TEXAS INSTRUMENTS

参 考 資 料

TMS320C64x DSP
2 レベル 内部メモリ

リファレンス・ガイド

2005

DSP Products

SPRU610 翻訳版

JAJU029 WAS SPRUE50

TMS320C64x DSP
2 レベル 内部メモリ
リファレンス・ガイド

TEXAS INSTRUMENTS

ご注意

日本テキサス・インスツルメンツ株式会社（以下TIJといいます）及びTexas Instruments Incorporated（TIJの親会社、以下TIJおよびTexas Instruments Incorporatedを総称してTIといいます）は、その製品及びサービスを任意に修正し、改善、改良、その他の変更をし、もしくは製品の製造中止またはサービスの提供を中止する権利を留保します。従いまして、お客様は、発注される前に、関連する最新の情報を取得して頂き、その情報が現在有効かつ完全なものであるかどうかご確認下さい。全ての製品は、お客様とTIとの間に取引契約が締結されている場合は、当該契約条件に基づき、また当該取引契約が締結されていない場合は、ご注文の受諾の際に提示されるTIの標準契約約款に従って販売されます。

TIは、そのハードウェア製品が、TIの標準保証条件に従い販売時の仕様に対応した性能を有していること、またはお客様とTIとの間で合意された保証条件に従い合意された仕様に対応した性能を有していることを保証します。検査およびその他の品質管理技法は、TIが当該保証を支援するのに必要とみなす範囲で行なわれております。各デバイスの全てのパラメーターに関する固有の検査は、政府がそれ等の実行を義務づけている場合を除き、必ずしも行なわれておりません。

TIは、製品のアプリケーションに関する支援もしくはお客様の製品の設計について責任を負うことはありません。TI製部品を使用しているお客様の製品及びそのアプリケーションについての責任はお客様にあります。TI製部品を使用したお客様の製品及びアプリケーションについて想定されうる危険を最小のものとするため、適切な設計上および操作上の安全対策は、必ずお客様にてお取り下さい。

TIは、TIの製品もしくはサービスが使用されている組み合せ、機械装置、もしくは方法に関連しているTIの特許権、著作権、回路配置利用権、その他のTIの知的財産権に基づいて何らかのライセンスを許諾するということは明示的にも黙示的にも保証も表明もしておりません。TIが第三者の製品もしくはサービスについて情報を提供することは、TIが当該製品もしくはサービスを使用することについてライセンスを与えるとか、保証もしくは是認するということを意味しません。そのような情報を使用するには第三者の特許その他の知的財産権に基づき当該第三者からライセンスを得なければならない場合もあり、またTIの特許その他の知的財産権に基づきTIからライセンスを得て頂かなければならない場合もあります。

TIのデータ・ブックもしくはデータ・シートの中にある情報を複製することは、その情報に一切の変更を加えること無く、且つその情報と結び付られた全ての保証、条件、制限及び通知と共に複製がなされる限りにおいて許されるものとします。当該情報に変更を加えて複製することは不公正で誤認を生じさせる行為です。TIは、そのような変更された情報や複製については何の義務も責任も負いません。

TIの製品もしくはサービスについてTIにより示された数値、特性、条件その他のパラメーターと異なる、あるいは、それを超えてなされた説明で当該TI製品もしくはサービスを再販売することは、当該TI製品もしくはサービスに対する全ての明示的保証、及び何らかの黙示的保証を無効にし、且つ不公正で誤認を生じさせる行為です。TIは、そのような説明については何の義務も責任もありません。

なお、日本テキサス・インスツルメンツ株式会社半導体集積回路製品販売用標準契約約款もご覧下さい。
http://www.tij.co.jp/jsc/docs/stdterms.htm

弊社半導体製品の取り扱い・保管について

半導体製品は、取り扱い、保管・輸送環境、基板実装条件によっては、お客様での実装前後に破壊/劣化、または故障を起こすことがあります。
弊社半導体製品のお取り扱い、ご使用にあたっては下記の点を遵守して下さい。

1. **静電気**
   - **素手で半導体製品単体を触らないこと。どうしても触る必要がある場合は、リストストラップ等で人体からアースをとり、導電性手袋等をして取り扱うこと。**
   - **弊社出荷梱包単位（外装から取り出された内装及び個装）又は製品単品で取り扱いを行う場合は、接地された導電性のテーブル上で（導電性マットにアースをとったもの等）、アースをした作業者が行うこと。また、コンテナ等も、導電性のものを使うこと。**
   - **マウンタやはんだ付け設備等、半導体の実装に関わる全ての装置類は、静電気の帯電を防止する措置を施すこと。**
   - **前記のリストストラップ・導電性手袋・テーブル表面及び実装装置類の接地等の静電気帯電防止措置は、常に管理されその機能が確認されていること。**
2. **温・湿度環境**
   - **温度：0～40℃、相対湿度：40～85%で保管・輸送及び取り扱いを行うこと。（但し、結露しないこと。）**
   - **直射日光があたる状態で保管・輸送しないこと。**
3. **防湿梱包**
   - **防湿梱包品は、開封後は個別推奨保管環境及び期間に従い基板実装すること。**
4. **機械的衝撃**
   - **梱包品（外装、内装、個装）及び製品単品を落下させたり、衝撃を与えないこと。**
5. **熱衝撃**
   - **はんだ付け時は、最低限260℃以上の高温状態に、10秒以上さらさないこと。（個別推奨条件がある時はそれに従うこと。）**
6. **汚染**
   - **はんだ付け性を損なう、又はアルミ配線腐食の原因となるような汚染物質（硫黄、塩素等ハロゲン）のある環境で保管・輸送しないこと。**
   - **はんだ付け後は十分にフラックスの洗浄を行うこと。（不純物含有率が一定以下に保証された無洗浄タイプのフラックスは除く。）**

以上

2001.11

まえがき

最初にお読みください

このマニュアルについて

TMS320C6000™ DSP ファミリーの TMS320C64x™ デジタル・シグナル・プロセッサ（DSP）は、プログラムとデータ用に 2 レベルのメモリ・アーキテクチャを備えています。1 次プログラム・キャッシュは L1P、1 次データ・キャッシュは L1D と呼ばれます。プログラムおよびデータ・メモリの両方とも、L2 と呼ばれる 2 次メモリを共用します。L2 はさまざまな容量のキャッシュおよび SRAM とする構成が可能です。本書では C64x™ 2 レベル内部メモリについて説明します。『TMS320C64x DSP Two-Level Internal Memory Reference Guide』（文献番号 SPRU610B）を翻訳しています。

表記規則

本書では、次の表記規則を使用しています。

- 16 進数は末尾に h を付けて表されています。たとえば、16 進数の 40（10 進数 64）は、40h と表されています。
- 本書では、レジスタは図で表され、表形式で説明されます。
  - レジスタの図は、複数のフィールドで構成される長方形で示されます。各フィールドには、ビット名が付けられています。フィールドの始まりと終わりを示すビットがその上に、またリード / ライト属性がその下に書かれています。凡例は、その属性を表すために使用される表記を示しています。
  - レジスタの図に示されている予約済みビットは、将来的なデバイスの拡張を考慮しているビットを表しています。

Texas Instruments 社からの関連文献

C6000™ デバイスおよびそのサポート・ツールを解説した関連文献は次のとおりです。関連文献は、www.ti.com から入手可能です。www.ti.com にアクセスして、検索ボックスに文献番号を入力してください。

TMS320C6000 CPU and Instruction Set Reference Guide（文献番号 SPRU189）では、TMS320C6000™ デジタル・シグナル・プロセッサの CPU アーキテクチャ、命令セット、パイプライン、および割り込みについて説明しています。

TMS320C6000 DSP Peripherals Overview Reference Guide（文献番号 SPRU190）では、TMS320C6000™ DSP 上で使用可能なペリフェラルについて説明しています。

TMS320C64x Technical Overview（文献番号 SPRU395）では、TMS320C64x™ DSP の概要について説明しています。また、TMS320C64x VelociTI™ により強化されるアプリケーション分野についても説明しています。

TSM320C6000 DSP Cache User's Guide（文献番号 SPRU656）では、メモリ・キャッシュの基礎について解説し、TMS320C6000™ DSP ファミリーのデジタル・シグナル・プロセッサ（DSP）において、2 レベルのキャッシュ・ベースのメモリ・アーキテクチャを効果的に活用する方法について説明します。外部メモリとのコヒーレンスを保つ方法、メモリ・レイテンシを軽減させる DMA の使用方法、およびコードを最適化し、キャッシュ効率を向上させる方法を示します。

TMS320C6000 Programmer's Guide（文献番号 SPRU198）では、TMS320C6000™ DSP 用に C およびアセンブラ・コードを最適化する方法について説明し、また、アプリケーション・プログラム例を示しています。

TMS320C6000 Code Composer Studio Tutorial（文献番号 SPRU301）では、Code Composer Studio™ 統合開発環境とソフトウェア・ツールの概要について説明しています。

Code Composer Studio Application Programming Interface Reference Guide（文献番号 SPRU321）では、Code Composer Studio™ アプリケーション・プログラミング・インターフェイス（API）について説明しています。この API を使用して、Code Composer 用のカスタム・プラグインを開発することができます。

TMS320C6x Peripheral Support Library Programmer's Reference（文献番号 SPRU273）では、TMS320C6000™ のペリフェラル・サポート・ライブラリの関数とマクロの内容について説明しています。ヘッダ・ファイル毎に、またアルファベット順に、関数とマクロを示しています。それぞれを詳しく説明するとともに、その使用方法を示すコード例を記述しています。

TMS320C6000 Chip Support Library API Reference Guide（文献番号 SPRU401）では、オンチップ・ペリフェラルの設定と制御のために使用するアプリケーション・プログラミング・インターフェイス（API）のセットについて説明しています。

商標

Code Composer Studio、C6000、C62x、C64x、C67x、TMS320C6000、TMS320C62x、TMS320C64x、TMS320C67x、および VelociTI は、Texas Instruments の商標です。

目次

図

表

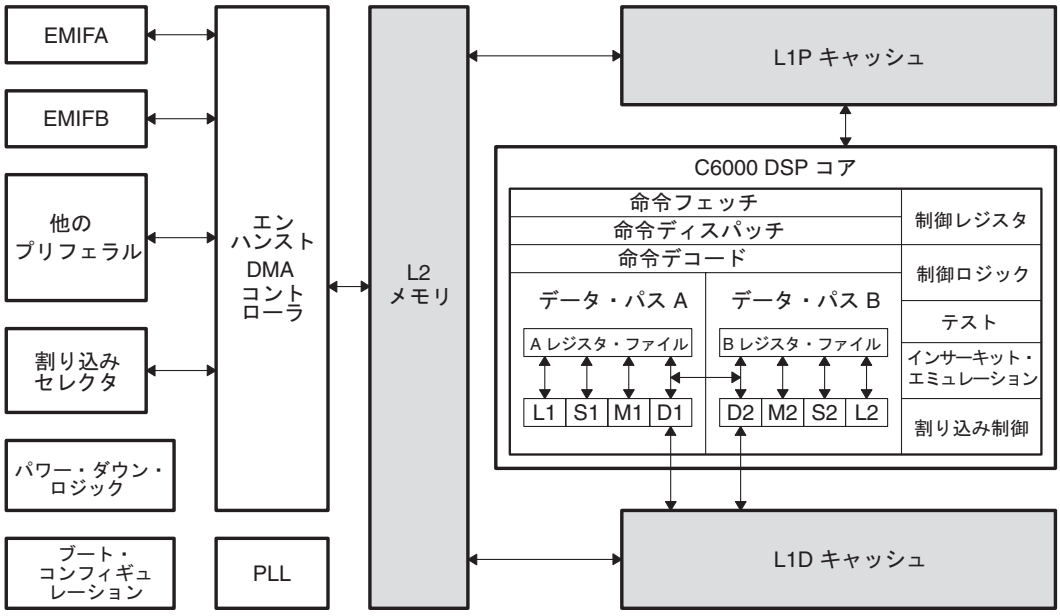
TMS320C64x 2 レベル内部メモリ

TMS320C6000™ DSP ファミリーの TMS320C621x、TMS320C671x、および TMS320C64x™ デジタル・シグナル・プロセッサ（DSP）は、プログラムとデータ用に 2 レベルのメモリ・アーキテクチャを備えています。1 次プログラム・キャッシュは L1P、そして 1 次データ・キャッシュは L1D と呼ばれます。プログラムおよびデータ・メモリは両方とも、L2 と呼ばれる 2 次メモリを共用します。L2 はさまざまな容量のキャッシュおよび SRAM とする構成が可能です。本書では C64x™ 2 レベル内部メモリについて説明します。C621x/C671x 2 レベル内部メモリの説明については、『TMS320C621x/C671x DSP Two-Level Internal Memory Reference Guide』（SPRU609）を参照してください。

1 メモリ階層の概要

図 1 は C64x DSP のブロック図を示します。表 1 は C621x/C671x と C64x の内部メモリの違いについてまとめたものです。図 2 は C6000 DSP™ の CPU、内部メモリ、およびエンハンスト DMA（EDMA）間のバス接続を図解しています。

図 1. TMS320C64x DSP のブロック図



注：EMIFB は特定の C64x デバイスでのみ使用可能です。使用可能なペリフェラル・セットについては、各デバイスのデータシートを参照してください。

表 1. TMS320C621x/C671x/C64x の内部メモリ比較

	TMS320C621x/C671xDSP	TMS320C64x DSP
内部メモリ構成	2 レベル	
L1P サイズ	4 K バイト	16 K バイト
L1P 構成	ダイレクト・マップ	
L1P CPU アクセス・タイム	1 サイクル	
L1P ライン・サイズ	64 バイト	32 バイト
L1P リード・ミス・アクション	L1P に 1 ライン・アロケート	
L1P リード・ヒット・アクション	L1P からのデータ・リード	
L1P ライト・ミス・アクション	L1P ライトはサポートされていない	
L1P ライト・ヒット・アクション	L1P ライトはサポートされていない	
L1P → L2 リクエスト・サイズ	2 フェッチ / L1P ライン	1 フェッチ / L1P ライン
L1P プロトコル	リード・アロケート	リード・アロケート パイプライン化ミス
L1P メモリ	シングル・サイクル RAM	
L1P → L2 シングル・ リクエスト・ストール	L2 ヒットで 5 サイクル	L2 ヒットで 8 サイクル
L1P → L2 パイプライン化ミスの 最少サイクル	パイプライン化ミスは サポートされていない	1 サイクル
L1D サイズ	4 K バイト	16 K バイト
L1D 構成	2 ウェイ・セット・アソシアティブ	
L1D CPU アクセスタイム	1 サイクル	
L1D ライン・サイズ	32 バイト	64 バイト
L1D 置換えストラテジー	2 ウェイ LRU	
L1D バンキング	64 ビット幅のデュアル・ポート RAM	8 × 32 ビット・バンク
L1D リード・ミス・アクション	L1D に 1 ライン・アロケート	
L1D リード・ヒット・アクション	L1D からのデータ・リード	
L1D ライト・ミス・アクション	L1D にアロケートなし、データは L2 に送られる	
L1D ライト・ヒット・アクション	L1D のデータ更新、ラインはダーティとマークされる	
L1D プロトコル	リード・アロケート	リード・アロケート パイプライン化ミス
L1D → L2 リクエスト・サイズ	2 フェッチ / L1D ライン	

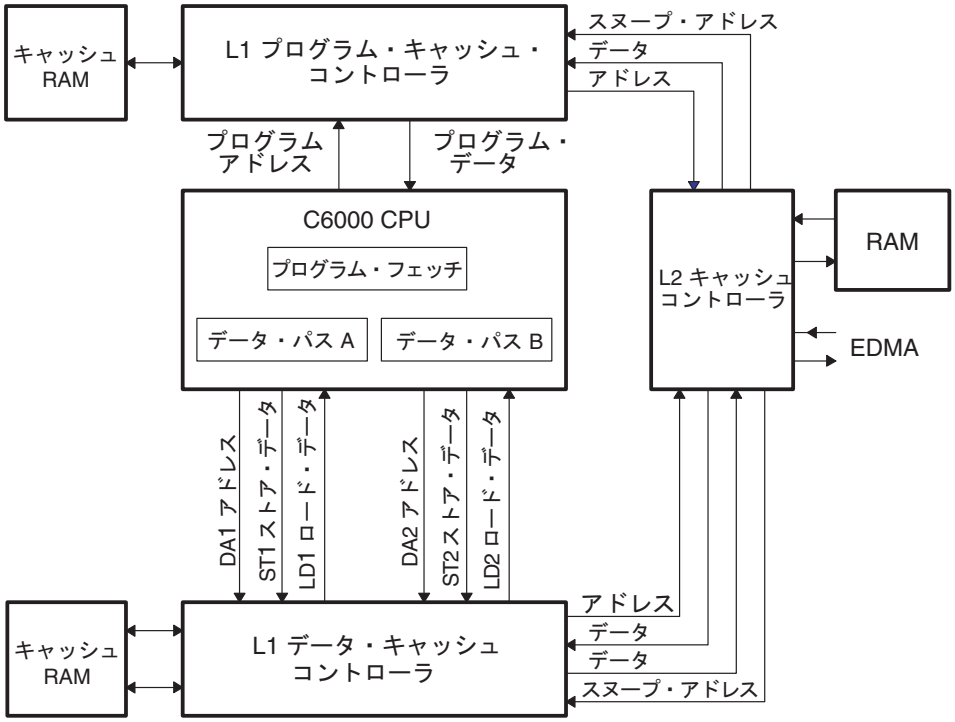
† C64x デバイスの一部は 256 K キャッシュ・モードをサポートしていません。各デバイスのデータシートを参照してください。

表 1. TMS320C621x/C671x/C64x の内部メモリ比較（続き）

	TMS320C621x/C671xDSP	TMS320C64x DSP
L1D → L2 シングル・リクエスト・ストール	L2 ヒットで 4 サイクル	6 サイクル / L2 SRAM ヒット 8 サイクル / L2 キャッシュ・ヒット
L1P → L2 パイプライン化ミスの最少サイクル	パイプライン化ミスはサポートされていない	2 サイクル
L2 合計サイズ	型番によって異なる。各デバイスのデータシートを参照してください。	
L2 SRAM サイズ	型番によって異なる。各デバイスのデータシートを参照してください。	
L2 キャッシュ・サイズ †	0/16/32/48/64 K バイト	0/32/64/128/256 K バイト
L2 構成	1/2/3/4 ウェイ・セット・アソシアティブ	4 ウェイ・セット・アソシアティブ・キャッシュ
L2 ライン・サイズ	128 バイト	
L2 置換えストラテジー	1/2/3/4 ウェイ LRU	4 ウェイ LRU
L2 バンキング	4 × 64 ビット・バンク	8 × 64 ビット・バンク
L2-L1P プロトコル	コヒーレンシ・インバリデート	
L2-L1D プロトコル	コヒーレンシ・スヌープ・インバリデート	コヒーレンシ・スヌープとスヌープ・インバリデート
L2 プロトコル	リード・ライト・アロケート	
L2 リード・ミス・アクション	データは EDMA 経由で、L2 に新しくアロケートされたラインにリードされる。リクエストされたデータは、L1 に渡される	
L2 リード・ヒット・アクション	L2 からのデータ・リード	
L2 ライト・ミス・アクション	データは EDMA 経由で、L2 に新しくアロケートされたラインにリードされる。ライト・データはその後、新しくアロケートされたラインにライトされる。	
L2 ライト・ヒット・アクション	データはヒットした L2 ロケーションにライトされる	
L2 → L1P リード・パス幅	256 ビット	
L2 → L1D リード・パス幅	128 ビット	256 ビット
L1D → L2 ライト・パス幅	32 ビット	64 ビット
L1D → L2 ビクティム・パス幅	128 ビット	256 ビット
L2 → EDMA リード・パス幅	64 ビット	
L2 → EDMA ライト パス幅	64 ビット	

† C64x デバイスの一部は 256 K キャッシュ・モードをサポートしていません。各デバイスのデータシートを参照してください。

図 2.　TMS320C64x 2 レベル内部メモリ・ブロック図



2 キャッシュの用語と定義

表 2 では、本書中で使用されている、C64x 2 レベル・メモリ階層の動作に関連する用語をリストしています。

表 2. 用語と定義

用語	定義
DMA	ダイレクト・メモリ・アクセス。典型的に、DMA はあるアドレス範囲からもう一方のアドレス範囲にメモリ・ブロックをコピーします。あるいはペリフェラルとメモリ間でデータを転送します。C64x DSP では、DMA 転送はエンハンスト DMA（EDMA）エンジンが行います。DMA 転送は、プログラム実行と並行して起こります。キャッシュ・コヒーレンスの観点から、EDMA アクセスは並列プロセッサによるアクセスと見なすことができます。
LRU	最低使用頻度。LRU 置換えポリシーの説明については、最低使用頻度アロケーションを参照してください。単体で使用されるとき一般に LRU は、セット の中で最低使用頻度のラインを識別するために、キャッシュが保持しているステータス情報として参照されます。たとえば、「あるキャッシュ・ラインをアクセスすると、そのラインの LRU を更新します。」というフレーズを考えてみてください。
アソシエイティビティ	各セット内のライン・フレーム数。この数値は、キャッシュ内のウェイ数ともいいます。
アロケーション	新しくキャッシュされるデータを格納するためのロケーションを検索するプロセス。このプロセスには、新しいデータ用の空きを作るために、現在キャッシュされているデータを追い出すことを含んでいます。
インバリデート	あるキャッシュ内の有効なキャッシュ・ラインをインバリデートとマークするプロセス。このアクションは単に該当するキャッシュ・ラインの内容を破棄するだけで、いかなる更新データもライトバックしません。ライトバックと組み合わせると、所定のレベルのメモリからキャッシュされたデータを完全に除去する一方に、そのデータを保持している次の下位レベルのメモリを効果的に更新します。ライトバックと組み合わされるインバリデートは、ライトバック・インバリデートとして参照され、一般にキャッシュ間のコヒーレンスを保持するために使用されます。
ウェイ	セット・アソシアティブ・キャッシュでは、キャッシュ内の各セットは複数のライン・フレームで構成されています。各セットのライン・フレーム数はキャッシュのウェイ数として参照されます。キャッシュ内のすべてのセットを通じて対応するライン・フレームの集合を、そのキャッシュのウェイと呼びます。たとえば、4 ウェイ・セット・アソシアティブ・キャッシュは 4 個のウェイをもっていて、キャッシュ内の各セットは関連する 4 個のライン・フレーム（4 ウェイのそれぞれと関連する）をもっています。結果として、メモリ・マップ内のどのキャッシュ可能なアドレスも、4 ウェイ・アソシアティブ・キャッシュにマップできる 4 つの可能なロケーションがあります。
追い出し（eviction）	新しくキャッシュされるデータ用に空きを作るために、キャッシュからラインを削除するプロセス。追い出しはまた、ユーザー制御により、キャッシュへアドレスまたはアドレス範囲にライトバック・インバリデートをリクエストすることにより発生させることもできます。追い出されたラインは、ビクティムとして参照されます。ビクティム・ラインがダーティな（すなわち、更新されたデータを含んでいる）場合は、コヒーレンシを保つためにそのデータを次のレベルのメモリに書き出します。
下位レベルのメモリ	階層的メモリ・システムでは、CPU からより遠くにあるメモリが下位レベルのメモリです。C64x システムでは、L2 より下のシステム・メモリ、およびすべてのメモリ・マップド・ペリフェラルが、階層における最下位レベルになります。

表 2.　用語と定義（続き）

用語	定義
競合性ミス	キャッシュ・ミスの一種で、容量上の制約ではなく、キャッシュのアソシエイティビティの制約により発生します。フル・アソシエイティブ・キャッシュは、新しくキャッシュされるラインを、キャッシュ内のどこにでもアロケートすることができます。ほとんどのキャッシュには、アソシエイティビティの制約があります（セット・アソシアティブ・キャッシュを参照してください）。そのため、データを置く場所に制限が加わり、柔軟性のあるキャッシュでは起こりえない、キャッシュ・ミスが生じます。
クリーン	有効で、かつ上位レベルのメモリまたは CPU によってライトされていないキャッシュ・ライン。有効なキャッシュ・ラインの逆の状態は、ダーティです。
コヒーレンス	簡略的には、あるデータ項目に対するいかなるリードでも、最後にライトされたデータの値を読み戻す場合は、メモリ・システムにはコヒーレンシがあります。これには、CPU および EDMA によるアクセスが含まれます。キャッシュ・コヒーレンスに関する詳細な説明は、8.1 節を参照してください。
コンパルソリー・ミス	初期参照ミスと言われることもあります。コンパルソリー・ミスとは、キャッシュ・ミスの一種で、データが以前にキャッシュにアロケートされる機会がなかったために必ず起こるキャッシュ・ミスです。通常は、特定のデータに対するコンパルソリー・ミスは、そのデータへの初回のアクセスで発生します。しかし、たとえそのデータに対する初回の参照でない場合でも、コンパルソリーと考えられるいくつかのケースがあります。そのようなケースには、ライト・アロケートされていないキャッシュの同一ロケーションに対して繰り返されるライト・ミス、およびキャッシュ不可能なロケーションに対するキャッシュ・ミスが含まれます。容量性ミスおよび競合性ミスと比較してください。
最低使用頻度（LRU）アロケーション	セットアソシアティブ・キャッシュおよびフルアソシアティブ・キャッシュでは、キャッシュ内でスペースをアロケートするとき、最低使用頻度アロケーションはセット内のライン・フレームの選択に使用される方法として参照されます。セット内のすべてのライン・フレームがアドレスにマップされ有効なデータを含んでいるとき、リードまたはライトを最も長い時間実行していない（前の実行から最も長い時間が経過した）セットのライン・フレームが、新しくキャッシュされるデータを保持するために選択されます。そして、新しいデータ用の空きを作るために、選択されたライン・フレームが追い出されます。
実行パケット	単一サイクル中に並行して実行を開始する命令から構成されるブロック。各実行パケットには、1 個から 8 個の命令が含まれています。
上位レベルのメモリ	階層的メモリ・システムでは、CPU により近いメモリが上位レベルのメモリです。メモリ階層において、最上位レベルは通常 1 次キャッシュです。このレベルのメモリは、CPU の直下にあります。上位レベルのメモリは、通常、下位レベルのメモリからのデータに対するキャッシュとして作動します。
初期参照ミス	あるデータに対する初回の参照で発生するキャッシュ・ミス。初期参照ミスは、コンパルソリー・ミス形式の一種です。
スヌープ	あるアドレスのデータを上位レベルのメモリがもっているかどうかを判別するために、下位レベルのメモリが上位レベルのメモリに問い合わせる手法。スヌープの第 1 の目的は、下位レベルのメモリが上位レベルのメモリに更新をリクエストすることによりコヒーレンシを保つことです。スヌープ動作は、ライトバック、またはより一般的にライトバック・インバリデートをトリガします。ライトバック・インバリデートをトリガするスヌープはスヌープ・インバリデートと呼ばれることがあります。

表 2.　用語と定義（続き）

用語	定義
スラッシュ	あるアクセス・パターンがキャッシュのパフォーマンスを著しく低下させる原因であるとき、そのアルゴリズムはキャッシュをスラッシュすると言われます。スラッシュは複数の理由により発生します。その 1 つは、短時間にアルゴリズムがデータ、あるいはプログラム・コードに再利用が非常に少ないか全くなしで、過多なまでにアクセスしていることです。すなわち、そのワーキング・セットが大き過ぎて、アルゴリズムが多数の容量性ミスを引き起こしています。ほかにも、アルゴリズムがアドレスが異なるが、すべてキャッシュ内の同じセットにマップされる小グループを、繰り返しアクセスしていると、大量の競合性ミスの原因になります。
セット	キャッシュ内である単一のアドレスが潜在的に存在する可能性のあるライン・フレームの集合。ダイレクトマップ・キャッシュはセットあたり 1 ライン・フレームで構成され、また N ウェイ・セット・アソシアティブ・キャッシュはセットあたり N 個のライン・フレームで構成されています。フルアソシアティブ・キャッシュは、キャッシュ内のすべてのライン・フレームで構成される 1 つのセットのみをもっています。
セット・アソシアティブ・キャッシュ	セット・アソシアティブ・キャッシュは、各下位レベルのメモリのロケーションを保持することが可能な、複数のライン・フレームで構成されています。新しいデータ・ラインに空きをアロケートするとき、その選択はキャッシュに対するアロケーション・ポリシーに基づいて行われます。C64x デバイスではセット・アソシアティブ・キャッシュに、最低使用頻度アロケーション・ポリシーを使用しています。
ダーティ	ライトバック・キャッシュでは、メモリ階層内のあるレベルに到達したライトは、そのレベルを更新しますが、その下位レベルまでは更新しません。したがって、キャッシュ・ラインが有効で、次の下位レベルに送られていない更新を含んでいる場合、そのラインはダーティであると言います。有効なキャッシュ・ラインで逆の状態は、クリーンと言います。
ダイレクト・マップ・キャッシュ	ダイレクト・マップ・キャッシュは、下位レベルのメモリの各アドレスをキャッシュの単一のロケーションにマップします。キャッシュ内の同一ロケーションに複数のロケーションがマップされることがあります。これは、キャッシュのロケーションのセットからデータを置く場所を 1 ヶ所選択する、マルチ・ウェイ・セット・アソシアティブ・キャッシュと対照的です。ダイレクト・マップ・キャッシュは、1 ウェイ・セット・アソシアティブ・キャッシュと考えることができます。
タグ	特定ラインにおいてストアされているアドレスの最上位ビットを含む記憶エレメント。タグ・アドレスは CPU から直接見ることのできない特別なタグ・メモリにストアされています。アクセスがヒットかミスかの判別をするために、アクセスごとにキャッシュはタグ・メモリに照会します。
タッチ	あるアドレスへのメモリ操作は、そのアドレスにタッチすると言われます。タッチはまた、その唯一の目的を、特定のレベルのキャッシュに配列エレメントまたはメモリ・アドレスの他の範囲をアロケートとしてリードすることでもあります。キャッシュにアロケートする目的で、メモリのある範囲をタッチするために使用される、CPU によるループはしばしばタッチ・ループといわれます。配列にタッチすることは、ソフトウェア制御によるデータのプリフェッチの一形式です。
ビクティム	新しいライン用のスペースがセットにアロケートされ、そのアドレスに対応したセット内のすべてのライン・フレームが有効なデータを保持しているとき、新しいデータ用の空きを作るために、キャッシュ・コントローラはラインを追い出すための有効ラインを 1 つ選択しなくてはなりません。通常は、最低使用頻度（LRU）ラインが選択されます。追い出されるラインは、ビクティム・ラインとして知られます。ビクティム・ラインがダーティの場合、その内容はビクティム・ライトバックを使用して、次の下位レベルのメモリにライトされます。

表 2. 用語と定義（続き）

用語	定義
ビクティム・バッファ	ビクティムがライトバックされるまで、それらを保持する特別なバッファ。新たに入ってくるデータ用にキャッシュに空きを作るために、ビクティム・ラインをビクティム・バッファに移動します。
ビクティム・ライトバック	ダーティ・ラインが追い出されるとき（すなわち、更新データをもつラインが追い出される）、更新データは下位レベルのメモリにライトされます。このプロセスはビクティム・ライトバックとして参照されます。
ヒット	キャッシュ・ヒットは、リクエストされたメモリのロケーションのデータがキャッシュ内にあるとき起こります。ヒットの反対は、ミスです。キャッシュ・ヒットにより、ソース・メモリからよりも格段に早くキャッシュからデータをフェッチできるので、ストールを最少に抑えることが可能です。ヒットとミスの決定は、メモリ階層の各レベルで個々に行われます。あるレベルでのミスが、それより下位のレベルではヒットの場合もあります。
フェッチ・パケット	単一サイクルでフェッチされる 8 個の命令のブロック。各フェッチ・パケットは複数の実行パケットから構成される場合があります。そのような場合、パケットを使いきるのに複数のサイクルを要します。
フルアソシアティブ・キャッシュ	キャッシュ内のいかなるロケーションにもすべてのメモリ・アドレスがストア可能なキャッシュ。そのようなキャッシュは柔軟性に富んでいますが、ハードウェアに作りこむのは通常、現実的ではありません。フルアソシアティブ・キャッシュは、より多くの制約のあるアロケーション・ポリシーをもつ、ダイレクト・マップ・キャッシュおよびセット・アソシアティブ・キャッシュとは明確な対照をなしています。ダイレクト・マップまたはセット・アソシアティブ・キャッシュのパフォーマンスを分析する場合に、フルアソシアティブ・キャッシュは競合性ミスと容量性ミスを概念的に区別するのに役に立ちます。セット・アソシアティブ・キャッシュの用語において、フルアソシアティブ・キャッシュはライン・フレームと同数のウェイで、1 つのセットだけをもつセット・アソシエイティブ・キャッシュと等価です。
ミス	リクエストされたメモリ・ロケーションのデータがキャッシュ内にないとき、キャッシュ・ミスが起こります。ミスによりライン・フレームがアロケートされ、データが次の下位レベルのメモリからフェッチされる間、リクエストがストールされます。L1D からの CPU ライト・ミスのような場合では、厳密に CPU をストールさせる必要がないこともあります。キャッシュ・ミスは、3 つのカテゴリー（コンパルソリー・ミス、競合性ミス、および容量性ミス）に分類されます。
ミス・パイプライン化	1 つのキャッシュ・ミスをサービスするプロセスは、数サイクルにわたりパイプライン化されます。ミスが連続して起こる場合、ミスをパイプライン化することで、いくつかのミスの処理をオーバーラップすることが可能になります。結果として、続いて起こるミスのオーバーヘッドは見かけ上なくなり、付加的なミスによるストール・ペナルティの増分は、1 つのミスを個別に処理する場合の増分よりもはるかに減少します。
メモリ・オーダリング	どのような順序でメモリ操作の効果がメモリ内に現われるか定義します（コンシステンシーと呼ばれることもあります）。メモリ階層内の特定レベルにおけるストロング・メモリ・オーダリングは、プログラムの順序と異なる順序でのそのレベルのメモリへのメモリ・アクセスの効果を観測することができないことを示しています。リラックス・メモリ・オーダリングでは、メモリ階層で異なる順序でのメモリ操作の効果が観測できるようにします。ストロング・メモリ・オーダリングが、メモリ・システムがプログラムの順序通りにメモリ操作を実行することをリクエストしているのではなく、プログラムの順序と一致する順序で、それらの効果が他のリクエスターから観測可能であることだけに注意してください。C64x のメモリ階層が提供するメモリ・オーダリングの保証については、8.3 節で説明します。

表 2. 用語と定義（続き）

用語	定義
有効	キャッシュ・ラインが次のレベルのメモリからフェッチされたデータを保持しているとき、そのライン・フレームは有効です。ライン・フレームがデータを保持していないとき、インバリデート状態が発生します。これは、まだ何もキャッシュされていないか、あるいはコヒーレンス・プロトコル、プログラム・リクエストなど何らかの理由により、以前キャッシュされたデータがインバリデートされた場合のどちらかが原因です。有効の状態は、下位レベルのメモリからフェッチされた後で、データが変更されているかどうかについては示しません。これは、ラインがダーティ、またはクリーンという状態によって表されます。
容量性ミス	キャッシュに、あるプログラムのワーキング・セット全体を保持するのに十分な空き領域がないために起こるキャッシュ・ミス。コンパルソリー・ミスおよび競合性ミスと比較してください。
ライト・アロケート	ライト・アロケート・キャッシュは、ライト・ミスが発生するとキャッシュ内にスペースをアロケートします。スペースはキャッシュのアロケーション・ポリシー（たとえば LRU ）に従ってアロケートされ、ラインのデータは次の下位レベルのメモリからリードされます。キャッシュにデータがあると、ライトが処理されます。ライトバック・キャッシュでは現在のレベルのメモリだけが更新されます。ライト・データは、すぐには次のレベルのメモリに渡されません。
ライト・マージング	ライト・マージングは、複数の独立したライトを、単一のより大きなライトに結合します。これは、処理の必要な個別のメモリ・アクセス数を削減することで、メモリ・システムのパフォーマンスを改善します。たとえば、C64x デバイスでは、同じダブルワード・アドレスへのライトの場合、L1D ライト・バッファはいくつかの条件のもとで複数のライトをマージすることが可能です。この例ではその結果として、より大きく効果的なライト・バッファのキャパシティが得られ、また L2 に対するバンド幅のインパクトを軽減することができます。
ライトスルー・キャッシュ	ライトスルー・キャッシュはすべてのライトを下位レベルのメモリに渡します。下位レベルのメモリに渡されなかった更新データは、ありません。結果として、ライトスルー・キャッシュ内のキャッシュ・ラインは、決してダーティになることはありません。C64x デバイスはライトスルー・キャッシュを利用していません。
ライトバック	有効だがダーティなキャッシュ・ラインから、下位レベルのメモリに更新データをライトする処理。ライトバックが起こった後、そのキャッシュ・ラインはクリーンと見なされます。インバリデートと組み合わされない限り（ライトバック・インバリデート）、ライトバック後そのラインは有効のままです。
ライトバック・インバリデート	ライトバック操作とそれに続くインバリデート。ライトバックおよびインバリデートを参照してください。C64x デバイスでは、キャッシュ・ラインのグループに対するライトバック・インバリデートは、ダーティなキャッシュ・ラインのデータのみ書き出し、対象キャッシュ・ラインのすべての内容がインバリデートされます。
ライトバック・キャッシュ	ライトバック・キャッシュは、ライト・ヒット時に自身のデータを変更するだけです。更新を直に次の下位レベルのメモリには送りません。そのデータは将来のある時点でライトバックされます。ライトバックされるのは、キャッシュ・ラインが追い出されるとき、または下位レベルのメモリが上位レベルのメモリからアドレスをスヌープするときのような場合です。また、キャッシュ・コントロール・レジスタを使用して、ある範囲のアドレスに対してライトバックを直接起動することも可能です。ライトバック・キャッシュに対するライト・ヒットは、対応するラインがダーティとしてマークされる原因になります。すなわち、そのラインには下位レベルのメモリにまだ送っていない更新が含まれています。

表 2. 用語と定義（続き）

用語	定義
ライン	キャッシュ・ラインはキャッシュが扱う最も小さなデータ・ブロックです。キャッシュ・ラインは、CPU またはすぐ上位レベルのメモリからのデータ・アクセスのサイズよりも大きいのが一般的です。たとえば、CPU がメモリから 1 バイトをリクエストする場合であっても、リード・ミス時のキャッシュは、そのリクエストを満たすために、1 ライン分のデータをリードします。
ライン・サイズ	1 キャッシュ・ラインのバイト単位でのサイズ。
ライン・フレーム	キャッシュ・データ（1 ライン）、関連するタグ・アドレス、およびそのラインのステータス情報を保持するキャッシュ内のロケーション。ステータス情報には、そのラインが有効 なのか、ダーティなのか、また LRU の現在状況を含みます。
リード・アロケート	リード・アロケート・キャッシュは、リード・ミス時にキャッシュ内にスペースをアロケートします。キャッシュがライト・アロケート・キャシュでない限り、ライト・ミスはアロケートを起こす要因にはなりません。ライト・アロケートを行わないキャッシュでは、ライト・データは次の下位レベルのキャッシュに渡されます。
ロード・スルー	CPU からのリクエストが 1 次、および 2 次キャッシュの両方でミスすると、そのデータは外部メモリからフェッチされ、そして 1 次、および 2 次キャッシュの両方に同時にストアされます。データをストアし、それと同時に上位レベルのキャッシュにそのデータを送るキャッシュがロード・スルー・キャッシュです。最初に下位レベルにデータをストアし、次のステップで上位レベルのキャッシュにデータを送るキャッシュと比較すると、ロード・スルー・キャッシュはストール時間を軽減します。
ロング・ディスタンス・アクセス	キャッシュ不可能なメモリに対して、CPU が行うアクセス。ロング・ディスタンス・アクセスは、キャッシュ可能とマークされていない外部メモリをアクセスするときに使用されます。
ワーキング・セット	プログラムまたはアルゴリズムのワーキング・セットとは、特定の時間周期内に参照されるデータ、およびプログラムの全体のことです。ワーキング・セットについて、上位レベルのメモリを分析するときは各アルゴリズムを基にし、また下位レベルのメモリを分析するときは全体のプログラムを基にして考えることはしばしば役に立ちます。

3 1 次データ・キャッシュ（L1D）

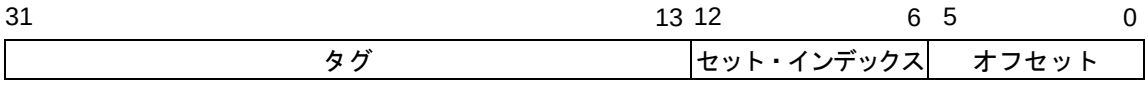
1 次データ・キャッシュ（L1D）は、CPU からのデータ・アクセスを処理します。以下の節では、L1D のパラメータと動作について説明します。L1D の動作は、第 7 章「メモリ・システム・コントロール」で説明する、様々なレジスタにより制御されます。

3.1 L1D パラメータ

L1D は、16 K バイトのキャッシュです。64 バイトのライン・サイズ、128 セットの 2 ウェイ・セット・アソシアティブ・キャッシュです。また、L1D と L2 メモリ間には、64 ビット、4 エントリのライト・バッファを備えています。

物理アドレスは、キャッシュ上に直接マップされます。物理アドレスは、図 3 に示すように 3 つのフィールドに分割されます。アドレスのビット 5-0 は、そのライン内のオフセットを指定します。アドレスのビット 12-6 は、キャッシュ内の 128 セットのうちの一つを選択します。アドレスのビット 31-13 は、ラインのタグとして使用されます。

図 3. L1D アドレス・アロケーション



L1D は 2 ウェイ・キャッシュなので、各セットは 2 つのキャッシュ・ライン（各ウェイに一つ）から構成されています。アクセスごとに、L1D はアクセスするアドレスのタグ部分を、適切なセット内の両方のラインのタグ情報と比較します。タグがそれらのラインの 1 つで一致し、そのラインが有効とマークされている場合、そのアクセスはヒットです。これらの条件が満たされない場合、そのアクセスはミスです。ミス・ペナルティについては、3.2 節で説明します。

L1D は、リード・アロケートのみのキャッシュです。これは、L1D ではリード・ミスに対して新しいラインがアロケートされることを意味します。しかし、ライト・ミスに対してはアロケートされません。そのため、4 エントリーのライト・バッファを L1D と L2 間に配置して、ライト・ミスのデータを取り込みます。このライト・バッファは、C621x/C671x デバイスのライト・バッファと比較して拡張されています。ライト・バッファについては、3.2.3 項で説明します。

L1D は、最低使用頻度（LRU）ライン・アロケーション・ポリシーを採用しています。つまり、L1D リード・ミス時、セット内で最も長い時間リードまたはライトされていないラインを追い出し、新たに入ってくるデータのための空きを作ることということです。インバリデート・ラインは、常に最低使用頻度として見なされることに注意してください。

選択されたラインがダーティの場合、つまり、その内容が更新されている場合、ビクティム・ラインのデータを、ビクティム・ライトバックとして L2 にライトバックするための準備が行われます。実際のビクティム・ライトバックは、新しいデータがフェッチされた後で、その新しくフェッチされたデータがキャッシュ可能とみなされる場合だけ起こります。新しくフェッチされたデータがキャッシュ不可能な場合、ビクティム・ライトバックはキャンセルされ、ビクティム・ラインは L1D キャッシュにとどまります。

3.2 L1D パフォーマンス

3.2.1 L1D メモリ・バンキング

C64x DSP は、C620x/C670x ファミリーが採用している構造と類似の最下位ビット（LSB）ベースのメモリ・バンキング構造を備えています。C64x デバイスの L1D は、8 個の 32 ビット幅のバンクに分割されています。それらのバンクは、シングルポートでサイクルあたり 1 アクセスのみが可能です。これは、シングルポート・メモリを複数バンクもつ代わりに、デュアル・ポート・メモリを単一バンクもつ C621x/C671x デバイスとは対照的です。図 4 は、アドレスのビット 4-2 はバンクを、また、ビット 1-0 はバンク内のバイトを選択することを示しています。

図 4. バンク・ナンバーとアドレスのマッピング

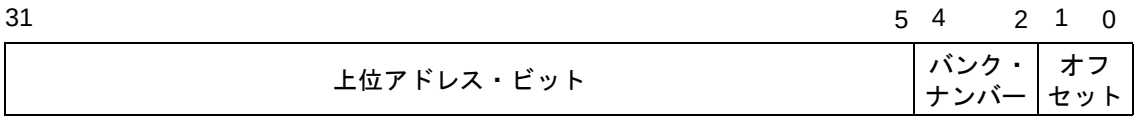
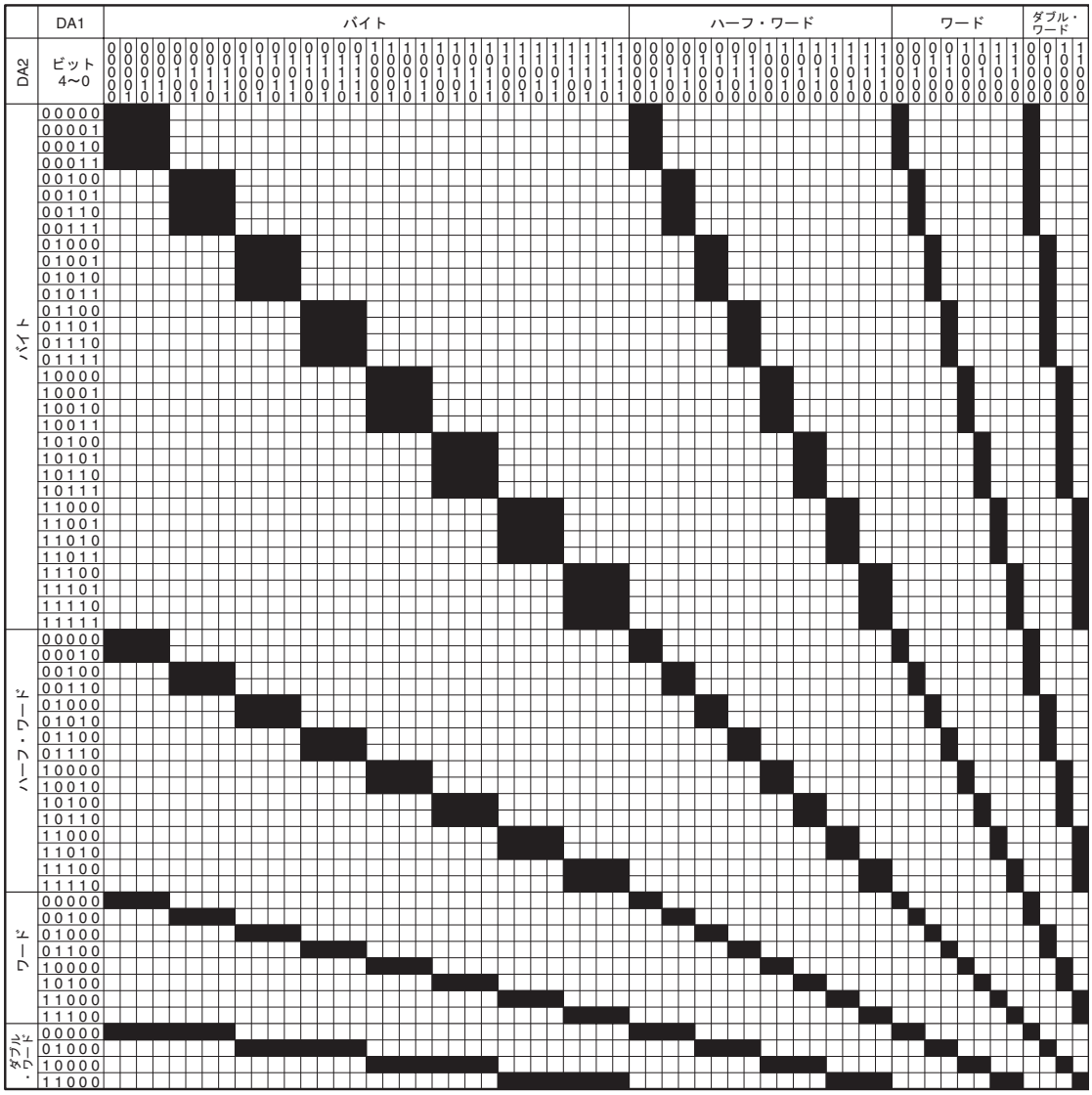


図 5 で斜線部分は、バンク競合ストールを起こす可能性がある同時アクセスにおけるアドレスの LSB の組み合わせを示しています。以下の特別な場合を除き、同じバンクに対する 2 つの同時アクセスには、1 サイクル・ストールのペナルティが発生します。

- ❏ メモリ・アクセスが両方とも、同一ワード内のオーバーラップしていないバイトに対するライト。すなわち、アドレスのビット 31-2 が同じ場合。
- ❏ メモリ・アクセスが両方とも、同一ワードのすべて、または一部をアクセスするリード。すなわち、アドレスのビット 31-2 が同じ場合。この場合、2 つのアクセスはオーバーラップします。
- ❏ メモリ・アクセスの片方または両方が L1D でミスするライト、代わりにライト・バッファが処理するライト。ライト・バッファに関する情報は 3.2.3 項を参照してください。
- ❏ メモリ・アクセスが単一のアラインなしアクセス。アラインなしアクセスは、たとえメモリ・システムがそれらを複数のアクセスに分割しても、バンク競合ストールの原因にはなりません。

同一バンクに対する同時のリード・アクセスとライト・アクセスは、常にストールの原因になることに注意してください。同一バンクに対する 2 つのリードまたは 2 つのライトは、上の条件が満たされている限りストールしません。

図 5.　　潜在的に競合するメモリ・アクセス



注：競合は、斜線部分で示されています。

C64x デバイスのバンキングは、C620x/C670x デバイスのメモリ・バンキングと類似していますが、次の 2 つの点で異なります。

- ❑ バンクのビット幅は、C64x デバイスでは 32 ビットで、C620x/C670x デバイスでは 16 ビットです。
- ❑ C620x/C670x デバイスでは、アクセスが同一ワードに対するものであるかどうかに関わらず、同一バンクに対する同時アクセスにより必ずストールします。C64x デバイスは、前に説明したように、ストールしないで続行する特別な場合が多くあります。

3.2.2 L1D ミス・ペナルティ

L1D は毎サイクル、CPU からのデータ・アクセス・リクエストを 2 つまで処理します。3.2.1 項で説明したようなバンク競合が起こらない限り、L1D にヒットするアクセスはストールなしで完了します。

L1D でミスしたリードは、リクエストされたデータがフェッチされている間は CPU をストールさせます。L1D は、リード・アロケート・キャッシュです。したがって、3.1 節で説明したように、リクエストされたデータのために新しいラインがアロケートされます。L2 SRAM にヒットする単独の L1D リード・ミスは、6 サイクルの間、CPU をストールさせます。また、L2 キャッシュにヒットする単独の L1D リード・ミスは、8 サイクルの間、CPU をストールさせます。これは L1D からのリクエスト処理を遅らせる他のメモリ・トラフィックが L2 内にないことを前提としています。5.4 節では、L2 にアクセスする様々なリクエスター間の相互の影響について説明します。

L2 でもミスになる L1D リード・ミスは、L2 が外部メモリからそのデータを取り出す間、CPU をストールさせます。データが取り出されると L2 にストアされ、L1D に転送されます。外部ミス・ペナルティは、システム負荷に関するその他の状況や、外部データを保持するために使用される外部メモリのタイプや幅によって変化します。5.2 節では、L1D に代わって L2 がどのようにキャッシュ・ミスを処理するかについて説明します。

同一サイクルの同一ラインで 2 つのリード・ミスがある場合、1 つのミス・ペナルティのみが課せられます。同様に、同一ラインに 2 回のアクセスが連続してあり、最初のアクセスがミスの場合、2 番目のアクセスに対して追加のミス・ペナルティは何ら与えられません。

L1D 内でラインをアロケートするプロセスは、ビクティム・ライトバックを引き起こします。ビクティム・ライトバックは、L1D の更新データを下位レベルのメモリに移動します。更新データが L1D から追い出されるとき、キャッシュはそのデータをビクティム・バッファに移動します。データがビクティム・バッファに移動されると、L1D は現行のリード・ミス処理を再開します。ビクティム・ライトバックの続きの処理は、バックグラウンドで行われます。しかし、続いて起こるリード・ミスやライト・ミスは、ビクティム・ライトバックが処理される間、待機しなくてはなりません。ビクティム・ライトバックは結果として、キャッシュ・ミスを処理する時間を著しく長引かせます。

L1D はリード・ミスをパイプライン化します。異なるラインに対する連続したリード・ミスはオーバーラップされることがあり、全体的なストール・ペナルティが軽減されます。ミスごとのストール・ペナルティの増加を 2 サイクルにまで抑えることができます。ミス・パイプライン化については、3.2.4 項で説明します。

ライト・ミスは直接 CPU をストールさせません。その代わり、ライト・ミスは L1D と L2 間にあるライト・バッファにキューされます。CPU はライト・ミスによりストールしませんが、ライト・バッファは様々な状況のもとで CPU をストールさせます。ライト・バッファの効果については、3.2.3 項で説明します。

3.2.3 L1D ライト・バッファ

L1D は、ライト・アロケートを行いません。その代わり、ライト・ミスは、L1D でのラインのアロケートなしに L2 に直接渡されます。これらのライト・ミスを取り込むために、ライト・バッファが L1D キャッシュと L2 メモリ間に存在します。ライト・バッファは L1D から L2 へのライト用に、64 ビットのパスを用意し、4 つの未処理のライト・リクエストを格納する空きを備えています。

ライト・バッファに空きがある場合、L1D をミスしたライトは CPU をストールさせません。ライト・バッファがフルの場合、バッファにライトのための空きができるまで、ライト・ミスは CPU をストールさせます。ライト・バッファはまた、リード・ミスの時間が延長されることで、間接的に CPU をストールすることもあります。L1D をミスしたリードは、ライト・バッファがエンプティでない間は、処理されません。ライト・バッファが空くと、リード・ミスが処理されます。これはリード・ミスのアドレスが、ライト・バッファ内でペンディングになっている、ライトのアドレスとオーバーラップする場合があるので必要です。

L2 は、リクエストされた L2 バンクがビジーでなければ、毎サイクル、ライト・バッファからの新しいリクエストを処理することができます。L2 のバンキング構造とパフォーマンスに対する影響については、5.3 節で説明します。

C64x のライト・バッファは、ライト・リクエストのマージングが可能です。次のすべてのルールに従う場合は、2 つのライト・ミスを単一のトランザクションにマージします。

- ❏ 2 つのアクセスのダブルワード・アドレス（すなわち、上位 29 ビット）が同じである。
- ❏ 2 つのライトが L2 SRAM のロケーション（L2 のキャッシュ内に保持されている可能性のあるロケーションではない）に対するものである。
- ❏ ちょうど最初のライトがライト・バッファのキューに置かれたところである。
- ❏ 現在、2 番目のライトがバッファのキューに置かれようとしている。
- ❏ 最初のライトが未だ L2 コントローラに渡されていない。

前述の条件は、プログラムが大規模で連続的なライトを行うとき、またはメモリ内の構造体に小規模のライトをバーストで行うときのような多くの状況で発生します。こういった場合、ライト・マージングは、ライト・バッファ内にある独立したストアの数を減少させることで、ライト・バッファの実質的な容量を増加させます。また、多数のライト・ミスのあるプログラムに対するストール・ペナルティを軽減します。

二次的な効果として、ライト・マージングは、L2 で実行されるメモリ操作の数を減らします。これは、L2 が処理する必要のある個々のライト操作の合計数を減らすことで、L2 メモリの全体的なパフォーマンスを向上させます。アクセス先が、隣接している場合、L2 バンクに複数回アクセスをするのではなく、複数のアクセスを 1 回にまとめます。これにより、他のリクエスターがそのバンクをより迅速にアクセスすることが可能になり、また、CPU が次のサイクルで直ちに次のバンクに移動できるようになります。

3.2.4 L1D ミス・パイプライン化

L1D キャッシュは、リード・ミスをパイプライン化します。L2 SRAM から処理される場合、単一の L1D リード・ミスは 6 サイクルを要します。また、L2 キャッシュから処理される場合は、8 サイクルを要します。ミス・パイプライン化は、いくつかのキャッシュ・ミスに対する処理をオーバーラップさせることで、オーバーヘッドの多くを見かけ上なくすことができます。

L1D ミス・パイプライン化が効率的に働くのは、複数の L1D リード・ミスが未処理のときです。C64x DSP のロード命令は、5 サイクルの深いパイプラインを持ち、C64x DSP はサイクルあたり 2 回までアクセスを出すことができます。このパイプラインで、L1D はパイプライン・ステージ（E2）でタグの比較を実行します。それに続くステージ（E3）では、キャッシュ・ヒットおよびミスを処理します。キャッシュ・リード・ミスにより、CPU はストールします。

L1D は、未処理のビクティム・ライトバックがなく、またライト・バッファがエンプティのときだけ単一のリード・ミスを処理します。2 つのキャッシュ・ミスが同時に起こるとき、L1D はプログラム順にそれらのミスを処理します（プログラム順については、8.3.1 項で説明します）。2 つのライト・ミスの場合、それらのミスはライト・バッファに入れられ、ライト・バッファがフルでない限り、CPU はストールしません（ライト・バッファについては、3.2.3 項で説明しています）。2 つのリード・ミス、またはリード・ミスとライト・ミスが 1 つずつのときは、異なるセット、すなわち、アドレスのビット 13-6 が異なるミスである限り、それらのミスはオーバーラップして処理されます。

キャッシュ・ミスは、E3 パイプライン・ステージで処理されます。L1D が E3 ですべてのキャッシュ・ミスに対してコマンドを L2 に出すと、L1D はキャッシュ・ミスが E2 パイプライン・ステージでのアクセスが原因であると見なして、パイプライン・ステージを内部的に一段階進めます。これにより L1D は、同時に起こるキャッシュ・ミス、また連続したサイクルで起こるキャッシュ・ミスのリクエストを、積極的にオーバーラップさせることが可能になります。ライト・バッファおよびビクティム・ライトバックがエンプティの場合のみ、L1D は E2 でのアクセスを考慮します。L1D の内部状態が進んだとしても、E3 ステージにあるアクセスに対するデータが戻るまで、CPU ストールは解除されません。

CPU ストールが解除されると、E2 ステージにあったメモリ・アクセスは E3 パイプライン・ステージに進みます。これにより 1 つまたは 2 つの新しいアクセスが、E2 パイプライン・ステージにもち込まれます。それはまた、1 つまたは 2 つの未処理のキャッシュ・ミスが E2 から E3 に受け渡されることになります。L1D は最初に、E3 内の未処理のキャッシュ・ミスに対してコマンドを出します。いったん E3 内のアクセスが処理されると、前述したように、L1D は E2 のアクセスを考えます。どの場合でも、まだ完了していないアクセスが E3 にあるとき、L1D は CPU をストールさせます。

結果的に、L1D は、L2 に対するリクエストを連続的に生成することができます。異なるキャッシュ・ラインに対する 2 つのメモリ・リードを毎サイクル発行するコードで、この効果が最大になります。前述のように、特に継続したリード・ミスがあるような場合、パイプライン化によりパフォーマンスが向上します。

L1D が新しいキャッシュ・ミスの処理を、前のミスの処理とオーバーラップすることが可能なとき、追加のミス・ペナルティをミスあたり 2 サイクルにまで抑えることができます。したがって、連続したミスが継続するシーケンスにおけるミス・ペナルティの平均は、理想的な場合で、ミスあたり 2 サイクルに近づきます。表 3 および表 4 は、L2 キャッシュ、および L2 SRAM にヒットする場合で、連続して起こる L1D リード・ミスの回数に対するパフォーマンスを説明しています。ここでは、すべてのミスがオーバーラップ可能と仮定しています。さらに、L1D キャッシュ・ミスの処理時間を延ばす可能性のある、その他のメモリ・トラフィックが L2 にないこと、そしてすべてのミスが L1D キャッシュ・ラインの半分のうち同じ側にあることを仮定しています。

表 3.　L2 キャッシュにヒットする L1D ミス数を変更した場合のミスあたりのサイクル

ミス数	合計ストール・サイクル	ミスあたりの平均サイクル
1	8	8
2	10	5
3	12	4
4	14	3.5
> 4（変化しない）	6 + (2 * M)	2 + (6 / M)

注： M = ミスの合計数

表 4.　L2 SRAM にヒットする L1D ミス数を変更した場合のミスあたりのサイクル

ミス数	合計ストール・サイクル	ミスあたりの平均サイクル
1	6	6
2	8	4
3	10	3.33
4	12	3
> 4（変化しない）	4 + (2 * M)	2 + (4 / M)

注：M = ミスの合計数

4 1 次プログラム・キャッシュ（L1P）

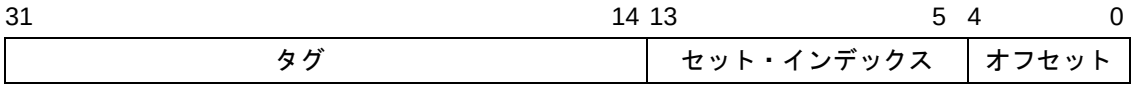
1 次プログラム・キャッシュ（L1P）は CPU からのプログラム・フェッチを処理します。以下の節では、L1P のパラメータと動作について説明します。L1P の動作は、第 7 章「メモリ・システム・コントロール」で説明する、様々なレジスタにより制御されます。

4.1 L1P パラメータ

L1P は、16 K バイトのキャッシュです。32 バイトのライン・サイズ、512 セットのダイレクトマップ・キャッシュです。

物理アドレスが、一定の方法でキャッシュ上にマップされます。物理アドレスは、図 6 に示すように 3 つのフィールドに分割されています。アドレスのビット 4-0 は、セット内の命令を指定します。アドレスのビット 13-5 は、キャッシュ内の 512 セットのうちの 1 つを選択します。アドレスのビット 31-14 は、ラインのタグの役割を果たします。

図 6. L1P アドレス・アロケーション



L1P はダイレクトマップ・キャッシュなので、各アドレスはキャッシュ内の固定のロケーションにマップされます。すなわち、各セットは、厳密に 1 ライン・フレームから構成されます。キャッシュ・ミスが起こると、キャッシュは新たに入ってくるデータに対応するラインをアロケートします。L1P は CPU からのライトをサポートしていないので、ラインにあった以前の内容は破棄されます。

4.2 L1P パフォーマンス

4.2.1 L1P ミス・ペナルティ

L1P にヒットするプログラム・フェッチは、CPU をストールさせることなく 1 サイクルで完了します。L2 内でヒットする L1P ミスは、そのミスを起こした近隣の実行パケットの並列度に依存し、最大で CPU を 8 サイクルストールさせます。これについては、4.2.2 項で詳しく説明します。

L2 キャッシュでもミスする L1P ミスは、L2 が外部メモリからそのデータを取り出して L1P にデータを転送するまで、CPU をストールさせます。そして、L1P は CPU にデータを戻します。この遅延は、システム負荷に関するその他の要因や、プログラムの保持に使用される外部メモリのタイプに依存します。

C64x DSP では、実行パケットが 2 つのフェッチ・パケットにまたがることが可能です。このまたがることによっても、1 つのミスに対するペナルティは変わりません。しかし、両方のフェッチ・パケットが L1P にない場合、キャッシュ・ミスが 2 回起こります。

4.2.2 L1P ミス・パイプライン化

L1P キャッシュは、キャッシュ・ミスをパイプライン化します。単一の L1P キャッシュ・ミスは、L2 からデータを取り出すのに 8 サイクル必要です。ミス・パイプライン化は、いくつかのキャッシュ・ミスの処理をオーバーラップすることで、オーバーヘッドの多くを見かけ上なくすことが可能です。さらに、キャッシュ・ミス・オーバーヘッドのいくつかは、フェッチ・パイプライン中で起こるディスパッチ・ストールとオーバーラップさせることができます。

複数の未処理の L1P キャッシュ・ミスがあると、L1P ミス・パイプライン化の効果が現れます。C64x DSP のフェッチ・パイプラインは、フェッチ・パイプラインに空きがある限り、新しいフェッチ・パケットのフェッチを毎サイクル試みることで、ミス・パイプライン化を実現します。この動作を理解するには、フェッチ・パイプライン自体について理解する必要があります。

フェッチおよびデコード・パイプラインは、最初の実行ステージ（E1）に至るまでに、次の 6 ステージに分けられています。

- ❏ PG － プログラム生成
- ❏ PS － プログラム送信
- ❏ PW － プログラム・ウェイト
- ❏ PR － プログラム・リード
- ❏ DP － ディスパッチ
- ❏ DC － デコード

C6000 DSP 命令は、フェッチ・パケットと実行パケットの 2 つのグループにまとめられます。CPU は、フェッチ・パケットという、8 命令の固定サイズの組みでメモリから命令をフェッチします。それらの命令はデコードされ、実行パケットという、並列に発行される命令の組みに分けられます。1 つの実行パケットは 1 個から 8 個の命令で構成されています。したがって、単一のフェッチ・パケットは複数の実行パケットで構成されることがあります。C64x DSP では、実行パケットが 2 つのフェッチ・パケットにまたがることもあります。パイプラインのプログラム・リード（PR）ステージには、フェッチ・パケットのシーケンス内にある、実行パケットのシーケンスを識別する役割があります。ディスパッチ（DP）ステージは、実行パケットを取り出して、各機能ユニットに送り出します。

フェッチ・パケットと実行パケットには相違があるため、フェッチ・パイプライン全体が毎サイクル進む必要はありません。むしろ PR パイプライン・ステージは、DP ステージが PR 内に保持されているフェッチ・パケットを完全に使い尽くした時点で、プログラム・ウェイト（PW）ステージがその内容を PR ステージに進めることを許します。PR 前のステージは、ギャップを埋めるために必要に応じて進みます。したがって、キャッシュ・ミスがないとき、DP ステージが現行のフェッチ・パケットから個々の実行パケットを引き出す間、フェッチ・パイプラインの初期ステージはストールされます。このストールを、ディスパッチ・ストールといいます。

C64x DSP は、パイプラインの初期ステージでキャッシュ・ミスが待ち状態の間に、それらのステージが DP に向かって進むのを許すことで、このディスパッチ・ストールを巧みに利用しています。キャッシュ・ミスは、PR 、PW 、および PS パイプライン・ステージを待たされることがあります。DP ステージは、パイプラインの PR ステージ内のフェッチ・パケットを処理している間、ディスパッチ・ストールによって PR ステージをストールさせるので、これらのキャッシュ・ストールを CPU に対して通知する必要はありません。しかし、フェッチ・パケットが完全に使い尽くされると、PW ステージの内容を PR ステージに進める必要があります。この時点で、DP が PR から実行パケットをリクエストし、そこにまだ未処理のキャッシュ・ミスがある場合、CPU はストールします。

分岐が実行されるとき、分岐先を含むフェッチ・パケットは、分岐先が E1 パイプライン・ステージに到達するまで、フェッチ・パイプラインを通してサイクルごとに進みます。分岐先は、前述のディスパッチ・ストールを無効にします。その結果、分岐先は、他の命令のようにはミス・パイプライン化から同じような恩恵を得ることはありません。しかし、分岐先の直後に続くフェッチ・パケットは恩恵を受けます。分岐先に続くフェッチ・パケットのコードが直ちに実行されなくとも、分岐によりいくつかの連続したフェッチがトリガされます。したがって、そのコードのすべてのミスはパイプライン化されます。さらに、分岐が実行される以前にリクエストされていたが、DP パイプライン・ステージに進めなかったフェッチ・パケットに対して、ストールが発生することはありません。

単一の L1P ミスに対するミス・ペナルティは 8 サイクルです。連続したミスにおいて 2 番目のミスでは、最大 2 サイクルの追加のストール・ペナルティが課せられます。直線的な（分岐のない）コードにおいて連続ミスが持続する場合、コードの平均的な並列度に基づき平均ミス・ペナルティが発生します。直線的なコードにおいて長くミスが持続する場合の平均ミス・ペナルティを表 5 にまとめます。

表 5.　多くの連続した実行パケットを実行する場合の平均ミス・ペナルティ

実行パケットあたりの命令数	実行パケットあたりの平均ストール数
1	0.125
2	0.125
3	0.688
4	1.500
5	1.813
6	2.375
7	2.938
8	4.000

5 2 次共用メモリ

2 次共用メモリ（L2）は、SRAM 、キャッシュ、またはその両方として動作します。また、EDMA コントローラを使用した DMA アクセス、L1P および L1D の両方からのキャッシュ・ミスを処理します。以下の節では、L2 のパラメータと動作について説明します。L2 の動作は、第 7 章「メモリ・システム・コントロール」で説明する、様々なレジスタにより制御されています。

5.1 L2 キャッシュおよび L2 SRAM

L2 は、モードにより SRAM、キャッシュ、あるいはその両方として動作することができます。L2 SRAM は、SRAM としてマップされる L2 の一部のことです。L2 キャッシュは、キャッシュとして動作する L2 の一部のことです。各デバイスで、L2 SRAM と L2 キャッシュを合わせた合計容量は、L2 のモードに関係なく固定です。

L2 の合計サイズは、C64x デバイスにより異なります。C6414、C6415、および C6416 デバイスは、1024 K バイトの L2 メモリを備えています。C6411、および C6412 デバイスは、256 K バイトの L2 メモリを備えています。その他の C64x デバイスの L2 サイズについては、各デバイスのデータシートでご確認ください。

リセット後、L2 メモリ全体は SRAM としてマップされています。L2 SRAM は、メモリ上で、常にアドレス 0000 0000h から始まる連続した領域です。C6414、C6415、および C6416 デバイスは、アドレス範囲 0000 0000h から 000F FFFFh 上に L2 SRAM をマップします。C6411 および C6412 デバイスは、アドレス範囲 0000 0000h から 0003 FFFFh 上に L2 SRAM をマップします。

L2 キャッシュは、4 ウェイ・セット・アソシアティブ・キャッシュで、モードにより容量は 32 K バイトから 256 K バイトの値をとります。キャッシュ・コンフィギュレーション・レジスタ（CCFG）内の L2MODE フィールドにより、L2 キャッシュはイネーブルされます。L2 キャッシュをイネーブルすると、利用できる L2 SRAM の量が減少します。使用するデバイスのキャッシュ・モードによって L2 メモリ・マップがどのように変化するかについては、7.1.3 項で説明します。

外部の物理メモリのアドレスは、キャッシュ・モードにより異なったかたちで、L2 キャッシュ上にマップされます。物理アドレスは、図 7、図 8、図 9、および図 10 に示すように 3 つのフィールドに分けられます。アドレスのビット 6-0 は、ライン内のオフセットを指定します。アドレスの次の 6 ビットから 9 ビット（モードに依存する）は、キャッシュ内のセットを指定します。アドレスの残りのビットは、ラインのタグとして使用されます。

図 7.　L2 アドレス・アロケーション、256 K キャッシュ（L2MODE = 111b）

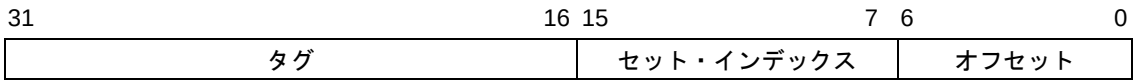


図 8.　L2 アドレス・アロケーション、128 K キャッシュ（L2MODE = 011b）

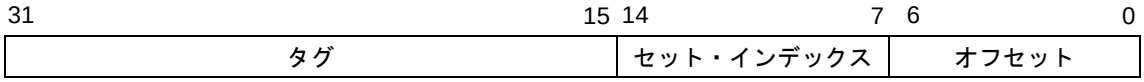


図 9.　L2 アドレス・アロケーション、64 K キャッシュ（L2MODE = 010b）

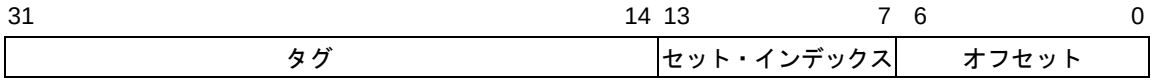
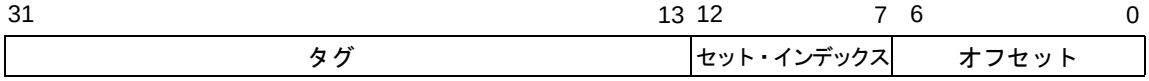


図 10.　L2 アドレス・アロケーション、32 K キャッシュ（L2MODE = 001b）



5.2 L2 の動作

L2 キャッシュは、イネーブルされている場合、L1P および L1D から外部アドレスへのリクエストを処理します。L2 キャッシュの動作は、L1P および L1D の動作と類似しています。リクエストが出されると、まず L2 はリクエストされたアドレスがキャッシュ内にあるかどうかを判別します。アドレスがキャッシュ内にある場合、そのアクセスはキャッシュ・ヒットと見なされ、L2 はそのリクエストを直接キャッシュ内で処理します。アドレスがキャッシュ内にない場合、そのアクセスはキャッシュ・ミスとなります。キャッシュ・ミスの場合、L2 はそのアドレスのキャッシュ可能性に合わせて、リクエストを処理します。

キャッシュ・ヒットの場合、L2 は、L2 キャッシュ内の対応するセットの LRU ステータスを更新します。アクセスがリードの場合、L2 はリクエストされたデータを返します。アクセスがライトの場合、L2 はキャッシュ・ラインの内容を更新し、そのラインにダーティのマークを付けます。L2 は、ライトバック・キャッシュです。したがって、L2 でのライト・ヒットは直ちに外部メモリには転送されません。外部メモリが更新されるのは、後にこのラインが追い出されるとき、またはこのラインがブロック・ライトバック・コントロール・レジスタを使用してライトバックされるときです。ブロック・キャッシュ操作については、7.3.2 項で説明します。

キャッシュ可能な外部メモリのロケーションに対するキャッシュ・ミスの場合、L2 は L2 キャッシュ内に新しいラインをアロケートします。L2 は L1D と異なり、リード・ミスおよびライト・ミスの両方でラインをアロケートします。L2 キャッシュは、キャッシュ・ミスでアロケートするセット内のラインを選択するために、最低使用頻度ポリシーを実行します。新しいデータにスペースがアロケートされると、L2 ライン 1 ライン分のデータが EDMA を通してアロケートされたラインに取り込まれます。

L2 からラインを追い出すには、ビクティムがクリーンかダーティかに関係なく、数ステップを必要とします。L2 内の各ラインに対して、L2 はそのラインが L1D でキャッシュされているかどうかも追跡します。ビクティム・ラインが L1D 内にあると検出されると、その L1D ラインを削除するためにスヌープ・インバリデート・リクエストを L1D に送ります。L1D は、対応するラインをインバリデートすることで対応します。L1D 内のそのラインがダーティの場合、更新データは L2 に渡され、追い出し中の L2 ラインとまとめられます。結合された結果は、外部メモリに書かれます。L1D と L2 のどちらもビクティム・ラインがダーティではない場合、その内容は破棄されます。これらの動作により、キャッシュ・ラインに対する最新のライトが、外部メモリにライトされることが保証されます。また、クリーン・ラインについては外部メモリにライトする必要がありません。

L2 からラインが追い出されるとき、L1P は参照されないことに注意してください。つまり、プログラム・コードが L2 から追い出されているにもかかわらず、L1P 内にとどまることができます。プログラム・コードは決して書き込まれないことを仮定しているためです。これに反し特異な状況では、キャッシュ済みのプログラム・コードを L1P から削除するために、プログラムは、7.3.2 項で説明のある、キャッシュ・コントロールを使用することができます。

キャッシュ・ミスがリードの場合、データが到着するとそのデータは L2 キャッシュにストアされます。また、そのデータは直ちにリクエスターに転送されます。この動作によって、全体的なストール時間が減少します。L1 からリクエストされた L2 キャッシュ・ラインの一部は、直接 L1 に送られ、L1 キャッシュ内でアロケートされます。また、L2 ライン全体は、到着した時に L2 キャッシュ内にストアされます。

キャッシュ・ミスがライトの場合、新たに入ってくるデータは、L1D からのライト・データとマージされます。そしてマージされた結果は、L2 にストアされます。L1D はライト・アロケートではないため、L1D 内へのラインのアロケートは直ちには行われません。

キャッシュ不可能なロケーションに対するキャッシュ・ミスは、ロング・ディスタンス・アクセスとなります。ロング・ディスタンス・リードでは、L2 はリクエストされたデータのリード転送を、EDMA コントローラに出します。リクエストされたデータが返されると、L2 はそのデータをリクエスターに転送します。L2 が、L2 キャッシュ内にこのデータ用にスペースをアロケートしません。L1D からのロング・ディスタンス・リードでは、CPU がリクエストするデータだけがリードされます。L1P からのロング・ディスタンス・リードでは、32 バイトだけをリードします。このデータ・リードは L1D、L1P、および L2 内にキャッシュされません。

ロング・ディスタンス・ライトでは、L2 はライトされたデータの一時的なコピーをストアします。その後、ライト・ミスに対するライト転送を EDMA コントローラに出します。ロング・ディスタンス・ライトは、L1D ライト・バッファを使用して L1D からのみ発生させることができます。ライトされたデータは特別な保持バッファにストアされるので、ロング・ディスタンス・ライトの処理中に、CPU をストールさせる必要はありません。また、これ以降の L2 SRAM アドレス空間やオンチップ・ペリフェラルに対するライトも、ロング・ディスタンス・アクセスの実行中に処理できます。

L2 キャッシュは、ロング・ディスタンス・アクセスを同時に 1 つだけ許します。前のロング・ディスタンス・ライトが処理中は、新たなロング・ディスタンス・ライトは CPU をストールする可能性があります。同様に、ロング・ディスタンス・リードは、前のすべてのロング・ディスタンス・ライトが完了するまで、処理されません。

ロング・ディスタンス・アクセスに課せられたオーダリングの制約は、ストロング・オーダリングなメモリのキャッシュ不可能領域に対するアクセスを可能にします。これらのアクセスのプログラム順序は、保持されます。したがって、これらのアクセスをペリフェラルとインターフェイスするときや、他の CPU と同期をとるときに使用可能です。

5.3 L2 バンク構造

L2 メモリは、CPU のクロック速度で動作する 8 個の 64 ビット幅のバンクで構成されています。しかし、パイプライン・アクセスに 2 サイクルを必要とします。各バンクは新しいリクエストを 2 サイクルに 1 回しか処理できませんが、L2 は毎サイクル新しいリクエストを処理します。

各バンクはリクエストを処理するのに 2 サイクルを必要とするため、あるバンクに対してあるサイクルでアクセスすると、そのバンクに対する次のサイクルでのすべてのアクセスはブロックされます。したがって、隣接サイクルでのアクセスではバンク競合が起こります。同一バンクに対する繰り返しのアクセスは、リクエスターに関係なく、2 サイクルに 1 回の割合で処理されます。これは、L1D からのライト・ミスをマージすることの重要性を示しています。

L2 は、以下のソースからのリクエストを処理する必要があります。

- ❏ L2 でヒットする L1P リード・ミス。
- ❏ L2 でヒットする L1D リード・ミスまたはライト・ミス。
- ❏ EDMA リードおよびライト。
- ❏ 内部キャッシュ動作（ビクティム・ライトバック、ライン・フィル、スヌープ）

内部キャッシュ機構は、L2 メモリに対するリクエスト中、4 番目のソースです。これらのリクエストは通常のキャッシュ動作がきっかけとなるデータ移動です。これらのリクエストのほとんどはキャッシュ・ミス、またはユーザー起動のキャッシュ・コントロール操作により引き起こされます。

各サイクルで、ただ 1 つのリクエスターだけが L2 をアクセスできます。L2 で競合が発生すると、L2 は上記の順序でリクエストにプライオリティを付けます。すなわち、L1P リード・ヒットが最高のプライオリティで、以下 L1D 、その他と続きます。

同時アクセス、またはバンク競合が原因の L2 競合は、L1P および L1D のキャッシュ・ミスを処理するのに必要な時間を長引かせてしまいます。上記のプライオリティは、CPU 起動によるアクセスを、キャッシュによるバックグラウンド動作や EDMA アクセスの前に処理することで、この競合による影響を最小に抑えています。

5.4 L2 インターフェイス

L2 は、L1D キャッシュ、L1P キャッシュ、および EDMA からのリクエストを処理します。L2 は、L2 自体のメモリ、ペリフェラル・コンフィギュレーション・バス（CFGBUS）、および第 6 章「レジスタ」で説明する様々なキャッシュ・コントロール・レジスタに対するアクセスを提供します。以下の節では、L2 と様々なリクエスター間の相互の影響について説明します。

5.4.1 L1D/L1P-to-L2 リクエストの処理

L2 コントローラは、L1P と L1D が L2 SRAM と L2 キャッシュの両方にアクセスすることを可能にします。L2 はまた、外部メモリのアドレスに対するロング・ディスタンス・アクセス、およびオンチップ・ペリフェラルに対するアクセスを仲介します。各アクセスにおいて、アドレスと L2 モードがその動作を決定します。

L2 SRAM 上にマップされているアドレスに対するメモリ・アクセスは、L2 によって直接処理され、また、L1P と L1D でキャッシュ可能として扱われます。L2 SRAM 以外のオンチップ上のアドレスに対するアクセスは、キャッシュ不可能として処理されます。これらのアクセスには、オンチップ・ペリフェラル、およびキャッシュ・コントロール・レジスタに対するアクセスが含まれています。

L1D による L2 SRAM へのアクセスは、L1P および L2 キャッシュのどちらのキャッシュ動作も引き起こしません。同様に、L2 SRAM に対する L1P アクセスは、L1D および L2 キャッシュのどちらのキャッシュ動作も引き起こしません。これは、8.1 節で概説するコヒーレンス・ポリシーに一致します。これは、5.4.2 項で説明する L2 SRAM への EDMA アクセスと対照的です。

外部アドレスに対する L1P と L1D のメモリ・アクセスは、キャッシュがイネーブルで、キャッシングがそのアドレス範囲で許されている場合、L2 キャッシュにより処理されます。特定の外部アドレス範囲でのキャッシュ可能性は、対応するメモリ・アトリビュート・レジスタ（MAR）にある、キャッシュ・イネーブル・ビット（CE）により決定されます。MAR 動作に関しては、7.2 節で説明します。

L2 が全 SRAM モード（L2MODE = 000b）の場合、L2 は、直接、メモリ・アクセスをキャッシュすることはありません。しかし、L1P および L1D からのリード・リクエストを処理するとき、L2 は取り込まれたデータのキャッシュ可能性のステータスを返します。したがって、L1P と L1D はキャッシュ可能とマークされているメモリのコピーを直接保持することが可能です。外部アドレスに対する L1D からのライトおよびライトバックは、このモードのときは、外部メモリに直接送られます。

キャッシュ不可能な外部ロケーションに対するアクセスは、ロング・ディスタンス・アクセスとして処理されます。ロング・ディスタンス・アクセスについては、5.2 節で説明しています。

注：

L2 は、L2 自身のコントロール・レジスタを含むオンチップ・ペリフェラル・アドレスに対しては、32 ビット・アクセスのみサポートしています。バイト、ハーフワード、およびダブルワード・アクセスでは、期待した動作にならないことがあります。

5.4.2 EDMA-to-L2 リクエストの処理

L2 コントローラは、L2 SRAM に対する EDMA アクセスを処理します。L2 SRAM への EDMA リードとライト・リクエストによって、8.1 節で説明のあるメモリ・コヒーレンス・ポリシーに対応した特定の動作が実行されます。

L2 コントローラは、L2 SRAM のどの部分が L1D キャッシュに保持されているかという情報を追跡します。EDMA が L2 SRAM のロケーションからリードする場合、必要に応じて、EDMA に送るリードを処理する前に、L2 コントローラは L1D の更新データをスヌープします。L1D に更新データがある場合、EDMA リードを開始する前に、その更新を L2 SRAM にライトします。L1D 内のそのラインは、有効のままですが、クリーンされます。L1P に関連する動作は、行われません。

EDMA が L2 SRAM のあるロケーションにライトする場合、必要に応じて、L2 コントローラは L1P にインバリデート・コマンド、および L1D にスヌープ・インバリデート・コマンドを送ります。L2 は L1P の状況を追跡していないので、インバリデート・コマンドは常に L1P に送られます。対応するアドレスが L1P 内にある場合、そのラインはインバリデートされます。L1P は更新データをもつことができないので、L2 は、これ以上 L1P に影響を与えません。L1D に関しては、ライトされるアドレスのコピーを L1D が保持していることを L2 が検出したときだけ、スヌープ・インバリデートが L1D に送られます。L1D に更新データがある場合、L1D から返されるデータよりも新たに入ってくるデータが優先されて、その更新データと EDMA ライトはマージされ、L2 にストアされます。その後、L1D のそのラインは無効とマークされます。

このスヌープとインバリデートの機構によって、L2 SRAM に対する EDMA と CPU のアクセス間でコヒーレンスが保たれます。8.1 節の例は、動作中のこのプロトコルについて詳しく説明しています。

5.4.3 EDMA を使用する L2 リクエストの処理

L2 コントローラは、EDMA を利用してリクエストを処理しています。キャッシュ・ミス、ビクティム・ライトバック、および外部アドレスに対するロング・ディスタンス・アクセスは、L2 コントローラが EDMA コントローラに対して DMA 転送リクエストを出すことで実行されます。

L2 リクエストは他の EDMA リクエストと共にキューに入れられ、EDMA でのポリシーに従って処理されます。L2 リクエストは、いずれの EDMA プライオリティ・キューにも置くことができます。リクエストのプライオリティ・レベル、および許される未処理リクエスト数を、7.4 節で説明があるように制御することが可能です。EDMA については、『TMS320C6000 DSP Enhanced Direct Memory Access (EDMA) Peripheral Reference Guide』（SPRU234）で説明されています。

5.4.4 キャッシュ・コントロールに対する EDMA アクセス

L2 コントローラは、メモリ・マップされたキャッシュ・コントロール・レジスタに対するアクセスを管理します。L2 コントローラは、CPU がコントロール・レジスタにアクセスすることを許しますが、EDMA がどのコントロール・レジスタにアクセスすることも許していません。EDMA がキャッシュ・コントロール・レジスタに対してアクセスすると、そのアクセスは捨てられます。EDMA ライトは無視され、EDMA リードは不定の値を返します。

これらの制限の結果、キャッシュ制御操作は CPU によってのみ起動することが可能です。キャッシュ制御操作のトリガを希望する外部デバイスは、割り込みハンドラまたは、他のメカニズムを使用してリクエスターに代わり適切なレジスタに対する CPU ライトをトリガする必要があります。

5.4.5 メモリ・サブシステムに対する HPI および PCI アクセス

ホスト・ポート・インターフェイス（HPI）およびペリフェラル・コンポーネント・インターコネクト（PCI）ペリフェラルは、2 レベル・メモリ・サブシステムに直接接続されていません。HPI および PCI リクエストは、EDMA が間接的に処理します。したがって、HPI および PCI アクセスは同じコヒーレンス・ポリシーに従い、EDMA アクセスと同じ制約を受けます。

EDMA と 2 レベル・メモリ・システム間の相互の影響については、5.4.2 から 5.4.4 項で説明します。第 8 章「メモリ・システム・ポリシー」では、メモリ・システムのコヒーレンスおよびオーダリング・ポリシーについて説明します。

6 レジスタ

2 レベル・メモリ階層は、表 6 にリストされているメモリ・マップされたコントロール・レジスタにより制御されます。また、CPU コントロール・ステータス・レジスタ（CSR）のデータ・キャッシュ・コントロール（DCC）フィールドおよびプログラム・キャッシュ・コントロール（PCC）フィールドにより制御されます。CSR については、7.1 節を参照してください。各レジスタのメモリ・アドレスについては、各デバイスのデータシートを参照してください。

表 6. 内部メモリ・コントロール・レジスタ

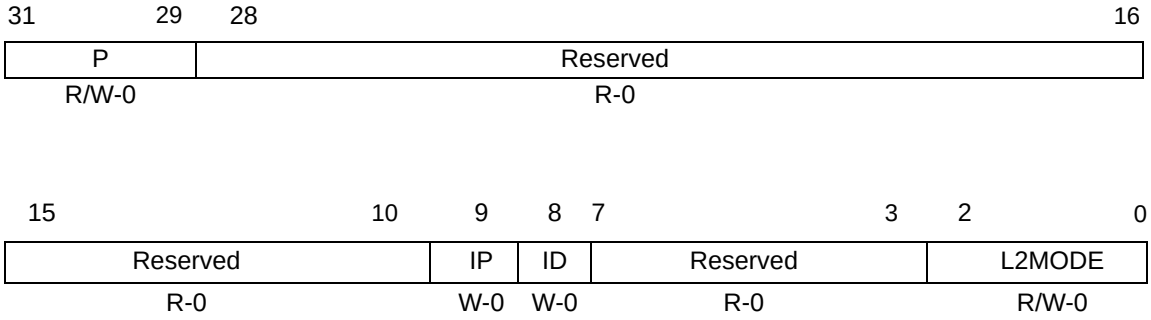
略称	レジスタ名	参照先
CCFG	キャッシュ・コンフィギュレーション・レジスタ	6.1
EDMAWEIGHT	L2 EDMA アクセス・コントロール・レジスタ	6.2
L2ALLOC0-3	L2 アロケーション・レジスタ 0-3	6.3
L2WBAR	L2 ライトバック・ベース・アドレス・レジスタ	6.4
L2WWC	L2 ライトバック・ワード・カウント・レジスタ	6.5
L2WIBAR	L2 ライトバック・インバリデート・ベース・アドレス・レジスタ	6.6
L2WIWC	L2 ライトバック・インバリデート・ワード・カウント・レジスタ	6.7
L2IBAR	L2 インバリデート・ベース・アドレス・レジスタ	6.8
L2IWC	L2 インバリデート・ワード・カウント・レジスタ	6.9
L1PIBAR	L1P インバリデート・ベース・アドレス・レジスタ	6.10
L1PIWC	L1P インバリデート・ワード・カウント・レジスタ	6.11
L1DWIBAR	L1D ライトバック・インバリデート・ベース・アドレス・レジスタ	6.12
L1DWIWC	L1D ライトバック・インバリデート・ワード・カウント・レジスタ	6.13
L1DIBAR	L1D インバリデート・ベース・アドレス・レジスタ	6.14
L1DIWC	L1D インバリデート・ワード・カウント・レジスタ	6.15
L2WB	L2 ライトバック・オール・レジスタ	6.16
L2WBINV	L2 ライトバック・インバリデート・オール・レジスタ	6.17
MAR0-MAR95	予約	-
MAR96-MAR111†	L2 メモリ・アトリビュート・レジスタ 96-111。EMIFB CE0-CE3 を制御します。	6.18
MAR112-MAR127	予約	-
MAR128-MAR191	L2 メモリ・アトリビュート・レジスタ 128-191。EMIFA CE0-CE3 を制御します。	6.18
MAR192-MAR255	予約	-

† MAR96-MAR111 は C6414/C6415/C6416 デバイスでのみ使用可能です。他のすべてのデバイスでは、これらのレジスタは予約されています。

6.1 キャッシュ・コンフィギュレーション・レジスタ（CCFG）

キャッシュ・コンフィギュレーション・レジスタ（CCFG）を図 11 に示し、表 7 で説明します。

図 11. キャッシュ・コンフィギュレーション・レジスタ（CCFG）



凡例：R = リードのみ。W = ライトのみ。R/W = リード / ライト。*-n* = リセット後の値。

表 7. キャッシュ・コンフィギュレーション・レジスタ（CCFG）フィールドの説明

ビット	フィールド	*symval* †	値	説明
31-29	P	OF（値）	0-7h	L2 リクエスターのプライオリティ・ビット
		DEFAULT URGENT	0	L2 コントローラ・リクエストはアージェント・プライオリティ・レベルに置かれます。
		HIGH	1h	L2 コントローラ・リクエストはハイ・プライオリティ・レベルに置かれます。
		MEDIUM	2h	L2 コントローラ・リクエストはミディアム・プライオリティ・レベルに置かれます。
		LOW	3h	L2 コントローラ・リクエストはロウ・プライオリティ・レベルに置かれます。
		-	4h-7h	予約
28-10	Reserved	-	0	予約。予約ビット・ロケーションは常に 0 としてリードされます。このフィールドのビットには常に 0 をライトしてください。
9	IP	OF（値）		インバリデート L1P ビット。
		DEFAULT NORMAL	0	通常の L1P 動作です。
		INVALIDATE	1	すべての L1P ラインをインバリデートします。

† CSL を使って実装する場合、表記 CACHE_CCFG_*field_symval* を使用してください。
‡ 256K よりも小さな L2 メモリをもつ C64x デバイスは、全 L2 キャッシュ・モードをサポートしません。サポートされるキャッシュ・サイズについては、各デバイスのデータシートを参照してください。

表 7.　キャッシュ・コンフィギュレーション・レジスタ（CCFG）フィールドの説明（続き）

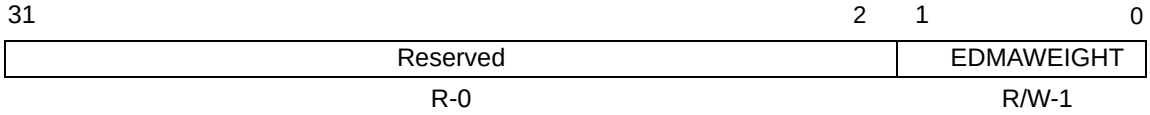
ビット	フィールド	*symval* †	値	説明
8	ID	OF（値）		インバリデート L1D ビット。
		DEFAULT NORMAL	0	通常の L1D 動作です。
		INVALIDATE	1	すべての L1D ラインをインバリデートします。
7-3	Reserved	-	0	予約。予約ビット・ロケーションは常に 0 としてリードされます。このフィールドのビットには常に 0 をライトしてください。
2-0	L2MODE‡	OF（値）	0-7h	L2 動作モード・ビット。使用可能なキャッシュ・モード、および各 L2 キャッシュ・モードによるメモリ・マップ上の L2 のサイズの違いについては、各デバイスのデータシートを参照してください。
		DEFAULT 0KC	0	L2 キャッシュはディスエーブル（全 SRAM モード）
		32KC	1h	4 ウェイ・キャッシュ（32K L2 キャッシュ）
		64KC	2h	4 ウェイ・キャッシュ（64K L2 キャッシュ）
		128KC	3h	4 ウェイ・キャッシュ（128K L2 キャッシュ）
		-	4h-6h	予約
		256KC	7h	4 ウェイ・キャッシュ（256K L2 キャッシュ）。256 K バイト・メモリよりも小さなサイズの L2 キャッシュのデバイスではサポートされません（例、C6410 デバイス）。

† CSL を使って実装する場合、表記 CACHE_CCFG_*field_symval* を使用してください。
‡ 256K よりも小さな L2 メモリをもつ C64x デバイスは、全 L2 キャッシュ・モードをサポートしません。サポートされるキャッシュ・サイズについては、各デバイスのデータシートを参照してください。

6.2 L2 EDMA アクセス・コントロール・レジスタ（EDMAWEIGHT）

L2 EDMA アクセス・コントロール・レジスタ（EDMAWEIGHT）を図 12 に示し、表 8 で説明します。

図 12. L2 EDMA アクセス・コントロール・レジスタ （EDMAWEIGHT）



凡例：R = リードのみ。R/W = リード / ライト。-*n* = リセット後の値。

表 8. L2 EDMA アクセス・コントロール・レジスタ（EDMAWEIGHT）フィールドの説明

ビット	フィールド	値	説明
31-2	Reserved	0	予約。予約ビット・ロケーションは常に 0 としてリードされます。このフィールドにどのような値をライトしても影響ありません。
1-0	EDMAWEIGHT	0-3h	EDMA ウェイトは、L1D が L2 への EDMA アクセスをブロックする時間を制限します。
		0	L1D アクセスは常に L2 への EDMA アクセスよりも優先されます。EDMA が優先されることはありません。
		1h	EDMA は 16 L1D プライオリティ・サイクル後に優先されます。
		2h	EDMA は 4 L1D プライオリティ・サイクル後に優先されます。
		3h	EDMA は 1 L1D プライオリティ・サイクル後に優先されます。

6.3 L2 アロケーション・レジスタ（L2ALLOC0-L2ALLOC03）

L2 アロケーション・レジスタ（L2ALLOC*n*）を図 13 に示し、表 9 で説明します。

図 13. L2 アロケーション・レジスタ（L2ALLOC0-L2ALLOC3）

（a）L2ALLOC0

31	3	2	0
Reserved		Q0CNT	
R-0		R/W-110	

（b）L2ALLOC1

31	3	2	0
Reserved		Q1CNT	
R-0		R/W-010	

（c）L2ALLOC2

31	3	2	0
Reserved		Q2CNT	
R-0		R/W-010	

（d）L2ALLOC3

31	3	2	0
Reserved		Q3CNT	
R-0		R/W-010	

凡例：R = リードのみ。R/W = リード / ライト。- *n* = リセット後の値。

表 9. L2 アロケーション・レジスタ（L2ALLOC0 ～ L2ALLOC3）フィールドの説明

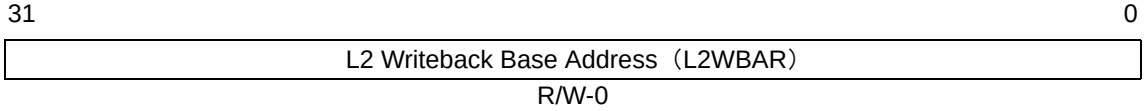
ビット	フィールド	*symval* †	値	説明
31-3	Reserved	—	0	予約。予約ビット・ロケーションは常に 0 としてリードされます。このフィールドのビットには常に 0 をライトしてください。
2-0	Q*n*CNT	OF（値）	0-7h	対応する EDMA プライオリティ・レベルで許される未処理の L2 および QDMA リクエストの合計数。そのプライオリティ・レベルにおけるそれ以上のリクエストは、未処理リクエスト数が Q*n*CNT の設定値より小さくなるまでストールされます。
		DEFAULT	2h	L2ALLOC1、L2ALLOC2、および L2ALLOC3 用。
		DEFAULT	6h	L2ALLOC0 用。

† CSL を使って実装する場合、表記 CACHE_L2ALLOC*n*_field_*symval* を使用してください。

6.4 L2 ライトバック・ベース・アドレス・レジスタ（L2WBAR）

L2 ライトバック・ベース・アドレス・レジスタ（L2WBAR）を図 14 に示し、表 10 で説明します。

図 14. L2 ライトバック・ベース・アドレス・レジスタ（L2WBAR）



凡例：R/W = リード / ライト。-*n* = リセット後の値。

表 10. L2 ライトバック・ベース・アドレス・レジスタ（L2WBAR）フィールドの説明

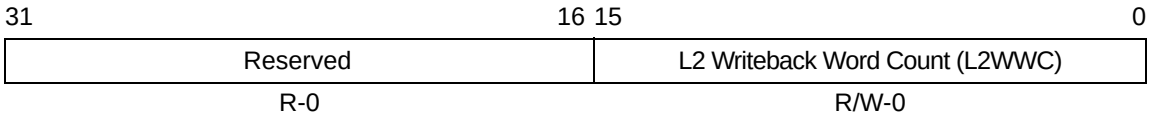
ビット	フィールド	*symval*†	値	説明
31-0	L2WBAR	OF（値）	0-FFFF FFFFh	L2 ライトバック・ベース・アドレス
		DEFAULT	0	

† CSL を使って実装する場合、表記 CACHE_L2WBAR_L2WBAR_*symval* を使用してください。

6.5 L2 ライトバック・ワード・カウント・レジスタ（L2WWC）

L2 ライトバック・ワード・カウント・レジスタ（L2WWC）を図 15 に示し、表 11 で説明します。

図 15. L2 ライトバック・ワード・カウント・レジスタ（L2WWC）



凡例：R = リードのみ。R/W = リード / ライト。- *n* = リセット後の値。

表 11. L2 ライトバック・ワード・カウント・レジスタ（L2WWC）フィールドの説明

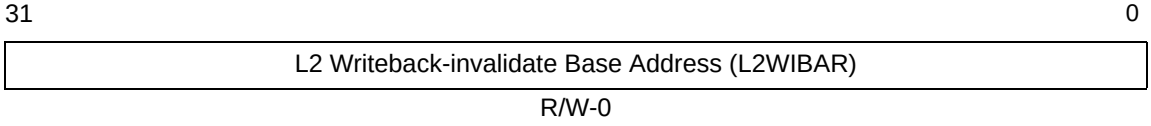
ビット	フィールド	*symval*†	値	説明
31-16	Reserved	-	0	予約。予約ビット・ロケーションは常に 0 としてリードされます。このフィールドのビットには常に 0 をライトしてください。
15-0	L2WWC	OF（値）	0-FFFFh	L2 ライトバック・ワード・カウント
		DEFAULT	0	

† CSL を使って実装する場合、表記 CACHE_L2WWC_L2WWC_*symval* を使用してください。

6.6 L2 ライトバック・インバリデート・ベース・アドレス・レジスタ（L2WIBAR）

L2 ライトバック・インバリデート・ベース・アドレス・レジスタ（L2WIBAR）を図 16 に示し、表 12 で説明します。

図 16. L2 ライトバック・インバリデート・ベース・アドレス・レジスタ（L2WIBAR）



凡例：R/W = リード / ライト。-*n* = リセット後の値。

表 12. L2 ライトバック・インバリデート・ベース・アドレス・レジスタ（L2WIBAR）フィールドの説明

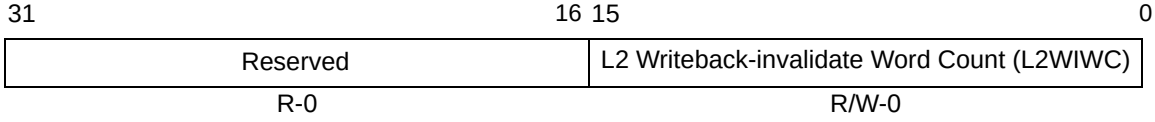
ビット	フィールド	*symval*†	値	説明
31-0	L2WIBAR	OF（値）	0-FFFF FFFFh	L2 ライトバック・インバリデート・ベース・アドレス
		DEFAULT	0	

† CSL を使って実装する場合、表記 CACHE_L2WIBAR_L2WIBAR_*symval* を使用してください。

6.7 L2 ライトバック・インバリデート・ワード・カウント・レジスタ（L2WIWC）

L2 ライトバック・インバリデート・ワード・カウント・レジスタ（L2WIWC）を図 17 に示し、表 13 で説明します。

図 17. L2 ライトバック・インバリデート・ワード・カウント・レジスタ（L2WIWC）



凡例：R = リードのみ。R/W = リード / ライト。-*n* = リセット後の値。

表 13. L2 ライトバック・インバリデート・ワード・カウント・レジスタ（L2WIWC）フィールドの説明

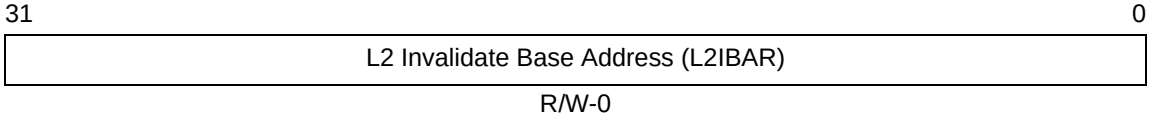
ビット	フィールド	*symval*†	値	説明
31-16	Reserved	-	0	予約。予約ビット・ロケーションは常に 0 としてリードされます。このフィールドのビットには常に 0 をライトしてください。
15-0	L2WIWC	OF（値）	0-FFFFh	L2 ライトバック・インバリデート・ワード・カウント
		DEFAULT	0	

† CSL を使って実装する場合、表記 CACHE_L2WIWC_L2WIWC_*symval* を使用してください。

6.8 L2 インバリデート・ベース・アドレス・レジスタ（L2IBAR）

L2 インバリデート・ベース・アドレス・レジスタ（L2IBAR）を図 18 に示し、表 12 で説明します。

図 18. L2 インバリデート・ベース・アドレス・レジスタ（L2IBAR）



凡例：R/W = リード / ライト。-*n* = リセット後の値。

表 14. L2 インバリデート・ベース・アドレス・レジスタ（L2IBAR）フィールドの説明

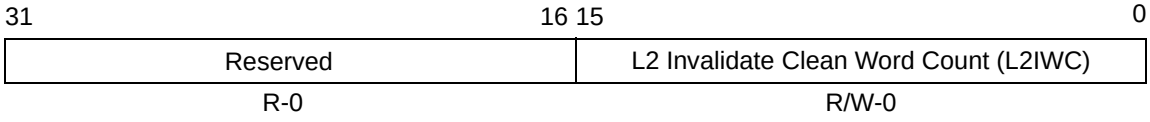
ビット	フィールド	*symval*†	値	説明
31-0	L2IBAR	OF（値）	0-FFFF FFFFh	L2 インバリデート・ベース・アドレス
		DEFAULT	0	

† CSL を使って実装する場合、表記 CACHE_L2IBAR_L2IBAR_*symval* を使用してください。

6.9 L2 インバリデート・ワード・カウント・レジスタ（L2IWC）

L2 インバリデート・ワード・カウント・レジスタ（L2IWC）を図 19 に示し、表 15 で説明します。

図 19. L2 インバリデート・ワード・カウント・レジスタ（L2IWC）



凡例：R = リードのみ。R/W = リード / ライト。-*n* = リセット後の値。

表 15. L2 インバリデート・ワード・カウント・レジスタ（L2IWC）フィールドの説明

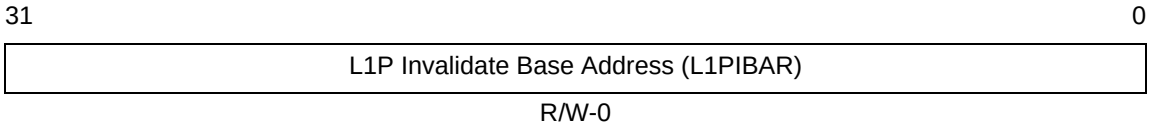
ビット	フィールド	*symval*†	値	説明
31-16	Reserved	-	0	予約。予約ビット・ロケーションは常に 0 としてリードされます。このフィールドのビットには常に 0 をライトしてください。
15-0	L2IWC	OF（値）	0-FFFFh	L2 インバリデート・ワード・カウント。
		DEFAULT	0	

† CSL を使って実装する場合、表記 CACHE_L2IWC_L2IWC_*symval* を使用してください。

6.10 L1P インバリデート・ベース・アドレス・レジスタ（L1PIBAR）

L1P インバリデート・ベース・アドレス・レジスタ（L1PIBAR）を図 20 に示し、表 16 で説明します。

図 20. L1P インバリデート・ベース・アドレス・レジスタ（L1PIBAR）



凡例：R/W = リード / ライト。*-n* = リセット後の値。

表 16. L1P インバリデート・ベース・アドレス・レジスタ（L1PIBAR）フィールドの説明

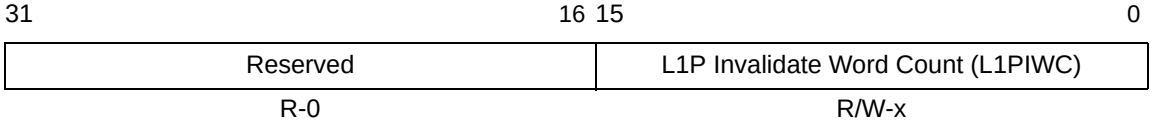
ビット	フィールド	*symval* †	値	説明
31-0	L1PIBAR	OF（値）	0-FFFF FFFFh	L1P インバリデート・ベース・アドレス
		DEFAULT	0	

† CSL を使って実装する場合、表記 CACHE_L1PIBAR_L1PIBAR_*symval* を使用してください。

6.11 L1P インバリデート・ワード・カウント・レジスタ（L1PIWC）

L1P インバリデート・ワード・カウント・レジスタ（L1PIWC）を図 21 に示し、表 17 で説明します。

図 21. L1P インバリデート・ワード・カウント・レジスタ（L1PIWC）



凡例：R = リードのみ。R/W = リード / ライト。*-n* = リセット後の値。

表 17. L1P インバリデート・ワード・カウント・レジスタ（L1PIWC）フィールドの説明

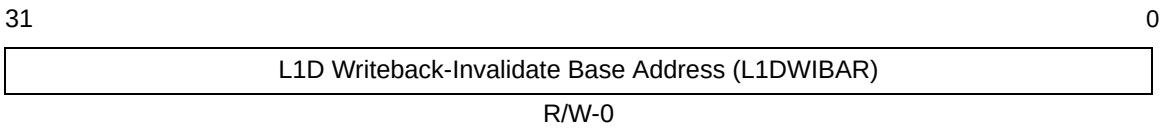
ビット	フィールド	*symval* †	値	説明
31-16	Reserved	-	0	予約。予約ビット・ロケーションは常に 0 としてリードされます。このフィールドのビットには常に 0 をライトししてください。
15-0	L1PIWC	OF（値）	0-FFFFh	L1P インバリデート・ワード・カウント
		DEFAULT	0	

† CSL を使って実装する場合、表記 CACHE_L1PIWC_L1PIWC_*symval* を使用してください。

6.12 L1D ライトバック・インバリデート・ベース・アドレス・レジスタ（L1DWIBAR）

L1D ライトバック・インバリデート・ベース・アドレス・レジスタ（L1DWIBAR）を図 22 に示し、表 18 で説明します。

図 22. L1D ライトバック・インバリデート・ベース・アドレス・レジスタ（L1DWIBAR）



凡例：R/W = リード / ライト。*-n* = リセット後の値。

表 18. L1D ライトバック・インバリデート・ベース・アドレス・レジスタ（L1DWIBAR）フィールドの説明

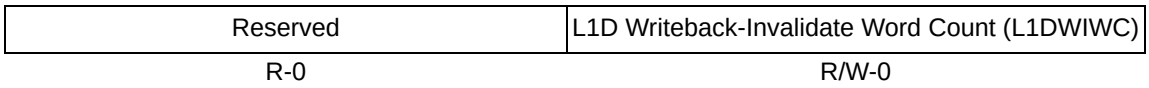
ビット	フィールド	*symval*†	値	説明
31-0	L1DWIBAR	OF（値）	0-FFFF FFFFh	L1D ライトバック・インバリデート・ベース・アドレス。
		DEFAULT	0	

† CSL を使って実装する場合、表記 CACHE_L1DWIBAR_L1DWIBAR_*symval* を使用してください。

6.13 L1D ライトバック・インバリデート・ワード・カウント・レジスタ（L1DWIWC）

L1D ライトバック・インバリデート・ワード・カウント・レジスタ（L1DWIWC）を図 23 に示し、表 19 で説明します。

図 23. L1D ライトバック・インバリデート・ワード・カウント・レジスタ （L1DWIWC）



凡例：R = リードのみ。R/W = リード / ライト。*-n* = リセット後の値。

表 19. L1D ライトバック・インバリデート・ワード・カウント・レジスタ（L1DWIWC）フィールドの説明

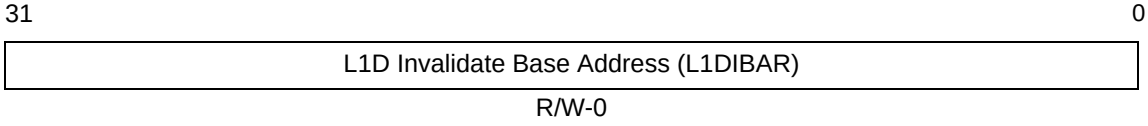
ビット	フィールド	*symval*†	値	説明
31-16	Reserved	-	0	予約。予約ビット・ロケーションは常に 0 としてリードされます。このフィールドのビットには常に 0 をライトしてください。
15-0	L1DWIWC	OF（値）	0-FFFFh	L1D ライトバック・インバリデート・ワード・カウント。
		DEFAULT	0	

† CSL を使って実装する場合、表記 CACHE_L1DWIWC_L1DWIWC_*symval* を使用してください。

6.14 L1D インバリデート・ベース・アドレス・レジスタ（L1DIBAR）

L1D インバリデート・ベース・アドレス・レジスタ（L1DIBAR）を図 24 に示し、表 20 で説明します。

図 24. L1D インバリデート・ベース・アドレス・レジスタ（L1DIBAR）



凡例：R/W = リード / ライト。-*n* = リセット後の値。

表 20. L1D インバリデート・ベース・アドレス・レジスタ（L1DIBAR）フィールドの説明

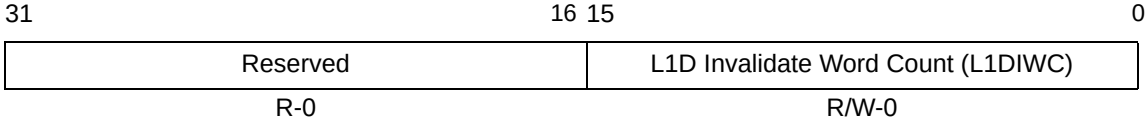
ビット	フィールド	*symval*†	値	説明
31-0	L1DIBAR	OF（値）	0-FFFF FFFFh	L1D インバリデート・ベース・アドレス。
		DEFAULT	0	

† CSL を使って実装する場合、表記 CACHE_L1DIBAR_L1DIBAR_*symval* を使用してください。

6.15 L1D インバリデート・ワード・カウント・レジスタ（L1DIWC）

L1D インバリデート・ワード・カウント・レジスタ（L1DIWC）を図 25 に示し、表 21 で説明します。

図 25. L1D インバリデート・ワード・カウント・レジスタ（L1DIWC）



凡例：R = リードのみ。R/W = リード / ライト。-*n* = リセット後の値。

表 21. L1D インバリデート・ワード・カウント・レジスタ（L1DIWC）フィールドの説明

ビット	フィールド	*symval*†	値	説明
31-16	Reserved	-	0	予約。予約ビット・ロケーションは常に 0 としてリードされます。このフィールドのビットには常に 0 をライトしてください。
15-0	L1DIWC	OF（値）	0-FFFFh	L1D インバリデート・ワード・カウント。
		DEFAULT	0	

† CSL を使って実装する場合、表記 CACHE_L1DIWC_L1DIWC_*symval* を使用してください。

6.16 L2 ライトバック・オール・レジスタ（L2WB）

L2 ライトバック・オール・レジスタ（L2WB）を図 26 に示し、表 22 で説明します。

図 26. L2 ライトバック・オール・レジスタ（L2WB）

31	1	0
Reserved		C
R-0		R/W-0

凡例：R = リードのみ。R/W = リード / ライト。-*n* = リセット後の値。

表 22. L2 ライトバック・オール・レジスタ（L2WB）フィールドの説明

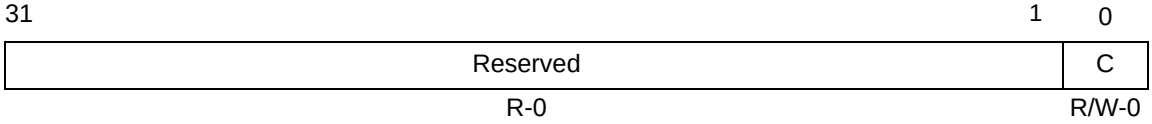
ビット	フィールド	*symval*†	値	説明
31-1	Reserved	-	0	予約。予約ビット・ロケーションは常に 0 としてリードされます。このフィールドのビットには常に 0 をライトしてください。
0	C	OF（値）		L2 ライトバック・オール。
		DEFAULT NORMAL	0	通常の L2 動作
		FLUSH	1	すべての L2 ラインをライトバックする

† CSL を使って実装する場合、表記 CACHE_L2WB_C_*symval* を使用してください。

6.17 L2 ライトバック・インバリデート・オール・レジスタ（L2WBINV）

L2 ライトバック・インバリデート・オール・レジスタ（L2WBINV）を図 27 に示し、表 23 で説明します。

図 27. L2 ライトバック・インバリデート・オール・レジスタ（L2WBINV）



凡例：R = リードのみ。R/W = リード / ライト。-*n* = リセット後の値。

表 23. L2 ライトバック・インバリデート・オール・レジスタ（L2WBINV）フィールドの説明

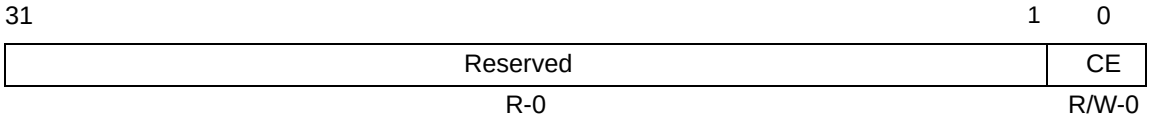
ビット	フィールド	*symval*†	値	説明
31-1	Reserved	-	0	予約。予約ビット・ロケーションは常に 0 としてリードされます。このフィールドのビットには常に 0 をライトしてください。
0	C	OF（値）		L2 ライトバック・インバリデート・オール。
		DEFAULT NORMAL	0	通常の L2 動作
		CLEAN	1	すべての L2 ラインをライトバック・インバリデートする

† CSL を使って実装する場合、表記 CACHE_L2WBINV_C_*symval* を使用してください。

6.18 L2 メモリ・アトリビュート・レジスタ（MAR0-MAR255）

L2 メモリ・アトリビュート・レジスタ（MAR）を図 28 に示し、表 24 で説明します。各 MAR は 16 M バイトのアドレス範囲についてキャッシュ可能性を制御します。7.2 節を参照してください。

図 28. L2 メモリ・アトリビュート・レジスタ（MAR）



凡例：R = リードのみ。R/W = リード / ライト。-*n* = リセット後の値。

表 24. メモリ・アトリビュート・レジスタ（MAR）フィールドの説明

ビット	フィールド	*symval*†	値	説明
31-1	Reserved	-	0	予約。予約ビット・ロケーションは常に 0 としてリードされます。このフィールドのビットには常に 0 をライトしてください。
0	CE	OF（値）		キャッシュ・イネーブル・ビットは L1D、L1P、および L2 が対応するアドレス範囲をキャッシュするかどうか決定します。
		DEFAULT DISABLE	0	メモリ範囲はキャッシュ可能ではありません。
		ENABLE	1	メモリ範囲はキャッシュ可能です。

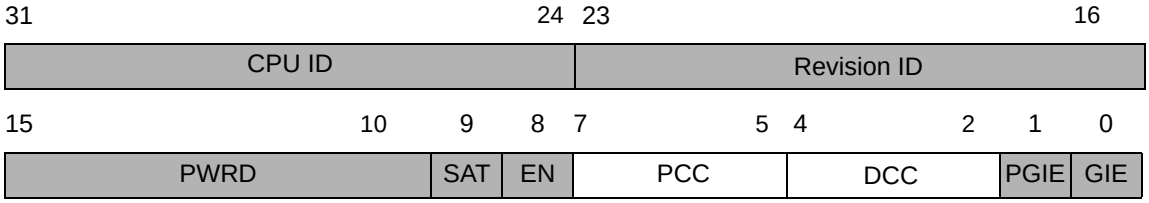
† CSL を使って実装する場合、表記 CACHE_MAR_CE_*symval* を使用してください。

7 メモリ・システム・コントロール

7.1 キャッシュ・モードの選択

2 レベル・メモリ階層のキャッシュ・モードは、キャッシュ・コンフィギュレーション・レジスタ、および CPU コントロール・ステータス・レジスタ（CSR）のデータ・キャッシュ・コントロール（DCC）フィールドとプログラム・キャッシュ・コントロール（PCC）フィールドにより決定されます。図 29 に CSR を示します。

図 29. CPU コントロール・ステータス・レジスタ（CSR）



7.1.1 CSR の DCC フィールドを使用した L1D モード選択

L1D は、キャッシュとしてのみ動作します。メモリにマップすることはできません。L1D は、フリーズおよびバイパス・モードをサポートしていません。データ・キャッシュ・コントロール（DCC）フィールドに許される値は、000b と 010b だけです。表 25 に示すように、他の DCC 値はすべて予約されています。

表 25. DCC フィールドを使用した L1D モード設定

ビット	フィールド	値	説明
4-2	DCC		データ・キャッシュ・コントロール・ビット。
		000	2 ウェイ・キャッシュをイネーブル。
		001	予約
		010	2 ウェイ・キャッシュをイネーブル。
		011-111	予約

7.1.2 CSR の PCC フィールドを使用した L1P モード選択

L1P は、キャッシュとしてのみ動作します。メモリにマップすることはできません。L1P は、フリーズおよびバイパス・モードをサポートしていません。プログラム・キャッシュ・コントロール（PCC）フィールドに許される値は、000b と 010b だけです。表 26 に示すように、他の PCC 値はすべて予約されています。

表 26. PCC フィールドを使用した L1P モード設定

ビット	フィールド	値	説明
7-5	PCC		プログラム・キャッシュ・コントロール・ビット。
		000	ダイレクトマップ・キャッシュをイネーブル。
		001	予約
		010	ダイレクトマップ・キャッシュをイネーブル。
		011-111	予約

7.1.3 CCFG の L2MODE フィールドを使用した L2 モード選択

L2 メモリは、マップされた SRAM、キャッシュ、あるいは両方の組み合わせで機能させることが可能です。キャッシュ・コンフィギュレーション・レジスタ（CCFG）の L2MODE フィールドは、L2 のどの部分が SRAM としてマップされ、どの部分がキャッシュとして機能するか決定します。L2MODE の設定値には、L2 SRAM または L2 キャッシュのどちらかが存在しないものがあります。L2 キャッシュのないモードは、全 SRAM モードといいます。CCFG は、図 11（39 ページ）および表 7 で説明しています。

L2MODE フィールドのリセット値は、000b です。したがって、L2 はリセット後、全 SRAM モードに設定されます。すなわち、L2 キャッシュはイネーブルではなく、L2 SRAM のアドレス範囲全体が使用可能です。L2 は、このモードにおいても L1P および L1D からの外部メモリ・リクエストを処理します。しかし、データそのもののコピーは保持しません。

デフォルトでは、外部メモリはキャッシュ可能になっていません。キャッシュ可能か否かは、キャッシュ・モードとは別に制御されます。外部メモリのキャッシュ可能性を制御する、メモリ・アトリビュート・レジスタ（MAR）については、7.2 節で説明します。

L2 キャッシュをイネーブルすると、その分、使用可能な L2 SRAM の容量は減ります。L2 SRAM アドレス空間の上位アドレスは、キャッシュに変わります。たとえば、C6414 DSP の場合、0000 0000h-000F FFFFh のアドレスに 1024K バイトの L2 メモリがあります。L2 が 256K キャッシュ・モード（L2MODE = 111b）のとき、アドレスの最上位 256K バイト範囲（000C 0000h-000F FFFFh）は、プログラムから使用できません。

注：

L2 キャッシュとして設定されている L2 アドレス範囲に対するリードおよびライトは、キャッシュ階層の不正操作となります。プログラムの 正しい動作を保証するには、L2 SRAM としてマップされている L2 アドレスのみに L2 アクセスを行うよう、プログラムで限定しなければなりません。

L2 コントローラは、アドレス範囲が 0000 0000h-0010 0FFFh であれは、たとえその中のアドレスが L2 キャッシュとして設定されていても、不正キャッシュ操作を起こさずにリードを処理できます。このアドレス範囲は、割り当てられている全 L2 アドレス範囲およびプログラムの最適化に使用できる追加の空間を意味します。したがって、前述の制約は、リードには適用されません。しかし、プログラムはリードの戻り値を解釈すべきではありません。また、プログラムは L2 SRAM として設定されていない L2 アドレスに対して、ライトを実行すべきではありません。

L2 キャッシュ・モードを切り換えたときに正しい動作が保証されるには、一連の操作を実行する必要があります。SRAM としてマップされる L2 メモリを追加、あるいは削除するときに必要な操作について、表 27 に示します。表のガイドラインに従わない場合、データ損失や予期しない L2 の動作が起こることがあります。

表 27.　L2 モード切り換え手順

切り換え前	切り換え後	以下の手順を実行します
全 SRAM モード	L2 キャッシュ付きモード	1）キャッシュに変換される L2 SRAM 空間から、EDMA を使用して必要なデータを転送する。 2）L2 キャッシュになる L2 SRAM アドレスに対して L1D でブロック・ライトバック・インバリデートを実行する。 3）ブロック・ライトバック・インバリデートが完了するのを待つ。 4）L1D 内にキャッシュされている可能性のあるすべての外部アドレス範囲に対してブロック・ライトバック・インバリデートを実行する（いずれの MAR でも CE ビットが 1 にセットされていない場合、この手順は必要ありません）。 5）手順 4 のブロック・ライトバック・インバリデートが完了するのを待つ。 6）CCFG にライトしてモード変更する。 7）CCFG をリードして、CCFG 変更の間、CPU を待機させる。 8）NOP を 8 サイクル実行する。
L2 SRAM と L2 キャッシュが混在するモード	マップされる L2SRAM が少なくなるモード	1）キャッシュに変換される L2 SRAM 空間から、EDMA を使用して必要なデータを転送する。 2）L2 キャッシュになる L2 SRAM アドレスに対して L1D でブロック・ライトバック・インバリデートを実行する。 3）ブロック・ライトバック・インバリデートが完了するのを待つ。 4）L2 のグローバル・ライトバック・インバリデート（L2WBINV）を実行する。 5）L2WBINV が完了するのを待つ。 6）CCFG にライトしてモード変更する。 7）CCFG をリードして、CCFG 変更の間、CPU を待機させる。 8）NOP を 8 サイクル実行する。
すべての L2 モード	マップされる L2SRAM が多くなるモード	1）L2 のグローバル・ライトバック・インバリデート（L2WBINV）を実行する。 2）L2WBINV が完了するのを待つ。 3）CCFG にライトしてモード変更する。 4）CCFG をリードして、CCFG 変更の間、CPU を待機させる。 5）NOP を 8 サイクル実行する。

7.2 キャッシュ可能性の制御

外部アドレス範囲のキャッシュ可能か否かは、メモリ・アトリビュート・レジスタ（MAR0-MA255）によって制御されます。各 MAR は表 28 にリストされているように、16 M バイト・アドレス範囲のキャッシュ可能性を制御します。MAR を図 28（51 ページ）に示し、表 24 で説明しています。各レジスタのメモリ・アドレスについては、各デバイスのデータシートを参照してください。

各 MAR のキャッシュ・イネーブル（CE）ビットは、L1D、L1P、および L2 が対応するアドレス範囲をキャッシュ可能にするかどうかを決定します。リセット後は、各 MAR の CE ビットは、ゼロクリアされます。そのため、外部メモリのキャッシングは、デフォルトではディスエーブルされています。この動作は、常にキャッシュ可能と見なされる L2 SRAM とは対照的です。

特定の外部アドレス範囲でキャッシングをイネーブルするには、アプリケーションはその範囲に対応する MAR の CE ビットを 1 に設定する必要があります。特別な手順は必要ありません。以降のそのアドレス範囲へのアクセスは、2 レベル・メモリ・システムによりキャッシュされます。

あるアドレス範囲のキャッシングをディスエーブルするには、以降のそのアドレス範囲に対するアクセスがキャッシュされないことを保証するために、プログラムは以下の手順に従わなければなりません。

1） この範囲内にあるすべてのアドレスを、L1 および L2 キャッシュから削除します。これは、次の方法の 1 つで実行できます。次の方法のうち、どちらか 1 つで十分です。

   a） L2 キャッシュがイネーブルされている場合、L2WBINV を使用してグローバル・ライトバック・インバリデートを起動する。L2WBINV の C ビットが 0 になるのを待つ。または、L2WIBAR/L2WIWC を使用して、この範囲に対するブロック・ライトバック・インバリデートを起動する。L2WIWC が 0 になるのを待つ。

   b） L2 が全 SRAM モードの場合、L1DWIBAR/L1DWIWC を使用してこの範囲に対するブロック・ライトバック・インバリデートを起動する。L1WIWC が 0 になるのを待つ。

   ブロック指向のキャッシュ制御は一度に 256K バイトのアドレス範囲でのみ操作が可能であることに注意してください。したがって、キャッシュからこのアドレス範囲全体を削除するために、複数のブロック・ライトバック・インバリデート操作が必要となることがあります。キャッシュ制御の詳細は、7.3 節で説明します。

2） 対応する MAR の CE ビットをゼロクリアします。

表 28. メモリ・アトリビュート・レジスタ

略称	レジスタ名
MAR0-MAR95	予約
MAR96†	L2 メモリ・アトリビュート・レジスタ 96：EMIFB CE0 範囲 6000 0000h-60FF FFFFh を制御します。
MAR97†	L2 メモリ・アトリビュート・レジスタ 97：EMIFB CE0 範囲 6100 0000h-61FF FFFFh を制御します。
MAR98†	L2 メモリ・アトリビュート・レジスタ 98：EMIFB CE0 範囲 6200 0000h-62FF FFFFh を制御します。
MAR99†	L2 メモリ・アトリビュート・レジスタ 99：EMIFB CE0 範囲 6300 0000h-63FF FFFFh を制御します。
MAR100†	L2 メモリ・アトリビュート・レジスタ 100：EMIFB CE1 範囲 6400 0000h-64FF FFFFh を制御します。
MAR101†	L2 メモリ・アトリビュート・レジスタ 101：EMIFB CE1 範囲 6500 0000h-65FF FFFFh を制御します。
MAR102†	L2 メモリ・アトリビュート・レジスタ 102：EMIFB CE1 範囲 6600 0000h-66FF FFFFh を制御します。
MAR103†	L2 メモリ・アトリビュート・レジスタ 103：EMIFB CE1 範囲 6700 0000h-67FF FFFFh を制御します。
MAR104†	L2 メモリ・アトリビュート・レジスタ 104：EMIFB CE2 範囲 6800 0000h-68FF FFFFh を制御します。
MAR105†	L2 メモリ・アトリビュート・レジスタ 105：EMIFB CE2 範囲 6900 0000h-69FF FFFFh を制御します。
MAR106†	L2 メモリ・アトリビュート・レジスタ 106：EMIFB CE2 範囲 6A00 0000h-6AFF FFFFh を制御します。
MAR107†	L2 メモリ・アトリビュート・レジスタ 107：EMIFB CE2 範囲 6B00 0000h-6BFF FFFFh を制御します。
MAR108†	L2 メモリ・アトリビュート・レジスタ 108：EMIFB CE3 範囲 6C00 0000h-6CFF FFFFh を制御します。
MAR109†	L2 メモリ・アトリビュート・レジスタ 109：EMIFB CE3 範囲 6D00 0000h-6DFF FFFFh を制御します。
MAR110†	L2 メモリ・アトリビュート・レジスタ 110：EMIFB CE3 範囲 6E00 0000h-6EFF FFFFh を制御します。
MAR111†	L2 メモリ・アトリビュート・レジスタ 111：EMIFB CE3 範囲 6F00 0000h-6FFF FFFFh を制御します。
MAR112-MAR127	予約
MAR128	L2 メモリ・アトリビュート・レジスタ 128：EMIFA CE0 範囲 8000 0000h-80FF FFFFh を制御します。
MAR129	L2 メモリ・アトリビュート・レジスタ 129：EMIFA CE0 範囲 8100 0000h-81FF FFFFh を制御します。
MAR130	L2 メモリ・アトリビュート・レジスタ 130：EMIFA CE0 範囲 8200 0000h-82FF FFFFh を制御します。
MAR131	L2 メモリ・アトリビュート・レジスタ 131：EMIFA CE0 範囲 8300 0000h-83FF FFFFh を制御します。
MAR132	L2 メモリ・アトリビュート・レジスタ 132：EMIFA CE0 範囲 8400 0000h-84FF FFFFh を制御します。
MAR133	L2 メモリ・アトリビュート・レジスタ 133：EMIFA CE0 範囲 8500 0000h-85FF FFFFh を制御します。
MAR134	L2 メモリ・アトリビュート・レジスタ 134：EMIFA CE0 範囲 8600 0000h-86FF FFFFh を制御します。
MAR135	L2 メモリ・アトリビュート・レジスタ 135：EMIFA CE0 範囲 8700 0000h-87FF FFFFh を制御します。
MAR136	L2 メモリ・アトリビュート・レジスタ 136：EMIFA CE0 範囲 8800 0000h-88FF FFFFh を制御します。

† MAR96-MAR111 は、C6414/C6415/C6416 のデバイスでのみ使用可能です。他のすべてのデバイスでは、これらのレジスタは予約されています。

表 28.　メモリ・アトリビュート・レジスタ（続き）

略称	レジスタ名
MAR137	L2 メモリ・アトリビュート・レジスタ 137：EMIFA CE0 範囲 8900 0000h-89FF FFFFh を制御します。
MAR138	L2 メモリ・アトリビュート・レジスタ 138：EMIFA CE0 範囲 8A00 0000h-8AFF FFFFh を制御します。
MAR139	L2 メモリ・アトリビュート・レジスタ 139：EMIFA CE0 範囲 8B00 0000h-8BFF FFFFh を制御します。
MAR140	L2 メモリ・アトリビュート・レジスタ 140：EMIFA CE0 範囲 8C00 0000h-8CFF FFFFh を制御します。
MAR141	L2 メモリ・アトリビュート・レジスタ 141：EMIFA CE0 範囲 8D00 0000h-8DFF FFFFh を制御します。
MAR142	L2 メモリ・アトリビュート・レジスタ 142：EMIFA CE0 範囲 8E00 0000h-8EFF FFFFh を制御します。
MAR143	L2 メモリ・アトリビュート・レジスタ 143：EMIFA CE0 範囲 8F00 0000h-8FFF FFFFh を制御します。
MAR144	L2 メモリ・アトリビュート・レジスタ 144：EMIFA CE1 範囲 9000 0000h-90FF FFFFh を制御します。
MAR145	L2 メモリ・アトリビュート・レジスタ 145：EMIFA CE1 範囲 9100 0000h-91FF FFFFh を制御します。
MAR146	L2 メモリ・アトリビュート・レジスタ 146：EMIFA CE1 範囲 9200 0000h-92FF FFFFh を制御します。
MAR147	L2 メモリ・アトリビュート・レジスタ 147：EMIFA CE1 範囲 9300 0000h-93FF FFFFh を制御します。
MAR148	L2 メモリ・アトリビュート・レジスタ 148：EMIFA CE1 範囲 9400 0000h-94FF FFFFh を制御します。
MAR149	L2 メモリ・アトリビュート・レジスタ 149：EMIFA CE1 範囲 9500 0000h-95FF FFFFh を制御します。
MAR150	L2 メモリ・アトリビュート・レジスタ 150：EMIFA CE1 範囲 9600 0000h-96FF FFFFh を制御します。
MAR151	L2 メモリ・アトリビュート・レジスタ 151：EMIFA CE1 範囲 9700 0000h-97FF FFFFh を制御します。
MAR152	L2 メモリ・アトリビュート・レジスタ 152：EMIFA CE1 範囲 9800 0000h-98FF FFFFh を制御します。
MAR153	L2 メモリ・アトリビュート・レジスタ 153：EMIFA CE1 範囲 9900 0000h-99FF FFFFh を制御します。
MAR154	L2 メモリ・アトリビュート・レジスタ 154：EMIFA CE1 範囲 9A00 0000h-9AFF FFFFh を制御します。
MAR155	L2 メモリ・アトリビュート・レジスタ 155：EMIFA CE1 範囲 9B00 0000h-9BFF FFFFh を制御します。
MAR156	L2 メモリ・アトリビュート・レジスタ 156：EMIFA CE1 範囲 9C00 0000h-9CFF FFFFh を制御します。
MAR157	L2 メモリ・アトリビュート・レジスタ 157：EMIFA CE1 範囲 9D00 0000h-9DFF FFFFh を制御します。
MAR158	L2 メモリ・アトリビュート・レジスタ 158：EMIFA CE1 範囲 9E00 0000h-9EFF FFFFh を制御します。
MAR159	L2 メモリ・アトリビュート・レジスタ 159：EMIFA CE1 範囲 9F00 0000h-9FFF FFFFh を制御します。
MAR160	L2 メモリ・アトリビュート・レジスタ 160：EMIFA CE2 範囲 A000 0000h-A0FF FFFFh を制御します。
MAR161	L2 メモリ・アトリビュート・レジスタ 161：EMIFA CE2 範囲 A100 0000h-A1FF FFFFh を制御します。
MAR162	L2 メモリ・アトリビュート・レジスタ 162：EMIFA CE2 範囲 A200 0000h-A2FF FFFFh を制御します。
MAR163	L2 メモリ・アトリビュート・レジスタ 163：EMIFA CE2 範囲 A300 0000h-A3FF FFFFh を制御します。
MAR164	L2 メモリ・アトリビュート・レジスタ 164：EMIFA CE2 範囲 A400 0000h-A4FF FFFFh を制御します。
MAR165	L2 メモリ・アトリビュート・レジスタ 165：EMIFA CE2 範囲 A500 0000h-A5FF FFFFh を制御します。

† MAR96-MAR111 は、C6414/C6415/C6416 のデバイスでのみ使用可能です。他のすべてのデバイスでは、これらのレジスタは予約されています。

表 28. メモリ・アトリビュート・レジスタ（続き）

略称	レジスタ名
MAR166	L2 メモリ・アトリビュート・レジスタ 166：EMIFA CE2 範囲 A600 0000h-A6FF FFFFh を制御します。
MAR167	L2 メモリ・アトリビュート・レジスタ 167：EMIFA CE2 範囲 A700 0000h-A7FF FFFFh を制御します。
MAR168	L2 メモリ・アトリビュート・レジスタ 168：EMIFA CE2 範囲 A800 0000h-A8FF FFFFh を制御します。
MAR169	L2 メモリ・アトリビュート・レジスタ 169：EMIFA CE2 範囲 A900 0000h-A9FF FFFFh を制御します。
MAR170	L2 メモリ・アトリビュート・レジスタ 170：EMIFA CE2 範囲 AA00 0000h-AAFF FFFFh を制御します。
MAR171	L2 メモリ・アトリビュート・レジスタ 171：EMIFA CE2 範囲 AB00 0000h-ABFF FFFFh を制御します。
MAR172	L2 メモリ・アトリビュート・レジスタ 172：EMIFA CE2 範囲 AC00 0000h-ACFF FFFFh を制御します。
MAR173	L2 メモリ・アトリビュート・レジスタ 173：EMIFA CE2 範囲 AD00 0000h-ADFF FFFFh を制御します。
MAR174	L2 メモリ・アトリビュート・レジスタ 174：EMIFA CE2 範囲 AE00 0000h-AEFF FFFFh を制御します。
MAR175	L2 メモリ・アトリビュート・レジスタ 175：EMIFA CE2 範囲 AF00 0000h-AFFF FFFFh を制御します。
MAR176	L2 メモリ・アトリビュート・レジスタ 176：EMIFA CE3 範囲 B000 0000h-B0FF FFFFh を制御します。
MAR177	L2 メモリ・アトリビュート・レジスタ 177：EMIFA CE3 範囲 B100 0000h-B1FF FFFFh を制御します。
MAR178	L2 メモリ・アトリビュート・レジスタ 178：EMIFA CE3 範囲 B200 0000h-B2FF FFFFh を制御します。
MAR179	L2 メモリ・アトリビュート・レジスタ 179：EMIFA CE3 範囲 B300 0000h-B3FF FFFFh を制御します。
MAR180	L2 メモリ・アトリビュート・レジスタ 180：EMIFA CE3 範囲 B400 0000h-B4FF FFFFh を制御します。
MAR181	L2 メモリ・アトリビュート・レジスタ 181：EMIFA CE3 範囲 B500 0000h-B5FF FFFFh を制御します。
MAR182	L2 メモリ・アトリビュート・レジスタ 182：EMIFA CE3 範囲 B600 0000h-B6FF FFFFh を制御します。
MAR183	L2 メモリ・アトリビュート・レジスタ 183：EMIFA CE3 範囲 B700 0000h-B7FF FFFFh を制御します。
MAR184	L2 メモリ・アトリビュート・レジスタ 184：EMIFA CE3 範囲 B800 0000h-B8FF FFFFh を制御します。
MAR185	L2 メモリ・アトリビュート・レジスタ 185：EMIFA CE3 範囲 B900 0000h-B9FF FFFFh を制御します。
MAR186	L2 メモリ・アトリビュート・レジスタ 186：EMIFA CE3 範囲 BA00 0000h-BAFF FFFFh を制御します。
MAR187	L2 メモリ・アトリビュート・レジスタ 187：EMIFA CE3 範囲 BB00 0000h-BBFF FFFFh を制御します。
MAR188	L2 メモリ・アトリビュート・レジスタ 188：EMIFA CE3 範囲 BC00 0000h-BCFF FFFFh を制御します。
MAR189	L2 メモリ・アトリビュート・レジスタ 189：EMIFA CE3 範囲 BD00 0000h-BDFF FFFFh を制御します。
MAR190	L2 メモリ・アトリビュート・レジスタ 190：EMIFA CE3 範囲 BE00 0000h-BEFF FFFFh を制御します。
MAR191	L2 メモリ・アトリビュート・レジスタ 191：EMIFA CE3 範囲 BF00 0000h-BFFF FFFFh を制御します。
MAR192-MAR255	予約

† MAR96-MAR111 は、C6414/C6415/C6416 のデバイスでのみ使用可能です。他のすべてのデバイスでは、これらのレジスタは予約されています。

7.3 プログラム起動によるキャッシュ操作

メモリ・システムは、キャッシュ制御操作を備えています。これにより、プログラムは特定のデータをライトバック、またはインバリデートすることをリクエストできます。キャッシュ操作は、次の 2 種類に分けられます。キャッシュ全体で動作するグローバル操作と、指定したアドレス範囲で動作するブロック操作です。グローバル操作については 7.3.1 項で、またブロック操作については 7.3.2 項で説明します。

メモリ・システムはプログラム起動によるキャッシュ操作を一度に 1 つだけ実行できます。これには、グローバル操作、ブロック操作、およびモード変更を含みます。この理由でメモリ・システムは、キャッシュ制御操作が進行中はキャッシュ・コントロール・レジスタに対するアクセスをストールさせることがあります。表 29 に可能な操作とメモリ・システムでの影響の概要を示します。

表 29. プログラム起動によるキャッシュ操作の概要

操作タイプ	キャッシュ操作	使用するレジスタ	L1P キャッシュへの影響	L1D キャッシュへの影響	L2 キャッシュへの影響
グローバル操作	L2 ライトバック・オール	L2WB	影響なし。	L2 キャッシュにも保持されているアドレスの更新ラインはライトバックされます。L2 キャッシュに保持されているアドレスと対応するすべてのラインがインバリデートされます。†	更新データがライトバックされます。すべてのラインは有効に保たれます。
	L2 ライトバック・インバリデート・オール	L2WBINV	すべての内容が破棄されます。†	L2 キャッシュにも保持されているアドレスの更新ラインはライトバックされます。L2 キャッシュに保持されているアドレスと対応するすべてのラインがインバリデートされます。†	更新ラインはライトバックされます。すべてのラインはインバリデートされます。
	L1P インバリデート・オール・	CCFG 内の IP ビット	すべての内容が破棄されます。	影響なし。	影響なし。
	L1D インバリデート・オール・	CCFG 内の ID ビット	影響なし。	すべての内容が破棄されます。更新データはライトバックされません。	影響なし。

† 7.3.3 項で説明されているように、これらの操作は L2 キャッシュがイネーブルされているときのみ、L1D と L1P に影響します。L2 キャッシュがイネーブルされているとき、これらの操作は L2 キャッシュにも保持される部分の L1P と L1D が影響します。将来の C6000 デバイスは、同一アドレスが L2 にキャッシュされるかどうか、あるいは L2 キャッシュがイネーブルされているかどうかに関係なく、これらの操作が、L1P と L1D のすべての内容に作用するようになる可能性があります。

‡ C621x/C671x デバイスの動作と対照的に、C64x デバイスの L1DWIBAR/L1DWIWC は、対応するブロックが L1P 内でインバリデートされることはありません。

表 29.　プログラム起動によるキャッシュ操作の概要（続き）

操作タイプ	キャッシュ操作	使用するレジスタ	L1P キャッシュへの影響	L1D キャッシュへの影響	L2 キャッシュへの影響
ブロック操作	L2 ブロック・ライトバック	L2WBAR、L2WWC	影響なし。	ブロック内の更新ラインは、L2 にライトされます。ブロック内のすべてのラインは、L1D でインバリデートされます。†	ブロック内の更新ラインは、外部メモリにライトされます。ブロック内のラインは、L2 で有効に保たれます。
	L2 ブロック・ライトバック・インバリデート	L2WIBAR、L2WIWC	ブロック内のすべてのラインがインバリデートされます。†	ブロック内の更新ラインは、L2 にライトされます。ブロック内のすべてのラインは、L1D でインバリデートされます。†	ブロック内の更新ラインは、外部メモリにライトされます。ブロック内のすべてのラインは、L2 でインバリデートされます。
	L2 ブロック・インバリデート	L2IBAR、L2IWC	ブロック内のすべてすべてのラインがインバリデートされます。†	ブロック内のすべてのラインは、L1D でインバリデートされます。ブロック内の更新データは、破棄されます。	ブロック内のすべてのラインは、L2 でインバリデートされます。ブロック内の更新データは、破棄されます。
	L1P ブロック・インバリデート	L1PIBAR、L1PIWC	ブロック内のすべてのラインがインバリデートされます。	影響なし。	影響なし。
	L1D ブロック・ライトバック・インバリデート	L1DWIBAR、L1DWIWC	影響なし。‡	ブロック内の更新ラインは、L2 にライトされます。ブロック内のすべてのラインは、L1D でインバリデートされます。	影響なし。
	L1D ブロック・インバリデート	L1DIBAR、L1DIWC	影響なし。	ブロック内のすべてのラインは、L1D でインバリデートされます。ブロック内の更新データは、破棄されます。	影響なし。

† 7.3.3 項で説明されているように、これらの操作は L2 キャッシュがイネーブルされているときのみ、L1D と L1P に影響します。L2 キャッシュがイネーブルされているとき、これらの操作は L2 キャッシュにも保持される部分の L1P と L1D が影響します。将来の C6000 デバイスは、同一アドレスが L2 にキャッシュされるかどうか、あるいは L2 キャッシュがイネーブルされているかどうかに関係なく、これらの操作が、L1P と L1D のすべての内容に作用するようになる可能性があります。

‡ C621x/C671x デバイスの動作と対照的に、C64x デバイスの L1DWIBAR/L1DWIWC は、対応するブロックが L1P 内でインバリデートされることはありません。

7.3.1 グローバル・キャッシュ操作

グローバル・キャッシュ操作は、キャッシュ内の内容全体に対して実行されます。グローバル操作は、キャッシュに対するプログラムからのアクセスよりも優先されます。あるキャッシュに対するプログラムからのアクセス(データまたはプログラム・フェッチのどちらも）は、キャッシュでグローバル・キャッシュ操作がアクティブになっている間、ストールします。

L1D および L1P へのグローバル操作は、CCFG の ID および IP ビット（図 11）で起動されます。L1D および L1P では、グローバル・インバリデートのみが提供されています。CCFG の ID または IP ビットに 1 をライトすることで、プログラムは対応する L1 キャッシュの全体の内容をインバリデートできます。グローバル・インバリデートが開始されると、対応するキャッシュの全体の内容が破棄されます。更新データは、ライトバックされません。ライト後の CCFG のリードは、インバリデート操作が完了するまでプログラムをストールさせます。

L1D グローバル・インバリデートは、L1D 内の更新データを、下位レベルのメモリにライトバックせずに、そのデータをすべて破棄してしまいます。そのため、更新データを下位レベルのメモリにライトされることを期待しているプログラムでは、誤った動作の原因となります。したがって、ほとんどのプログラムでは、L1D グローバル・インバリデートを使用するのではなく、L1D ブロック・ライトバック・インバリデート（7.3.2 項を参照）か、グローバル L2 操作を使用します。

L2 は、グローバル・ライトバックとグローバル・ライトバック・インバリデート操作の両方を提供します。L2 のグローバル・キャッシュ操作は、L2WB (図 26（49 ページ）および、表 22 を参照）および L2WBINV（図 27（50 ページ）および表 23 を参照）のどちらかの C ビットに 1 をライトすることにより起動されます。L2WB の C ビットに 1 をライトすると、L2 のグローバル・ライトバックを起動し、L2WBINV の C ビットに 1 をライトすると L2 のグローバル・ライトバック・インバリデートを起動します。キャッシュ操作が完了するまで、C ビットは 1 としてリードされます。プログラムはポーリングすることで、キャッシュ操作がいつ完了するのかを判別することができます。

7.3.3 項で説明するように、L2 のグローバル操作は L1 キャッシュに間接的な影響を与えます。

7.3.2 ブロック・キャッシュ操作

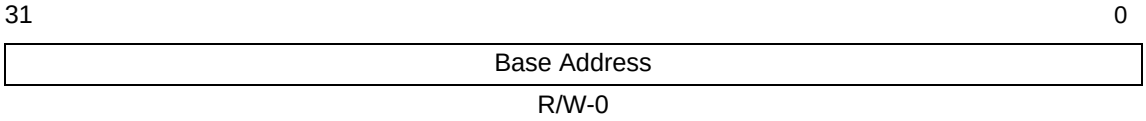
ブロック・キャッシュ操作は、キャッシュ内で保持されているアドレス範囲に対して実行します。ブロック操作は、バックグランドで実行されるので、他のプログラムによるアクセスとブロック・キャッシュ操作が交互に行われます。

プログラムは、2 回のライトでブロック・キャッシュ操作を起動します。プログラムは最初にベース・アドレス・レジスタ（BAR）に先頭アドレスをライトします（図 30 を参照）。次にプログラムは対応するワード・カウント・レジスタに、操作する分のワードの合計数をライトします（図 31 を参照）。ワード・カウント・レジスタにゼロではない値がライトされると、直ちにキャッシュ操作が始まります。キャッシュは BAR/WC 擬似レジスタ・ペアのセットを、そのキャッシュがサポートする各ブロック操作に対して 1 セット提供します。サポートされるキャッシュ操作の完全なリストを表 29 に示します。

WC のワード・カウント・フィールドは、16 ビット幅しかないことに注意してください。ブロック・サイズは 65535 ワード（おおよそ 256K バイト）までに制限されます。制限範囲を越える場合には、複数のブロック・コマンドを発行する必要があります。

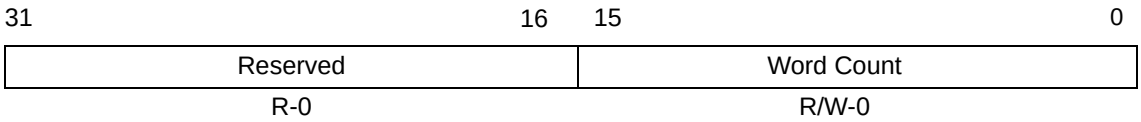
ブロック操作はワード・アドレスとワード・カウントでそのブロックを指定しますが、ブロック操作は常にキャッシュ・ライン全体に対して動作します。各々の影響を受けるキャッシュについて、常にライン全体がライトバックおよびインバリデートされます。そのため、プログラムは、キャッシュ・ライン境界に配列をアラインさせること、およびキャッシュ・ライン・サイズの倍数になるように配列にパッドを入れることに注意を払う必要があります。これらの配列に関してインバリデートのみのコマンドを起動するときは、特に注意が必要です。

図 30. ブロック・キャッシュ・オペレーション・ベース・アドレス・レジスタ（BAR）



凡例：R/W = リード / ライト。*-n* = リセット後の値

図 31. ブロック・キャッシュ・オペレーション・ワード・カウント・レジスタ（WC）



凡例：R = リードのみ。R/W = リード / ライト。*-n* = リセット後の値。

現行のデバイスの実装では、ブロック・コマンド・レジスタ（BAR と WC）は、ハードウェア内で単一のレジスタ・ペアとして実装されています。ブロック・コマンド・レジスタは、アドレス・マップ内の複数アドレスで擬似レジスタとして見えます。したがって、プログラムは、異なるコマンド・レジスタ・ペアと交互にライト「しないで」ください。BAR/WC ペアへのライトは、他の BAR/WC ライトに対してアトミックに実行されなくてはなりません。プログラムによっては、BAR と WC にライトする間、割り込みが起きないようにする必要があります。

この制限にもかかわらず、プログラムは前のコマンドの実行中に、新しいブロック・コマンドの起動を試みることもできます。キャッシュ・コントローラは、前のキャッシュ操作が完了するまで BAR に対するライトをストールさせます。そのため、正しい動作が維持されます。ストールを避けるために、プログラムは WC をポーリングして、ブロック・キャッシュ操作が完了したかどうかを判別することができます。キャッシュ操作が進行中、WC は、非ゼロの値を、操作が完了するとゼロを返します。返される非ゼロの値は、デバイスの実装により変化します。したがって、プログラムはレジスタが非ゼロかどうかだけを確認してください。

プログラムでは、BAR の値が、ブロック・キャッシュ操作の間で保持されていると仮定するべきではありません。各キャッシュ操作に対して、プログラムは、常に BAR にアドレス、そして WC にワード・カウントをライトしなくてはなりません。またプログラムでは、複数の BAR と WC が同一の物理レジスタにマップされているとは限りません。各キャッシュ操作では、プログラムはその操作用の BAR と WC にライトしなくてはなりません。

LIP ブロック・インバリデートを L1D ブロック・ライトバック・インバリデートと連動して使用すると、L1D と LIP 間でソフトウェア制御によってコヒーレンスを維持することができます。メモリ・システムのコヒーレンス・ポリシーについては、8.1 節で説明します。あらかじめ CPU でライトされたコードを実行するには、プログラムは、L1P 内のブロックをインバリデートするために L1PIBAR/L1PIWC を、また L1D 内のブロックをライトバック・インバリデートするために L1DWIBAR/L1DWIWC を使用する必要があります。これらの操作はどちらの順序でも実行できます。プログラム・フェッチに対してこれらの操作に関するタイミングは、明確に定義されていません。したがって、この方法でインバリデートされたアドレス範囲にブランチする前に、プログラムは L1DWIWC と L1PIWC でゼロがリードされるまで待機しなくてはなりません。C621x/C671x デバイスでは、L1DIBAR/L1DIWC の動作が異なることに注意してください。詳細については、『TMS320C621x/C671x DSP Two-Level Internal Memory Reference Guide』（SPRU609）を参照してください。

表 29 で説明しているように、L2 のブロック・キャッシュ操作は間接的に L1D および L1P に影響を与えます。ここでの相互の影響の詳細については、7.3.3 項で説明します。

注：

ブロック・キャッシュ操作が進行中のブロック内のアドレスに対するリードまたはライトは、リクエストどおりのライトバックやインバリデートが実行されないことがあります。この事態を避けるには、プログラムは、進行中のブロック・キャッシュ操作の影響を受けるキャッシュ・ラインの範囲内にあるアドレスにアクセスすべきでありません。プログラムは、適切な WC を参照することにより、ブロック操作の完了を確認できます。

7.3.3 L1 キャッシュへの L2 コマンドの影響

注：

この節で説明する動作は、2 レベル・メモリ・サブシステムを実装する将来の C6000 デバイスでは、変更される可能性があります。将来のデバイスでは、現行の L2 キャッシュ・モードに関わらず、L2 での L1D への影響が取り除かれる可能性もありますし、L2 キャッシュ操作は L1 で実行される可能性もあります。上位互換性をもたせるプログラムには、前述のメモリ・システム内の特定の動作に依存性をもたせるべきではありません。

L2 のキャッシュ操作は、間接的に L1D と L1P で動作します。結果として L2 が全 SRAM モードの場合、L2 キャッシュ操作はいずれのキャッシュに対しても影響を与えません。これはグローバルおよびブロック・コマンドの両方に適用されます。一方、L2 キャッシュがイネーブルされている場合、プログラム起動による L2 でのキャッシュ操作は L1D と L1P の対応する内容に対して動作します。

通常の状況のもとでは、L1D キャッシュは L2 に「包含」され、L1P は包含されません。「包含」されているとは、L1D のすべての内容が L2 SRAM または L2 キャッシュのどちらかに保持されていることを意味します。L2 キャッシュ操作は、これらの特性を考慮して設計されています。

L1P は L2 に「包含」されていないため、L2 のラインをインバリデートする L2 キャッシュ操作は、L1P に対して明示的なインバリデート・コマンドを送ります。L2 のグローバル・ライトバック・インバリデート（L2WBINV）は、L1P を完全にインバリデートするトリガになります。L2 でのブロック・インバリデートおよびライトバック・インバリデート操作は、対応する L1P キャッシュ・ラインへのインバリデート・コマンドを何も考慮せずに送ります。これにより、キャッシュ操作完了後に、L1P が常にメモリの最も新しい内容をフェッチすることを保証します。

通常 L1D は、L2 に包含されているので、L2 は通常のキャッシュ・プロトコル動作によって L1D が間接的に動作することに依存しています。L2 が L1D も対応するアドレスのコピーを保持していることを検出すると、L2 内のラインに対してトリガされるライトバックおよびインバリデートは、結果として、L1D に送られるスヌープ・インバリデート・コマンドを発生させます。したがって、L2 グローバル・ライトバック（L2WB）および L2 グローバル・ライトバック・インバリデート（L2WBINV）は、L1D に保持されているすべての外部アドレスが、L1D からライトバックおよびインバリデートされる原因になります。同様に、L2 でのブロック操作は、間接的なスヌープ・インバリデートによって L1D の対応する範囲を L2 にライトさせ、L1D でインバリデートさせる原因にもなります。

結果として、L2 キャッシュは L2 SRAM のコピーを保持することはないので、L1D 内にキャッシュされる L2 SRAM アドレスが、L2 へのプログラム起動によるキャッシュ操作の影響を受けることはありません。L1D から L2 SRAM アドレスを除去するには、プログラムは L1D ブロック・キャッシュ操作を直接使用しなくてはなりません。通常、L1D から L2 SRAM アドレスを直接削除することは、L2 キャッシュ・モードを変更するときにのみ必要とされます。8.1 節で説明しているコヒーレンス・ポリシーにより、プログラムが L1D の一部分を L2 SRAM に手動でライトバックする必要性はほとんどありません。

もう 1 つの結果として、L1D が L2 に包含されていないとき、非直感的な動作が生じます。通常の状況のもとでは、L1D は L2 に包含されています。したがって、ほとんどのプログラムはこの状況に対して注意を払う必要はありません。実際に、7.1.3 項にある推奨される L2 キャッシュ・モードの変更手順は、メモリ・システムがこの状況にならないことを保証しています。ここでの手順に厳密に「従わない」とき、L2 では保持されない外部メモリのコピーを L1D が保持することが可能です。L1D が L2 に包含されていない状況は、以下のシーケンスで発生します。

- ❏ L2 が全 SRAM モードにある間に、プログラムが外部アドレス範囲のキャッシングをイネーブルした。
- ❏ プログラムが、この外部アドレス範囲から直接リードした。
- ❏ プログラムが、まずブロック・キャッシュ操作を使用して L1D からの外部アドレス範囲の除去をしないで、L2 キャッシュをイネーブルした。

この状況を避けるには、プログラムは、L2 キャッシュをイネーブルするとき、外部アドレスが L1D にキャッシュされないことを保証しなくてはなりません。L2 キャッシュをイネーブルする前に外部アドレスのキャッシングをイネーブルしないか、または 7.3.2 項で説明するブロック・キャッシュ操作を使用して外部アドレスを L1D から除去することで、これを実現できます。

7.4 L2-to-EDMA リクエストの制御

5.4.3 項での説明のように、L2 コントローラは、リクエストを処理するのに EDMA を利用しています。L2 リクエストは、4 つの EDMA プライオリティ・キューのいずれにも置くことができます。キャッシュ・サービス・リクエストに対するプライオリティ・キューの選択は、CCFG の P フィールド（図 11（39 ページ）を参照）を使用して行われます。

キャッシュの適切な動作を保証するためには、プログラムは、L2 リクエストのプライオリティティ・レベルを変更するときに注意を払わなくてはなりません。以下の順序に、従ってください。

1） EDMA プライオリティ・キュー・ステータス・レジスタ（PQSR）をポーリングして、現行のプライオリティ・レベルに対応する PQ ビットが 1 になるのを待つ。PQSR は『TMS320C6000 DSP Enhanced Direct Memory Access (EDMA) Peripheral Reference Guide』（SPRU234）で説明されています。

   このステップでは、外部トリガによる EDMA 転送のような、同一のプライオリティ・キューを使用する他の転送をディスエーブルすることが必要になるかもしれません。そうしないと、負荷の重いシステムでは、長時間 PQSR の PQ ビットが 1 にならないことがあります。

2） 新しい値を CCFG 内の P フィールドにライトする。

3） CCFG をリードして、ライトが完了したことを確認する。

システム内で L2 リクエストと他の EDMA リクエストの間の公平性を保つために、L2 は 4 つのプライオリティ・キュー・アロケーション・レジスタを備えています。それらは、L2ALLOC0、L2ALLOC1、L2ALLOC2 および L2ALLOC3（図 13（42 ページ）を参照）です。これらのレジスタは、EDMA に発行される L2 リクエストの割合を制御します。L2 はまた、QDMA コントロール・レジスタを持っており、L2ALLOC0-3 によって設定される制限は、QDMA リクエストにも適用されます。QDMA については、『TMS320C6000 DSP Enhanced Direct Memory Access (EDMA) Peripheral Reference Guide』（SPRU234）で説明されています。

L2ALLOC の設定は、対応する EDMA プライオリティ・レベルで許される未処理の L2 および QDMA リクエストの合計数を決定します。そのプライオリティ・レベルにおけるそれ以上のリクエストは、未処理リクエスト数が L2ALLOC の設定値より小さくなるまでストールされます。ここでの未処理の転送とは、所定のプライオリティ・キューに発行したが、そのキューから EDMA チャネル・レジスタに取り出されていない転送のことです。L2ALLOC0-3 は、『TMS320C6000 DSP Enhanced Direct Memory Access (EDMA) Peripheral Reference Guide』（SPRU234）で説明している、プライオリティ・キュー・アロケーション・レジスタ（PQAR）と同様に動作します。

表 30 では、デフォルトのキュー・アロケーションをリストしています。L2ALLOC 設定値ではまた、転送を損失しないように、PQAR の現行の設定値を考慮に入れる必要があります。L2ALLOC および PQAR の設定値を変更する正しい手順は、『TMS320C6000 DSP Enhanced Direct Memory Access (EDMA) Peripheral Reference Guide』（SPRU234）で説明されています。

表 30.　L2ALLOC のデフォルトのキュー・アロケーション

プライオリティ・レベル	アロケーション・レジスタ	デフォルトの L2/QDMA アロケーション
アージェント	L2ALLOC0	6
ハイ	L2ALLOC1	2
ミディアム	L2ALLOC2	2
ロウ	L2ALLOC3	2

7.5　L2 への EDMA アクセスの制御

C64x DSP には、L2 キャッシュ・レジスタのメモリ・マップ内に、L2 EDMA アクセス・コントロール・レジスタ（EDMAWEIGHT）が組み込まれています。これは、L2 に対する EDMA と L1D アクセスについてプライオリティの相対的な重み付けを制御します。EDMAWEIGHT は、L1D が L2 に対する EDMA アクセスをブロックする時間の長さを制限することで、EDMA アクセスのプライオリティを一時的に引き上げます。このプライオリティ引き上げは、L1D 内ではミスするが、L2 キャッシュまたは L2 SRAM でヒットする CPU からのライト・データと競合するときにのみ適用されます。通常のライン・アロケーションおよび追い出し動作では、L1D-to-L2 のアクセス・パターン内に EDMA アクセスが使用可能なギャップが生まれます。EDMAWEIGHT を使用すると、このプライオリティ引き上げを発生させる回数を制御できます。EDMA のプライオリティが上がると、プライオリティが CPU データ側に戻される前に、1 回のアクセスを完了することが可能です。EDMAWEIGHT については、図 12（41 ページ）および表 8 で説明しています。

L1D が、*n* サイクル連続して L2 をブロックすると、EDMA は 1 サイクルだけプライオリティが上げられます。

8 メモリ・システム・ポリシー

この章では、メモリ・システムに関する各種ポリシーについて説明します。説明する内容は、CPU と EDMA またはホスト・アクセス間のコヒーレンス、およびメモリ更新が行われる順序などです。

8.1 メモリ・システムのコヒーレンス

キャッシュ・メモリは、メモリ階層の下位レベルからのデータのコピーを、上位レベルで保持することで機能します。これにより、メモリ階層の上位レベルは下位レベルよりも高速にアクセスすることが可能で、またアクセス毎に下位レベルを参照する必要がないので、パフォーマンス上有利になります。メモリに対するアクセスの多くは階層の上位のレベルで取り込まれるので、CPU や他のデバイスは、メモリの内容を異なった形態で参照することができます。

メモリ・システムがコヒーレントであるとは、メモリへのすべてのリクエスターから、個々のメモリのロケーションに対する更新が同じ順序で行われていると観測できることです。リクエスターとは、CPU または EDMA などのデバイスです。コヒーレントなメモリ・システムは、あるメモリ・ロケーションに対するすべてのライトが、どのリクエスターからも、そのロケーションを上書きするライトがない限り、そのロケーションのそれ以降のリードに対して参照可能なことを保証します。同一のリクエスターがライトおよびリードする場合、ライトの結果は直ちに参照可能になります。あるリクエスターがライトして、別のリクエスターがリードする場合、そのリード側は直ちに更新された値を参照することができないかもしれません。しかし、十分な時間が経過した後では更新された値を参照することは可能です。コヒーレンスはまた、あるメモリ・ロケーションに対するすべてのライトが順次処理されることを暗示しています。すなわち、複数のリクエスターが 1 つのロケーションにライトするときでも、すべてのリクエスターがメモリのロケーションに対するライトを同じ順番で参照することができます。

2 レベル・キャッシュ・メモリ・システムは、種々の条件下でコヒーレントです。C64x デバイスは、プログラマに対し制限付きでコヒーレンスを保証しています。この制限付きコヒーレンス・モデルは、リーズナブルなプログラミング・モデルを提供しつつ、設計を単純化し、かつハードウェアのパフォーマンスを向上させます。

メモリ・システムのコヒーレンスは、キャッシュ可能なデータにのみ適用されます。キャッシュ不可能なデータでは、メモリ・システムはアクセスを単にその最終の宛先に、いかなる中間的なコピーも保持せずに渡すだけです。この場合は、コヒーレンスは問題になりません。したがって、ここでの説明は、キャシュ可能なメモリだけに絞っています。一般にキャッシュ可能なメモリは、RAM や ROM の各種形式に限定されます。

C64x メモリ・システムのコヒーレンス・モデルは、リクエスターおよびメモリの場所の観点から説明されます。2 レベル・メモリ・システムは、3 つのソースからのリクエストをサポートします。ソースとは、CPU プログラム・フェッチ、CPU データ・アクセス、そして EDMA アクセスです（2 レベル・メモリ・システムへの HPI および PCI アクセスは、EDMA を使用して処理されます）。ここで話題にしているキャッシュ可能なエリアには、内部および外部 RAM があります。ROM は一般的にライト可能でないため、RAM についてのみ説明します。表 31 では、メモリ・システムがコヒーレンシを保証する場合について示しています。

表 31.　2 レベル・メモリ・システムにおけるコヒーレンス保証

リクエスター	CPU プログラム・フェッチ	CPU データ・アクセス	EDMA アクセス
CPU プログラム・フェッチ	コヒーレント	ソフトウェア管理による、L2 レベルのみ	オンチップ SRAM のみ
CPU データ・アクセス	ソフトウェア管理による、L2 レベルのみ	コヒーレント	オンチップ SRAM のみ
EDMA アクセス	オンチップ SRAM のみ	オンチップ SRAM のみ	コヒーレント

ハードウェアは、CPU と EDMA による内部 SRAM アドレスに対するアクセスがコヒーレントであることを保証しますが、外部アドレスについては保証しません。EDMA がアクセスする外部アドレスは、EDMA がアクセスするときキャッシュに保持されないように、ソフトウェアが保証しなくてはなりません。ソフトウェアがそのような保証をしていない場合、影響範囲内のアドレスでデータ破壊が発生することがあります。アドレスの特定範囲がキャッシュに保持されないことを保証するための必要な手順については、7.3 節を参照してください。

また、CPU データとプログラム・アクセスとがコヒーレントでないことも重要です。両者がコヒーレントでない理由は、L1P によるプログラム・フェッチ時に L1D に照会を行わないので、L1D 内で取り込まれる CPU ライトが長時間、L1P から参照できないからです。したがって、後で実行されるロケーションにライトするプログラムは、少なくとも L2 に対して、L1D からの更新データが実行前にライトされることを保証しなくてはなりません。7.3 節で説明している L1DWIBAR/L1DWIWC および L1PIBAR/L1PIWC は、この目的で使用することができます。

たとえ、メモリ・システムがコヒーレントであるとしても、キャッシュ可能なメモリに対する更新を、すべてのリクエスターが同じ順序で観測することを保証するものではありません。これが重要になるのは、メモリ内のバッファが CPU および EDMA またはホスト・ポート・アクセスによりアクセスされるときです。注意を払わない限り、CPU 更新がメモリ・システムにより処理されている間にアクセスが発生する場合、EDMA または ホスト・ポート・アクセスは新旧データを混在して参照することがあります。CPU アクセスがメモリ・システムで観測される順序については、8.3 節で説明します。

8.2 L2 SRAM における EDMA コヒーレンスの例

8.1 節で説明しているように、メモリ・システムは、L2 SRAM に対する CPU および EDMA アクセス間のコヒーレンスを保証します。外部メモリに対する CPU および EDMA アクセス間のコヒーレンスは提供されていません。ここでは、ハードウェアがどのようにして L2 SRAM のコヒーレンスの保証を実現しているか、例を示して説明します。

DSP の一般的なプログラミング技法では、ダブル・バッファリングを使用してデータ・ストリームを処理します。この手法では、デバイスの外部にある大規模データ・セットは、小さな塊にして転送され、その塊に対して演算することで処理されます。その後、その結果は外部メモリに転送されます。これらの転送は、通常、EDMA コントローラによって実行されます。擬似コードを使って典型的な処理の順序を、図 32 に示します。

EDMA は、CPU と独立して動作するため、EDMA 転送とデータ処理はオーバーラップさせることが可能です。ダブル・バッファリングでは、入出力バッファのセットを 2 つもたせることでオーバーラップを可能にしています。1 つは処理用、1 つは転送用です。CPU と EDMA はそれらの間でバッファのセットをピンポンのように交互に使用します。そのため、ダブル・バッファリングは、しばしばピンポン・バッファリングと言われます。擬似コードでは、ダブル・バッファによる処理の順序は、図 33 に示されています。ここで「ピン」と「ポン」は、ダブル・バッファに対するポインタの組みです。「ピン」は CPU が処理するバッファを指し、「ポン」が EDMA が書き込むバッファを指します。

内部入力および内部出力バッファのタイム・シーケンスは、図示すると、図 34 のようになります。EDMA リードは、処理の一段前に、EDMA ライトはそれから一段後に行われます。図 34 では、ステップ 2 はステップ 3 と 4 と同時に、そして、ステップ 5 と 6 はステップ 7 と 8 にオーバーラップして動作します。

図 35 では、ソフトウェア内のリード / 処理 / ライトのパイプラインを、処理とオーバーラップする EDMA 転送とともに示しています。

図 32.　　ストリーミング・データの擬似コード

```
while（処理するデータが残っている）
{
   次のブロックを内部入力バッファにリード

   内部入力バッファを処理、結果を内部出力バッファに出力

   内部出力バッファを外部出力へライト
}
```

図 33.　　ダブル・バッファリングの擬似コード

```
はじめのブロックを内部「ピン」入力バッファにリード
while（処理するデータが残っている）
{
   次のブロックの内部「ポン」入力バッファへのリード開始

   「ピン」入力および出力バッファの準備完了まで待機

   内部「ピン」入力バッファを処理、結果を内部「ピン」出力バッファに出力

   「ピン」および「ポン」バッファへのポインタを交換

   「ポン」入力バッファからの書き出しを開始
}
```

図 34.　ダブル・バッファリングのタイム・シーケンス

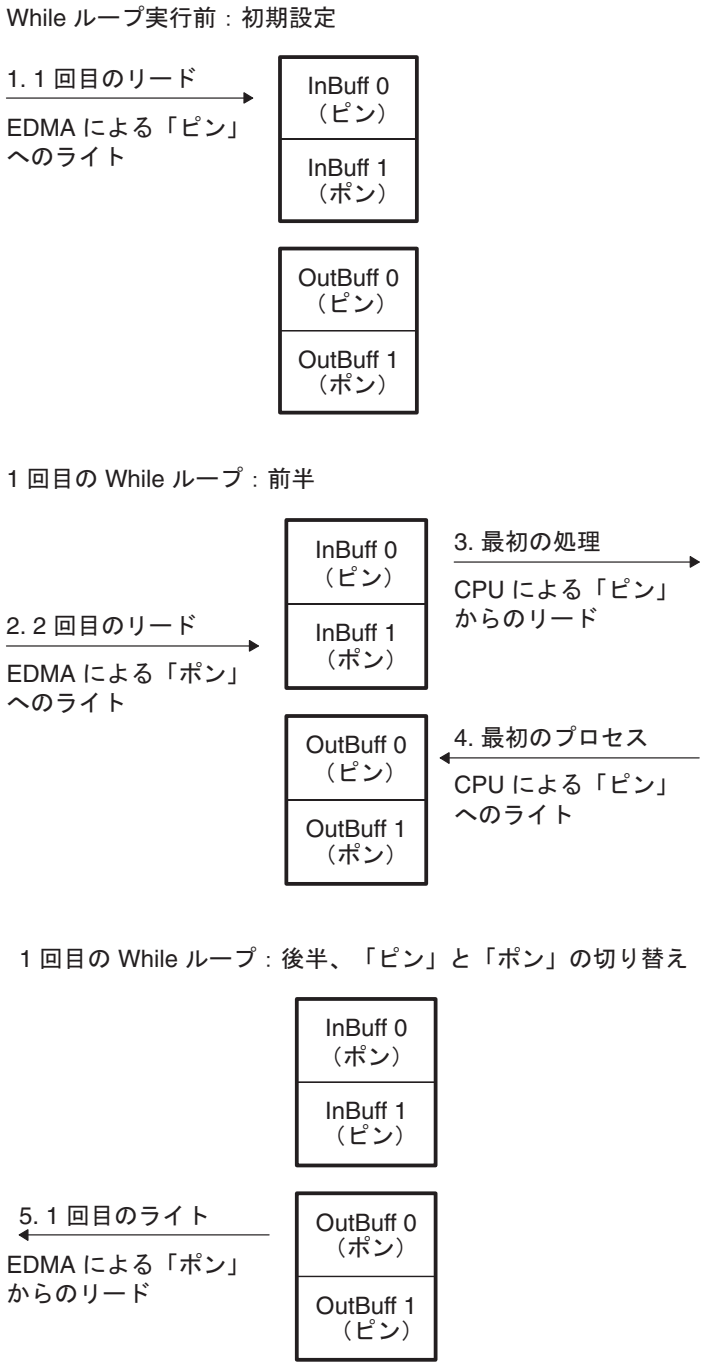
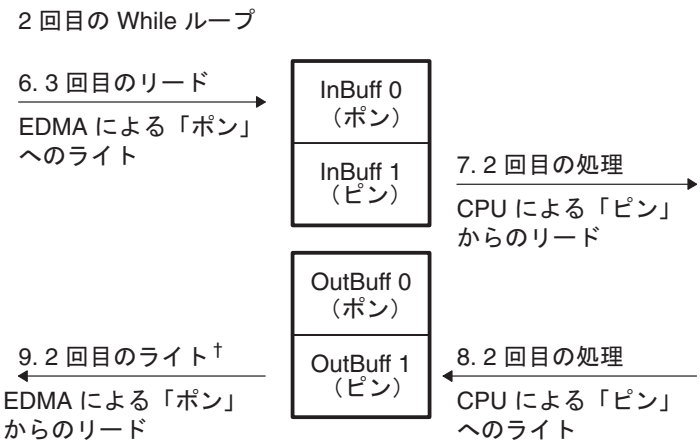
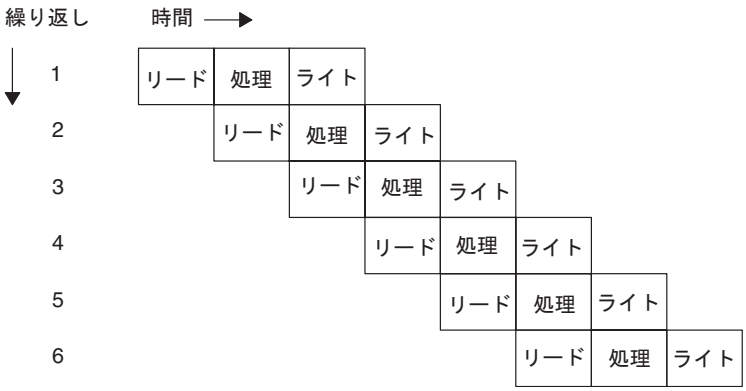


図 34 ダブル・バッファリングのタイム・シーケンス（続き）



† 「ピン」と「ポン」のラベルはステップ 9 の前に切り替えます。

図 35.　パイプライン処理としてのダブル・バッファリング



メモリ・システムのコヒーレンスは、直接このプログラミング技法をサポートします。スヌープとインバリデートにより（5.4.2 項で説明のあるように)、プログラマの介在なしに L2 SRAM と L1D の同期を保ちます。この例では、メモリ・システムは、図 34 の各番号のステップに応じて、次の動作を行います。

1） EDMA は、InBuff 0 にライトします。EDMA が InBuff 0 にライトするとき、これらのアドレスを L1D がキャッシングしていることを検出すると、L2 コントローラは L1D の対応するラインにスヌープ・インバリデートを送ります。L2 が最新のデータを保持するために、L1D からライトバックされる更新データと EDMA がライトするデータがマージされます。対応するラインは L1D でインバリデートされます。

2） このステップは、ステップ 1 と同様に、EDMA が InBuff 1 にライトします。

3） CPU は、InBuff 0 をリードします。これらのラインは、ステップ 1 で L1D でスヌープ・インバリデートされました。したがって、これらのアクセスは L1D でミスし、L1D に L2 SRAM から新しいデータをリードします。

4） CPU は、OutBuff 0 にライトします。実際には、このステップはステップ 3 とオーバーラップ、またはインターリーブする場合があります。はじめに、OutBuff 0 が L1D に存在していると仮定します。この場合、ライトは L1D にヒットし、そしてそのラインをダーティとしてマークします。OutBuff 0 が L1D に存在していない場合、ライトは直接 L2 SRAM に送られます。

5） EDMA は、OutBuff 0 をリードします。L2 における EDMA リードは、L1D で保持される各キャッシュ・ラインへのスヌープをトリガします。そのラインのダーティなデータは、EDMA のリードが処理される前に L2 にライトされます。したがって、EDMA は最新のデータを参照します。そのラインは、クリーンとマークされ、有効のまま残されます。C621x/C671x デバイスでは、そのラインは L1D でインバリデートされます。

6） EDMA は、InBuff 0 にライトします。このステップは、ステップ 1 と同様に処理されます。すなわち、InBuff 0 は必要に応じて L1D からスヌープ・インバリデートされます。ダーティなデータが L2 SRAM にライトバックされた後に、EDMA ライトが処理されます。

7） CPU は、InBuff 1 をリードします。これらのラインは、ステップ 6 で L1D でスヌープ・インバリデートされています。したがって、これらのアクセスは L1D でミスし、L1D に L2 SRAM から新しいデータをリードします。

8） CPU は、OutBuff 1 にライトします。このステップはステップ 4 と同様に処理されます。

9） EDMA は、OutBuff 1 をリードします。このステップはステップ 5 と同様に処理されます。

スヌープおよびスヌープ・インバリデートにより、入力バッファおよび出力バッファに対して、EDMA と CPU の同期が自動的に保たれています。さらに、ダブル・バッファリングは、EDMA および CPU アクセスが同時に発生することを可能にします。

8.3 メモリ・アクセス・オーダリング

8.3.1 メモリ・アクセスのプログラム順序

C6000 DSP のアーキテクチャは、スループット、消費電力、およびプログラミングの容易さを最適化するために、ストロング・オーダー・メモリ・モデルと、リラックス・オーダー・メモリ・モデルを組み合わせてサポートしています。これらの用語は、特定のプログラム・シーケンスが示すメモリ操作の順序に対して相対的に定義されます。このオーダリングは、メモリ・アクセスのプログラム順序といわれます。

C6000 DSP コアは、メモリ操作をサイクル毎に 2 つまで並列して起動できます。メモリ・アクセスのプログラム順序は、アーキテクチャの仮想的な順次実行におけるメモリ・アクセスがもたらす結果を定義します。すなわち、特定の操作のシーケンスに対して、「先に」や「後で」のようなタイム・シーケンス用語を正確に使用して、並列メモリ動作の処理順序を記述します。

プログラム順序は、実行パケット内の命令に関して、また実行パケットのシーケンスに関して定義されます。メモリ操作（並列に発行されるものを含む）は相互に「先に」または「後で」と記述されます。用語の「先に」と「後で」は、厳密に反対の状態を表します。X が Y より「先に」実行されない場合、そのとき X は Y より「後で」実行されます。

異なる実行パケットから起動された複数のメモリ・アクセスは、実行パケット自体の順序と同じ順序を一時的にもちます。すなわち、定義されるプログラム順序は、サイクル *i* で発行されるメモリ操作は、サイクル *i* + 1 で発行されるメモリ操作より常に「先に」あります。また、サイクル *i*-1 で発行されるメモリ操作よりも「後に」あります。

並列に発行されるアクセスでは、操作のタイプ（リードまたはライト）、および操作を実行するデータ・アドレス・ポートがそのオーダリングを決定します。表 32 では、オーダリング規則を説明しています。

表 32. 単一実行パケットから発行されるメモリ操作のプログラム順序

データ・アドレス 1（DA1）操作	データ・アドレス 2（DA2）操作	アクセスのプログラム順序
ロード	ロード	DA1 は、DA2 より「先」
ロード	ストア	DA1 は、DA2 より「先」
ストア	ロード	DA2 は、DA1 より「先」
ストア	ストア	DA1 は、DA2 より「先」

ロードまたはストア命令に対するデータ・アドレス・ポートは、メモリ操作にデータ（アドレスではなく）を供給するデータパスにより決定されます。データパス A でデータを操作するロードとストア命令は DA1 を使用します。データパス B でデータを操作するロードとストア命令は DA2 を使用します。操作するデータを供給するデータパスにより、DA1 または DA2 のどちらが使用されるかが決定されることに注意してください。アクセス用のアドレスを供給するデータパスとは、無関係です。

C64x DSP は、LDNW、STNW、LDNDW および STNDW 命令を使用して、メモリに対するアラインなしのメモリ・アクセスをサポートしています。メモリ・システムは、これらのメモリ・アクセスがアトミックであることを保証していません。むしろ、これらの命令のアクセスは、複数の動作に分割されます。メモリ・アクセスのプログラム順序は、単一のアラインなしアクセスの、個々のメモリ操作の順序を定義していません。プログラム順序は、アラインなしアクセス全体が先または後のアクセスに対してどのように順序付けられているかのみを定義しています。したがって、たとえアラインなしアクセス全体が CPU 自体に対して、前述に定義したプログラム順序に従ったとしても、他のリクエスターはアラインなしメモリ・アクセスが分割されて発生するのを見ることになります。

前述の定義は、メモリ・システムのプログラム動作を説明しています。メモリ・システムは、メモリ・アクセスのプログラム順序に対するプログラム動作が、CPU アクセスの相対関係を保持することを保証します。しかし、プログラム動作が正しく保持される限り、それらがメモリ階層内で実行されるとき、メモリ・システムは操作のオーダリングを緩和することがあります。また、メモリ・システムに対する他のリクエスターが、元のプログラム順序と異なる順序で発生するアクセスを観測することが可能になります。詳細は、8.3.2 項で説明します。

8.3.2 ストロングおよびリラックス・メモリ・オーダリング

メモリ・アクセスのプログラム順序は（8.3.1 項で説明のように）、メモリ・アクセスの特定シーケンスにおいて望ましいプログラム動作を表現しています。ほとんどの状況において、メモリ・システムは、これらのプログラム動作を保持するために、メモリ階層のすべてのレベルにおいてこの厳密な順序でメモリ・アクセスを実行する必要はありません。実際には、パフォーマンスや消費電力の理由で、アクセスの順序を緩和することはメモリ階層にとってきわめて有利になります。

ペリフェラルと通信する場合、および同一メモリをアクセスする他のデバイス（EDMA、他の CPU）と協調動作する場合、プログラム順序を保持することは潜在的により重要になります。これらの多様なニーズに応えるために、メモリ階層はストロング・オーダーのメモリ操作と、リラックス・オーダーのメモリ操作の両方をサポートします。

メモリ・システムのコヒーレンスとは、単一のメモリ・ロケーションに対するライトはすべてのリクエスターに関して順次処理されること、およびすべてのリクエスターがそのロケーションに対して同じシーケンスでライトを観測できることを意味します。異なるロケーションに対するアクセスのオーダリング、または同じロケーションへの複数のリードのオーダリングと、コヒーレンスは一切関係ありません。むしろ、メモリ・システムのオーダリング規則（ストロング、またはリラックス）に、別のロケーションに対するアクセスに適用されるオーダリングの保証について記述しています。

システム内のある場所からメモリ操作の異なるシーケンスを観測することが可能ではない場合、メモリ操作のシーケンスはストロング・オーダーであるといいます。たとえば、あるリクエスターがロケーション Y にライトする前に別のロケーション X にライトする場合、2 番目のリクエスターが更新された X より前に、更新された Y を観測することがない場合、そのアクセスはストロング・オーダーであると見なされます。この例では、メモリ・システムが最初のリクエスターからのライト、そして 2 番目のリクエスターからのリードを処理する順序に関係なく、2 番目のリクエスターは X に対するライトの前に Y に対するライトの発生を観測することができてはなりません。しかし、ストロング・オーダリングでは必ずしもライトを順次処理する必要はありません。2 番目の CPU にとって、X より前に更新された Y を観測することが可能ではない限り、ライトは同時にまたは反対の順序で起こっても差し支えないということです。

アラインなしアクセス（LDNW、STNW、LDNDW、および STNDW 命令によって発行される）は、他のアクセスに対してストロング・オーダーです。アラインなしアクセスはアトミックであることが保証されないので、ストロング・オーダリングの保証はアラインなしアクセス全体にのみ適用されます。単一のアラインなしアクセスを構成する個々の動作の間ではオーダリングの保証がなされません。メモリ・システムがアラインなしアクセスを複数のメモリ操作に分割するためです。

メモリ階層は、すべてのキャッシュ不可能なロング・ディスタンス・アクセスに対してはストロング・オーダリングを提供します。このようなアクセスは通常、ペリフェラルや対応する MAR のキャッシュ・イネーブル・ビットがセットされていない、外部メモリに対して行われます。

外部メモリのキャッシュ可能なロケーションには、メモリ階層はリラックス・オーダリングのみを提供します。したがって他のリクエスターは、元々のメモリ・アクセスのプログラム順序と異なる順序でメモリでの更新の発生を観測することがありえます。このリラックス・オーダリングであっても、CPU は、8.3.1 項で説明した望ましいメモリ・システムのプログラム動作を観測できます。

L2 SRAM 内のキャッシュ可能なロケーションについて、C64x デバイスは、同一 L1D キャッシュ・ラインに関連付けられたロケーションで、L1D 内にないロケーションに対する CPU アクセスに対してストロング・オーダリングを提供します（C621x/C671x デバイスでは、これは上位 27 ビットが等しいアドレスに対するデータ・アクセスがストロング・オーダリングであることを意味します）。C64x デバイスでは、これは上位 26 ビットが等しいアドレスに対するデータ・アクセスがストロング・オーダリングであることを意味します。

同一のキャッシュラインに関連付けられていない L2 SRAM のロケーションでは、ライトで、かつ関連するアドレスが L1D 内にない場合に限ってのみ、ストロング・オーダリングが提供されます。これは 7.3.2 項で説明している L1DWIBAR および L1DWIWC 制御レジスタを使用することで保証されます。その他の場合は、L2 SRAM に対する CPU アクセスに対して、リラックス・オーダリングが提供されます。

L2 は L2 SRAM に対する EDMA アクセスについて、限定されたオーダリングの保証を提供します。L2 に対する EDMA リードは、他の EDMA リードとの間でリオーダリングされません。L2 に対する EDMA ライトは、他の EDMA ライトとの間でリオーダリングされません。しかし、リードとライトは、相互の間でリオーダリングされることもあります。

索引

数

C

E

I

L

M

P

か

き

く

し

す

ひ

ふ

め

れ

日本テキサス・インスツルメンツ株式会社

本　　社　〒160-8366　東京都新宿区西新宿６丁目２４番１号　西新宿三井ビルディング３階　☎０３（４３３１）２０００（番号案内）

西日本ビジネスセンター　〒530-6026　大阪市北区天満橋１丁目８番３０号　ＯＡＰオフィスタワー26階　☎０６（６３５６）４５００（代　　表）

■お問い合わせ先

プロダクト・インフォメーション・センター（PIC）　URL: http: //www.tij.co.jp/pic/

TMS320C64x DSP
2 レベル内部メモリ
リファレンス・ガイド

第 1 版 2005 年 12 月

発行所 **日本テキサス・インスツルメンツ株式会社**
〒160-8366
東京都新宿区西新宿 6-24-1（西新宿三井ビルディング）

Made in Japan
’05.12

日本テキサス・インスツルメンツ株式会社

SPRUE50