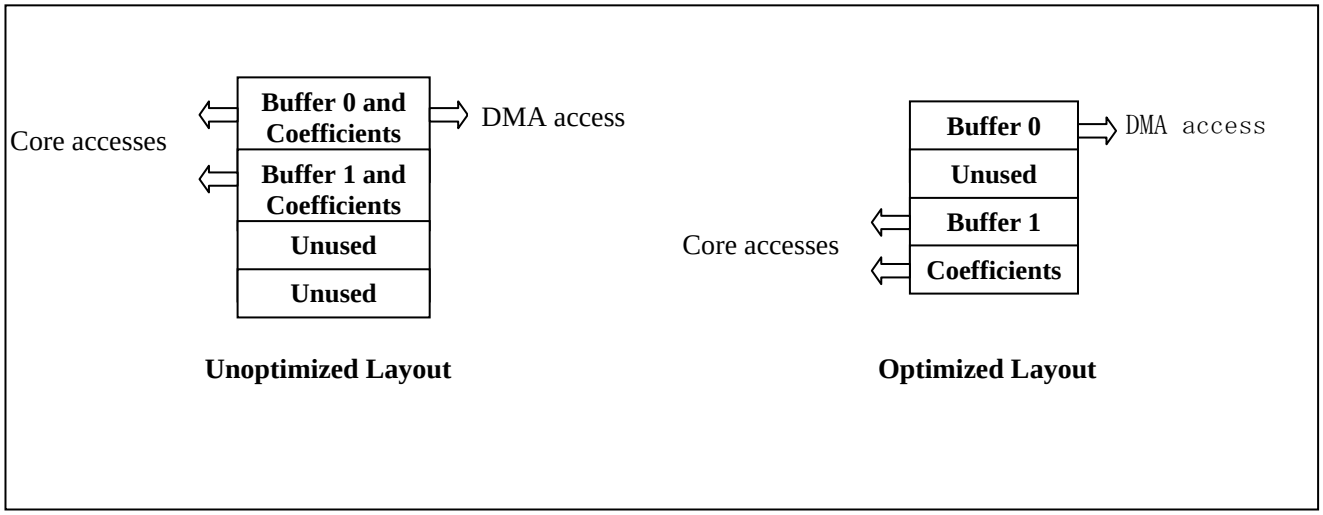

図 2.マルチポート・サブバンク内部メモリ・アーキテクチャの効率良い使用方法

L1 SRAM のみの使用

コードとデータを L1 SRAM に配置すると、メモリ・アクセス遅延が最小になります。コードとデータの合計所要メモリが L1 SRAM で使用可能なサイズを超える場合には、最も頻繁に使用される実行コードと最も頻繁にアクセスされるデータのみを L1 メモリに配置します。L1 SRAM に効率良くコードを配置するためには、自動 PGO リンカー・ツール(「*PGO Linker - A Code Layout Tool for the Blackfin Processors (EE-306)*」[8]で説明)を使います。次の数セクションで幾つかの技術を推奨しますが、データ・レイアウトはプログラマが行う必要があります。

L1 SRAM メモリは、すべての要求に対してシングル・サイクル・アクセスを保証するため、リアルタイム性のクリティカルなコード項目とデータ項目を配置するメモリとして使うこともできます。キャッシュ・メモリにはキャッシュ・ミスのペナルティがあるため、要求されたコード項目またはデータ項目のアクセス・タイムを事前に決めることはできません。これはシステムのリアルタイム条件を満たすためにクリティカルになることがあります。

L1 SRAM とキャッシュの使用

アプリケーションのコード・サイズが L1 命令 SRAM 空間より大きい場合には、命令キャッシュを使うと、平均メモリ・アクセス遅延を短くすることができます。VisualDSP++ 4.5 ツールでは、`Project Options` ダイアログ・ボックスを使うか、またはプロジェクト・ソース・ファイル内に`_cplb_ctrl` 定数を定義することにより、キャッシュを有効にすることができます。

Blackfin プロセッサでのキャッシュ・メモリの使い方については、「*Using Cache Memory on Blackfin Processors (EE-271)*[9] 」を参照してください。VisualDSP++ インストレーション・フォルダ(`VisualDSP++ 4.5/Blackfin/Examples/ADSP-BF533 EZ-Kit Lite/Cache/`)内にあるプロジェクト例にも、Blackfin プロセッサ上でキャッシュを有効にする方法を説明してあります。

また、Blackfin プロセッサはメモリ階層内でコードとデータを効率良く移動させる別の方法も提供しています。DMA エンジンは、キャッシュ・メカニズムを使うことなくコードとデータを管理することができます。DMA を使うと、コードとデータを前もって移動できるため、キャッシュ・ミスの犠牲を回避し、キャッシュ・ラインの無効化によるサイクル損失を少なくすることができます。また、DMA を使うと、コアが処理中にバックグラウンドでメモリ転送を行うことができるため、貴重なコア・サイクルを節約することができます。ただし DMA を使うと、その為のソフトウェア開発時間の増大や、コードとデータの不規則またはランダムなアクセスを要求するアプリケーションにおける実装の難しさといった懸念も存在します。

キャッシュ・ミス弊害を回避するためのDMAの使用

DMA を効果的に使用するためには、プログラムの動作とデータ・アクセス・パターンを知ることが重要です。コードとデータの管理は、すべてプログラマの役割であるため、それによりソフトウェア開発時間が増える可能性があります。DMA を使ったデータ・オブジェクトの管理の実現性と、キャッシュを使用した場合の優位性については、「*Video Templates for Developing Multimedia Applications on Blackfin Processors (EE-301)*[7]」を参照してください。キャッシュと DMA との間のトレードオフについては、「*The best way to move multimedia data*[10]」を参照してください。

L2 SRAMの説明

BlackfinプロセッサのL2 メモリは、SRAMメモリとしてのみ使用することができます。L2 のアクセス・タイムは、L1 より大きいですが、外付けSDRAMのアクセス性能より優れています。L2 メモリは、ADSP-BF561 とADSP-BF54xの両Blackfinプロセッサ上でのみ使用できることに注意してください。表 1に、コア・クロック・サイクル数(CCLK)および/またはシステム・クロック・サイクル数(SCLK)でL2 メモリ性能を示します。

Access Type	Number of Cycles	Comments
Direct core access instruction fetch (64-bit)	9 CCLKs	
Direct core access data fetch	9 CCLKs (1st 32-bit Fetch) 2 CCLKs (2nd 32-bit Fetch)	Can be 8-, 16-, or 32-bit access
Cache line fill request Instruction and data (32-byte)	15 CCLKs	First 8 bytes available after 9 CCLKs, and 2 CCLKs for each successive 8 bytes
L2 to L1 memory DMA transfers (8-byte)	2 CCLKs	
L2 system read (e.g., L2 to external memory transfer)	1 SCLK + 2 CCLKs	
L2 system write (e.g., external memory to L2 transfer)	1 SCLK	

表1. L2 メモリの性能

ガイドライン

L2 SRAM は、L1 SRAM メモリに入らないコードとデータを配置する際に使うことができます。ADSP-BF561 デュアルコア・プロセッサを使う場合は、2 個のコア間で共用するデータ・オブジェクトを L2 メモリ空間に配置することができます。L2 メモリも、マルチポート・サブバンクとしてデザインされています。このため、コアと DMA による L2 メモリへの同時アクセスが可能になります(ただしアクセスは別々のサブバンクを対象とする場合)。

外部メモリの説明

ADSP-BF561 Blackfin プロセッサの場合、SDRAM コントローラ(SDC)が 16 または 32 ビットの SDRAM メモリ・バンク幅を、ADSP-BF53x と ADSP-BF52x プロセッサの場合は SDC が 16 ビット SDRAM メモリ・バンク幅を、それぞれサポートします。SDRAM メモリは、最大 SCLK 周波数である 133MHz でデータを転送することができます。SDC は 4 個の内部メモリ・バンクを持つ標準 SDRAM メモリ・デバイスに対して外付け部品不要のインターフェースを提供し、さらにインターリーブ・バンク・メモリ・アクセスも可能にします。コアと DMA からの外部メモリに対する同時アクセスに対しては調停ロジックが必要です。このセクションでは、SDRAM メモリ・バンキングとその性能について詳しく説明します。

SDRAMバンキング

標準的な SDRAM メモリ・デバイスはメモリ・バンクを内蔵しています。SDC は、SDRAM の内部バンクを利用するために、インターリーブ・メモリ・バンク・アクセスをサポートしています。SDC インターフェースは、最大 4 個の内部メモリ・バンクをサポートします。

SDRAM 内部バンク・アドレスは、行アドレスから構成されます。内部バンクはメモリ・ページのセットに分割され、各メモリ・ページは SDC コントロール・レジスタを使って設定されます。ページ数は、SDRAM コンフィギュレーション設定値と内部メモリ・バンク・サイズにより指定されます。SDC は、各内部バンク内で 1 回に 1 つのオープン・ページしか持つことができません。内部メモリ・バンク内で閉じているページを開くときは、プリチャージ・コマンドとアクチベーション・コマンドが必要です。

メモリ内のオープン・ページに対するアクセスは、*オンページ・アクセス*と呼ばれます。オンページ・アクセスでは、プリチャージ・コマンドまたはアクチベーション・コマンドは不要です。閉じているページに対するアクセスの場合は、*オフページ・アクセス*と呼ばれ、プリチャージ・コマンドとアクチベーション・コマンドに要する遅延サイクル数が増えます。SDRAM 性能を向上させるためには、オフページ・アクセスを少なくすることが必要です。

SDRAMの性能

表 2に、オンページ・アクセスに対するSDRAM性能を示します。

Access Type	Number of SCLK Cycles for a Word* Access
Instruction/data cache line fill request	1.1
Direct core instruction fetch	1.1
DAG read access	8
DAG write access	1
MemDMA write access. (e.g., L1 to SDRAM memory transfer)	1
MemDMA read access (e.g., SDRAM to L1 memory transfer)	1.1

表2. オンページ・ワード・アクセスに対する外部メモリの性能*(* ADSP-BF56x プロセッサの場合 32 ビット、ADSP-BF53x と ADSP-BF52x プロセッサの場合 16 ビット)*

表 3に、オフページ・アクセスに対する外部メモリ・アクセスの性能を示します。

Access Type	Number of SCLK Cycles for a Word* Access
Read	$t_{RP} + t_{RCD} + CL$
Write	$t_{WR} + t_{RP} + t_{RCD}$

表3. オフページ・ワード・アクセスに対する外部メモリの性能*(* ADSP-BF56x プロセッサの場合 32 ビット、ADSP-BF53x と ADSP-BF52x プロセッサの場合 16 ビット)*

表 3で:

t_{RP} –プリチャージ・コマンドとアクチベーション・コマンドとの間の遅延(SCLK で 1～7 サイクル)

t_{RCD} –アクチベーションと最初のリード/ライト・コマンドとの間の遅延(SCLK で 1～7 サイクル)

t_{WR} –書き込みとプリチャージ・コマンドとの間の遅延(SCLK で 1～2 サイクル)

CL (CAS 遅延) –読み出しコマンドからデータ出力までの遅延(SCLK で 2～3 サイクル)

ガイドライン

オフページ・アクセスの最小化

局在性の高いコードとデータが、内部メモリ・バンクのページ境界を跨いで配置されているほどオフページ・アクセス遅延が大きくなります。オフページ・アクセスを少なくするためには、局在性の高いコードとデータのブロックを内部メモリ・バンクの同じページに配置するようにします。最大 4 個のメモリ・ページを SDRAM メモリ空間で同時にオープンできるという利点を利用するため、コードとデータの空間ブロックをページ・サイズの 4 倍にして、各内部バンク内の異なるページにこれらを配置するのが最適です。

アプリケーションの空間特性の把握が困難な場合は、アプリケーションに特有なトレードオフについて次の手法を検討することができます。

1. 命令とデータのミックスの空間特性の測定と使用はさらに困難です。プログラミング労力を軽減するため、場合によっては、L2 と外部メモリを別々のコード・メモリ空間とデータ・メモリ空間として使うことができます。例えば、アプリケーション・コード(または最も頻繁に使用されるアクセスされるコード)をすべてL1 メモリまたはL2 メモリに配置できる場合は、すべての外部メモリをデータ・オブジェクトに割り当てることができます。データ・オブジェクトを外部メモリだけに配置すると、外部メモリに対するアクセスの制御性と予測可能性が向上します。ただし、ここでのトレードオフはデータ・アクセス遅延の増加についてであることに注意してください。図 3に、メモリ割り当ての例を示します。

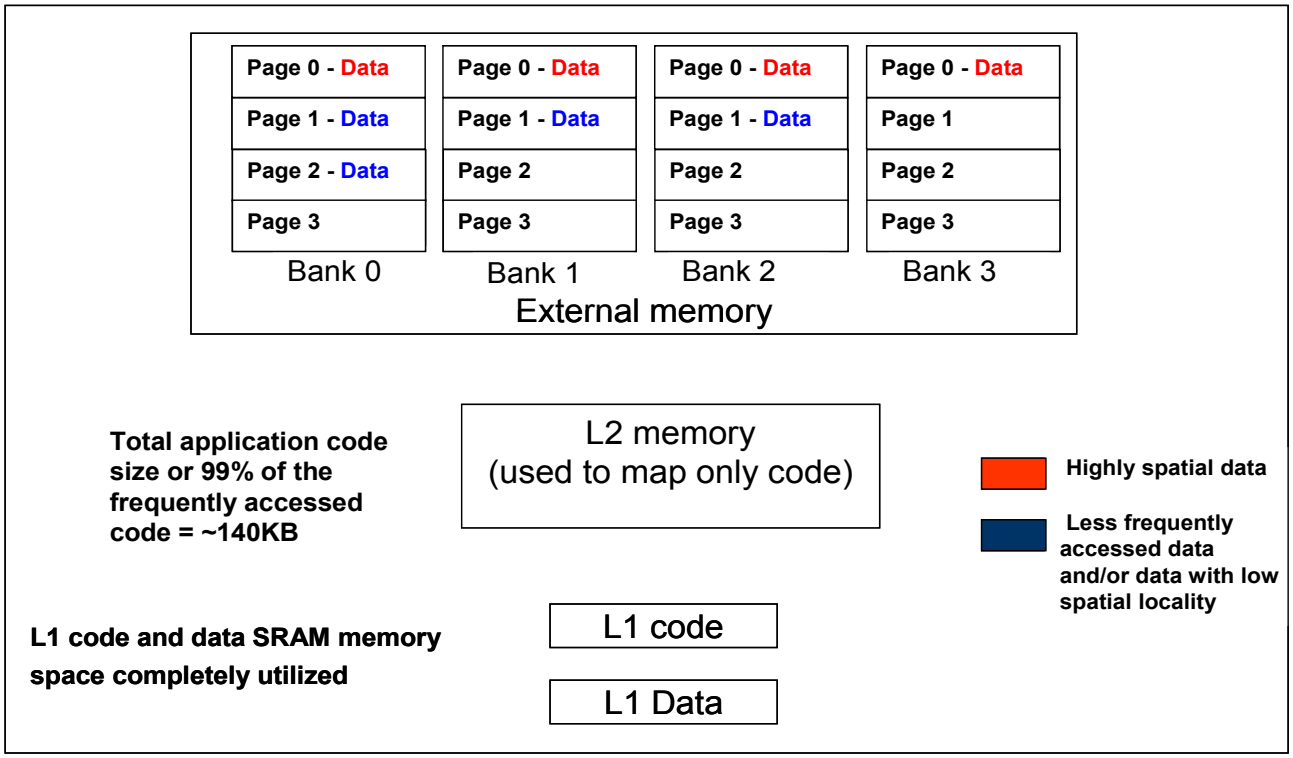

図 3.データ専用に外部メモリを使用

2. コード/データを別々の内部メモリ・バンクに割り当てるように外部メモリを分割することができます。この方法は、L1 メモリとL2 メモリから溢れたコードを局在性の高いコードが内部バンクのページ境界を跨がないように割り当て可能な場合に役立ちます。図 4にメモリ・マップの例を示します。

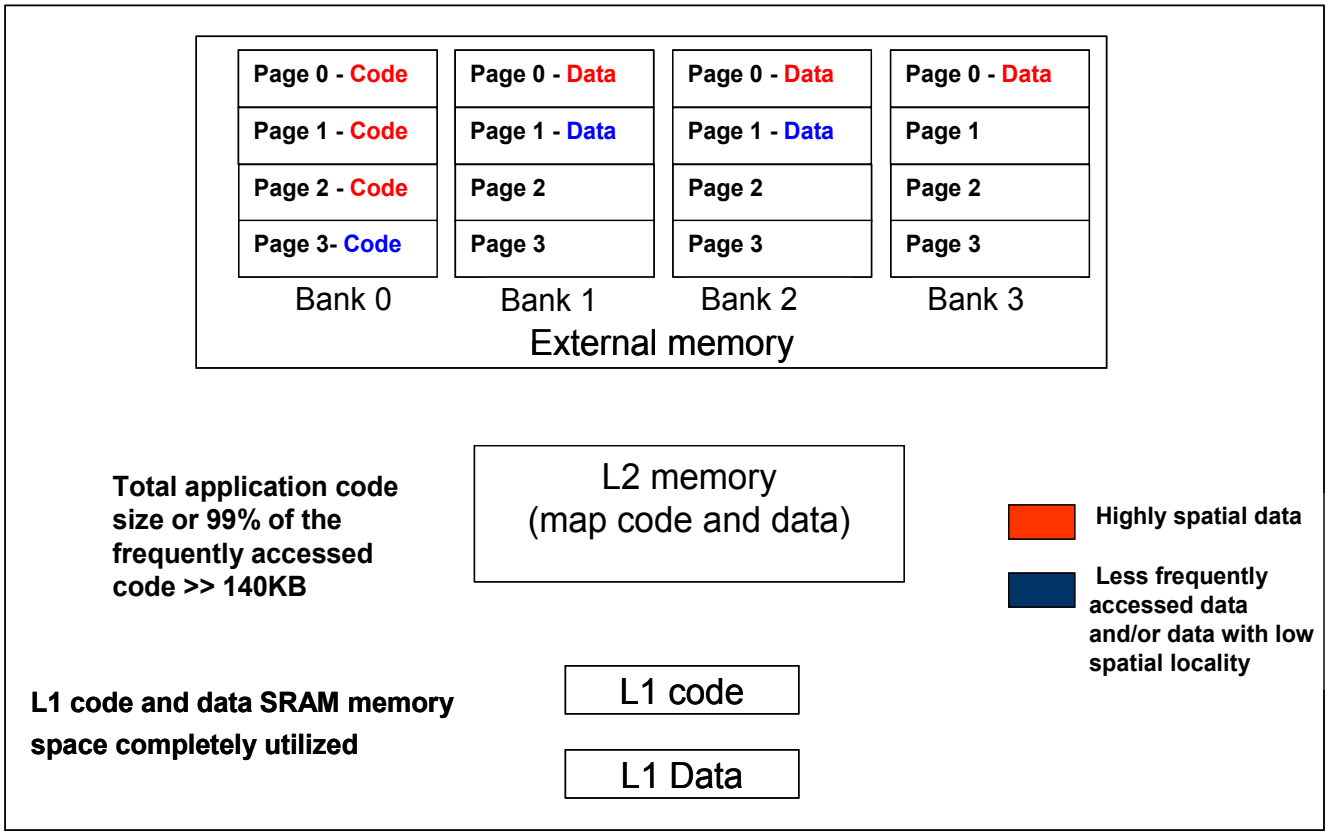

図 4.SDRAM 内部メモリ・バンクの 1 ページのみにコードを配置

3. 一般に、複数のバッファがペリフェラル・データ用に維持されています。これらのバッファは、別々の内部バンクに配置することができます。図 5にメモリ・マップの例を示します。複数のデータ・バッファを管理する他の方法については、「Video Templates for Developing Multimedia Applications on Blackfin Processors (EE-301)[7]」を参照してください。

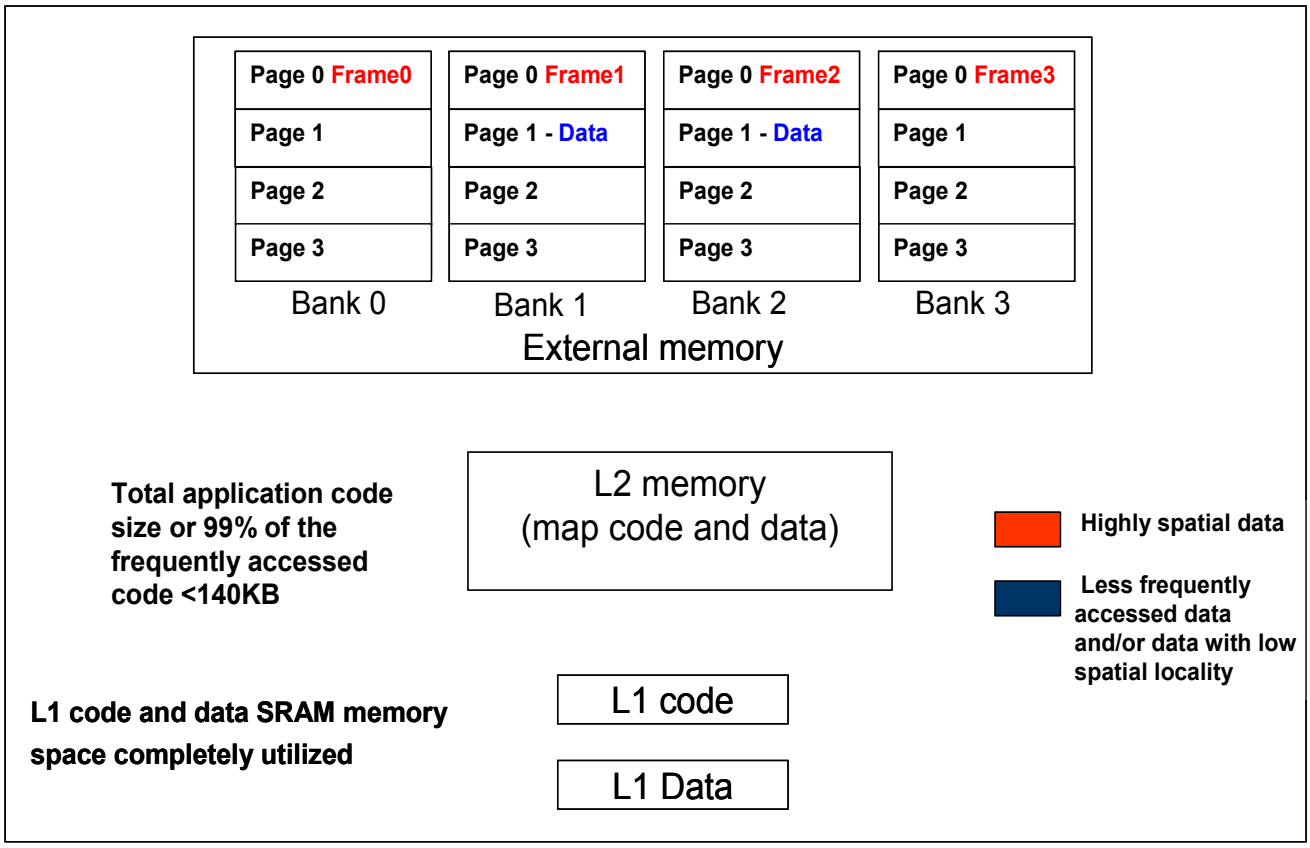

図 5.4 個の異なる SDRAM 内部バンクへのフレーム・バッファの配置

その他の推奨事項

一般に、頻繁にアクセスされるコード・オブジェクトとデータ・オブジェクトは局在しています。このため、頻繁にアクセスされるコードとデータを内部バンク内のページ・サイズの 4 倍の大きさのセクションに均等に分けて、これらを各内部バンク内の別々のページに配置することができます。

もう 1 つの可能性は、外部メモリに対するコアからの直接アクセスを回避することです。表 2に示すようにDAG読み出しアクセスではSCLKで 8 サイクル要しますが、このために、外部に割り当てた小さいデータのアクセスでも大きな帯域幅を使ってしまいます。したがって、外部メモリに割り当てたデータ・オブジェクトをアクセスするときはキャッシュまたはDMAを使うようにします。

システム・アーキテクチャ

このセクションでは、システム・バス、DMA コントローラ、ペリフェラル、外部バス調停回路などの、Blackfin プロセッサのシステム・アーキテクチャについて説明し、これらの最適な使い方のガイドラインを提供します。アーキテクチャは主に性能の点から説明するため、詳細については述べません。

システム・バス

次のシステム・バスが DMA チャンネル、ペリフェラル、コアに接続されています。

ペリフェラル・アクセス・バス(PAB)

PAB バスは、システム MMR レジスタを持つすべてのコア外のペリフェラルを接続します。PAB バス上でのレジスタ読み出しアクセスとレジスタ書き込みアクセスには、それぞれ SCLK で 3 サイクルおよび 2 サイクルの遅延があります。

DMAバス

ADSP-BF561 プロセッサの場合、次の DMA バスがペリフェラルとメモリ階層を接続します。

- DMA アクセス・バス(32 ビット DAB1 と 16 ビット DAB2)は、ペリフェラルとの間の DMA データ転送を提供します。
- DMA コア・バス(32 ビット DCB1、DCB2、DCB3、DCB4)は、コア(A と B) L1 メモリとの間の DMA データ転送、または L1 メモリと L2 メモリとの間の DMA データ転送を提供します。
- DMA 外部バス(DEB)は、外部メモリとの間の DMA データ転送を提供します。

ペリフェラルとメモリとの間の転送遅延は、最大 32 ビットのアクセスに対してSCLKで 2 サイクルです。これに対して任意のペリフェラルは、各SCLKサイクルでバスをアクセスすることができるため、使用可能な帯域幅をフルに使用することができます。異なるレベルのメモリ間の転送では、各 32 ビット・ワードに対してSCLKで 1 サイクルの遅延を実現することができます。L1 メモリとL2 メモリとの間の転送では、各CCLKサイクルで 32 ビット・ワードを転送することができます。表 4に、異なるレベルのメモリ階層間でのメモリ―メモリ間DMA転送の性能を示します。

Source	**Destination**	**Approximate SCLKs for n Words (from start of DMA to interrupt upon completion)**
32-bit SDRAM	L1 Data memory	n+14
L1 Data memory	32-bit SDRAM	n+11
32-bit ASYNC memory	L1 Data memory	xn+12 x is the number of wait states + setup/hold SCLK cycles (minimum x =2)
L1 Data memory	32-bit ASYNC memory	xn+9 x is the number of wait states + setup/hold SCLK cycles (minimum x =2)
32-bit SDRAM	32-bit SDRAM	10+(17n/7)
32-bit ASYNC memory	32-bit ASYNC memory	10+2xn x is the number of wait states + setup/hold SCLK cycles (minimum x =2)
L1 Data memory	L1 Data memory	2n+12

表4. メモリ―メモリ間転送での *DMA* バスの性能

ADSP-BF53x と ADSP-BF52x の両プロセッサの場合、次の DMA バスがペリフェラルとメモリ階層を接続します。

- DMA アクセス・バス(16 ビット DAB)は、ペリフェラルとの間の DMA データ転送を提供します。
- DMA コア・バス(16 ビット DCB)は、L1 メモリとの間の DMA データ転送、または L1 メモリと外部メモリとの間の DMA データ転送を提供します。
- DMA 外部バス(DEB)は、外部メモリとの間の DMA データ転送を提供します。

ADSP-BF53x プロセッサの性能と転送遅延は ADSP-BF561 プロセッサと同じですが、各 SCLK サイクルでの最大ワード・サイズが 16 ビットです。

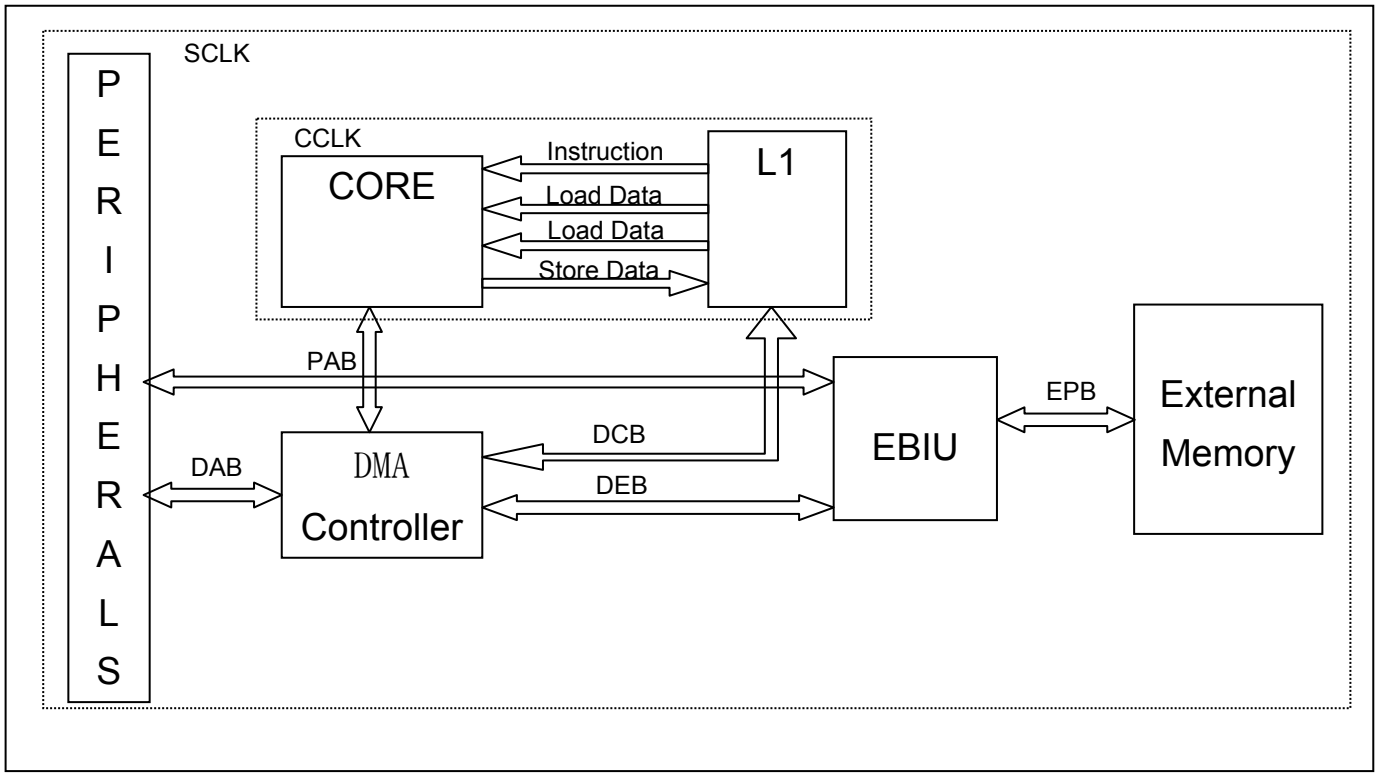

図 6.ADSP-BF53x/ADSP-BF52x Blackfin プロセッサの内部バス構造 ADSP-BF561 プロセッサの場合、コアが 2 個、DMA コントローラが 2 個、L2 メモリのブロックが 1 個になります。

外部アクセス・バス(EAB)

EAB の使用により、コアはチップ外部のメモリを直接アクセスすることができるようになります。ADSP-BF561 デュアルコア・プロセッサの場合、8、16、または 32 ビット・ワードの転送を各 SCLK サイクルで実行することができます。ADSP-BF53x プロセッサの場合、8 または 16 ビット・ワードの転送を各 SCLK サイクルで実行することができます。EAB は 133 MHz の最大周波数で動作します。

ガイドライン

ペリフェラルでの最大バス幅とパッキングの使用

各転送で最大バス幅を使用すると、システム・スループットを大幅に増やすことができます。ADSP-BF561 プロセッサの 32 ビット DMA アクセス、および ADSP-BF53x/ADSP-BF52x プロセッサの 16 ビット DMA アクセスをパッキングと組み合わせて使用すると(ペリフェラル・インターフェースで使用可能な場合)、システム・バスを他の動作に解放できるため、システムのスループットを大幅に増やすことができます。

例えば、PPI は ADSP-BF561 プロセッサでは 32 ビット・パッキングを、ADSP-BF53x/ADSP-BF52x プロセッサでは 16 ビット・パッキングを、それぞれ提供します。

DMAトラヒックの制御

Blackfinプロセッサでは、すべてのシステム・バスでトラヒック制御を提供しています。バス上のトラヒック方向が頻繁に変わると、バンクのターンアラウンド・タイムのために遅延が増えます。トラヒック・コントロール・レジスタを使用する方法は、システム・バス・トラヒックを最適化して、帯域幅使用率を向上させる最善な方法の 1 つです。各DMAバスのトラヒック期間を一方向の転送にグループ化するように指定できるため、バンク・ターンアラウンド・タイムを小さくすることができます。図 7に、DABバスで最適化されたトラヒック・パターンを示します。

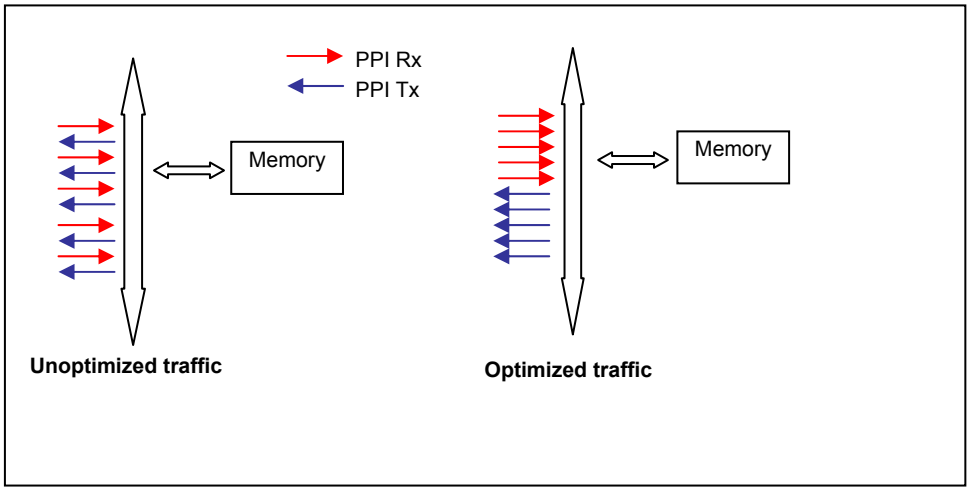

図 7.システム・バスでの DMA トラヒックの最適化

図 8にDMAトラヒック・コントロール・レジスタを示します。各バスは、一方向の最大 16 転送にグループ化するように設定することができます。また、メモリDMAチャンネルを、トラヒック・コントロール・レジスタを使ってラウンドロビン優先順位に設定することができます。

トラヒック期間の値は、システム内のペリフェラル数とバス・トラヒック量に応じてアプリケーションごとに評価することができます。経験則として、DMA コントローラ上に 4 個以上のペリフェラルが存在する場合は、3 に近いトラヒック期間の値が最適で、DMA コントローラ上に 3 個以下のペリフェラルが存在する場合には 7 に近い値が最適です。また、DEB のトラヒック期間値はシステム性能に最大の影響を与えるため、十分な評価を行う必要があることに注意してください。

また、メモリ DMA ラウンドロビン期間の値を小さくすると、コントローラが数ワードごとに DMA チャンネル間で切り替わるため、DMA 転送のスループットが低下します。一般に、大部分のアプリケーションでは、値が大きいほどリソースを均等に共用して性能をバランスさせます。

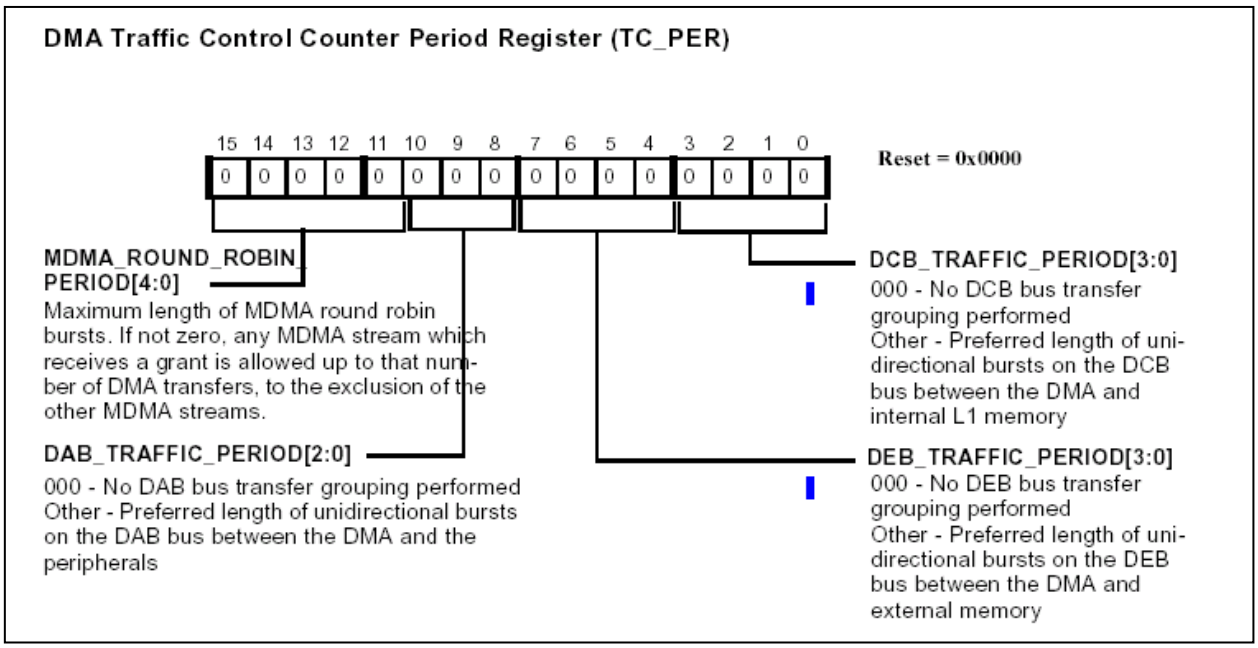

図 8.DMA トラヒック・コントロール・レジスタ

CCLK/SCLK比

バスの性能は、コアとシステム・クロックの周波数比から影響を受けます。約 2.5:1 より低い比では、同期とパイプラインの遅延により、システム・クロック・ドメインでバス使用率が低下します。例えば、2:1 のクロック比では DMA はシステム・クロック・レートの 2/3 で動作します。コアとシステム・クロックの比を高くすると、フル帯域幅を使うことができます。

DMAアーキテクチャ

DMA コントローラは、ペリフェラル・インターフェースと内蔵メモリおよび外付けメモリとの間の DMA 動作を管理します。複数のペリフェラル DMA チャンネルとメモリ DMA チャンネルがメモリ・リソースを取り合います。ペリフェラル DMA チャンネル(DAB バス)のバスと、メモリ―コア間アクセス・バス(DCB バス)を分離すると、アクセス遅延を小さくすることができます。また、各ペリフェラルは自分自身の FIFO バッファを持っており、このためにフル・パイプライン化されたペリフェラル―メモリ間 DMA 転送に比べてさらに遅延が小さくなります。

DMA 転送は、ストップ、自動バッファ、ディスクリプタなどの複数のモードで起動することができます。詳細については、プロセッサの*ハードウェア・リファレンス*・マニュアルの DMA の章を参照してください。ディスクリプタ・モードでは、DMA コントローラは DMA チャンネル・レジスタへロードする値をデータ・メモリ空間にあるディスクリプタ・オブジェクトからフェッチします。ディスクリプタまたはディスクリプタ配列は、DMA レジスタへの MMR 書き込みで発生する遅延を隠すことができます。

ADSP-BF561 プロセッサは 2 個の DMA コントローラを、ADSP-BF53x プロセッサは 1 個の DMA コントローラを、それぞれ内蔵しています。複数の DMA コントローラを持つプロセッサでは、各 DMA コントローラは別々の 2 本のバスに接続されるため、転送を DMA コントローラ間で分担できるのでスループットが向上します。

ガイドライン

1. 高いデータ・レートを持つペリフェラルに高い優先順位を与えるとスループットが向上します。DMA ペリフェラル・マップ・レジスタを使って、DMA コントローラに割り当てられたデフォルトのペリフェラル割り当てを変更してください。

2. 複数の DMA コントローラを使用できる場合は、各 DMA コントローラに分担させるとシステムの帯域幅要求を軽減できます。
3. 自動バッファ・モードまたはディスクリプタ・モードを使用すると、システム・クロック・サイクル数を節約することができます。

バス調停

EBIU は、外部メモリに対するコア・アクセスと DMA アクセスの競合を調停します。外部メモリ・アクセスの調停ポリシーを説明する前に、システム内で DMA 要求が*アージェント*と呼ばれるときはどんなタイミングかを知ることが重要です。

アージェントDMA

ペリフェラル・チャンネルが外部デバイスからデータを受信している次のケースについて考えてみます。ペリフェラル側が受信のとき(メモリ書き込み)、ペリフェラル FIFO がフルで、かつ FIFO の内容をメモリへ書き込むために DMA がバスをアクセスできないとき、バッファ・オーバーフロー状態が発生します。この状況で、*アージェント DMA 要求*が発行されます。

同様に、ペリフェラル FIFO がエンプティで、かつメモリから読み出して FIFO へ書き込むために DMA がバスのアクセスを取得できない場合に、アンダーフロー状態を防止するため、ペリフェラル側送信(メモリ読み出し)でアージェント DMA 要求が発行されます。

両ケースとも、DMA コントローラからアージェント DMA 要求が自動的に発行されるため、ユーザの介入は不要です。ADSP-BF534/BF536/BF537 プロセッサでは、1 つの DMA 要求がアージェントになると、システム内のすべての待ち状態の DMA 要求がアージェントになります。ADSP-BF531/BF532/BF533 と ADSP-BF561 プロセッサでは、1 つのアージェント要求が発生しても待ち状態の要求はアージェントになりません。これらのプロセッサですべての待ち状態の要求をアージェントにするときは、`EBIU_AMGCTL` レジスタの `CDPRIO` ビットをセットすることができます。これについては次のセクションで詳しく説明します。

調停方式

外部メモリ・アクセスでは、デフォルトでアージェント DMA 要求が最高の優先順位を持ち、この下にコア要求と DMA 要求が続きます。この優先順位方式は、外部メモリ・アクセスに対して設定可能であり、ガイドラインのセクションで説明します。

L1 メモリと L2 メモリでは、DMA 要求とコア要求の調停は固定されています。内部メモリ・アクセスでは、DMA アクセスとコア・アクセスがメモリの同じサブバンクを対象とする場合にのみ調停が必要です。同じサブバンクをアクセスすると、デフォルトで DMA がコアより高い優先権を取得します。ロック・コア・アクセスでは、ロックされたアクセスが完了するまで DMA が待たされます。

ガイドライン

外部メモリ・アクセスの場合、コア要求が DMA 要求より高い優先順位を持ちます。これにより、複数のコアが外部メモリをアクセスすると、スループットに大きな影響が生じることがあります。この影響は、すべての DMA アクセスをアージェント DMA 要求に昇格させて、すべての DMA アクセスにコア・アクセスより高い優先順位を与えることにより回避することができます。これは、`EBIU_AMGCTL` レジスタの `CDPRIO` ビットまたは `DMAPRIO` ビットをセットすることにより実施することができます。

その他の一般的なガイドライン

アプリケーションが必要とする合計帯域幅の計算は、帯域幅の面でシステムの実現性を調査する開始点として適していると考えられます。リソースが効率良く使用されていないシステムでは、使用率が合計最大帯域幅理論値の 20%以下の場合でもバッファのアンダーフローまたはオーバーフローが発生します。

バッファのアンダーフロー/オーバーフロー・エラーは、デバッグ時に見逃されることがあります。Blackfin プロセッサでサポートされているエラー割り込みを使うと、システムが帯域幅不足で動作しているか否か調べることができます。ペリフェラル・ステータス・レジスタが、オーバーフローまたはアンダーフロー・エラーを表示します。また、このエラー割り込みはデフォルトで有効にされていないことに注意してください。すべての Blackfin プロセッサでは、エラー割り込みはデフォルトで IVG7 に割り当てられています。エラーが発生しているペリフェラルを割り込みサービス・ルーチン内で調べるのは、ユーザが行う必要があります。

エラー割り込みは、アプリケーション内の他の割り込みベクタと同様に有効にすることができます。エラー処理では特に、システム内のすべてのペリフェラルのステータス・レジスタをチェックして、エラーを引き起こしている原因を探す必要があります。

システム最適化技術

これまで説明した技術は次のようにまとめることができます。

1. コードとデータのレイアウト技術
2. パッキングとフル・バス幅の使用
3. 効率良いバンク配置
4. DMA トラヒック制御を使ったバス・トラヒックの最適化
5. 高い CCLK/SCLK 比の維持
6. 複数の DMA コントローラ(使用可能な場合)によるデータ転送の分担
7. プログラマブルなバス優先順位方式の使用

このセクションではこれらの技術を評価し、各々を特定のテスト・ケースを使って定量化します。技術を順次テスト・ケースに追加して、システムのスループットを各ステップで評価します。コード・レイアウトとデータ・レイアウトは、それぞれ「*PGO Linker - A Code Layout Tool for the Blackfin Processors (EE-306)*[8]」と「*Video Templates for Developing Multimedia Applications on Blackfin Processors (EE-301)*[7]」で説明しているため、この EE ノートでの解析では省略してあります。

評価方法

システム帯域幅を評価するため、2 つのベンチマークを用意しました。1 つ目のベンチマークは、外部メモリに対してメモリ DMA 読み出しと書き込みのみを実行し、このベンチマークに対して他のシステム・バス動作が発生しないシンプルなプログラムです。2 つ目のベンチマークは少し複雑なプログラムであり、システム内で複数のメモリ DMA チャンネル、ペリフェラル DMA チャンネル、コア動作を使います。このベンチマークは前述の最適化技術の利点をデモストレーションするために使用します。この解析の後半では、トラヒック・コントロール・レジスタの設定と帯域幅使用率の CCLK/SCLK 比の設定の効果を別々に評価します。

この解析の目標は、種々の最適化技術を使うことにより、システム性能を順次向上させることができることを示すことです。効果を定量化するため、最適化技術の評価の指標として平均スループットを選択しました。平均システム・スループットは次のように測定されました。

平均システム・スループット= (外部メモリに対するデータ読み出し数または書き込み数)/sec

システム・バス動作の時間間隔は、内部コア・タイマを使って設定します。タイマは、1 sec 経過後に割り込みを発生するように設定します。ペリフェラル/MDMA チャンネルを有効にする直前にタイマを起動し、コア・タイマの ISR 内でペリフェラル/MDMA チャンネルを無効にします。転送されたデータ量は、ペリフェラルの割り込みサービス・ルーチン内でカウンタを基準として測定します。各バッファ転送に割り込みが発生し、ペリフェラル/MDMA ISR が起動されるごとにカウンタをインクリメントします。すべてのペリフェラルと MDMA チャンネルが自動バッファ・モードで動作するため、スループットの最終計算ではペリフェラル割り込み遅延を考慮する必要はありません。計算ではタイマ割り込み遅延が無視されていることに注意してください。

ADSP-BF561 プロセッサ上でベンチマークを評価しましたが、この結果から得られた推論は、特に注記がないかぎり、すべての Blackfin プロセッサに適用することができます。

システム帯域幅の解析

このセクションでは、シンプルなメモリ DMA の例と、ビデオとオーディオの入力/出力およびファイル共有機能を持つ実用的な組込型アプリケーションの 2 つのベンチマークについて説明します。

例 1—メモリDMA

このベンチマークでは、1 対のメモリDMAチャンネルを使ってL1 メモリと外部メモリとの間でバッファが転送されます。DMAバッファは 8 KBに、CCLKとSCLKはそれぞれ 600 MHzと 120 MHzに設定されています。表 5に、SDRAMアクセスのスループット解析を示します。システム内では 1 個のメモリDMAチャンネルのみが動作し、外部メモリに対してペリフェラル・アクセスまたはコア・アクセスは発生しないことに注意してください。

DMA Operation	Packet Size	Buffers Transferred Per Second	Average Throughput (MB/s)	% of Max Theoretical Throughput (of 480 MB/s)
Write access L1 to external memory	8 KB	57303	469	98%
Read access External memory to L1	8 KB	51165	419	87%

表5. 外部メモリに対する *DMA* 読み出しと書き込みアクセスのスループット

メモリDMAチャンネルは自動バッファ・モードで動作し、コア・タイマがタイムアウトすると割り込みが発生します。このベンチマークでは、システム最適化技術は何も使用していませんが、表 5に示すように、スループットは最大理論帯域幅に近い値になっています。これは、外部バスではメモリDMAチャンネル 1 のリード/ライト・アクセス以外の他の動作がないためです。さらに、アクセスは外部メモリ内の同じページのみが対象です。このために、ページ・ミスのペナルティが発生しません。外部メモリからの読み出しは、各読み出しサイクルでCAS遅延の影響を受けます。DMA読み出しと書き込みアクセスの外部メモリ遅延については 表 2を参照してください。

例 2—ファイル共用機能を持つオーディオ/ビデオアプリケーション

2 つ目のベンチマークでは、より実用的なアプリケーションの解析が可能で、ビデオ入力/出力のペリフェラル・インターフェース、オーディオ入力/出力、ネットワークまたは USB インターフェースを介するファイル共用が含まれます。このベンチマークは基本的に、設定済みのシングル・メモリ DMA ベンチマークに対するインターフェースの追加により構成されます。外部デバイスとのインターフェースでは、次のペリフェラル割り当てを使っています。

External Device	Peripherals Assigned	DMA Controller (DMAC)	Comments
Video encoder	PPI0	DMAC 1	ITU-656 format video input
Video decoder	PPI1	DMAC 1	
Audio in	SPORT0 RX	DMAC 2	
Audio out	SPORT0 TX	DMAC 2	
File write to PC (USB to ASYNC memory transfers)	MDMA1_1	DMAC 1	Write to ASYNC memory
Internal memory DMA transfers	MDMA1_0	DMAC 1	To move data from external memory to L1 memory

表6. システムにインターフェースする外部デバイスのペリフェラル/*DMA* 割り当て

最適化技術と推奨ガイドラインを評価するため、次の 4 つのアプリケーション・ケースを使用しました。

ケース 1: 図 9に示す設定を使用。これは基本ケース・モデルで、システム最適化技術は何も使用していません。

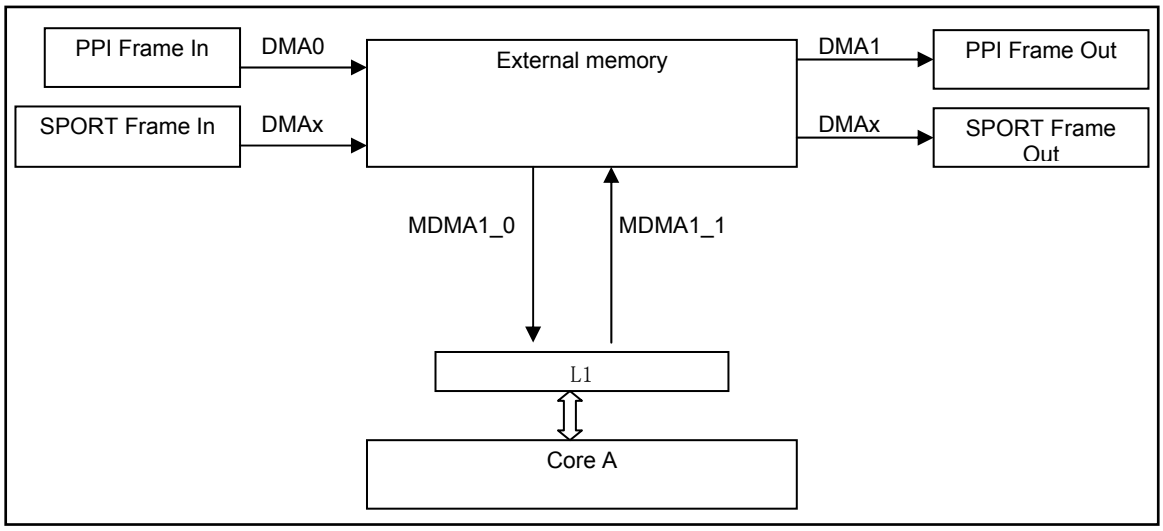

図 9.例 1 (L2 メモリは省略)

ケース 2:メモリ DMA 転送(MDMA1_0 MDMA1_1)を DMA コントローラ 2 (MDMA2_0 MDMA2_1)へ移動。

ケース 3:両 DMA コントローラをメモリ DMA 転送に使用。

ケース 4:厳しいコア・アクセス・ループを外部メモリに追加。

MDMA チャンネルは、各 SCLK サイクルで 32 ビット・ワードを外部メモリへ転送することができるため、最大バス使用率になることに注意してください。MDMA チャンネルは割り込みのない自動バッファ・モードで動作するため、ペリフェラル DMA チャンネル転送間の空き帯域幅を使います。

Peripheral	Packet Size	Packets Transferred Per Second	MB/s
PPI – In (30 MHz-NTSC video in)	1716 * 525 = 900900	16	14
PPI – Out (30 MHz - NTSC video out)	1716 * 525 = 900900	9	8
MDMA	8192	14	1
SPORT_RX/TX (4 MHz)	32	14649	0.1
TOTAL			23

表7. 基本ケース *1* のスループット計算

Optimization Technique	Scenario 1		Scenario 2 (Spreading to 2 DMA Controllers)		Scenario 3 (Using Both DMA Controllers)	
	MB/s	Improvement	MB/s	Improvement	MB/s	Improvement
Baseline	23	1x	23	1x	23	1x
Bus width and PPI packing	54	2.3x	148	6.4x	148	6.4x
Efficient bank placement	96	4.2x	348	15.2x	351	15.3x
DMA traffic control	230	10.0x	330	15.0x	342	15.5x

表8. *3* つのアプリケーション・ケースに対する種々の最適化技術によるスループットの改善

表 8に示すように、フル・バス幅を使用し、パッキングを有効にすると、性能は最初のケースでは 2.3 倍に、2 番目と 3 番目のケースでは 6.4 倍に、それぞれ改善されます。効率良いバンク配置により、さらに 8 倍(15.2～6.4 倍)の改善が得られます。2 個のDMAコントローラにデータ転送を分担させると、スループットは基本ケースのスループットの 5 倍になります。トラヒックが 1 個のDMAコントローラでスケジュールされる場合は(最初のケース)、DMAトラヒック制御により性能が改善されます。2 個のDMAコントローラを使うと、バス幅パッキングと効率良いバンク配置のみにより、帯域幅使用率が最大になるため、トラヒック制御を追加してもスループットはケース 2 と 3 ほど増加しません。

外部メモリに対して厳しいコア・アクセスを追加する 4 番目のケースも調べました(表 9)。

Optimization	Scenario 1		Scenario 4	
	MB/s	Improvement	MB/s	Improvement
Baseline	23	1x	21	1x
Bus width and PPI packing	54	2.3x	52	2.5x
Efficient bank placement	96	4.2x	54	2.6x
DMA traffic control	230	10.0x	55	2.6x
Set CDPRIO	230	10.0x	195	9.3x

表9. コアが外部メモリをアクセスするとき *CDPRIO* ビットをセットすることによるスループットの改善

表 9に示すように、厳しいコア・アクセスにより、システム・バスがロックされてしまい、DMAチャンネルにアクセスが与えられないようになります。DMAがバスに対するアクセスを取得できない場合は、有効な技術はありません。`CDPRIO`ビットをセットすると、スループットが改善されます。

トラヒック・コントロール・レジスタの評価

前述のように、DMA トラヒック制御を使うと、システム・スループットを大幅に改善できますが、トラヒック期間値を大きくすると、他のペリフェラルが長時間データを得られなくなります。例えば、アプリケーションで PPI0 と PPI1 の両インターフェースを使う場合を考えます(PPI1 を受信モードで、PPI1 を送信モードで使用)。DEB トラヒック期間を 16 に設定すると、いずれの PPI もバスのアクセスを再度取得するまで 16 転送待つ必要があります。これにより、待ち状態の PPI の FIFO でアンダーフローまたはオーバーフローが発生してしまいます。

これをデモストレーションするため、添付 ZIP ファイル[11]のテスト・ケースを使用しました。このプロジェクトでは、2 個の PPI チャンネルと DMA コントローラ 1 の 2 個の MDMA チャンネルが使用されています。このテスト・ケースでは、トラヒック制御がオフのとき、PPI1 ステータス・レジスタがアンダーフロー・エラーを表示します。トラヒック制御に 0x0777 を設定すると、バンクのターンアラウンド・タイムが小さくなるため、PPI1 のアンダーフロー・エラーが消えます。スループットをさらに上げるために、値を 0x07ff に増やすと、PPI1 で再びアンダーフロー・エラーが発生します。これは、PPI1 を長時間保持すると、その FIFO のデータが不足するのでアンダーフローが発生するためです。

前のセクションで説明したように、一般的経験則は、DMA コントローラ上のペリフェラル数が 3 個を下回る場合はトラヒック期間値 0x7 を、DMA コントローラ上のペリフェラルが 4 個以上の場合はトラヒック期間値 0x3 を、それぞれ使用することです。また、外部メモリのバンク・ターンアラウンド・タイムが L1 SRAM メモリのターンアラウンドよりペナルティが大きいため、DEB トラヒック期間は DCB または DAB のバス・トラヒック期間より大きな影響をスループット性能に与えることは興味深いことです。

トラヒック・コントロール・レジスタの上位 5 ビットがMDMAラウンドロビン期間を制御します。ラウンドロビンを使うと、高い優先順位のMDMAチャンネルが低い優先順位チャンネルを阻止しないように、MDMAチャンネル間で均等に共用できるようになりますが、ラウンドロビンを使用すると、チャンネル間での切り替えによる遅延が増えるためDMA性能が低下します。図 10に、スループット性能と、割り込みが発生しないDMAコントローラ 1 で動作する 2 個のMDMAチャンネルに対するMDMAラウンドロビン・カウントとの関係を示します。このスループットは前の実験と同じ方法で測定しました。

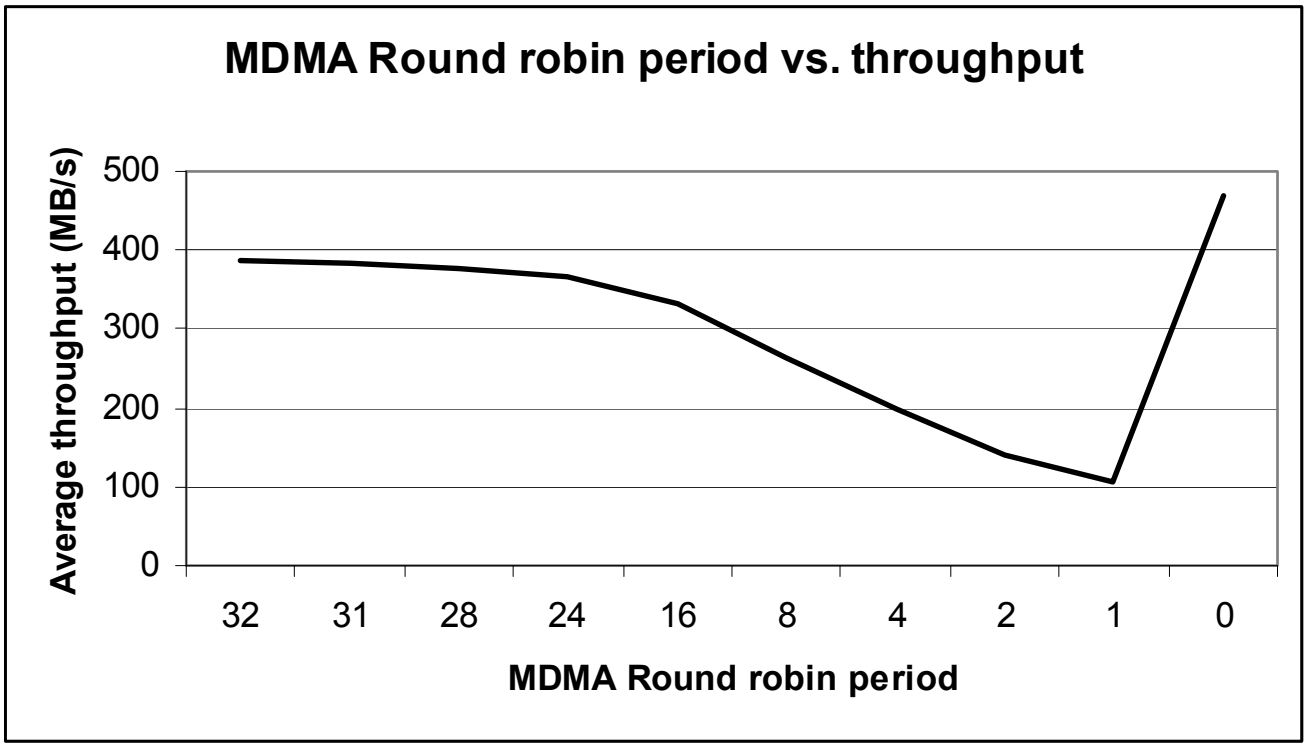

図 10. MDMA ラウンドロビン期間対スループット

図から分かるように、ラウンドロビン・カウントが減ると、システムのスループットも減少します。ラウンドロビン期間をゼロに設定すると、最大のシステム・バス使用率が得られますが、このケースでは、高い優先順位のチャンネル(MDMA1)のために、低い優先順位のチャンネル(MDMA2)は実行中のトランザクションが完了するまでバスに対するアクセスが得られなくなります。

CCLK/SCLK比の評価

低いコア・クロックでは、同期とコア・バスの遅延増加により、システム・バス使用率が低下するため、CCLKとSCLKとの比もシステム・スループットに影響を与えます。表 10に、CCLK/SCLK比のシステム・スループットに対する影響を示します。使用したテスト・ケースは、1 個のMDMAチャンネルでL1 メモリと外部メモリとの間で 8KBのDMAバッファを転送する場合と同じで、外部メモリに対する読み出しアクセスおよび書き込みアクセスに対して種々のCCLK/SCLK比を使って実験を繰り返しました。

DMA Access	CCLK/SCLK Ratio	Transfers Per Second	Average Throughput (Transfers/sec * buffer size)	% of Theoretical Throughput (of 480 MB/s)
Write	5	57303	469	97.71
	4	46265	379	78.96
	3	46265	379	78.96
	2	33301	272	56.67
	1	23378	191	39.79
Read	5	51165	419	87.29
	4	51165	419	87.29
	3	51165	419	87.29
	2	38757	317	66.04
	1	29196	239	49.79

表10. *CCLK/SCLK* 比のシステム・スループットに対する影響

表から分かるように、2:1 より小さい CCLK/SCLK 比では性能が大幅に低下します。外部メモリに対する読み出しアクセスでは CAS 遅延が加わるためにコア・バス遅延が隠れてしまうので、読み出しへの影響は小さくなります。バス使用率の低下は DCB 遅延によるものですが、システム・バス上のペリフェラル動作が増えて DCB がボトルネックにならないと、見えなくなってしまいます。

結論

Blackfin プロセッサは、EBIU の帯域幅のフル利用に役立つ幾つかの最適化技術を提供しています。このアプリケーション・ノートで説明したメモリとシステムの最適化技術は、効率良いコード/データ・レイアウトとシステム性能の最適化に役立ちます。

参考

[1] *ADSP-BF533 Blackfin Processor Hardware Reference*. Rev 3.2, July 2006. Analog Devices, Inc.

[2] *ADSP-BF561 Blackfin Processor Hardware Reference*. Rev 1.0, July 2005. Analog Devices, Inc.

[3] *ADSP-BF537 Blackfin Processor Hardware Reference*. Rev 2.0, December 2005. Analog Devices, Inc.

[4] *Embedded Media Processing*. David Katz and Rick Gentile. Newnes Publishers., Burlington, MA, USA, 2005.

[5] *Video Framework Considerations for Image Processing on Blackfin Processors (EE-276)*. Rev 1, September 2005. Analog Devices Inc.

[6] *VisualDSP++ 4.5 Device Drivers and System Services Manual for Blackfin Processors*. Rev 2.0, March 2006. Analog Devices, Inc.

[7] *Video Templates for Developing Multimedia Applications on Blackfin Processors (EE-301)*. Rev 1, September 2006. Analog Devices Inc.

[8] *PGO Linker - A Code Layout Tool for the Blackfin Processors (EE-306)*. Rev 1, December 2006. Analog Devices Inc.

[9] *Using Cache Memory on Blackfin Processors (EE-271*). Rev 1, June 2005. Analog Devices Inc.

[10] *The Best way to move multimedia data*. David Katz and Rick Gentile. http://www.embedded.com/showArticle.jhtml?articleID=16700107. December 2003. Embedded.com.

[11] Associated ZIP File. Rev 1, May 2007. Analog Devices, Inc.

ドキュメント改訂履歴

Revision	Description
Rev 1 – July 10, 2007 *by Kaushal Sanghai*	Initial Release