

DDR3 SDRAM 高性能コントローラ ユーザガイド



101 Innovation Drive
San Jose, CA 95134
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・デート: 2007 年 12 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-01021-1.0

第1章 この 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	1-4
OpenCore Plus 評価機能	1-5
OpenCore Plus タイム・アウト動作	1-6

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

Memory Settings	3-1
PHY Settings	3-1
Controller Settings	3-1

第4章 機能の説明

ブロック説明	4-1
コントロール・ロジック	4-1
レイテンシ	4-3
デザイン例	4-4
インタフェースおよび信号	4-6
インタフェースの説明	4-6
信号	4-12

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、DDR3 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。

表 1-1. DDR3 SDRAM 高性能コントローラのリリース情報	
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/DDR3
プロダクト ID	00C2 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、DDR3 SDRAM 高性能コントローラによる各アルテラ・デバイス・ファミリへのサポートのレベルを示します。

表 1-2. サポートされるデバイス・ファミリ	
デバイス・ファミリ	サポートの種類
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

- パワー・アップ・キャリブレーション付き On-Chip Termination (OCT: チップ内終端) のサポート
- Stratix III デバイスのハーフ・レート・サポート
- SOPC Builder への対応
- 自動的に生成されるメモリ・シミュレーション・モデルによるシミュレーション・フローの簡素化
- altmemphy メガファンクションのサポート
- 業界標準の DDR3 SDRAM デバイスおよびモジュールのサポート
- 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® Memory-Mapped (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

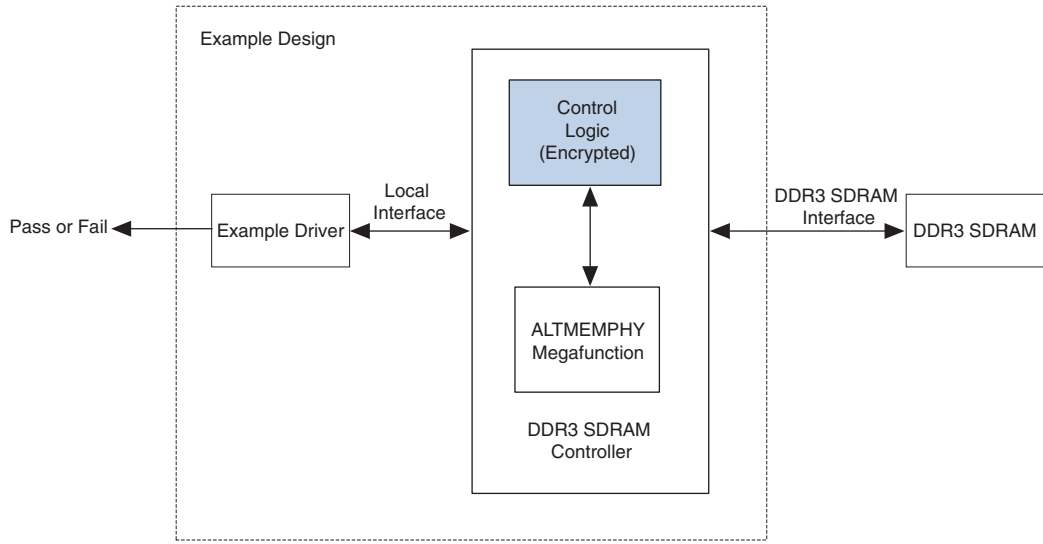
アルテラの DDR3 SDRAM 高性能コントローラ MegaCore ファンクションは、業界標準の DDR3 SDRAM への簡素化されたインタフェースを提供します。この MegaCore ファンクションは、アルテラの altmemphy メガファンクションと連携して動作します。



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、DDR3 SDRAM 高性能コントローラの MegaCore ファンクションを作成するデザイン例を含むシステム・レベル図を示します。

図 1-1. システム・レベル図



MegaWizard Plug-In Manager は、ドライバ例および DDR3 SDRAM 高性能コントローラのカスタム・バリエーションをインスタンス化するデザイン例を生成します。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、DDR3 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 DDR3 SDRAM 高性能コントローラのさまざまなゲート・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・バージョンを使用した、DDR3 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。

デバイス	System f_{MAX} (MHz)
Stratix III	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、Stratix III デバイスにおける DDR3 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	1,292	1,389	1	62
64	16	2,156	2,199	3	108
256	64	3,901	3,880	17	128
288	72	4,336	4,200	19	142

インストール および ライセンス

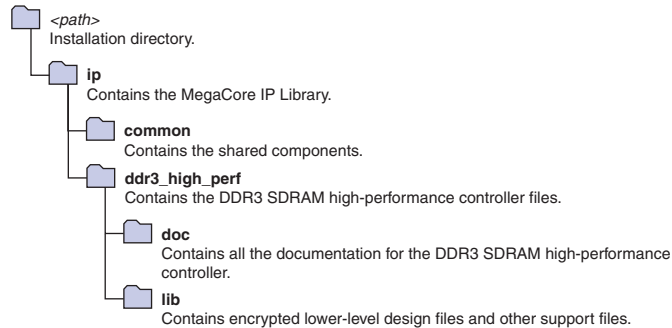
DDR3 SDRAM 高性能コントローラユーザガイド MegaCore ファンクションは、MegaCore IP ライブラリの一部であり、Quartus® II ソフトウェアとともに配布されます。また、アルテラのウェブサイト (www.altera.co.jp) からダウンロードすることもできます。



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、DDR3 SDRAM 高性能コントローラユーザガイド 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。

DDR3 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。



DDR3 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。



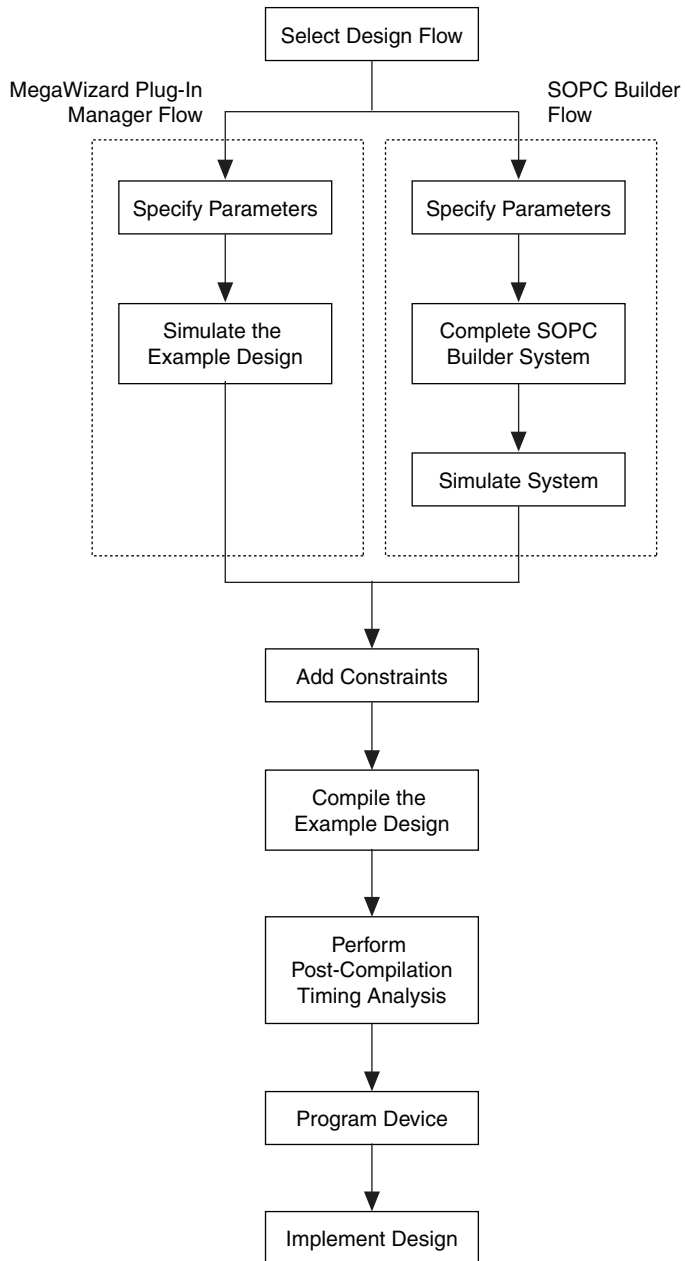
MegaCore ファンクションの場合、アンテザード・タイムアウトは 1 時間、テザード・タイムアウト値は無制限です。

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、DDR3 SDRAM 高性能コントローラユーザガイド 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、DDR3 SDRAM 高性能コントローラユーザガイド MegaCore function 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。

- DDR3 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、DDR3 SDRAM 高性能コントローラユーザガイド MegaCore function 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 DDR3 SDRAM 高性能コントローラユーザガイドを搭載する 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 DDR3 SDRAM 高性能コントローラユーザガイド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。



Do you want to assign a specific device? と表示されたら、**Yes** を選択して特定のデバイスを選びます。

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 **DDR3 SDRAM 高性能コントローラユーザガイド** を追加します。



DDR3 SDRAM 高性能コントローラユーザガイド 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、DDR3 SDRAM 高性能コントローラユーザガイド MegaCore function 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. **Finish** をクリックします。
10. SOPC Builder で、**Interface Protocols** および **Serial** を拡張します。
11. **JTAG UART** を選択して、**Add** をクリックします。

12. **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。

13. 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。

14. **Generate** をクリックします。

15. 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。サンプル・ドライバおよび DDR3 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、独自のトップ・レベル・デザイン・ファイルを作成するか、または `altmemddr_example_top.v` ファイルを編集することができます。

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

MegaWizard Plug-In Manager フロー

MegaWizard Plug-In Manager フローでは、DDR3 SDRAM 高性能コントローラユーザガイド MegaCore function 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、DDR3 SDRAM 高性能コントローラユーザガイド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。



Do you want to assign a specific device? と表示されたら、**Yes** を選択して特定のデバイスを選びます。

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。



DDR3 SDRAM 高性能コントローラユーザガイド MegaCore function 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 のいずれか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 (1 / 2) 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 Quartus II 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。

表 2-1. 生成されるファイル (2 / 2) 注 (1)	
ファイル名	説明
<variation name>.ppf	この XML ファイルは、Quartus II Pin Planner に対する MegaCore ピン属性を記述しています。MegaCore ピン属性には、ピンの方向、位置、I/O 規格のアサインメント、およびドライブ強度などがあります。IP Toolbench を Pin Planner アプリケーションの外側で起動する場合、Pin Planner を使用するにはこのファイルを明示的にロードしなければなりません。
<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に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
<variation name>_example_top_tb.vhd または .v	テストベンチ例
<variation name>_mem_model.v	メモリ・モデル
<variation name>_pin_assignments.tcl	ピン・アサインメント制約スクリプト

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、ターゲットとするメモリのデザイン例およびシミュレーション・モデルの VHDL または Verilog HDL テストベンチを生成します。これらはプロジェクト・ディレクトリの **testbench** ディレクトリにあります。



テストベンチについて詳しくは、「[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。

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。
3. 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。
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. テストベンチのトップ・レベル・モジュールに、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. テストベンチのトップ・レベル・インスタンスの名前、`dut`を**Design instance name in test bench**に入力します。
10. **Run for**を**500 μs**に変更します。
11. テストベンチ・ファイルの追加**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。
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 `<variation name>_pin_assignments.tcl` 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。



さらに、複数の IP コアのトライアル・ピン配置を実行するために、Assignments メニューの **Pins** をクリックします。**Groups** または **All Pins** 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。`<variation name>_pin_assignments.tcl` および `<variation name>_dq_groups.tcl` スクリプトが自動的に実行されます。

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 and Synthesis** をクリックします。
5. デザインのピンにピン位置を割り当てます。
 - a. 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。

または

- b. すべての DQ および DQS ピンを手動で指定して、プロジェクトと PCB 要件を整合させます。

または

- c. プロジェクトのすべてのピン位置を手動で指定して、プロジェクトと PCB 要件を整合させます。



ピンを割り当てるときは、例えばクロック・ソース、リセット、アドレスおよびコマンド信号で、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. すべてのメモリ・インタフェース・ピンに対する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 DDR3 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>_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、<variation name>_phy_report_timing.tcl を選択します。



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、およびストロープ信号が含まれます。



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。

図 3-1. Controller Settings

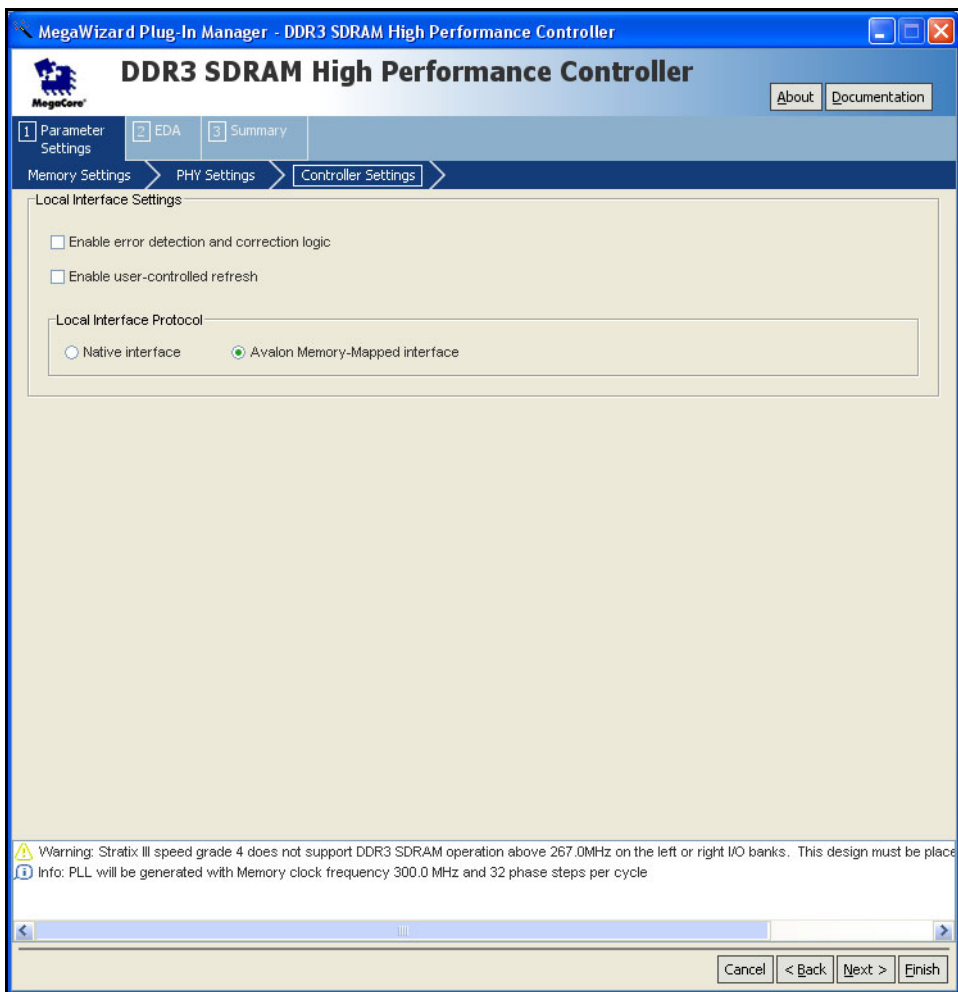


表 3-1 に、コントローラ設定を示します。

表 3-1. Controller Settings		
パラメータ	範囲	説明
Enable 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。4-10 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 Memory-Mapped	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® Memory-Mapped (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。

DDR3 SDRAM 高性能コントローラ MegaCore ファンクションは、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。

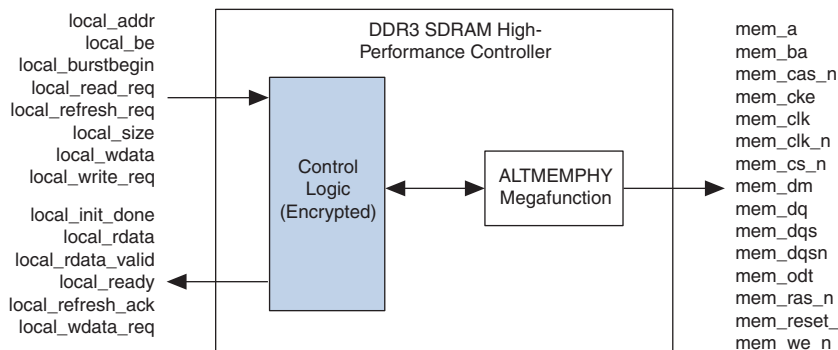


altmemphy メガファンクションについて詳しくは、「[altmemphy メガファンクション・ユーザガイド](#)」を参照してください。

ブロック説明

図 4-1 に、DDR3 SDRAM 高性能コントローラのブロック図を示します。

図 4-1. DDR3 SDRAM 高性能コントローラのブロック図



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 信号を組み合わせて使用して、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
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

DDR3 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。DDR3 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、5～11 クロック・サイクル後にデータ・バスに現れます。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 (6) 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。最初の RD または WR コマンドの後、バースト長に達するまで、順次リードおよびライトが続きます。DDR3 SDRAM デバイスは、4 または 8 データ・サイクルの固定バースト長または実行時モードをサポートします。実行時モードの場合、コントローラはリードまたはライト・コマンドごとに 4 または 8 のバーストを要求できます。この実行時モードは唯一サポートされているモードです。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定期的に発行されます。このファンクションは、DDR3 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、使用する 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関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・サイドの周波数はメモリ・インタフェース周波数の半分です。

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 4-3 に、ハーフ・レート・コントローラおよび 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。

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)
ハーフ	400	読み出し	$6 + 4 + 1.5 + 8.5 = 20$	50
		書き込み	$8 + 0 + 2 = 10$	25

 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。

デザイン例

MegaWizard® Plug-In Manager は、DDR3 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、DDR3 SDRAM 高性能コントローラと、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

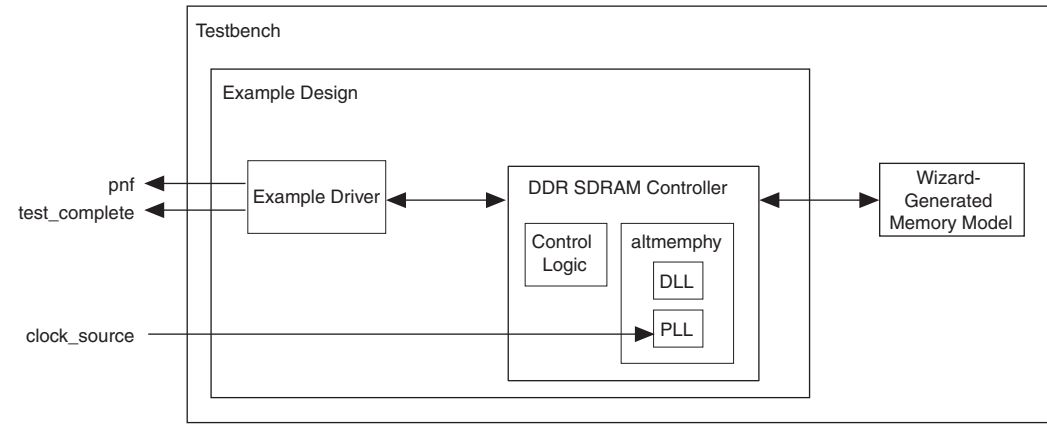


表 4-4 に、このデザイン例とテストベンチに関連するファイルを示します。

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>_example_driver.v または .vhd	サンプル・ドライバ。
<variation name>_mem_model.v または .vhd	ウィザードが生成したメモリ・モデル。
<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。

このサンプル・ドライバは、DDR3 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。

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。



シミュレーション・スクリプトの実行方法について詳しくは、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。

- 4-6 ページの「[インタフェースの説明](#)」
- 4-12 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。

- 4-6 ページの「[ライト](#)」
- 4-9 ページの「[リード](#)」
- 4-10 ページの「[ユーザ・リフレッシュ・コントロール](#)」
- 4-11 ページの「[初期化タイミング](#)」

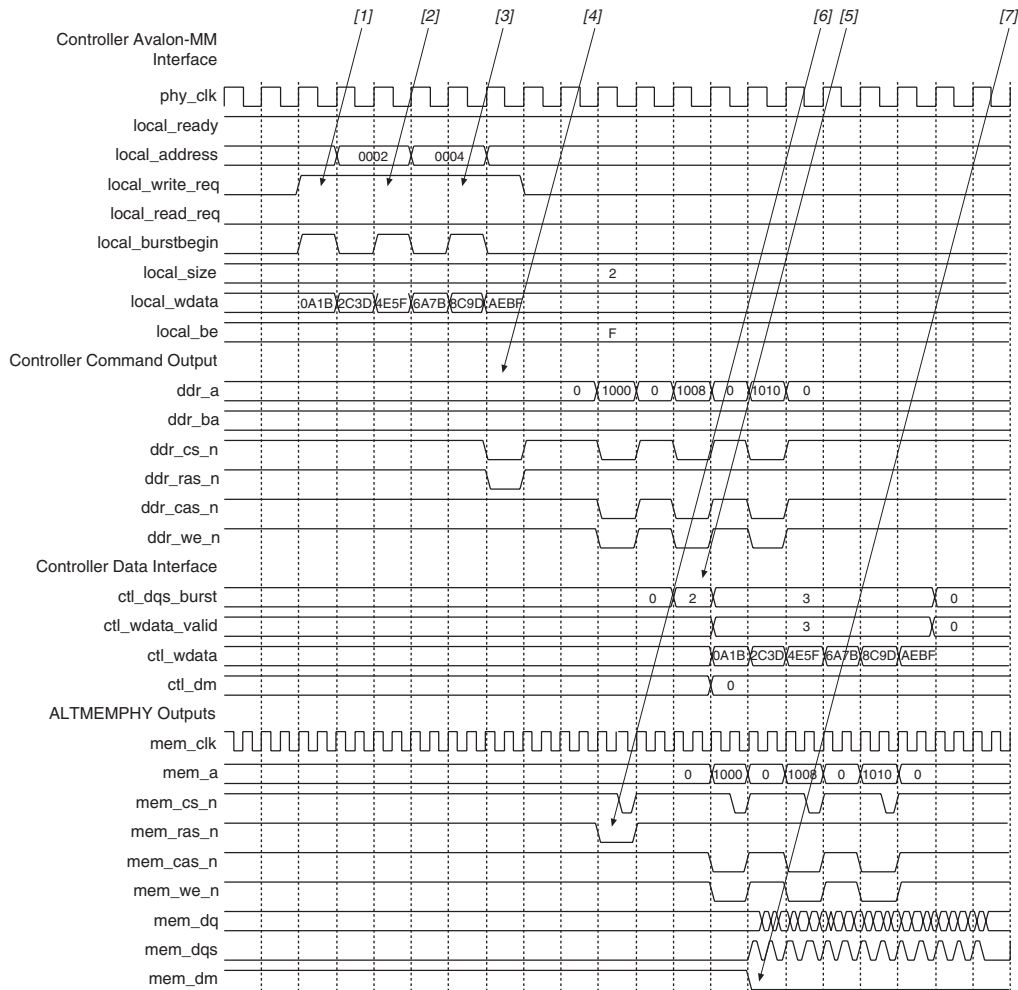


これらのインタフェース要求は、ネイティブ・インタフェース用です。Avalon™ Memory-Mapped (Avalon-MM) インタフェースについては、「[Avalon Memory-Mapped Interface Specification](#)」を参照してください。

ライト

4-7 ページの図 4-3 に、シーケンシャル・アドレスに対する 3 つのバック・ツー・バック・ライト要求（サイズはすべて 2）を示します。DDR3 SDRAM コントローラは、実行時バースト・モードをサポートしています。このモードでは、ローカル・サイドのインタフェースで 1 または 2 のバースト長（DDR3 SDRAM サイドのインタフェースでの 4 または 8 と同じ）を要求できます。

図 4-3. ライト



1. ユーザ・ロジックは、このライトの `local_write_req`、`local_burstbegin`、サイズ、およびアドレス信号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、要求はアドレス 0 に対する長さ 2 (DDR3 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い場合、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。この長さ 2 のバーストの場合は、次のクロック・サイクルでライト・データの 2 回目のビートを提示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>
```

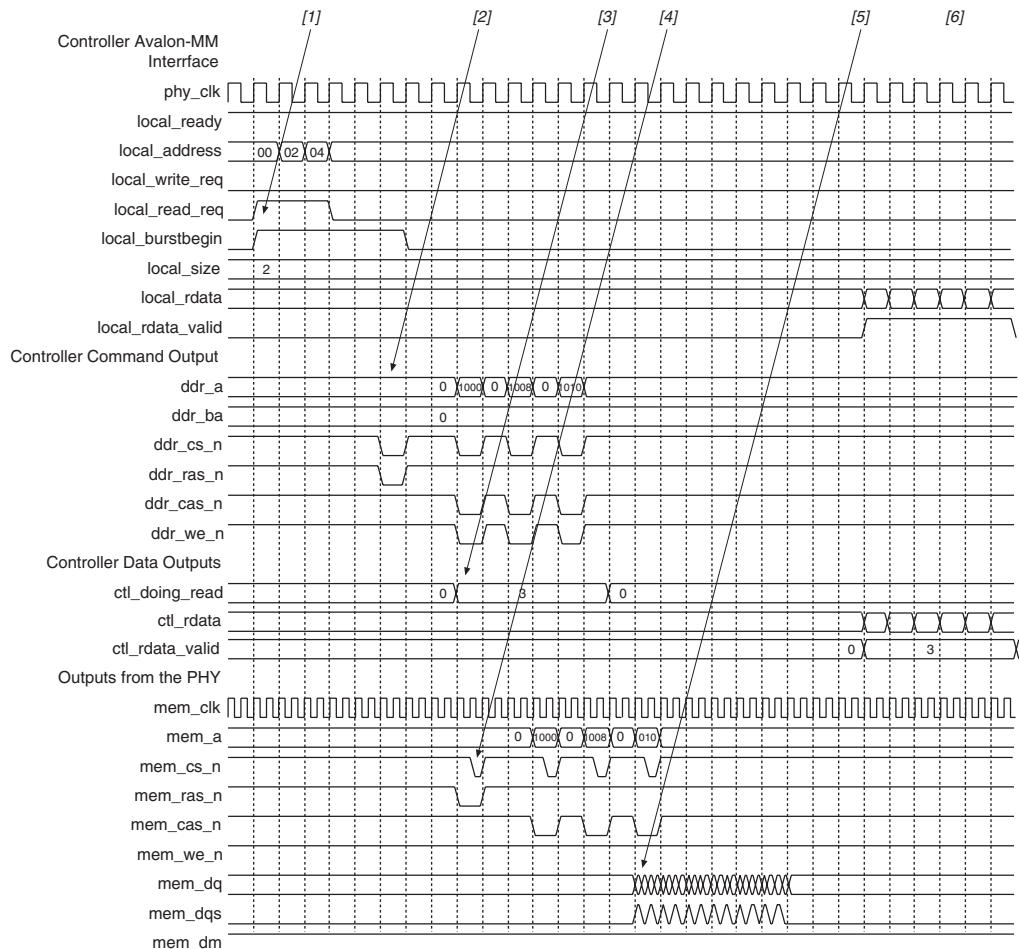
2. ユーザ・ロジックは、シーケンシャル・アドレスへの 2 回目のライトを要求します。今回のサイズは 2 (DDR3 SDRAM サイドでは 8) です。local_ready 信号は、アサートされたままで、コントローラが要求を受け付けたことを示します。
3. ユーザ・ロジックは、3 回目のライトを要求します。コントローラは最大 4 つの要求をバッファできるため、local_ready 信号は High のまま維持され、この要求は受け付けられます。
4. コントローラは、必要なバンク・アクティベーション・コマンドと 3 つのライト・コマンドを ALTMEMPHY に順次発行します。これらのコマンドは ALTMEMPHY でハーフ・レートからフル・レートに変換され、メモリ・デバイスに発行されます。
5. コントローラは、DQS (ctl_dqs_burst) および DQ (ctl_wdata_valid) 出力をイネーブルする期間を制御する信号をアサートします。ctl_dqs_burst および ctl_wdata_valid 信号は 2 ビット幅のため、コントローラがハーフ・レート・クロックで動作している場合でも、DQS および DQ 信号がイネーブルされるフル・レートの mem_clk サイクル数を制御できます。この例では、DQS 出力は (DQS プリアンブルを考慮して) 13 フル・レート・クロック・サイクルの間イネーブルされ、DQ は 12 フル・レート・クロック・サイクルの間イネーブルされます。ライト・データ (ctl_wdata) とマスク (ctl_dm) は、ctl_wdata_valid と同時に発行されます。
6. ALTMEMPHY は、メモリ・デバイスにバンク・アクティベーションとライト・コマンドを発行します。

7. ALTMEMPHY は、DQS、DQ、および DM 信号を発行して、メモリ・デバイスにデータを書き込みます。

リード

4-9 ページの図 4-4 に、サイズ 2 の 3 つのリード要求を示します。DDR3 SDRAM コントローラは、実行時バースト・モードをサポートしています。このモードでは、ローカル・サイドのインタフェースで 1 または 2 のバースト長 (DDR3 SDRAM サイドのインタフェースにおける 4 または 8 と同じ) を要求できます。

図 4-4. リード



1. ユーザ・ロジックは、各リードの `local_read_req`、`local_burstbegin`、`local_size`、および `local_address` 信号をアサートすることによって、サイズ 2 (DDR3 SDRAM サイドでは 8) の 3 つのバック・ツー・バック・リードを要求します。`local_ready` 信号がアサートされており、これはコントローラが各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. コントローラは、必要なバンク・アクティベーション・コマンドと 3 つのリード・コマンドを `ALTMEMPHY` に順次発行します。これらのコマンドは `ALTMEMPHY` でハーフ・レートからフル・レートに変換され、メモリ・デバイスに発行されます。
3. コントローラは、`ctl_doing_read` 信号をアサートし、`ALTMEMPHY` にキャプチャ・レジスタに対するイネーブルのタイミングとその期間を示します。
4. `ALTMEMPHY` は、メモリ・デバイスにバンク・アクティベーションとリード・コマンドを発行します。
5. メモリ・デバイスは、CAS レイテンシ後に要求されたアドレスに対するリード・データを、`ALTMEMPHY` がリード・データをキャプチャするのに使用する `DQS` ストロープ信号と共に返します。
6. コントローラは、ユーザ・ロジックにリード・データを発行し、`local_rdata_valid` 信号で有効とマークします。後続のリード要求の間、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。

ユーザ・リフレッシュ・コントロール

図 4-5 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 4-5. ユーザ・リフレッシュ・コントロール

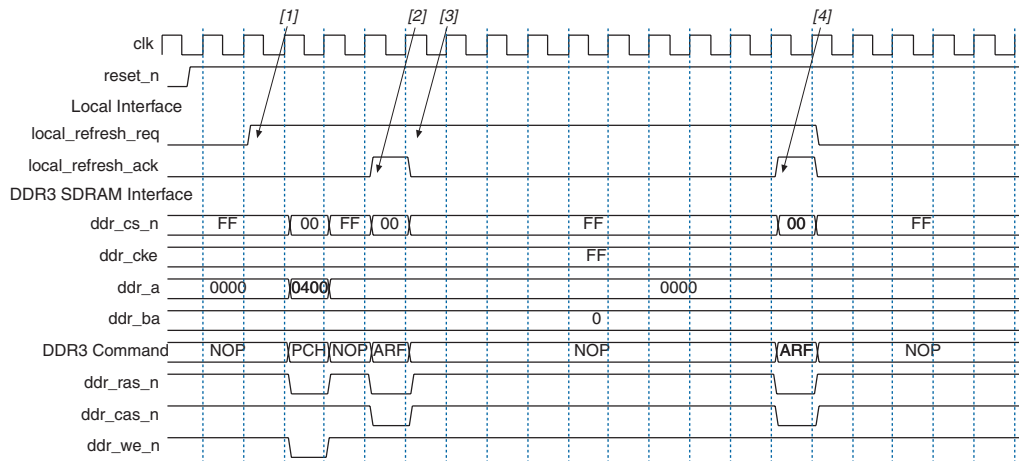


図 4-5 の注:

(1) DDR3 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。

初期化タイミング

DDR3 SDRAM 高性能コントローラは、altmemory メガファンクションを利用して初期化を行います。



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。

altmemory 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 4-5 に、DDR3 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。

信号名	入力 / 出力	説明
local_addr[]	入力	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。 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 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 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 つの最下位ビット (LSB) は無視されます。
local_be[]	入力	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_burstbegin	入力	Avalon バースト開始ストロープで、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 インタフェースの場合のみです。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。
local_read_req	入力	リード要求信号。
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。

表 4-5. ローカル・インタフェース信号 (2 / 2)

信号名	入力 / 出力	説明
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。DDR3 SDRAM 高性能コントローラは、ローカル・サイドのインタフェースで 1 および 2 のバースト長をサポートします。
local_wdata[]	入力	ライト・データ・バス。local_wdata の幅は、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 4 倍です。
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せん。
local_ready	出力	local_ready 信号は、DDR3 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ないことを示します。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、DDR3 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 4-6 に、DDR3 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・ストロープ信号。DDR3 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_dqs_n[]	双方向	メモリ・データ・ストロープ信号。DDR3 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・ストロープ信号。
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。
mem_ras_n	出力	メモリ・ロウ・アドレス・ストロープ信号。
mem_reset_n	出力	メモリ・リセット信号。
mem_we_n	出力	メモリ・ライト・イネーブル信号。

表 4-6 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バス構造をサポートする必要があります。

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	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
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」

書体	意味
Courier フォント	<p>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）。</p> <p>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。</p>
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。