

DDR および DDR2 SDRAM 高性能コントローラ・ユーザガイド



101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
www.altera.com

この資料は英語版を翻訳したもので、内容に相違が生じる場合には原文を優先します。こちらの日本語版は参考用としてご利用ください。設計の際には、最新の英語版で内容をご確認ください。

MegaCore バージョン: 7.2
ドキュメント・バージョン: 7.2
ドキュメント・デート: 2007 年 10 月

Copyright © 2007 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.



UG-01010-4.0

第1章 これらの MegaCore ファンクションについて

リリース情報	1-1
サポートされるデバイス・ファミリ	1-1
特長	1-2
概要	1-2
MegaCore 検証	1-3
パフォーマンスおよびリソース使用率	1-4
インストールおよびライセンス	1-5
OpenCore Plus 評価機能	1-6
OpenCore Plus タイム・アウト動作	1-7

第2章 使用法

デザイン・フロー	2-1
フローの選択	2-3
SOPC Builder フロー	2-3
パラメータの指定	2-4
SOPC Builder システムの終了	2-4
システムのシミュレーション	2-5
MegaWizard Plug-In Manager フロー	2-6
パラメータの指定	2-6
デザイン例のシミュレーション	2-9
デザイン例のコンパイル	2-10
デバイスのプログラム	2-12
ユーザ・デザインの実装	2-12

第3章 パラメータの設定

メモリ設定	3-1
PHY 設定	3-1
コントローラ設定	3-1

第4章 機能の説明

ブロックの説明	4-1
コントロール・ロジック	4-1
レイテンシ	4-3
ECC	4-5
デザイン例	4-10
インタフェースおよび信号	4-12
インタフェースの説明	4-12
信号	4-24

付録 A. ECC レジスタの説明

追加情報

改訂履歴	Info-i
アルテラへのお問い合わせ	Info-i
表記規則	Info-i

リリース情報

表 1-1 に、DDR および DDR2 SDRAM 高性能コントローラ MegaCore® ファンクションのリリースに関する情報を示します。

項目	説明
バージョン	7.2
リリース月	2007 年 10 月
製品コード	IP-SDRAM/HPDDR (DDR SDRAM) IP-SDRAM/HPDDR2 (DDR2 SDRAM)
プロダクト ID	00BE (DDR SDRAM) 00BF (DDR2 SDRAM) 00CO (altmemphy メガファンクション)
ベンダ ID	6AF7

サポートされる デバイス・ ファミリ

下記で説明されているように、MegaCore ファンクションは、ターゲットのアルテラ・デバイス・ファミリに対し、フル・サポートあるいは暫定サポートを提供しています。

- フル・サポートとは、MegaCore ファンクションがデバイス・ファミリの機能およびタイミング要件をすべて満たしており、製造デザインで使用可能であることを意味します。
- 暫定サポートとは、MegaCore ファンクションがデバイス・ファミリの機能要件はすべて満たしているが、タイミング要件については評価中であるため、生産デザインでの使用は注意が必要なことを意味します。

表 1-2 に、DDR および DDR2 SDRAM 高性能コントローラにより各アルテラ・デバイス・ファミリに提供されるサポートのレベルを示します。

デバイス・ファミリ	サポートの種類
Arria™ GX	フル・サポート
Cyclone® III	暫定サポート
HardCopy® II	暫定サポート

表 1-2. サポートされるデバイス・ファミリ (2 / 2)

デバイス・ファミリ	サポートの種類
Stratix® II	フル・サポート
Stratix II GX	フル・サポート
Stratix III	暫定サポート
その他のデバイス・ファミリ	サポートなし

特長

- 誤り訂正コード (ECC) 機能を統合
- パワーアップ・キャリブレーション付き On-Chip Termination (チップ内終端) で Cyclone III および Stratix III デバイスをサポート
- Arria GX、Cyclone III、HardCopy II、Stratix II、および Stratix II GX の各デバイスをフル・レート・サポート
- SOPC Builder への対応
- altmemphy メガファンクションをサポート
- 業界標準の DDR および DDR2 SDRAM デバイスおよびモジュールをサポート
- オプションのユーザ・コントローラ・リフレッシュ
- オプションの Avalon® メモリ・マップド (Avalon-MM) ローカル・インタフェース
- 使いやすい MegaWizard® インタフェース
- OpenCore Plus 評価をサポート
- アルテラでサポートしている VHDL、Verilog HDL シミュレータ上で使用可能な IP ファンクション・シミュレーション・モデル

概要

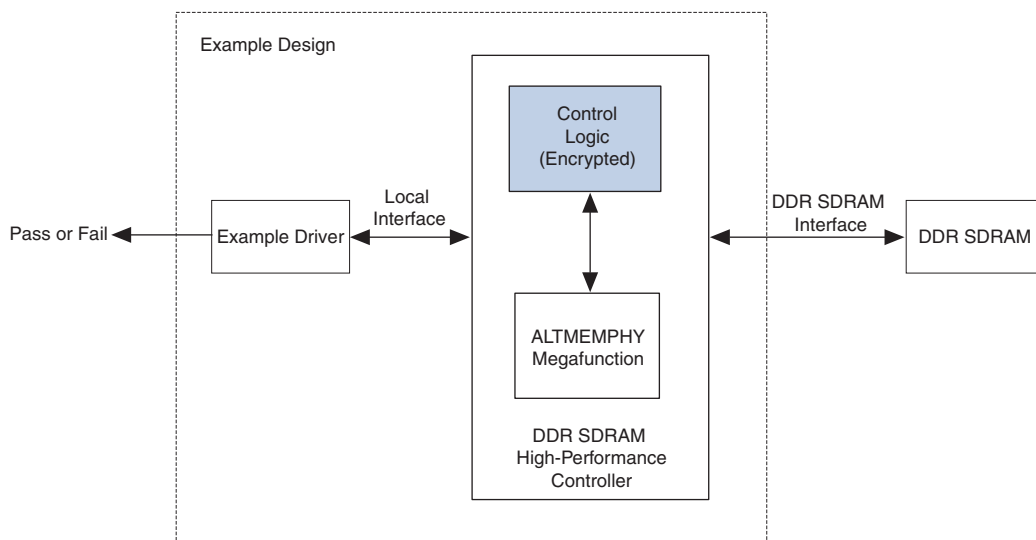
アルテラの DDR および DDR2 SDRAM 高性能コントローラ MegaCore ファンクションは、業界標準の DDR SDRAM および DDR2 SDRAM への簡略化されたインタフェースを提供します。この MegaCore ファンクションは、アルテラの altmemphy メガファンクションと連携して動作します。



altmemphy メガファンクションについて詳しくは、「[altmemphy メガファンクション・ユーザガイド](#)」を参照してください。

図 1-1 に、DDR または DDR2 SDRAM 高性能コントローラのシステム・レベルのブロック図を示します。

図 1-1. システム・レベル図



MegaWizard Plug-In Manager では、デザイン例が生成され、PLL(Phase-Locked Loop)、サンプル・ドライバ、DDR または DDR2 SDRAM 高性能コントローラのカスタム・バリエーション、およびオプションの DLL (Stratix シリーズの場合のみ) がインスタンス化されます。デザイン例は、ハードウェア上でシミュレーション、合成、および使用可能な完全に動作するデザインです。サンプル・ドライバは、セルフ・テスト・モジュールでコントローラにリードとライトのコマンドを発行し、リード・データをチェックして、パス / フェイルおよびテスト完了の信号を出力します。

MegaCore 検証

MegaCore 検証では、シミュレーション・テストを実行します。アルテラでは、DDR および DDR2 SDRAM 高性能コントローラの機能を保証するために、業界標準の Denali モデルを使用して、機能テストをカバーする徹底したランダムなダイレクト・テストを実施しました。さらに、アルテラでは DDR および DDR2 SDRAM 高性能コントローラの幅広いゲート・レベル・テストを実施して、コントローラのコンパイル後の機能を検証しました。

パフォーマンス およびリソース 使用率

表 1-3 に、Quartus® II ソフトウェア・バージョン 7.2 を使用した DDR SDRAM 高性能コントローラの標準的なパフォーマンスの結果を示します。

表 1-3. 標準的なパフォーマンス		
デバイス	System f_{MAX} (MHz)	
	DDR SDRAM	DDR2 SDRAM
Cyclone III	167 (1)	200 (1)
Stratix II	200	333
Stratix III	200 (1)	400 (1)

表 1-3 の注：

(1) デバイス特性評価待ちです。



デバイス性能について詳しくは、該当するデバイス・ハンドブックを参照してください。

表 1-4 に、Cyclone III デバイスにおける DDR SDRAM 高性能コントローラの標準的なサイズを示します。

表 1-4. 標準的なサイズ—Cyclone III デバイス			
ローカル・データ幅 (ビット)	メモリ幅 (ビット)	等価 LE 数	メモリ (M9K)
32	8	1,632	3
64	16	1,871	6
256	64	3,199	21
288	72	3,431	22

表 1-5 に、Stratix II デバイスにおける DDR SDRAM 高性能コントローラの標準的なサイズを示します。

ローカル・データ幅 (ビット)	メモリ幅 (ビット)	組み合わせ ALUT 数	ロジック・レジスタ数	メモリ・ブロック数 (M4K)
32	8	1,163	1,123	3
64	16	1,229	1,269	6
256	64	1,573	2,135	20
288	72	1,649	2,278	22

表 1-6 に、Stratix III デバイスにおける DDR SDRAM 高性能コントローラの標準的なサイズを示します。

ローカル・データ幅 (ビット)	メモリ幅 (ビット)	組み合わせ ALUT 数	ロジック・レジスタ数	メモリ	
				M9K	MLAB
32	8	1,199	1,131	2	3
64	16	1,246	1,347	4	6
256	64	2,673	2,746	13	24

インストール およびライ センス

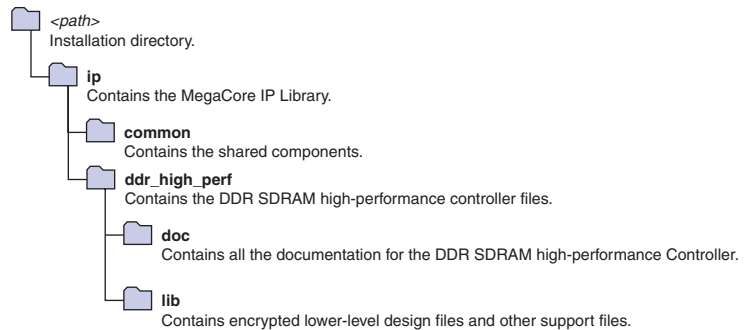


DDR および DDR2 SDRAM 高性能コントローラ MegaCore ファンクションは、MegaCore IP ライブラリの一部であり、Quartus® II ソフトウェアとともに配布されます。また、アルテラのウェブサイト (www.altera.co.jp) からダウンロードすることもできます。

システム要求およびインストールの手順については、「[Quartus II Installation & Licensing for Windows](#)」または「[Quartus II Installation & Licensing for UNIX & Linux Workstations](#)」を参照してください。

図 1-2 に、DDR および DDR2 SDRAM 高性能コントローラ MegaCore ファンクションをインストールした後のディレクトリ構造を示します。ここで、*<path>* がインストール・ディレクトリです。Windows でのデフォルトのインストール・ディレクトリは、**c:\altera\72** です。UNIX および Solaris では、**/opt/altera/72** です。

図 1-2. ディレクトリ構造



MegaCore ファンクションを製品に組み込む場合にのみ、ライセンスを購入していただく必要があります。

DDRまたはDDR2 SDRAM 高性能コントローラ MegaCore ファンクションのライセンス購入後は、アルテラのウェブサイト（www.altera.co.jp/licensing）からライセンス・ファイルを要求して、コンピュータにインストールできます。ライセンス・ファイルを要求すると、アルテラから電子メールで **license.dat** ファイルが送信されます。インターネットをご利用いただけないお客様は、アルテラの販売代理店にお問い合わせください。

OpenCore Plus 評価機能

アルテラの無償 OpenCore Plus 評価版機能により、以下の処理を実行することができます。

- システム内のメガファンクション（アルテラ MegaCore ファンクションまたは AMPPSM メガファンクション）の動作のシミュレーション
- デザインの機能を検証したり、サイズやスピードを素早く簡単に評価する。
- MegaCore ファンクションを含むデザインに対し、実行時間に制限のあるデバイス・プログラミング・ファイルを生成
- デバイスをプログラムし、評価されるメガファンクションを含むデザインを実機上で検証する。

メガファンクションのライセンスは、お客様が機能と性能に満足し、かつデザインを製品化する場合にのみ、ご購入いただく必要があります。



DDR および DDR2 SDRAM 高性能コントローラを使用した OpenCore Plus ハードウェア評価について詳しくは、「[Application Note 320: OpenCore Plus Evaluation of Megafunctions](#)」を参照してください。

OpenCore Plus タイム・アウト動作

OpenCore Plus ハードウェア評価機能は、以下の 2 種類の動作モードでメガファンクションの実機評価をサポートします。

- **Untethered** (アンテザード) — デザインは限定時間のみ実行されます。
- **Tethered** (テザード) — ボードとホスト・コンピュータ間に接続が必要です。デザイン内のすべてのメガファンクションが **Tethered** モードをサポートしている場合、デバイスはより長時間または無制限に動作できます。

最も限定的な評価時間に達すると、デバイス内のすべてのメガファンクションが同時にタイム・アウトします。デザイン内に複数のメガファンクションがある場合、特定のメガファンクションのタイム・アウト動作は、他のメガファンクションのタイム・アウト動作によって隠されることがあります。



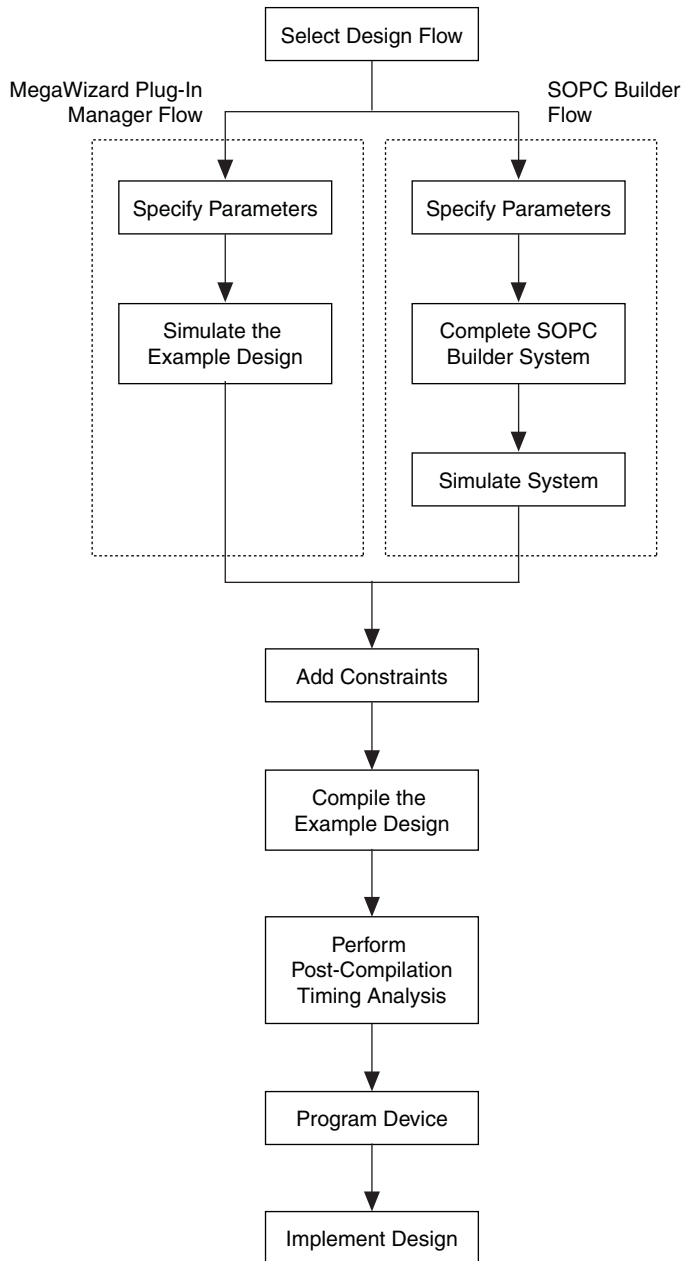
MegaCore ファンクションの場合、アンテザード・タイムアウトは 1 時間、テザード・タイムアウト値は無制限です。

ハードウェア評価期限経過後にデザインは動作を停止し、`local_ready` 出力が **Low** になります。

デザイン・ フロー

図 2-1 に、DDR および DDR2 SDRAM 高性能コントローラ MegaCore® ファンクションおよびQuartus® IIソフトウェアを使用してシステムを構築するためのステージを示します。この章の項では、各ステージについて説明します。

図 2-1. デザイン・フロー



フローの選択

以下のいずれかのフローを使用して、DDR および DDR2 SDRAM 高性能コントローラ MegaCore ファンクションをパラメータ化することができます。

- SOPC Builder フロー
- MegaWizard Plug-in Manager フロー

SOPC Builder フローは、以下の利点を提供します。

- 自動的に生成されるシミュレーション環境
- カスタム・コンポーネントを作成し、それらをコンポーネント・ウィザードを介して統合
- すべてのコンポーネントを Avalon-MM インタフェースと自動的に相互接続

MegaWizard Plug-in Manager フローは、以下の利点を提供します。

- DDR または DDR2 SDRAM インタフェースから直接ペリフェラル・デバイスへのデザイン
- より高い周波数動作を達成

SOPC Builder フロー

SOPC Builder フローでは、DDR および DDR2 SDRAM 高性能コントローラ MegaCore ファンクションを新規または既存の SOPC Builder システムに直接追加することができます。また、他の使用可能なコンポーネントを簡単に追加して、Nios II プロセッサ、外部メモリ・コントローラ、および Scatter/Gather DMA コントローラなどの DDR および DDR2 SDRAM 高性能コントローラを搭載する SOPC Builder システムを素早く構築することもできます。SOPC Builder は、システム・インタコネクタ・ロジックおよびシステム・シミュレーション環境を自動的に構築します。



関連情報	参照先
SOPC Builder	「Quartus II ハンドブック」の Volume 4
Quartus II ソフトウェア	Quartus II Help

パラメータの指定

SOPC Builder フローを使用して DDR および DDR2 SDRAM 高性能コントローラパラメータを指定するには、以下のステップに従います。

1. Quartus II ソフトウェアで、**New Project Wizard** を使用して新規 Quartus II プロジェクトを作成します。
2. Tools メニューで、**SOPC Builder** をクリックします。
3. 新しいシステムの場合、システム名と言語を指定します。
4. **System Contents** タブからシステムに、**DDR or DDR2 SDRAM High-Performance Controller** を追加します。



DDR or DDR2 SDRAM High-Performance Controller は、**Memories and Memory Controllers > SDRAM** ディレクトリにあります。

5. **Parameter Settings** タブのすべてのページで必要なパラメータを指定します。



パラメータについて詳しくは、[3-1 ページの「パラメータの設定」](#)を参照してください。

6. **Finish** をクリックして、DDR および DDR2 SDRAM 高性能コントローラ MegaCore ファンクションを完了し、それをシステムに追加します。

SOPC Builder システムの終了

SOPC Builder システムを終了するには、以下のステップに従います。

7. SOPC Builder で **Nios II processor** を選択し、**Add** をクリックします。
8. **Reset Vector** および **Exception Vector** の場合は、**altmemddr** を選択します。
9. **Reset Vector Offset** を **0x20** に、**Exception Vector Offset** を **0x40** に変更します。



altmemphy メガファンクションは、リセットされるごとにメモリ・インタフェース・キャリブレーションを実行し、アドレス $0x0 \sim 0x1f$ に書き込みます。システム・リセット中にメモリの内容を保持するには、 $0x20$ 以下のメモリ・アドレスを使用しないようにします。リセットするたびにフラッシュから SDRAM メモリの内容をリロードする場合、このステップは不要です。

10. **Finish** をクリックします。
11. SOPC Builder で、**Interface Protocols** および **Serial** を拡張します。
12. **JTAG UART** を選択して、**Add** をクリックします。
13. **Finish** をクリックします。



アドレスのオーバーラップを警告するメッセージが表示される場合、System メニューで **Auto Assign Base Addresses** をクリックします。



ECC をイネーブルして、IRQ のオーバーラップを警告するメッセージが表示される場合は、System メニューで **Auto Assign IRQs** をクリックします。

14. このシステム例では、不要なクロック・ドメイン・クロス・ロジックを回避するために、他のすべてのモジュールが `altmemddr_sysclk` でクロックされるようにします。
15. **Generate** をクリックします。
16. 自動的に生成された制約を機能させるには、ピン名とピン・グループ・アサインメントが一致しなくてはなりません。一致しない場合は、デザインをコンパイルするときにフィットが得られません。ピン名を編集するには、`altmemddr_example_top.v` ファイルを編集します。また、サンプル・ドライバおよび DDR SDRAM 高性能コントローラを置き換え、SOPC Builder で生成されたシステムをインスタンス化するには、`altmemddr_example_top.v` ファイルを編集します。

システムのシミュレーション

システム生成時、SOPC Builder はシステム全体のシミュレーション・モデルおよびテストベンチをオプションで生成します。これらを使用して、アルテラがサポートする任意のシミュレーション・ツールでシステムを簡単にシミュレートすることができます。また、SOPC Builder は

ModelSim Tcl スクリプトおよびマクロのセットを生成することもでき、これらを使用して ModelSim シミュレーション・ソフトウェアで、テストベンチ、IP 機能シミュレーション・モデル、およびシステムを記述するプレーン・テキスト RTL デザイン・ファイルをコンパイルできます。



関連情報	参照先
SOPC Builder システムのシミュレーション	「Quartus II ハンドブック」の Volume 4 「AN 351 : Simulating Nios II Systems」



デザインをコンパイルする前に、Tcl 制約スクリプト `<variation name>_pin_assignments.tcl` を実行する必要があります。

MegaWizard Plug-In Manager フロー

MegaWizard Plug-In Manager フローでは、DDR および DDR2 SDRAM 高性能コントローラ MegaCore ファンクションをカスタマイズし、ファンクションを手動でデザインに組み込むことができます。



関連情報	参照先
MegaWizard Plug-In Manager	Quartus II Help
Quartus II ソフトウェア	

パラメータの指定

MegaWizard Plug-in Manager フローを使用して、DDR および DDR2 SDRAM 高性能コントローラのパラメータを指定するには、以下のステップに従います。

1. Quartus II ソフトウェアで、**New Project Wizard** を使用して新規 Quartus II プロジェクトを作成します。
2. MegaWizard Plug-In Manager を起動するには、Tools メニューで **MegaWizard Plug-In Manager** をクリックして、以下のステップに従います。



DDR or DDR2 SDRAM High-Performance Controller は、**Interfaces > Memory Controllers** ディレクトリにあります。

3. **Parameter Settings** タブのすべてのページでパラメータを指定します。



パラメータについて詳しくは、[3-1 ページ](#)の「**パラメータの設定**」を参照してください。

4. **EDA** タブで、**Generate Simulation Model** をオンにして、選択した言語で MegaCore ファンクション用の IP 機能シミュレーション・モデルを生成します。

IP 機能シミュレーション・モデルは、Quartus II ソフトウェアで生成するサイクル単位の正確な VHDL または Verilog HDL モデルです。



これらのシミュレーション・モデルは、シミュレーション目的にのみ使用し、合成やその他の目的には使用しないでください。これらのモデルを合成に使用すると、機能しないデザインが作成されます。



一部サードパーティ合成ツールでは、詳細なロジックは含まず MegaCore ファンクションの構造のみを含むネットリストを使用して、MegaCore ファンクションを含むデザインの性能を最適化することができます。合成ツールでこの機能がサポートされている場合、**Generate netlist** をオンにします。

5. **Summary** タブで、生成するファイルを選択します。グレイのチェックマークは、自動的に生成されるファイルを示します。その他のファイルはすべてオプションです。



プロジェクト・ディレクトリで生成されるファイルについて詳しくは、[表 2-1](#)を参照してください。

6. **Finish** をクリックして、MegaCore ファイルおよびサポートするファイルを生成します。
7. 生成レポートを表示した後、**Exit** をクリックして MegaWizard Plug-In Manager を閉じます。

表 2-1 に、生成されたファイルおよびプロジェクト・ディレクトリに存在する可能性があるその他のファイルを示します。MegaWizard Plug-In Manager レポートに指定されるファイルの名前とタイプは、デザインを VHDL または Verilog HDL のいずれで作成したかによって異なります。

表 2-1. 生成されるファイル 注 (1)	
ファイル名	説明
<variation name>.bsf	MegaCore ファンクションのバリエーション用 QuartusII シンボル・ファイル。Quartus II ブロック図エディタでこのファイルを使用できます
<variation name>.html	MegaCore ファンクション・レポート・ファイルです。
<variation name>.vo または .vho	VHDL または Verilog HDL のゲート・レベル・シミュレーション・モデルです。
<variation name>.vhd または .v	カスタム MegaCore ファンクションの VHDL または Verilog HDL トップレベルの記述を定義する MegaCore ファンクション・バリエーション・ファイルです。デザイン内部のこのファイルによって定義されたエンティティをインスタンスします。QuartusII ソフトウェアでのデザインのコンパイル時にこのファイルがインクルードされます。
<variation name>_auk_ddr_hp_controller_wrapper.vo または .vho	VHDL または Verilog HDL の IP 機能シミュレーション・モデルです。
<variation name>_bb.v	MegaCore ファンクション・バリエーションの Verilog HDL ブラックボックス・ファイルです。サードパーティ製 EDA ツールを使用してデザインを合成するときこのファイルを使用します。
<variation name>_example_driver.vhd または .v	サンプル・ドライバ。
<variation name>_example_top.vhd または .v	デザイン例。

表 2-1 の注:

(1) <variation name> は、バリエーション名です。

8. 生成レポートを表示した後、**Exit** をクリックして MegaWizard Plug-In Manager を閉じます。
9. <variation name>_example_top.v または .vhd ファイルがプロジェクトのトップ・レベル・デザイン・ファイルになるように設定します。

デザイン例をシミュレーション (2-9 ページの「デザイン例のシミュレーション」参照) し、次にコンパイル (2-10 ページの「デザイン例のコンパイル」参照) します。

デザイン例のシミュレーション

MegaWizard Plug-In Manager が生成する IP 機能シミュレーション・モデルを使用して、デザイン例をシミュレーションできます。MegaWizard Plug-In Manager は、デザイン例の VHDL または Verilog HDL テストベンチをプロジェクト・ディレクトリの **testbench** ディレクトリに生成します。



テストベンチについて詳しくは、4-10 ページの「[デザイン例](#)」を参照してください。

IP 機能シミュレーション・モデルは、アルテラがサポートする任意の VHDL または Verilog HDL シミュレータで使用できます。

シミュレーションは、NativeLink を使用して Quartus II ソフトウェアからサードパーティ製シミュレーション・ツールを使用して実行できます。



NativeLink について詳しくは、「[Quartus II ハンドブック Volume 3](#)」の「[Simulating Altera IP Using NativeLink](#)」章を参照してください。

Quartus II ソフトウェアで NativeLink を使用してシミュレーションの設定を行うには、以下のステップに従ってください。

1. IP 機能シミュレーション・モデルを使用して、カスタム・バリエーションを作成します。
2. サードパーティ・シミュレータ実行ファイルへの絶対パスが設定済みかどうかチェックします。Tools メニューで、Options をクリックし、EDA Tools Options を選択します。
3. サンプル・プロジェクトに、トップレベル・エンティティを設定します。
 - a. File メニューの **Open** をクリックします。
 - b. `<variation name>_example_top` を表示して、**Open** をクリックします。
 - c. Project メニューの **Set as top-level entity** をクリックします。
4. Processing メニューで、**Start** をポイントして **Start Analysis & Elaboration** をクリックします。
5. Assignments メニューの **Settings** をクリックして、**EDA Tool Settings** を展開し、**Simulation** を選択します。Tool Name でシミュレータを選択して、**NativeLink Settings** で **Compile Test Bench** を選択し、**Test Benches** をクリックします。
6. **New** をクリックします。
7. **Test bench name** に名前を入力します。

8. **Test bench entity** に、自動的に生成されたテストベンチの名前、`<variation name>_example_top_tb`を入力します。
9. **Instance** に、トップ・レベル・インスタンスの名前を入力します。
10. **Run for** を **500 μ s** に変更します。
11. テストベンチ・ファイルおよび自動的に生成されたメモリ・モデル・ファイルを追加します。**File name** フィールドでメモリ・モデルとテストベンチの位置を参照して、**OK** をクリックし、**Add** をクリックします。テストベンチは `<variation name>_example_top_tb.v`、メモリ・モデルは `<variation name>_mem_model.v` です。
12. **OK** をクリックします。
13. **OK** をクリックします。
14. Tools メニューで **EDA Simulation Tool** をポイントして、**Run EDA RTL Simulation** をクリックします。

デザイン例の コンパイル

Quartus II ソフトウェアを使用してデザイン例をコンパイルし、コンパイル後のタイミング解析を実行するには、以下のステップに従います。

1. TimeQuest をイネーブルにします。
 - a. Assignments メニューで、**Settings** をクリックし、**Timing Analysis Settings** を展開し、**Use TimeQuest Timing Analyzer during compilation** を選択して、**OK** をクリックします。
 - b. Synopsys Design Constraints ファイル (`<variation name>_phy_ddr_timing.sdc`) をプロジェクトに追加します。Project メニューで、**Add/Remove Files in Project** をクリックし、ファイルを参照します。
2. ピンの I/O 規格アサインメントを追加します。



デザインをコンパイルする前に、I/O 規格アサインメント・スクリプトを手動で実行しなければなりません。このスクリプトは、メモリ・インタフェース・ピンおよび正しい I/O 規格を割り当て、Quartus II フィッタが動作しなくなる問題を回避します。

- a. DDR2 SDRAM インタフェースが 1 つで、デザイン例に示すようにトップ・レベル・ピンにデフォルトの名前が付いている場合は、`<variation name>_pin_assignments.tcl` を実行します。

または

- b. 以下のステップに従います。
 - Assignments メニューで、**Pin Planner** をクリックします。Quartus II Pin Planner でトップ・レベル・デザインを編集して、すべての DDR または DDR2 信号名にプリフィックスを付加します。例えば、**mem_ddr** を **core1_mem_addr** に変更します。
 - Assignments メニューの **Pins** をクリックします。ウィンドウ内で右クリックし、**Create/Import Megafunction** をクリックします。**Import an existing custom megafunction** を選択し、**<variation name>.ppf** にアクセスします。
 - トップレベルの DDR または DDR2 信号名に追加したプリフィックスを **Instance name** ボックスに入力し、**OK** をクリックします。



Cyclone III デバイスでは、メモリ・インタフェース・ピンに兼用 VREF ピンを使用してはなりません。

3. サンプル・プロジェクトに、トップレベル・エンティティを設定します。
 - a. File メニューの **Open** をクリックします。
 - b. **<variation name>_example_top** を表示して、**Open.** をクリックします。
 - c. Project メニューの **Set as top-level entity** をクリックします。
4. Processing メニューで、**Start** をポイントして **Start Analysis and Synthesis** をクリックします。
5. DQ および DQS ピン・グループを割り当てます。
 - a. Quartus II ソフトウェアがデザインを正しくフィットするように、すべてのピンを割り当てる必要があります。タイミング解析はすべてのピンを制約した場合にのみ正しくなります。
 - b. Pin Planner または Assignment Editor のいずれかを使用して、クロック・ソース・ピンを手動で割り当てます。さらに、各 DQS ピンを必要なピンに割り当て、使用する必要がある DQS ピン・グループを選択します。次に Quartus II フィッタは、対応する DQ 信号を各グループ内の適切な DQ ピンに自動的に配置します。



DDR2 SDRAM でピンを割り当てる場合、例えばクロック・ソースやリセットなどで、I/O 規格が LVTTTL ではなく SSTL18-II に設定されていることを確認します。また、QuartusII ソフトウェアでピンを配置したいデバイスのバンクまたはサイドを選択します。

6. Stratix III デザインまたは高度な I/O タイミングを使用する場合、**Device and pin options** でボード・トレース・モデルを指定します。あるいは、他のデバイスを使用し、高度な IO タイミングを使用しない場合、すべてのメモリ・インタフェース・ピンに出力ピンの負荷を設定します。
7. 必要な I/O ドライブ強度（シミュレーションから得られたもの）を選択して、各信号または ODT 設定が正しくドライブされ、オーバーシュートやアンダシュートが生じないようにします。
8. Processing メニューで **Start Compilation** をクリックして、デザインをコンパイルします。
9. オプション。レポート・タイミングを実行して、詳細な DDR SDRAM インタフェース・タイミング・レポートを取得します。<variation name>_phy_report_timing.tcl を実行します。
 - a. Tools メニューの **Tcl Scripts** をクリックします。
または
 - b. Tools メニューの **TimeQuest Timing Analyzer** をクリックします。Script メニューの **Run Tcl Script** をクリックします。



SignalTap II ロジック・アナライザをデザインに接続するには、「AN 380: Test DDR or DDR2 SDRAM Interfaces on Hardware Using the Example Driver」を参照してください。

デバイスのプログラム

デザイン例をコンパイルした後は、ゲート・レベル・シミュレーションを実行したり（2-9 ページの「[デザイン例のシミュレーション](#)」を参照）、ターゲットのアルテラ・デバイスをプログラムしてハードウェアでデザイン例を検証することができます。

ユーザ・デザインの実装

デザイン例に基づいてユーザ・デザインを実装するには、デザイン例のサンプル・ドライバをユーザ独自のロジックに置き換えます。

メモリ設定

メモリ設定ページは、altmemphy メガファンクションのメモリ設定ページと同じです。



メモリ設定について詳しくは、「[altmemphy メガファンクション・ユーザガイド](#)」を参照してください。

PHY 設定

ボード・スキューは、すべてのメモリ・インタフェース信号のスキューで、クロック、アドレス、コマンド、データ、マスク、およびストロブ信号が含まれます。



PHY 設定について詳しくは、「[altmemphy メガファンクション・ユーザガイド](#)」を参照してください。

コントローラ 設定

図 3-1 に、コントローラの設定を示します。

☒ 3-1. Controller Settings

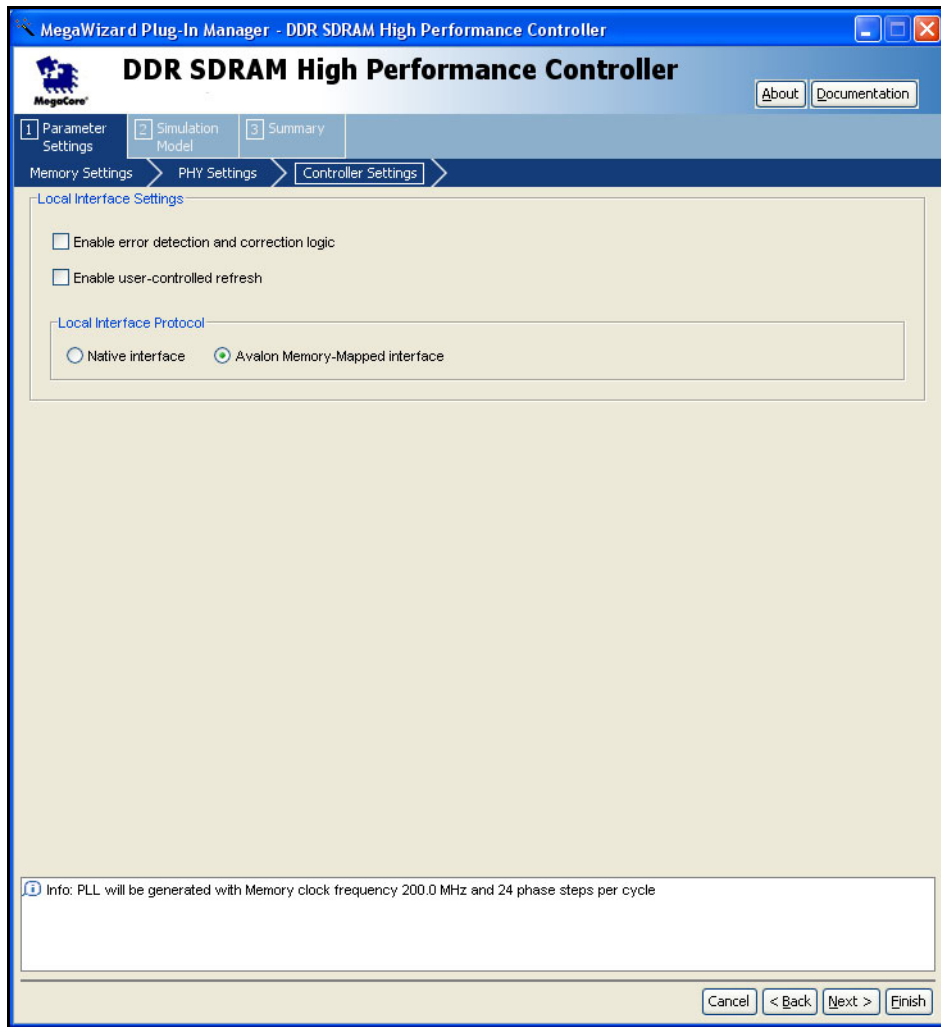


表 3-1 に、コントローラ設定を示します。

表 3-1. Controller Settings		
パラメータ	範囲	説明
Enable error correction and detection logic	オンまたはオフ	オンにして、デザインに ECC を追加します。4-5 ページの「ECC」を参照してください。
Enable user controlled refresh	オンまたはオフ	リフレッシュのユーザ制御に対してオンにします。4-19 ページの「ユーザ・リフレッシュ・コントロール」を参照してください。
Local Interface Protocol	ネイティブまたは Avalon メモリ・マップド	ユーザ・ロジックとメモリ・コントローラ間のローカル・インタフェースを指定します。Avalon® メモリ・マップド (MM) インタフェースを使用して、他の Avalon-MM ペリフェラルに簡単に接続することができます。

ブロックの 説明

DDR および DDR2 SDRAM 高性能コントローラは、暗号化されたコントロール・ロジックと `altmemphy` メガファンクションをインスタンス化します。

`altmemphy` メガファンクションについて詳しくは、「[altmemphy メガファンクション・ユーザガイド](#)」を参照してください。

図 4-1 に、DDR & DDR2 SDRAM 高性能コントローラのブロック図を示します。

図 4-1. DDR & DDR 2 SDRAM 高性能コントローラのブロック図

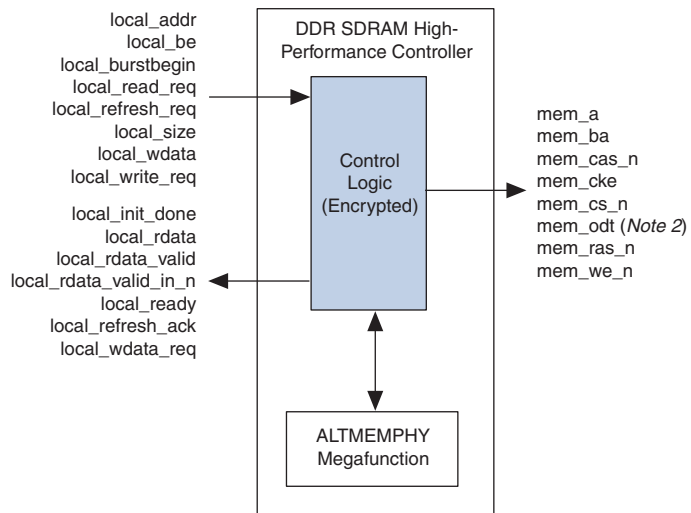


図 4-1 の注：

(1) DDR2 SDRAM 高性能コントローラのみ。

コントロール・ロジック

バス・コマンドは、`mem_ras_n`、`mem_cas_n`、および `mem_we_n` 信号を組み合わせ使用して、SDRAM デバイスを制御します。例えば、3 つの信号がすべて High のクロック・サイクルでは、関連コマンドは

ノー・オペレーション (NOP) です。また、NOP コマンドはチップ・セレクト信号がアサートされていないときも示されます。表 4-1 に、標準 SDRAM バス・コマンドを示します。

表 4-1. バス・コマンド				
コマンド	コード	ras_n	cas_n	we_n
No operation (ノー・オペレーション)	NOP	High	High	High
Active (アクティブ)	ACT	Low	High	High
Read (読み出し)	RD	High	Low	High
Write (書き込み)	WR	High	Low	Low
Burst terminate (バースト終了)	BT	High	High	Low
Precharge (プリチャージ)	PCH	Low	High	Low
Auto refresh (オート・リフレッシュ)	ARF	Low	Low	High
Load mode register (ロード・モード・レジスタ)	LMR	Low	Low	Low

DDR および DDR2 SDRAM 高性能コントローラは、SDRAM バンクを開いてから、その SDRAM バンクのアドレスにアクセスする必要があります。開かれるロウとバンクは、アクティブ (ACT) コマンドと同時に取り込まれます。DDR および DDR2 SDRAM 高性能コントローラは、別のロウにアクセスする必要がある場合にはバンクを閉じてから再度開きます。プリチャージ (PCH) コマンドは、バンクのみ閉じます。

SDRAM へのアクセスに使用される主なコマンドは、リード (RD) およびライト (WR) です。WR コマンドが発行されると、最初のカラム・アドレスとデータ・ワードが取り込まれます。RD コマンドが発行されると、最初のアドレスが取り込まれます。最初のデータは、2 ~ 3 クロック・サイクル後にデータ・バスに現れます (DDR2 SDRAM の場合は 3 ~ 5 クロック・サイクル後)。この遅延はカラム・アドレス・ストロープ (CAS) レイテンシであり、内部 DRAM コアの読み出しからバス上にデータが現れるまでに時間を要するためです。CAS レイテンシは、SDRAM のスピードとメモリ・クロックの周波数によって異なります。一般に、クロックが速いほど、多くの CAS レイテンシのサイクルが必要です。初期 RD または WR コマンドの後、バースト長に到達するまで、またはバースト終了 (BT) コマンドが発行されるまで、順次リードおよびライトが継続します。DDR および DDR2 SDRAM デバイスは、

2、4、または 8 データ・サイクルのバースト長をサポートしています。オート・リフレッシュ・コマンド (ARF) は、データを確実に保持するために周期的に発行されます。このファンクションは、DDRまたはDDR2 SDRAM 高性能コントローラによって実行されます。

ロード・モード・レジスタ・コマンド (LMR) は、SDRAM モード・レジスタをコンフィギュレーションします。このレジスタは、CAS レイテンシ、バースト長、およびバースト・タイプを格納します。



詳しくは、使用している SDRAM の仕様を参照してください。

レイテンシ

メモリ・コントローラのデザインで検討する必要があるレイテンシは、リード・レイテンシとライト・レイテンシの 2 種類です。リード・レイテンシとライト・レイテンシの定義は、以下のとおりです。

- リード・レイテンシは、リード要求を開始した後、リード・データがローカル・インタフェースに現れるのに要する時間です。
- ライト・レイテンシは、ライト要求を開始した後、ライト・データがメモリ・インタフェースに現れるのに要する時間です。

レイテンシの計算は、以下の前提条件に基づきます。

- 読み出しと書き込みは既に開いているロウに対して行う。
- local_ready 信号が High にアサートされている (ウェイト・ステートなし)
- レイテンシはローカル・サイドの周波数と絶対時間 (ns) を使用して定義されている。



ハーフ・レート・コントローラの場合、ローカル・サイド周波数はメモリ・インタフェース周波数の 1/2 で、フル・レート・コントローラの場合は、メモリ・インタフェース周波数と同じです。

アルテラでは、リード・レイテンシとライト・レイテンシを、ローカル・インタフェース・クロック周波数とメモリ・コントローラの絶対時間で定義しています。

リード・レイテンシは以下のように定義されます。

$$\text{リード・レイテンシ} = \text{コントローラ・レイテンシ} + \text{コマンド出力レイテンシ} + \text{CAS レイテンシ} + \text{PHY リード・データ入力レイテンシ}$$

ライト・レイテンシは以下のように定義されます。

$$\text{ライト・レイテンシ} = \text{コントローラ・レイテンシ} + \text{コマンド出力レイテンシ} + \text{ライト・データ・レイテンシ}$$

表 4-2 に、リード・レイテンシとライト・レイテンシのコンポーネントの定義を示します。

用語	説明
コントローラ・レイテンシ	local_read_req から control_doing_rd まで。
コマンド出力レイテンシ	control_doing_rd から mem_cs_n まで。
CAS レイテンシ	リード・コマンドからバス上に DQ データが現れるまで。
PHY リード・データ入力レイテンシ	ローカル・インタフェースに現れるリード・データ。
ライト・データ・レイテンシ	メモリ・インタフェースに現れるライト・データ。

表 4-2 に、ハーフ・レートおよびフル・レート・コントローラおよび Cyclone III、Stratix II、および Stratix III デバイスのライト・レイテンシおよびリード・レイテンシ定義から派生するリード・レイテンシおよびライト・レイテンシを示します。

デバイス	コントローラ・レート	周波数 (MHz)	レイテンシ・タイプ	レイテンシ (サイクル)	レイテンシ (ns)
Cyclone III	ハーフ	200	読み出し	$6 + 2.5 + 1.5 + 7 = 17$	85
			書き込み	$7 + 1.5 + 1 = 9.5$	47
	フル・サポート	167	読み出し	$5 + 2 + 3 + 9 = 19$	114
			書き込み	$7 + 0 + 2 = 9$	60
Stratix II	ハーフ	333	読み出し	$6 + 2.5 + 1.5 + 7 = 17$	51
			書き込み	$7 + 1.5 + 1 = 9.5$	28.5
	フル・サポート	267	読み出し	$5 + 2 + 3 + 9 = 19$	71
			書き込み	$7 + 0 + 2 = 9$	30
Stratix III	ハーフ	400	読み出し	$6 + 3.5 + 1.5 + 9 = 20$	50
			書き込み	$7 + 2 + 1.5 = 10.5$	26
Stratix III	フル・サポート	333	読み出し	$5 + 3 + 3 + 9 = 20$	60
			書き込み	$7 + 1 + 2 = 11$	33



レイテンシはコンフィギュレーションの正確さに依存します。正確なレイテンシはシミュレーションから取得する必要がありますが、この値は自動キャリブレーション・プロセスのためハードウェアでは異なる可能性があります。

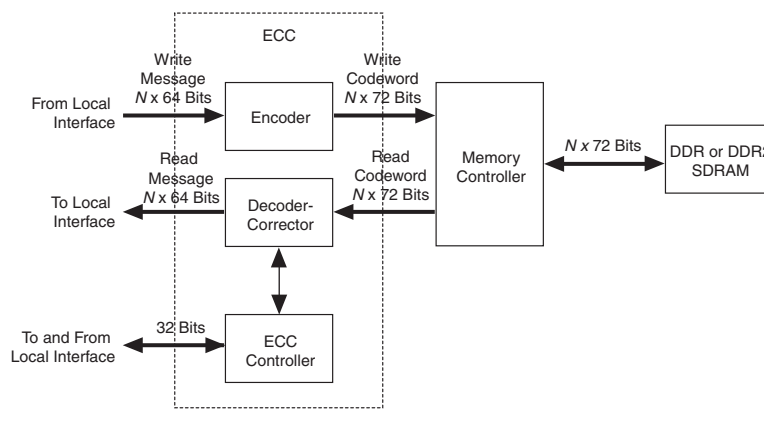
ECC

オプションの誤り訂正コード (ECC) は、エンコーダとデコーダ・コレクタで構成されており、シングル・ビット・エラーの検出と訂正、ダブル・ビット・エラーの検出を行います。ECC は各 64 ビット・メッセージに 8 ビット ECC を使用します。ECC は以下の特長を備えています。

- Hamming コード ECC: 64 ビットの各データを、8 ビットの Hamming コード・パリティ・ビットを持つ 72 ビットのコードワードにエンコードします。
- レイテンシ:
 - 書き込み時、最大 1 または 2 クロック遅延
 - 読み出し時、最大 1 または 3 クロック遅延
- すべてのシングル・ビット・エラーを検出し、訂正します。また、ECC はシングル・ビット・エラーのユーザ定義スレッシュホールドに達すると割り込みを送信します。
- すべてのダブル・ビット・エラーを検出します。また、ECC はダブル・ビット・エラーのユーザ定義スレッシュホールドに達すると、ダブル・ビット・エラー数をカウントして、割り込みを送信します。
- 部分的書き込みを受け入れます。
- 強制エラーを作成して、ECC の動作をチェックします。
- 安定した状態でパワー・アップします。

図 5 に、ECC のブロック図を示します。

図 5. ECC のブロック図



ECC は以下のブロックで構成されます。

- エンコーダ — 64 ビット・メッセージを 72 ビット・コードワードにエンコードします。
- デコーダ・コレクタ — 可能な場合、72 ビット・コードワードをデコードおよび訂正します。
- ECC コントローラ — 複数のエンコーダとデコーダ・コレクタを制御するため、ECC はさまざまなバス幅を扱うことができます。また、エンコーダおよびデコーダ・コレクタの以下の機能も制御します。
 - 割り込み：
 - 検出および訂正されたシングル・ビット・エラー
 - 検出されたダブル・ビット・エラー
 - シングル・ビット・エラー・カウンタ・スレッショルドの超過
 - ダブル・ビット・エラー・カウンタ・スレッショルドの超過
 - コンフィギュレーション・レジスタ：
 - シングル・ビット・エラー検出カウンタ・スレッショルド
 - ダブル・ビット・エラー検出カウンタ・スレッショルド
 - 最初に検出されたエラーまたは最新エラーのステータスのキャプチャ
 - テストを目的とする ECC の意図的な破損
 - ステータス・レジスタ：
 - エラー・アドレス
 - エラー・タイプ：シングル・ビット・エラーまたはダブル・ビット・エラー
 - 対応するバイト・エラー ECC シンドローム

- エラー信号 — データ・ワードに対応するエラー信号がデータと共に供給され、訂正不能なダブル・ビット・エラーがリターン・データ・ワードで発生した場合に High になります。
- カウンタ：
 - 検出および / または訂正されたシングル・ビット・エラー
 - 検出されたダブル・ビット・エラー



ECC レジスタについて詳しくは、[付録 A、ECC レジスタの説明](#)を参照してください。

ECC は、それぞれパラレルに動作する複数のエンコーダをインスタンス化し、データ・ワードの幅が 64 の整数倍であると仮定して任意のデータ・ワード幅をエンコードすることができます。

ECC は、ローカル（ネイティブまたは Avalon-MM インタフェース）およびメモリ・コントローラ間で動作します。

ECC は、ローカル・インタフェースと ECC 間に $N \times 64$ ビット（ N は整数）幅のインタフェースを持ち、ローカル・インタフェースからデータを受信および返信するために使用されます。このインタフェースは、ネイティブ・インタフェースまたは Avalon-MM スレーブ・インタフェースのどちらも可能であり、インタフェースのタイプは MegaWizard Plug-In で選択します。

ECC は、ローカル・インタフェースと ECC 間に第 2 インタフェースとして、ECC コントローラの動作ステータスを制御およびレポートする 32 ビット幅の Avalon-MM スレーブ・インタフェースを持っています。

ECC からのエンコードされたデータは、ECC とメモリ・コントローラ間にある $N \times 72$ ビット幅の Avalon-MM マスタ・インタフェースを使用して、メモリ・コントローラに送信されます。

DDR SDRAM 高性能コントローラをテストするときには、ECC をオフにできます。

割り込み

ECC は以下のいずれかのシナリオが発生すると、割り込みを発行します。

- シングル・ビット・エラー・カウンタが、設定された最大シングル・ビット・エラー・スレッショルド値に達した。
- ダブル・ビット・エラー・カウンタが、設定された最大ダブル・ビット・エラー・スレッショルド値に達した。

リターン・データ・ワードの N 個のすべての部分に対して、対応するイベントが発生するたびに、エラー・カウンタがインクリメントされます。このインクリメント値は、最大スレッシュホールドと比較され、最大スレッシュホールドと同じときは割り込み信号が送信されます。対応するステータス・レジスタに 1 を書き込むと、ECC は割り込みをクリアします。コントロール・ワードを使用して、いずれかのカウンタからの割り込みをマスクできます。

部分的書き込み

ECC は部分的書き込みをサポートします。アドレス信号、データ信号、およびバースト信号に加え、Avalon-MM インタフェースは、バイト・イネーブルのための信号ベクタもサポートします。この信号ベクタの各ビットは、データ・バス上の 1 バイトを表します。したがって、これらの任意のビットの 0 は、コントローラがその特定の場所へ書き込みを行えないようにする信号です（部分的書き込み）。部分的書き込みの場合、ECC は以下のステップを実行します。

- 部分的書き込み条件を受信すると、Avalon-MM インタフェースからのそれ以降のリード・コマンドまたはライト・コマンドを停止します。
- 同時に、部分的書き込みアドレスに対する自己生成リード・コマンドをメモリ・コントローラに送信します。
- 特定のアドレスに対して、メモリ・コントローラからリターン・データを受信すると、ECC はデータをデコードし、エラーをチェックし、そのデータを ECC コントローラに送信します。
- ECC コントローラは、訂正されたデータワードまたは正しいデータワードと受信情報をマージします。
- 更新されたデータワードを、エンコーディングのためにエンコーダに送信し、次にライト・コマンドでメモリ・コントローラに送信します。
- Avalon-MM インタフェースからのコマンドの停止を解除します。これにより、Avalon-MM インタフェースは新しいコマンドを受信できるようになります。

以下のコーナー・ケースが発生する可能性があります。

- リード・モディファイ・ライト・プロセスのリード・フェーズ中のシングル・ビット・エラーこの場合、シングル・ビット・エラーが最初に訂正され、シングル・ビット・エラー・カウンタがインクリメントされ、この訂正済みのデコードされたデータ・ワードに部分的書き込みが実行されます。
- リード・モディファイ・ライト・プロセスのリード・フェーズ中のダブル・ビット・エラーこの場合、ダブル・ビット・エラー・カウンタがインクリメントされ、Avalon-MM インタフェースを介して割り込みが送信されます。新しいライト・ワードは、その場所へ書

き込まれません。この条件は割り込みステータス・レジスタの個別フィールドにハイライトされます。

部分的バースト

一部の DIMM には DM ピンがないため部分的バーストをサポートしません。最低 4 ワードを同時にメモリに書き込む必要があります。部分的バースト書き込みの場合、ECC は、部分的書き込みと同様のメカニズムを提供します。

部分的バーストの場合、ネイティブ・インタフェースからの書き込みデータは最大バースト・サイズ深度の 64 ビット幅 FIFO バッファに格納され、パラレルの場合には対応するアドレスのリード・コマンドが DIMM に送信されます。ネイティブ・インタフェースからの以降のコマンドは、現在のバーストが読み出し、修正、およびメモリ・コントローラへの書き戻しが行われるまで停止します。

ECC レイテンシ

ECC を使用すると、以下のレイテンシの変更が生じます。

バースト長 1

ローカル・バースト長 1 の場合、ライト・レイテンシは 1 クロック・サイクルだけ増加し、リード・レイテンシは 1 クロック・サイクルだけ増加します (チェックおよび訂正を含む)。

部分的書き込みでは、ECC コントローラで読み出しとそれに続く書き込みが発生するため、レイテンシはコントローラが特定のアドレスからデータをフェッチするのに要する時間によって異なります。

表 4-4 に、バースト長とレートの関係を示します。

ローカル・バースト長	レート	メモリ・バースト長
1	ハーフ	4
2	フル・サポート	4

バースト長 2

ローカル・バースト長 2 の場合、ライト・レイテンシは 2 クロック・サイクルだけ増加し、リード・レイテンシは 1 クロック・サイクルだけ増加します (チェックおよび訂正を含む)。

部分的書き込みでは、ECC コントローラで読み出しとそれに続く書き込みが発生するため、レイテンシはコントローラが特定のアドレスからデータをフェッチするのに要する時間によって異なります。

シングル・ビット・エラーの場合、メモリの自動訂正は、リード・サイクルを停止せずに行われ（可能な場合）、訂正中は ECC コントローラへの追加コマンドは停止されます。

デザイン例

MegaWizard® Plug-In Manager は、DDR または DDR2 SDRAM 高性能コントローラのインスタンス化および接続方法を示すデザイン例を作成します。このデザイン例は、DDR または DDR2 SDRAM 高性能コントローラ、そのコントローラにリードおよびライト要求を発行するいくつかのドライバ・ロジック、必要なクロックを作成する PLL、および DLL (Stratix シリーズのみ) で構成されます。このデザイン例は、コンパイルしてスタティック・タイミング・チェックとボード・テストの両方に使用できるワーキング・システムです。

図 4-2 に、テストベンチとデザイン例を示します。

図 4-2. テストベンチとデザイン例

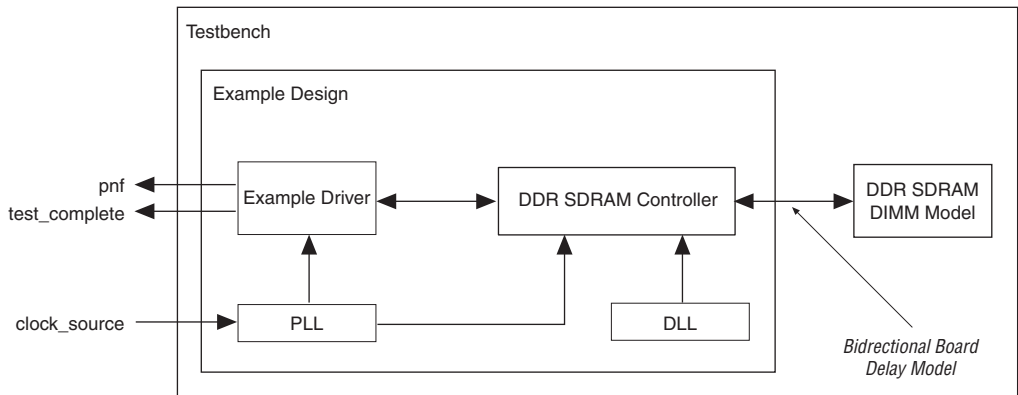


表 4-5 に、このデザイン例とテストベンチに関連するファイルを示します。

ファイル名	説明
<variation name>_example_top_tb.v または .vhd	デザイン例のテストベンチ。
<variation name>_example_top.v または .vhd	デザイン例。
<variation name>_phy_alt_mem_phy_pll_<device>.v または .vhd (1)	サンプル PLL。
<variation name>_example_driver.v または .vhd	サンプル・ドライバ。
<variation name>.v または .vhd	カスタム MegaCore ファンクションのトップレベルの記述。

表 4-5 の注：

(1) <device> は、デバイスの短縮名で、例えば Stratix II デバイスは sii です。

このサンプル・ドライバは、DDR または DDR2 SDRAM 高性能コントローラのセルフ・チェック・テスト・ジェネレータです。このサンプル・ドライバは、ステート・マシンを使用して、すべてのメモリ・バンクにおいて一定のロウ・アドレス範囲内で、一定のカラム・アドレス範囲にデータ・パターンを書き込みます。次に、同じ場所からデータを読み戻し、データが一致することを確認します。リード・データの比較に失敗した場合は、pass not fail (pnf) 出力が Low に遷移します。また、バイト単位での比較を示す pnf_per_byte 出力もあります。ライトまたはリード・テスト・シーケンスの最後のクロック・サイクルの間、test_complete 出力は High に遷移します。この遷移後、テストは最初から再開されます。

使用するデータ・パターンは、バイトあたり 8 ビットの LFSR を使用して生成され、LFSR ごとに異なる初期化シードを持ちます。

テストベンチは、DDR または DDR2 SDRAM モデルのプレースホルダ、PLL の基準クロックをインスタンス化します。test_complete が High で検出されると、テストに合格したかどうかを示すテスト終了メッセージが出力されます。



アルテラはメモリ・シミュレーション・モデルを提供していません。メモリ・シミュレーション・モデルは、メモリ・ベンダから入手する必要があります。



シミュレーション・スクリプトの実行方法については、2-9 ページの「デザイン例のシミュレーション」を参照してください。

インタフェースおよび信号

この項では、以下の内容について説明します。

- 4-12 ページの「インタフェースの説明」
- 4-24 ページの「信号」

インタフェースの説明

この項では、以下のローカル・サイドのインタフェース要求について説明します。

- 4-12 ページの「ライト」
- 4-15 ページの「リード」
- 4-17 ページの「リード-ライト-リード-ライト」
- 4-17 ページの「リード-ライト-リード-ライト」
- 4-20 ページの「DDR SDRAM の初期化タイミング」
- 4-22 ページの「DDR2 SDRAM の初期化タイミング」



これらのインタフェース要求は、ネイティブ・インタフェース用です。Avalon™ Memory-Mapped (Avalon-MM) インタフェースについては、「Avalon Memory-Mapped Interface Specification」を参照してください。



図に示すのは、すべてフル・レート・コントローラです。

ライト

4-13 ページの図 4-3 に、サイズの異なる 3 つのライト要求を示します。最初の 2 つは、シーケンシャル・アドレスへのライト要求で、3 つ目は新しいロウおよびバンクへのライト要求です。コントローラでは、メモリ・デバイスで設定される最大バースト長以下の任意のバースト長を使用できます。例えば、DDR SDRAM メモリにバースト長 8 を選択した場合、コントローラではバースト長 1、2、3、および 4 (DDR SDRAM サイドでは 2、4、6、および 8) を使用できます。



コンセプトは DDR2 SDRAM と同様ですが、バースト長 1 と 2 (DDR2 SDRAM サイドでは 2 と 4) しか使用できません。

図 4-3. ライト

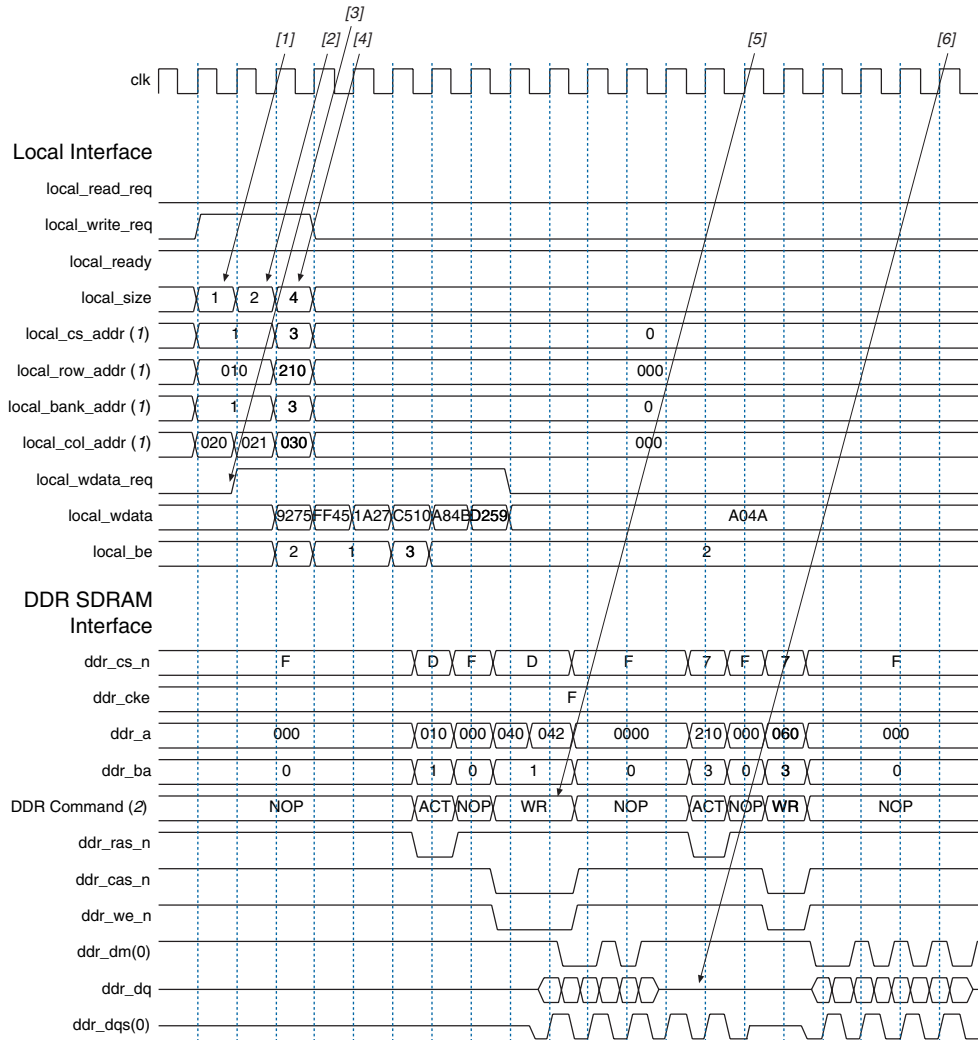


図 4-3 の注 :

- (1) local_cs_addr、local_row_addr、local_bank_addr、および local_col_addr 信号は、local_addr 信号の一部です。
- (2) DDR Command は、コマンド信号(mem_ras_n、mem_cas_n、および mem_we_n)が発行するコマンドを示します。

1. ユーザ・ロジックは、`local_write_req` 信号とこのライトのサイズおよびアドレスをアサートすることによって、最初のライトを要求します。この例では、要求はチップ・セレクト 1 への長さ 1 (DDR SDRAM サイドでは 2) のバーストです。`local_ready` 信号がアサートされており、これはコントローラがこの要求を受け付けたことを示し、ユーザ・ロジックは次のクロック・サイクルで、別のリードまたはライトを要求できます。`local_ready` 信号がアサートされていなかった場合、ユーザ・ロジックはアサートされた、ライト要求、サイズ、およびアドレス信号を保持する必要があります。



`local_be` はアクティブ High、`mem_dm` はアクティブ Low です。`local_wdata` および `local_be` を `mem_dq` および `mem_dm` にマップするには、32 ビット `local_wdata` および 16 ビット `mem_dq` を持つフル・レート・デザインを検討します。

```
Local_wdata = <22334455> <667788AA> <BBCCDDEE>
Local_be = <1100> <0110> <1010>
```

これらの値は以下のようにマップします。

```
Mem_dq = <4455><2233><88AA><6677><DDEE><BBCC>
Mem_dm = <1 1> <0 0> <0 1> <1 0> <0 1> <0 1>
```

2. ユーザ・ロジックは、シーケンシャル・アドレスへの 2 回目のライトを要求します。今回のサイズは 2 (DDR SDRAM サイドでは 4) です。`local_ready` 信号は、アサートされたままで、コントローラが要求を受け付けたことを示します。
3. コントローラはユーザ・ロジックからの最初の書き込み要求に対して、ライト・データとバイト・イネーブルを要求します。ライト・データとバイト・イネーブルは、要求の 1 クロック・サイクル後に提示する必要があります。この例では、コントローラは以降の書き込みに対してライト・データも継続して要求します。ユーザ・ロジックは、ライトを要求するときには、バースト全体に対してライト・データを供給できなければなりません。
4. ユーザ・ロジックは、異なるチップ・セレクトへの 3 回目のライトを要求します。コントローラは最大 4 つの要求をバッファできるため、`local_ready` 信号は High のまま維持され、この要求は受け付けられます。
5. 必要なバンク・アクティベーション・コマンドを発行済みの場合、コントローラは最初の 2 つのライト要求を順次、メモリ・デバイスに発行します。

6. メモリに書き込むデータがない場合でも、mem_dqs 信号は、メモリ・デバイスのバースト長（この例では 8）の全長に対して、トグルを継続する必要があります。

リード

4-16 ページの図 4-4 に、サイズの異なる 3 つのリード要求を示します。コントローラでは、メモリ・デバイスで設定される最大バースト長以下の任意のバースト長を使用できます。例えば、DDR SDRAM メモリにバースト長 8 を選択した場合、コントローラではバースト長 1、2、3、および 4（DDR SDRAM サイドでは 2、4、6、および 8）を使用できます。



コンセプトは DDR2 SDRAM と同様ですが、バースト長 1 と 2（DDR2 SDRAM サイドでは 2 と 4）しか使用できません。

図 4-4. リード

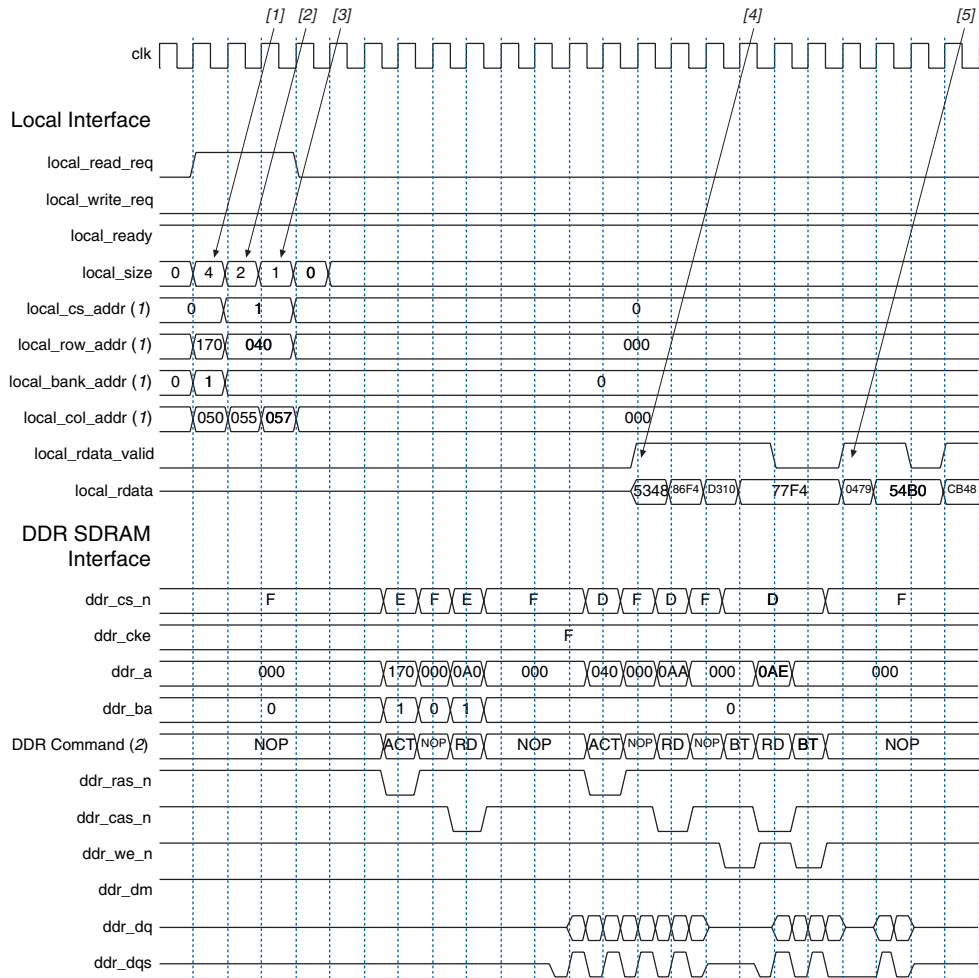


図 4-4 の注：

- (1) local_cs_addr、local_row_addr、local_bank_addr、および local_col_addr 信号は、local_addr 信号の一部です。
- (2) DDR Command は、コマンド信号が発行するコマンドを示します。

1. ユーザ・ロジックは、local_read_req 信号とこのリードのサイズおよびアドレスをアサートすることによって、最初のリードを要求します。この例では、要求は長さ 4 (DDR SDRAM サイドでは 8) のバーストです。local_ready 信号がアサートされており、これはコントローラがこの要求を受け付けたことを示し、ユーザ・ロジック

クは次のクロック・サイクルで、別のリードまたはライトを要求できます。local_ready 信号がアサートされていなかった場合、ユーザ・ロジックはアサートされた、リード要求、サイズ、およびアドレス信号を保持する必要があります。

2. ユーザ・ロジックは、別のアドレスへの 2 回目のリードを要求します。今回のサイズは 2 (DDR SDRAM サイドでは 4) です。local_ready 信号は、アサートされたままで、コントローラが要求を受け付けたことを示します。
3. ユーザ・ロジックは、別のアドレスへの 3 回目のリードを要求します。今回のサイズは 1 (DDR SDRAM サイドでは 2) です。local_ready 信号は、アサートされたままで、コントローラが要求を受け付けたことを示します。
4. コントローラは、local_rdata_valid 信号をアサートすることによって、最初の要求のリード・データを返します。コントローラが要求を受け付けて、データを返すまでの正確なクロック・サイクル数は、コントローラ内で保留になっている他の要求数、メモリの状態、およびメモリのタイミング要件 (CAS レイテンシなど) によって異なります。
5. コントローラは後続のリード要求のリード・データを返します。

リード - ライト - リード - ライト

4-18 ページの図 4-5 に、インタリーブド・リードおよびライトのシーケンスを示します。

図 4-5. リード - ライト - リード - ライト

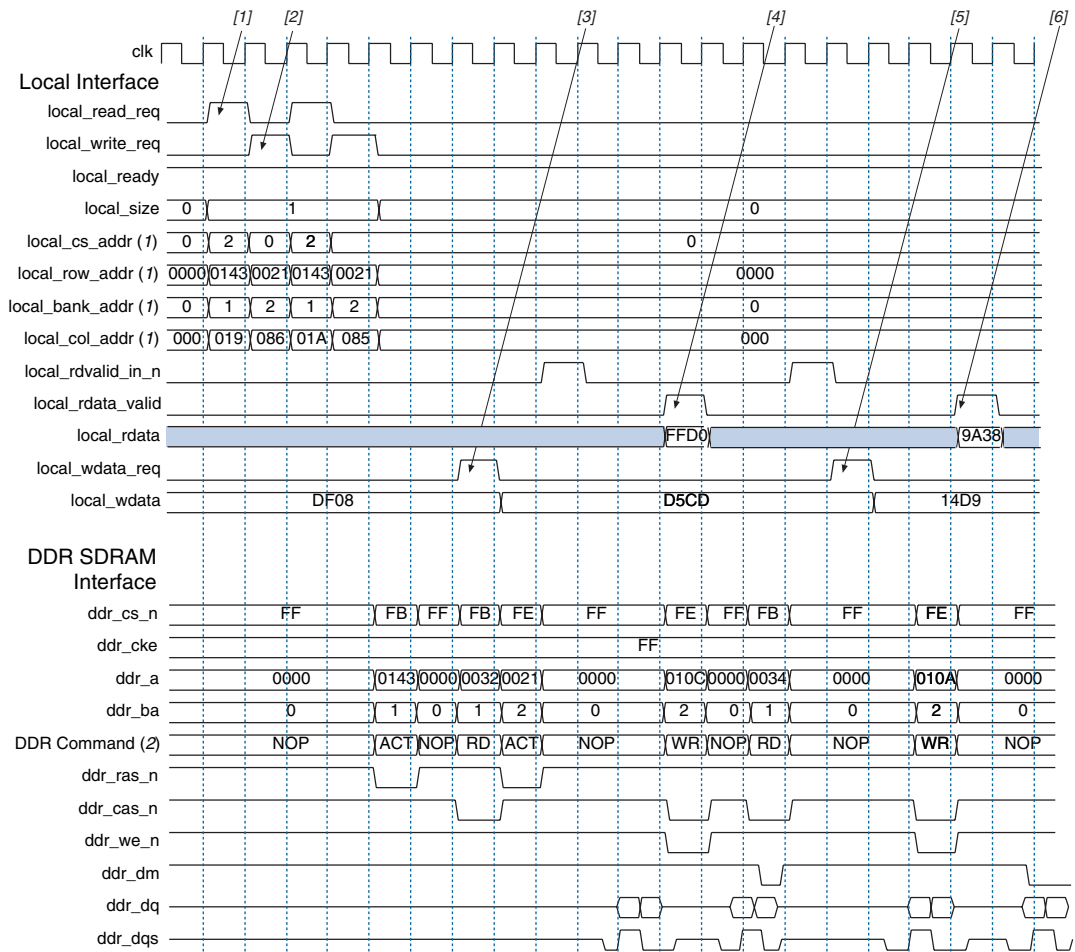


図 4-5 の注：

- (1) local_cs_addr、local_row_addr、local_bank_addr、および local_col_addr 信号は、local_addr 信号の一部です。
- (2) DDR Command は、コマンド信号が発行するコマンドを示します。

1. ユーザ・ロジックは、local_read_req 信号とそのリードのサイズおよびアドレスをアサートすることによって、リードを要求します。local_ready 信号が High なので、この要求は受け付けられたとみなすことができます。
2. ユーザ・ロジックは、ライト、リード、および他のライト要求を要求し、これらが受け付けられます。

3. コントローラは、ライト・データ要求信号をアサートし、ユーザ・ロジックに対して、次のクロック・エッジで有効なライト・データとバイト・イネーブルを提示するよう要求します。
4. 最初のリード要求からのリード・データが返され、リード・データ有効信号によって有効としてマークされます。
5. コントローラは、再び 2 回目のライト要求のためにライト・データ要求をアサートします。
6. 2 回目のリード要求からのリード・データが返されます。

ユーザ・リフレッシュ・コントロール

図 4-6 に、ユーザ・リフレッシュ・コントロール・インタフェースを示します。この機能により、コントローラがメモリにリフレッシュを発行するタイミングを制御できます。この機能により、ワースト・ケース・レイテンシをさらに細かく制御し、リフレッシュをバーストで発行してアイドル期間を利用することができます。

図 4-6. ユーザ・リフレッシュ・コントロール

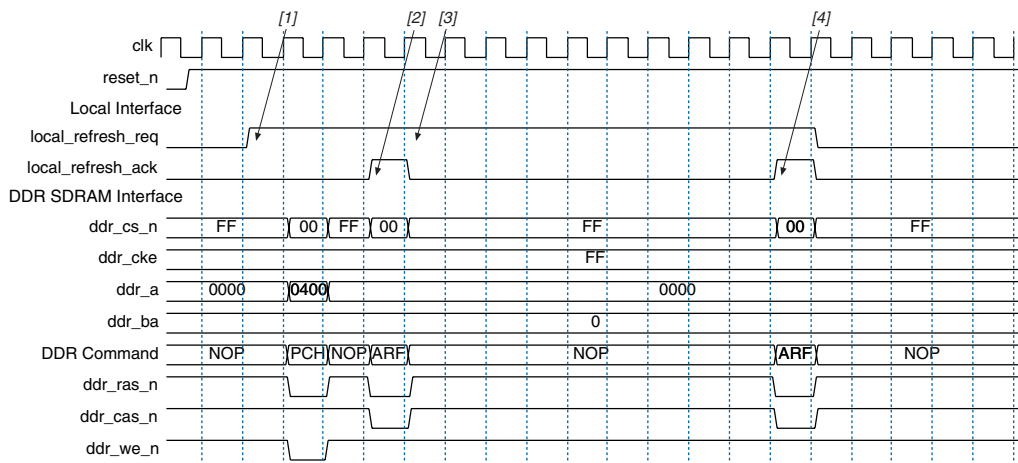


図 4-6 の注：

(1) DDR Command は、コマンド信号が発行するコマンドを示します。

1. ユーザ・ロジックは、リフレッシュ要求信号をアサートして、コントローラにリフレッシュを実行する必要があることを示します。コントローラがリフレッシュ要求に優先権を設定するため、リードおよびライト要求信号の状態は問題ではありません（ただし、コントローラは現在アクティブなリードまたはライトを完了します）。

2. コントローラはリフレッシュ確認信号をアサートして、リフレッシュを発行したことを示します。この信号は、ユーザ・リフレッシュ・コントロール・オプションがオンになっていない場合でも使用でき、ユーザ・ロジックはコントローラがリフレッシュを発行するタイミングを追跡できます。
3. ユーザ・ロジックは、アサートされたリフレッシュ要求信号を維持して、別のリフレッシュ要求を実行する希望を示します。

コントローラは再びリフレッシュ確認信号をアサートして、リフレッシュを発行したことを示します。この時点で、ユーザ・ロジックはリフレッシュ要求信号をデアサートし、コントローラはバッファのリードおよびライトを継続します。

DDR SDRAM の初期化タイミング



DDR SDRAM と DDR2 SDRAM では、初期化タイミングが異なります。DDR2 SDRAM の初期化タイミングについては、[4-22 ページの「DDR2 SDRAM の初期化タイミング」](#)を参照してください。

DDR SDRAM 高性能コントローラは、以下のメモリ・コマンド・シーケンスを発行することによって、SDRAM デバイスを初期化します。

- NOP (200 μ s、プログラム可能)
- PCH
- 拡張 LMR (ELMR)
- LMR
- NOP (200 クロック・サイクル、固定)
- PCH
- ARF
- ARF
- LMR

最後の LMR コマンドの発行後、メモリ・コントローラはメモリの制御を `altmemory` メガファンクションに渡し、`altmemory` メガファンクションがキャリブレーション・プロセスを実行できるようになります。



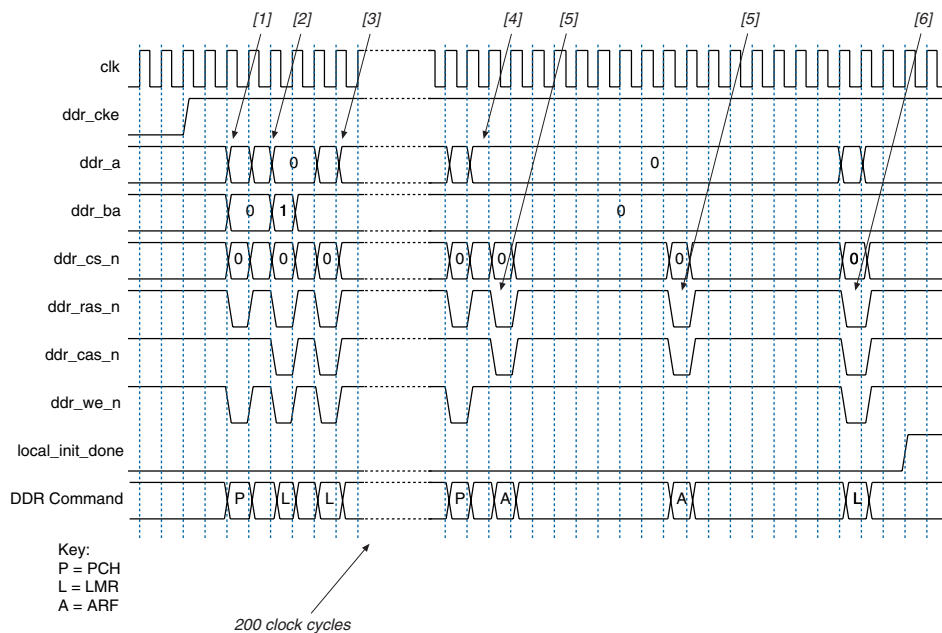
詳しくは、「[altmemory メガファンクション・ユーザガイド](#)」を参照してください。

[4-21 ページの図 4-7](#) に、標準的な初期化タイミング・シーケンスを示します。これについては以下で説明します。リセットから最初の PCH コマンドまでの時間は、200 μ s でなければなりません。シミュレーション・テストでは、MegaWizard Plug-In Manager の **memory initialization time at power up t(INIT)** パラメータを設定して、この時間を短縮できます。



tINIT をゼロに設定しないでください。

図 4-7. DDR SDRAM デバイスの初期化タイミング



1. PCH コマンドは、プリチャージ・ピン、アドレス・ビット a[10]、または a[8] を High に設定すると、すべてのバンクに送信されます。
2. ELMR コマンドは、メモリ・デバイスの内部 DLL (Delay-Locked Loop) をイネーブルするために発行されます。ELMR コマンドは、バンク・アドレス・ビットが拡張モード・レジスタを指定するように設定された LMR コマンドです。
3. LMR コマンドは、CAS レイテンシやバースト長などのメモリの動作パラメータを設定します。この LMR コマンドは、メモリ・デバイスの内部 DLL をリセットするのにも使用されます。DDR SDRAM 高性能コントローラでは、DLL リセットの後、メモリに次のコマンドを発行する前に 200 クロック・サイクルが経過します。
4. もう一度 PCH コマンドですべてのバンクをアイドル状態にします。
5. PCH コマンドの後に、2つの ARF コマンドを発行する必要があります。
6. 最後の LMR コマンドは、DLL をリセットすることなく動作パラメータをプログラムします。

7. 最後の LMR コマンドの発行後、メモリ・コントローラはメモリの制御を `altmemphy` メガファンクションに渡し、`altmemphy` メガファンクションがキャリブレーション・プロセスを実行できるようになります。



詳しくは、「[altmemphy メガファンクション・ユーザガイド](#)」を参照してください。

`altmemphy` がキャリブレーションを終了すると、メモリ・コントローラは、メモリ・デバイスが初期化されたことを示す `local_init_done` 信号をアサートします。

DDR2 SDRAM の初期化タイミング

DDR2 SDRAM 高性能コントローラは、以下のコマンド・シーケンスを発行することによって、メモリ・デバイスを初期化します。

- NOP (200 μ s、プログラム可能)
- PCH
- ELMR、レジスタ 2
- ELMR、レジスタ 3
- ELMR、レジスタ 1
- LMR
- PCH
- ARF
- ARF
- LMR
- ELMR、レジスタ 1
- ELMR、レジスタ 1

最後の ELMR コマンドの発行後、メモリ・コントローラはメモリの制御を `altmemphy` メガファンクションに渡し、`altmemphy` メガファンクションがキャリブレーション・プロセスを実行できるようになります。



詳しくは、「[altmemphy メガファンクション・ユーザガイド](#)」を参照してください。

4-23 ページの図 4-8 に、標準的な DDR2 SDRAM 初期化タイミング・シーケンスを示します。これについては以下で説明します。リセットからクロック・イネーブル信号が High になるまでの時間は、200 μ s でなければなりません。シミュレーション・テストでは、MegaWizard Plug-In Manager の **memory initialization time at power up t(INIt)** パラメータを設定して、この時間を短縮できます。

5. LMR コマンドは、CAS レイテンシやバースト長などのメモリの動作パラメータを設定するために発行されます。この LMR コマンドは、メモリ・デバイスの内部DLLをリセットするのにも使用されます。
6. もう一度 PCH コマンドですべてのバンクをアイドル状態にします。
7. PCH コマンドの後に、2つの ARF コマンドを発行する必要があります。
8. 最後の LMR コマンドは、DLL をリセットすることなく動作パラメータをプログラムするために発行されます。
9. ステップ 5 の後の 200 クロック・サイクルで、メモリ・デバイス・オフチップ・ドライバ (OCD) のインピーダンスをデフォルト値に設定するために、2つの ELMR コマンドが発行されます。
10. 最後の ELMR コマンドの発行後、メモリ・コントローラはメモリの制御を `altmemphy` メガファンクションに渡し、`altmemphy` メガファンクションがキャリブレーション・プロセスを実行できるようになります。



詳しくは、「[altmemphy メガファンクション・ユーザガイド](#)」を参照してください。

`altmemphy` がキャリブレーションを終了すると、メモリ・コントローラは、メモリ・デバイスが初期化されたことを示す `local_init_done` 信号をアサートします。

信号

表 4-6 に、クロックおよびリセット信号を示します。

表 4-6. クロックおよびリセット信号 (1 / 2)		
シンボル	入力 / 出力	説明
<code>global_reset_n</code>	入力	コントローラへの非同期リセット入力。その他のリセット信号はすべて、再同期化されたこの信号から派生したものです。この信号は Low の間、PLL を含む完全な ALTMEMPHY をリセット状態に保持します。
<code>phy_clk</code>	出力	ユーザに供給される ALTMEMPHY クロック。ALTMEMPHY に対するすべてのユーザ入力および出力は、このクロックに同期します。
<code>pll_ref_clk</code>	入力	PLL への基準クロック入力。
<code>reset_phy_clk_n</code>	出力	非同期リセット。関連クロック・ドメインに対して非同期でアサートされ、同期的にディアサートされます。

表 4-6. クロックおよびリセット信号 (2 / 2)

シンボル	入力 / 出力	説明
reset_request_n	出力	PLL 出力がロックされていないことを示すリセット要求出力。任意のシステム・レベル・リセット・コントローラへのリセット要求入力としてこれを使用します。PLL がロックしている間、この信号は常に Low です。したがって、この信号を使用するどのリセット・ロジックにも、レベル検出ではなく、立ち下がりエッジでリセット要求を検出すべきです。
soft_reset_n	入力	コントローラをリセットするための非同期リセット入力。このリセットにより、PHY は立ち下がりエッジを検出すると PLL にリセット信号をパルスし、信号が Low の間 ALTMEMPHY がリセット状態の場合は残りの部分を保持します。

表 4-7 に、DDR および DDR2 SDRAM 高性能コントローラのローカル・インタフェース信号を示します。

表 4-7. ローカル・インタフェース信号 (1 / 3)		
信号名	入力 / 出力	説明
local_addr[]	入力	<p>バーストを開始する必要があるメモリ・アドレス。このバスの幅は以下の式で決まります。</p> <p>1 チップ・セレクトの場合: 幅 = バンク・ビット + ロウ・ビット + カラム・ビット - 1</p> <p>複数のチップ・セレクトの場合: 幅 = チップ・ビット + バンク・ビット + ロウ・ビット + カラム・ビット - 1</p> <p>フル・レート・コントローラの場合、ローカル・データ幅はメモリ・データ・バス幅の 2 倍であるため、メモリ・サイドのカラム・アドレスの最下位ビット (LSB) は無視されます。</p> <p>ハーフ・レート・コントローラの場合、ローカル・データ幅はメモリ・データ・バス幅の 4 倍であるため、メモリ・サイドのカラム・アドレスの 2 つの LSB は無視されます。</p>
local_be[]	入力	<p>ライト時に個々のバイトをマスクするために使用するバイト・イネーブル信号。local_be はアクティブ High、mem_dm はアクティブ Low です。</p> <p>local_wdata および local_be を mem_dq および mem_dm にマップするには、32 ビット local_wdata および 16 ビット mem_dq を持つフル・レート・デザインを検討します。 Local_wdata = < 22334455 > < 667788AA > < BBCCDDEE > Local_be = 1100 0110 1010</p> <p>これらの値は以下のようにマップします。 Mem_dq = <4455><2233><88AA><6677><DDEE><BBCC> Mem_dm = 1 1 0 0 0 1 1 0 0 1 0 1</p>
local_burstbegin	入力	<p>Avalon バースト開始ストローブで、Avalon バーストの開始を示します。この信号を使用できるのは、ローカル・インタフェースが Avalon-MM インタフェースで、メモリ・バースト長が 2 より大きい場合のみです。他のすべての Avalon-MM 信号とは異なり、local_ready がデアサートされた場合、バースト開始信号はアサートされたままにはなりません。</p>
local_read_req	入力	<p>リード要求信号。</p>

表 4-7. ローカル・インタフェース信号 (2 / 3)

信号名	入力 / 出力	説明
local_refresh_req	入力	ユーザ制御リフレッシュ要求。User Controlled Refresh がオンの場合、local_refresh_reqが使用可能になり、メモリ要件を満たすために十分なリフレッシュ要求を発行する必要があります。このオプションにより、複数のリフレッシュ・コマンドを同時に動作させる場合も含めて、メモリにリフレッシュが発行されるタイミングを完全に制御することができます。リフレッシュ要求は、リード要求およびライト要求がすでに処理中でない限り、これらの要求よりも優先されます。
local_size[]	入力	要求された、メモリへのリードまたはライト・アクセスのビット数を制御します。この値は 2 進数としてエンコードされます。値の範囲は、メモリ・バースト長と、ウィザードでフル・レートまたはハーフ・レートのどちらを選択するかによって決まります。 メモリ・バースト長が 4 で、ハーフ・レートの場合、ローカル・バースト長は 1 になるため、local_size は常に 1 でドライブする必要があります。 メモリ・バースト長が 4 で、フル・レートの場合、ローカル・バースト長は 2 で、リード要求またはライト要求ごとに local_size に 1 または 2 を設定する必要があります。
local_wdata[]	入力	ライト・データ・バス。local_wdata の幅は、フル・レート・コントローラの場合はメモリ・データ・バスの 2 倍で、ハーフ・レート・コントローラの場合はメモリ・データ・バスの 4 倍です。
local_write_req	入力	ライト要求信号。
local_init_done	出力	メモリ初期化完了信号。コントローラがメモリの初期化を完了するとアサートされます。リード要求とライト要求は、local_init_done がアサートされる前でも受け付けられますが、それらの要求は安全に発行できるようになるまではメモリに発行されません。
local_rdata[]	出力	リード・データ・バス。local_rdata の幅は、メモリ・データ・バスの 2 倍です。
local_rdata_error	出力	現在のリード・データにエラーがある場合にアサートされます。
local_rdata_valid	出力	リード・データ有効信号。local_rdata_valid 信号は、リード・データ・バス上に有効なデータが存在することを示します。local_rdata_validのタイミングは、選択された再同期およびパイプライン・オプションに対応するように自動的に調整されます。
local_rdvalid_in_n	出力	リード・データ有効信号の早期バージョンで、それより 3 サイクル前に現れます。必ずネイティブ・インタフェースでのみ提供されます。

信号名	入力 / 出力	説明
local_ready	出力	local_ready 信号は、DDR または DDR2 SDRAM 高性能コントローラが要求信号を受け付ける準備ができていることを示します。local_readyが、リードまたはライト要求がアサートされるクロック・サイクルでアサートされる場合、その要求はすでに受け付けられています。local_ready 信号がデアサートされると、DDR または DDR2 SDRAM 高性能コントローラがこれ以上要求を受け付けることができないことを示します。
local_refresh_ack	出力	リフレッシュが発行されるたびに1クロック・サイクルの間アサートされるリフレッシュ要求確認です。User Controlled Refresh オプションが選択されていない場合でも、local_refresh_ack は、コントローラがリフレッシュ・コマンドを発行したことをローカル・インタフェースに示します。
local_wdata_req	出力	ライト・データ要求信号。次のクロック・エッジで有効なライト・データを提示する必要があることをローカル・インタフェースに示します。

表 4-8 に、DDR および DDR2 SDRAM インタフェース信号を示します。

信号名	入力 / 出力	説明
mem_dq[]	双方向	メモリ・データ・バス。このバスはローカル・リードおよびライト・データ・バスの幅の半分です。
mem_dqs[]	双方向	メモリ・データ・ストロブ信号。DDR または DDR2 SDRAM にデータを書き込み、アルテラ・デバイスへのリード・データをキャプチャします。
mem_clk (1)	双方向	メモリ・デバイスのクロック。
mem_clk_n (1)	双方向	メモリ・デバイスの反転クロック。
mem_a[]	出力	メモリ・アドレス・バス。
mem_ba[]	出力	メモリ・バンク・アドレス・バス。
mem_cas_n	出力	メモリ・カラム・アドレス・ストロブ信号。
mem_cke[]	出力	メモリ・クロック・イネーブル信号
mem_cs_n[]	出力	メモリ・チップ・セレクト信号。
mem_dm[]	出力	メモリ・データ・マスク信号。ライト時に個々のバイトをマスクします。
mem_odt[]	出力	メモリ On-Die Termination コントロール信号 (DDR2 SDRAM のみ)。
mem_ras_n	出力	メモリ・ロウ・アドレス・ストロブ信号。

表 4-8. DDR & DDR2 SDRAM インタフェース信号 (2 / 2)

信号名	入力 / 出力	説明
mem_we_n	出力	メモリ・ライト・イネーブル信号。

表 4-8 の注:

- (1) mem_clk 信号は、FPGA からの出力専用信号です。ただし、Quartus II ソフトウェアでは、これらは双方向 (INOUT) IO として定義し、ALTMEMPHY メガファンクションが使用する模擬パス構造をサポートする必要があります。

表 4-9 に、ECC コントローラの信号を示します。

表 4-9. ECC コントローラ信号

信号名	入力 / 出力	説明
ecc_addr[]	入力	ECC コントローラのアドレス。
ecc_be[]	入力	ECC コントローラのバイト・イネーブル。
ecc_interrupt	出力	ECC コントローラからの割り込み。
ecc_rdata[]	出力	ECC コントローラからのリターン・データ。
ecc_read_req	入力	ECC コントローラのリード要求。
ecc_wdata[]	入力	ECC コントローラのライト・データ。
ecc_write_req	入力	ECC コントローラのライト要求。

この付録では、ECC レジスタおよびレジスタ・ビットについて説明します。

表 A-1 に、ECC レジスタを示します。

表 A-1. ECC レジスタ (1 / 3)					
名称	アドレス	サイズ (ビット)	属性	デフォルト	説明
Control word specifications	00	32	R/W	0000000F	このレジスタはECC機能のすべてのコマンドを保持します。
Maximum single-bit error counter threshold	01	32	R/W	00000001	シングル・ビット・エラー・カウンタは (シングル・ビット・エラーが発生すると)、このレジスタで定義されるとおり最大スレッシュホールドになるまでインクリメントされます。このスレッシュホールドを越えると、ECC が割り込みを生成します。
Maximum double-bit error counter threshold	02	32	R/W	00000001	ダブル・ビット・エラー・カウンタは (ダブル・ビット・エラーが発生すると)、このレジスタで定義されるとおり最大スレッシュホールドになるまでインクリメントされます。このスレッシュホールドを越えると、ECC が割り込みを生成します。
Current single-bit error count	03	32	RO	00000000	シングル・ビット・エラー・カウンタは (シングル・ビット・エラーが発生すると)、最大スレッシュホールドになるまでインクリメントされます。このステータス・レジスタを読み出すと、カウンタの値を知ることができます。
Current double-bit error count	04	32	RO	00000000	ダブル・ビット・エラー・カウンタは (ダブル・ビット・エラーが発生すると)、最大スレッシュホールドになるまでインクリメントされます。このステータス・レジスタを読み出すと、カウンタの値を知ることができます。

表 A-1. ECC レジスタ (2 / 3)					
名称	アドレス	サイズ (ビット)	属性	デフォルト	説明
Last or first single-bit error error address	05	32	RO	00000000	このステータス・レジスタは、最後のシングル・ビット・エラーのエラー・アドレスを格納します。これはコントロール・ワード・クリアを使用してクリアできます。コントロール・ワードのビット 10 が High に設定されている場合、最初に発生したアドレスが格納されます。
Last or first double-bit error error address	06	32	RO	00000000	このステータス・レジスタは、最後のダブル・ビット・エラーのエラー・アドレスを格納します。これはコントロール・ワード・クリアを使用してクリアできます。コントロール・ワードのビット 10 が High に設定されている場合、最初に発生したアドレスが格納されます。
Last single-bit error error data	07	32	RO	00000000	このステータス・レジスタは、最後のシングル・ビット・エラーのエラー・データ・ワードを格納します。データ・ワードは 64 の N 乗倍なので、データ・ワードは深度 $2N$ 、32 ビット幅の FIFO バッファに、最下位 32 ビット・サブ・ワードを先頭にして格納されます。これは、コントロール・ワード・クリアを使用して個別にクリアできます。
Last single-bit error syndrome	08	32	RO	00000000	このステータス・レジスタは、最後のシングル・ビット・エラー・シンδροームを格納します。これにより、64 ビット・データ・ワードのエラー・ビットの位置が指定されます。データ・ワードは 64 の N 乗数なので、シンδροームは深度 N 、8 ビット幅の FIFO バッファに格納されます。ここで、各シンδροームはデータ・ワードの各 64 ビット部分のエラーを表します。このレジスタは、データ・ワードのどの部分が最後のシングル・ビット・エラー・データ・レジスタに表示されるかに応じて、正しいシンδροームでアップデートされます。これはコントロール・ワード・クリアを使用して個別にクリアできます。

表 A-1. ECC レジスタ (3 / 3)

名称	アドレス	サイズ (ビット)	属性	デフォルト	説明
Last double-bit error error data	09	32	RO	00000000	このステータス・レジスタは、最後のダブル・ビット・エラーのエラー・データ・ワードを格納します。データ・ワードは 64 の N 乗なので、データ・ワードは深度 $2N$ 、32 ビット幅の FIFO バッファに、最下位 32 ビット・サブ・ワードを先頭にして格納されます。これはコントロール・ワード・クリアを使用して個別にクリアできます。
Interrupt status register	0A	5	RO	00000000	このステータス・レジスタは、割り込みステータスを 4 つのフィールドに格納します (表 A-3 参照)。これらのステータス・ビットは、対応する位置に 1 を書き込むことによってクリアできます。
Interrupt mask register	0B	5	WO	00000001	このレジスタは、割り込みマスクを 4 つのフィールドに格納します (表 A-4 参照)。
Single-bit error location status register	0C	32	R/W	00000000	このステータス・レジスタは、データ・ワードの各 64 ビット部分で発生したシングル・ビット・エラーを各ビットに格納します (表 A-5 を参照)。これらのステータス・ビットは、対応する位置に 1 を書き込むことによってクリアできます。
Double-bit error location status register	0D	32	R/W	00000000	このステータス・レジスタは、データ・ワードの各 64 ビット部分で発生したダブル・ビット・エラーを各ビットに格納します (表 A-6 を参照)。これらのステータス・ビットは、対応する位置に 1 を書き込むことによってクリアできます。

表 A-2 にコントロール・ワード仕様レジスタを示します。

表 A-2. コントロール・ワード仕様レジスタ			
ビット	名称	入力 / 出力	説明
0	Count single-bit error	デコーダ - コレクタ	1 の場合、シングル・ビット・エラー数をカウントします。
1	Correct single-bit error	デコーダ - コレクタ	1 の場合、シングル・ビット・エラーを訂正します。
2	Double-bit error enable	デコーダ - コレクタ	1 の場合、すべてのダブル・ビット・エラーを検出し、ダブル・ビット・エラー・カウンタをインクリメントします。
3	Hamming code enable	エンコーダ	1 の場合、着信データをエンコードします。
4	Clear all status registers	コントローラ	1 の場合、カウンタ・シングル・ビット・エラーおよびダブル・ビット・エラー・ステータス・レジスタの最初と最後のエラー・アドレスをクリアします。
5	Single-bit error interrupt mask	コントローラ	1 の場合、シングル・ビット・エラー割り込みをマスクします。
6	Double-bit error interrupt mask	コントローラ	1 の場合、ダブル・ビット・エラー割り込みをマスクします。シングル・ビット・エラーおよびダブル・ビット・エラー割り込みは、OR がとられ、出力として Avalon-MM インタフェースに送信されます。
7	Counter clear on read	コントローラ	1 の場合、カウンタのリード時クリア機能をイネーブルします。
8	Corrupt ECC enable	コントローラ	1 の場合、エンコーディング時の意図的 ECC 破壊をイネーブルにし、ECC をテストします。
9	ECC corruption type	コントローラ	0 の場合はすべての ECC コードワードでシングル・ビット・エラーを作成し、1 の場合はすべての ECC コードワードでダブル・ビット・エラーを作成します。
10	First or last error	コントローラ	1 の場合、シングル・ビット・エラーまたはダブル・ビット・エラーの最後のエラー・アドレスではなく最初のエラー・アドレスを格納します。
11	Clear interrupt	コントローラ	1 の場合、割り込みをクリアします。

表 A-3 に、割り込みステータス・レジスタを示します。

ビット	名称	説明
0	Single-bit error	1 の場合、シングル・ビット・エラーが発生しました。
1	Double-bit error	1 の場合、ダブル・ビット・エラーが発生しました。
2	Maximum single-bit error	1 の場合、シングル・ビット・エラーの最大スレッショルドを超えました。
3	Maximum double-bit error	1 の場合、ダブル・ビット・エラーの最大スレッショルドを超えました。
4	Double-bit error during read-modify-write	1 の場合、リード・モディファイ・ライト状態時にダブル・ビット・エラーが発生しました。(部分書き込み)。
その他	Reserved	Reserved

表 A-4 に、割り込みマスク・レジスタを示します。

ビット	名称	説明
0	Single-bit error	1 の場合、シングル・ビット・エラーをマスクします。
1	Double-bit error	1 の場合、ダブル・ビット・エラーをマスクします。
2	Maximum single-bit error	1 の場合、シングル・ビット・エラーの最大スレッショルドを超える状態をマスクします。
3	Maximum double-bit error	1 の場合、ダブル・ビット・エラーの最大スレッショルドを超える状態をマスクします。
4	Double-bit error during read-modify-write	1 の場合、リード・モディファイ・ライト状態時にダブル・ビット・エラーが発生すると割り込みをマスクします。(部分書き込み)。
その他	Reserved	Reserved

表 A-5 に、シングル・ビット・エラー位置スタータス・レジスタを示します。

表 A-5. シングル・ビット・エラー位置スタータス・レジスタ		
ビット	名称	説明
ビット $N-1 \sim 0$	Interrupt	0 の場合、シングル・ビット・エラーは発生していません。1 の場合、この 64 ビット部分でシングル・ビット・エラーが発生しました。
その他	Reserved	Reserved

表 A-6 に、ダブル・ビット・エラー位置スタータス・レジスタを示します。

表 A-6. ダブル・ビット・エラー位置スタータス・レジスタ		
ビット	名称	説明
ビット $N-1 \sim 0$	Cause of Interrupt	0 の場合、ダブル・ビット・エラーは発生していません。1 の場合、この 64 ビット部分でダブル・ビット・エラーが発生しました。
その他	Reserved	Reserved

改訂履歴

以下の表に、このユーザガイドの改訂履歴を示します。

日付	バージョン	変更内容
2007年10月	7.2	<ul style="list-style-type: none"> ● ハーフ・レートおよびフル・レート情報を修正 ● local_wdata_req、mem_clk、および mem_clk_n の説明を修正 ● 標準レイテンシ値を更新
2007年5月	7.1	<ul style="list-style-type: none"> ● デバイスのサポートを更新 ● ECC の説明を追加 ● 付録 A を追加
2006年12月	7.0	Cyclone® III サポートを追加。
2006年12月	6.1	初版








アルテラへのお問い合わせ

アルテラ製品に関する最新情報は、アルテラのウェブサイト、www.altera.co.jp をご覧ください。テクニカル・サポートについては、www.altera.co.jp/mysupport にアクセスしてください。また、アルテラの販売代理店にもお問い合わせいただけます。

表記規則

本書では、以下の表記規則を使用しています。

書体	意味
太字かつ文頭が大文字	コマンド名、ダイアログ・ボックス・タイトル、チェックボックス・オプション、およびダイアログ・ボックス・オプションは、太字かつ文頭が大文字で表記されています。例: Save As ダイアログ・ボックス
太字	外部タイミング・パラメータ、ディレクトリ名、プロジェクト名、ディスク・ドライブ名、ファイル名、ファイルの拡張子、およびソフトウェア・ユーティリティ名は、太字で表記されています。例: f_{MAX} , lqdesigns ディレクトリ、 d: ドライブ、 chiptrip.gdf ファイル
斜体かつ文頭が大文字	資料のタイトルは、斜体かつ文頭が大文字で表記されています。例: <i>AN 75: High-Speed Board Design</i>
斜体	内部タイミング・パラメータおよび変数は、斜体で表記されています。例: <i>t_{PIA}</i> , <i>n + 1</i> 変数は、山括弧 (<>) で囲み、斜体で表記されています。例: <ファイル名>、<プロジェクト名>.pof ファイル

書体	意味
文頭が大文字	キーボード・キーおよびメニュー名は、文頭が大文字で表記されています。 例: Delete キー、Options メニュー
「小見出しタイトル」	資料内の小見出しおよびオンライン・ヘルプ・トピックのタイトルは、鉤括弧で囲んでいます。例: 「表記規則」
Courier フォント	信号およびポート名は、Courier フォントで表記されています。 例: data1、tdi、input。アクティブ Low 信号は、サフィックス n で表示されています (例: resetn)。 表示されているとおりに入力する必要があるものは、Courier フォントで表記されています (例: c:\qdesigns\tutorial\chiptrip.gdf)。また、Report ファイルのような実際のファイル、ファイルの構成要素 (例: AHDL キーワードの SUBDESIGN)、ロジック・ファンクション名 (例: TRI) も Courier フォントで表記されています。
1.、2.、3. および a.、b.、c. など	手順など項目の順序が重要なものは、番号が付けられリスト形式で表記されています。
	箇条書きの黒点などは、項目の順序が重要ではないものに付いています。
	チェックマークは、1 ステップしかない手順を表します。
	指差しマークは、要注意箇所を表しています。
	CAUTION マークは、特別な配慮および理解が必要であり、手順またはプロセスを始める前、または続ける際に確認すべき情報を示しています。
	注意マークは、手順またはプロセスを始める前、または続ける際に確認すべき情報を示しています。
	矢印は、Enter キーを押すことを示しています。
	足跡マークは、詳細情報の参照先を示しています。