

はじめに

多くのシステムやアプリケーションでは、データ・ストレージまたはバッファとして外部メモリ・インタフェースを使用します。システム・アプリケーションが広い帯域幅を要求し、さらに複雑になり、デバイスが高価になると、1つのデバイス内に複数のメモリ・インタフェースを内蔵して、コストとボード面積を節約し、分割の複雑さを回避することが好まれるようになります。FPGAアーキテクチャ向けに効率的で最適化された複数のメモリ・インタフェースを実装するには、デバイスとフィジカル・インタフェース (PHY) の両方の機能に注意する必要があります。

このアプリケーション・ノートでは、メモリ・トランザクションは独立に発生しますが、同じ周波数で動作する複数のメモリ・インタフェースを実装するための手順について説明します。このような実装では PHY の複数インスタンス化が必要になるため、DLL、PLL、クロック・ネットワークなどのFPGAリソースの共用が必要になることがあります。メモリ・インタフェースの幅と深さの拡張は、PHYによりネイティブにサポートされています。このドキュメントで説明するメモリ・インタフェースでは、Stratix® III、Stratix II、Arria™ GX、HardCopy® II、Cyclone® III の各デバイスで使用可能な ALTMEMPHY メガファンクションを使用します。



ALTMEMPHY メガファンクションは、アルテラの DDR および DDR2 SDRAM High Performance Controller™ Megacore ファンクションのデータ・パスを実装します。ALTMEMPHY メガファンクションは、初期化時に製造プロセス (P) 変動を除去するために再同期化クロックをダイナミックにキャリブレートするため、また、システム動作時に電圧と温度の (VT) 変動をトラッキングするために必要なすべてのロジックを生成します。この手法により、システムのすべての PVT 条件で、再同期化データがセットアップ時間とホールド時間の最適マージンを持つことが保証されます。また、ALTMEMPHY メガファンクションをユーザ固有のメモリ・コントローラと組み合わせて使用することもできます。Stratix III、Arria GX、Cyclone III の各メモリ・インタフェースでは、ALTMEMPHY メガファンクションの使用が必要です。



スタティック再同期化クロックを使用する複数メモリ・インタフェースについては、[「AN 392: Multiple DDR and DDR2 SDRAM Controllers on One Device」](#)を参照してください。



Stratix II デバイス、Stratix II GX デバイス、または HardCopy II デバイスをターゲットとして、DDR および DDR2 SDRAM 高性能コントローラのように ALTMEMPHY メガファンクションをデータ・パスとして使用できるか否かについて不明な場合には、「[テクニカル・ブリーフ 091: External Memory Interface Options for Stratix II Devices](#)」を参照してください。



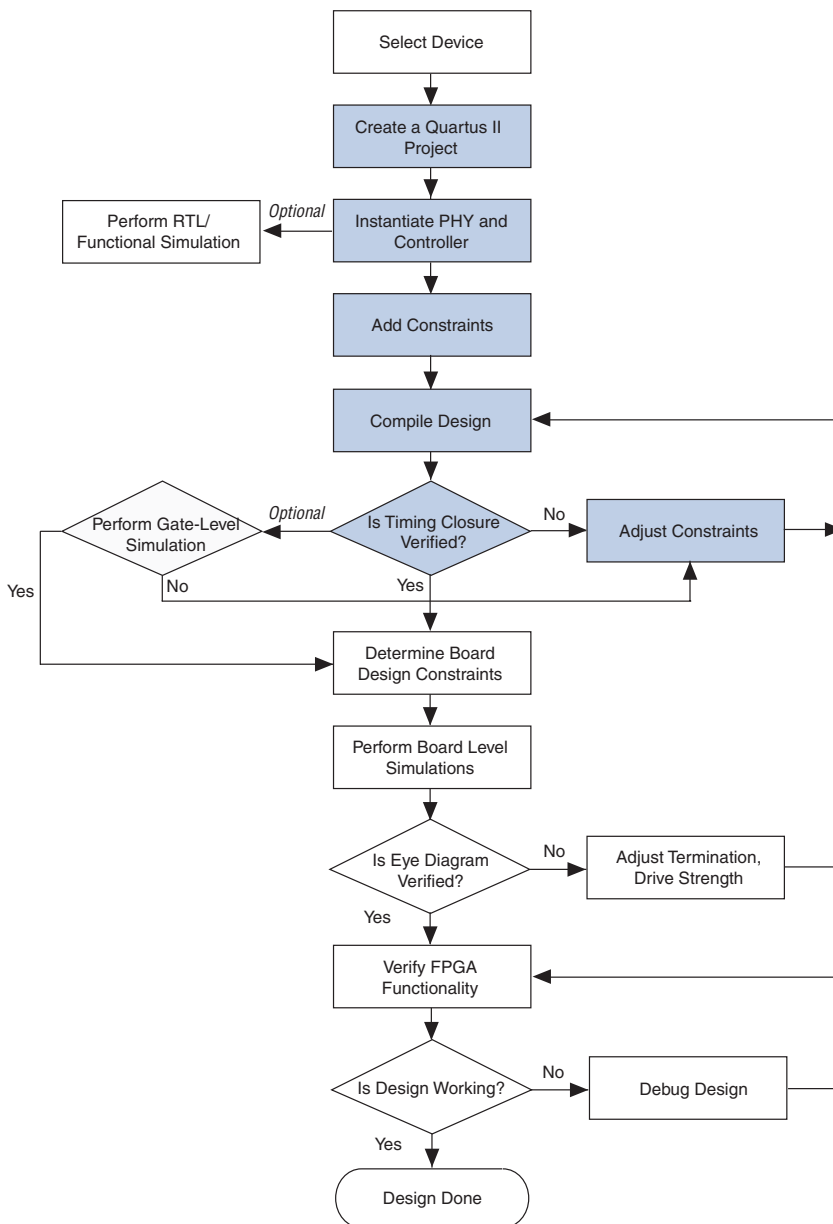
Stratix II デバイスをターゲットとしたデザインは、HardCopy II デバイスへ移行することができます。このアプリケーション・ノートにおいて、Stratix II デバイスに関するアルテラ FPGA の説明は、Stratix II GX デバイスまたは HardCopy II デバイスにも適用できます。HardCopy II での ALTMEMPHY メガファンクションの使用については、「[AN 463: Using the ALTMEMPHY Megafunction with HardCopy II Structured ASICs](#)」を参照してください。



デザイン内の複数のメモリ・インタフェースがリソースを共用しない場合は、これらをデザイン内で独立したモジュールとして扱うことができます。ユーザ固有のメモリ・コントローラを生成するときには「[ALTMEMPHY メガファンクション・ユーザガイド](#)」、メモリ・コントローラとしてアルテラ MegaCore ファンクションを使うときには「[DDR および DDR2 SDRAM 高性能コントローラ・ユーザガイド](#)」を参照してください。

図 1 に、FPGA デバイス内に外部メモリ・インタフェースを実装するデザイン・フローを示します。このアプリケーション・ノートでは、図 1 に青で示す各ステップ (Quartus® II プロジェクトの生成から複数メモリ・インタフェース・デザインのタイミング検証まで) について説明します。ただし、メモリ・デバイスと FPGA は選択済みとします。このドキュメントでは、例に示すように、まずデバイス・リソースをレビューし、デザイン回路を生成し、次に DDR および DDR2 高性能コントローラ MegaCore ファンクションを使用して Quartus II ソフトウェア内で複数メモリ・インタフェース・デザインを生成します。

図 1. アルテラ FPGA デバイスに外部メモリ・インタフェースを実装するデザイン・フロー



このアプリケーション・ノートは、ALTMEMPHY メガファンクションとアルテラの FPGA リソースを理解している方を対象にしています。参考のため、このアプリケーション・ノートは以下の資料と併用してご利用ください。

- Stratix III、Stratix II、Stratix II GX、Arria GX、および Cyclone III の各デバイス・ハンドブックの「クロック・ネットワークおよび PLL」の章
- Stratix III、Stratix II、Stratix II GX、Arria GX、および Cyclone III の各デバイス・ハンドブックの「外部メモリ・インタフェース」の章
- 「ALTMEMPHY メガファンクション・ユーザガイド」
- 「AN 328: Stratix II デバイスによる DDR2 SDRAM インタフェース」
- 「AN 435: Stratix III デバイスの DDR および DDR2 SDRAM インタフェース実装のためのデザイン・ガイドライン」
- 「AN 436: Stratix III デバイスの DDR3 SDRAM インタフェース実装のためのデザイン・ガイドライン」
- 「AN 445: Cyclone III デバイスの DDR & DDR2 SDRAM インタフェース実装のためのデザイン・ガイドライン」
- 「AN 463: Using the ALTMEMPHY Megafunction with HardCopy II Structured ASICs」
- デバイス・ピン配置

Quartus II で デザインを 生成する前に

図 2 に、ALTMEMPHY メガファンクションを使用したシングル・メモリ・インタフェースを示します。ユーザ・ロジックから構成されているインタフェースがメモリ・コントローラに接続され、メモリ・コントローラが ALTMEMPHY メガファンクションに接続されます。ALTMEMPHY メガファンクションは、データ読み出し・パス・モジュール、アドレス / コマンド・パス・モジュール、データ書き込みパス・モジュールを介してメモリ・デバイスとインタフェースします。これらのモジュールは、メモリ・インタフェースに固有であるため、共用できません。



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。



デザイン・ファイル例 `top.qar` は、アルテラ・ウェブサイトの資料ページのアプリケーション・ノート・セクションで提供されています。

図 2. ALTMEMPHY メガファンクション・データ・パスを使用したメモリ・コントローラのブロック図

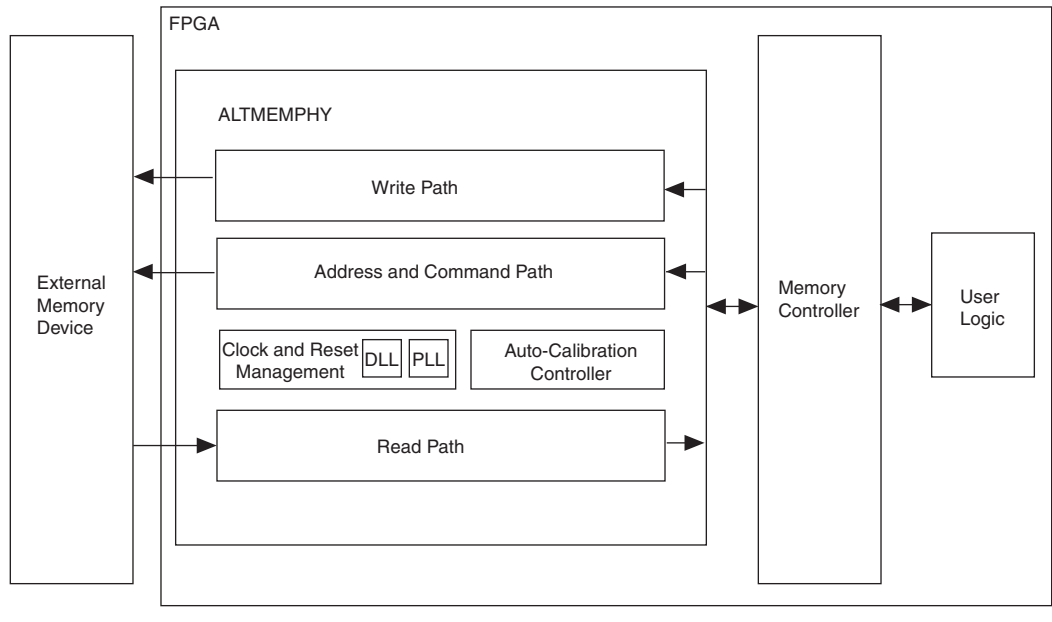


図 2 に示すオートキャリブレーション・コントローラ・ブロックを使用して、再同期化クロックのキャリブレーションと調整を行います。Quartus II バージョン 7.1 の ALTMEMPHY メガファンクションでは、このブロックの共用はできません。

クロックおよびリセット・マネジメント・ブロックは、ALTMEMPHY メガファンクションのすべてのクロック信号とリセット信号を発生します。このブロックは、複数のメモリ・インタフェース間で共用される DLL、PLL、PLL リコンフィギュレーション回路、リセット信号をインスタンス化します。



DLL、PLL、またはクロックの回路が共用される場合、各コントローラは同じ周波数で動作する必要があることに注意してください。

1 個のデバイスにフィットする複数のメモリ・インタフェースを生成する場合、まず各インタフェースに対して十分なデバイス・リソースがあることを確認してください。これらのデバイス・リソースとしては、ピン数、DLL 数、PLL 数、クロック・ネットワーク数などがあります。クロックおよびリセット・マネジメント・ブロックからの DLL、PLL、クロック・ネットワーク・リソースは、複数のメモリ・インタフェース間で共用できますが、注意しなければならない共用上の制約があります。これらの制約については以後の項で説明します。

FPGA について

ターゲット FPGA について次の事項を知っておく必要があります。

- 複数インタフェース・デザインの要求と FPGA 内で使用可能なリソースの比較
- 複数インタフェースを制限するリソースは何か
- FPGA にフィットできるメモリ・インタフェースを増やすために共用できるリソースは何か

次の項では、複数メモリ・インタフェース・デザインで使用できるデバイス・リソースについて説明します。必要なリソースと使用可能なリソースの比較を行った後に、共用するリソースを決定することができます。この結果により、システムの回路が決定されます。共用するリソースがない場合は、デバイス内に2つのモジュールとしてメモリ・コントローラを生成することができます。リソースを共用する場合について、DLL および / または PLL クロック出力の共用についてのガイドを次の項に示します。

I/O ピン数

DQS ピンと DQ ピンは、デバイスのピン・テーブルに記載されており、デバイスごとに固有な位置に固定されています。これらの位置は、配線のスキューを小さくし、マージンを大きくするように最適化されています。特定のデバイスでサポートされている DQS/DQ グループ数については、ピン・テーブルと外部メモリ・インタフェースの章で必ず確認してください。



複数メモリ・インタフェースに対しては I/O ピンを共用することはできません。

メモリ・インタフェースが FPGA にフィットするか否かを調べるときは、FPGA で使用可能なピン数とメモリ・インタフェースに必要なピン数を比較します。一般に、各ピンを1つの I/O バンクにフィットできない場合でも、デバイスの一方の側に配置すべきです。十分なピン数がない場合には、アドレス・ピンとコマンド・ピンをデバイスの異なる側に配置することができます。表 1 に、各メモリ・インタフェースに必要なピン数の一覧を示します。



表 1 では、キャリブレーション付きの直列 OCT またはキャリブレーション付きの並列 OCT、またはダイナミックにキャリブレートされる OCT の使用を想定しています。これは、R_{UP} ピンと R_{DN} ピンの使用で示してあります。

表 1. メモリ・インタフェースに対して必要なピン数 注 (1)、(2) (1 / 2)									
メモリ・インタフェース	バス幅	DQ ピン数	DQS ピン数	DM/BWSn ピン数	アドレス・ ピン数 (3)	コマンド・ ピン数	クロック・ ピン数	R _{UP} /R _{DN} ピン数	合計 ピン数
DDR3 SDRAM (4)、(5)	×4	4	2	0 (6)	14	10	2	2	34
	×8	8	2	1	14	10	2	2	39
	×16	16	4	2	14	10	2	2	50
	×72	72	18	9	16	15	4	2	136
DDR2 SDRAM (7)	×4	4	1	0 (6)	15	9	2	2	33
	×8	8	1	1	15	9	2	2	38
	×16	16	2	2	14	9	2	2	47
	×72	72	9	9	14	12	6	2	124
DDR SDRAM (5)	×4	4	1	0 (6)	14	7	2	2	28
	×8	8	1	1	14	7	2	2	33
	×16	16	2	2	14	7	2	2	43
	×72	72	9	9	13	9	6	2	118
QDRII+ SRAM	×9	18	2	1	19	3 (8)	4	2	49
	×18	36	2	2	18	3 (8)	4	2	67
	×36	72	2	4	17	3 (8)	4	2	104
QDRII SRAM	×9	18	2	1	19	2	4	2	48
	×18	36	2	2	18	2	4	2	66
	×36	72	2	4	17	2	4	2	103
RLDRAMII CIO	×9	9	2	1	22	7	4	2	47
	×18	18	2	1	21	7	6	2	57
	×36	36	2	1	20	7	8	2	76

メモリ・インタフェース	バス幅	DQ ピン数	DQS ピン数	DM/BWSn ピン数	アドレス・ ピン数 (3)	コマンド・ ピン数	クロック・ ピン数	R _{UP} /R _{DN} ピン数	合計 ピン数
RLDRAM II SIO	×9	18	2	1	22	7	4	2	56
	×18	36	2	1	21	7	6	2	75
	×36	72	2	1	20	7	8	2	112

表 1 の注:

- (1) これらのピン数の例は、メモリ・ベンダのデータ・シートから取得したものです。使用する構成のメモリ・デバイスの正確なアドレス・ピン数とコマンド・ピン数については確認が必要です。
- (2) PLL と DLL の入力基準クロック・ピンは、このカウントに含まれていません。
- (3) アドレス・ピン数は、メモリ・デバイスの集積度に依存します。
- (4) TDQS ピンと TDQS# ピンは、このカウントに含まれていません。
- (5) 数値は、1 GB メモリ・デバイスに基づいています。
- (6) アルテラの FPGA は、×4 モードで DM ピンをサポートしていません。
- (7) 数値は、差動 DQS、RDQS、RDQS# ピンのサポートを使用しない 2 GB メモリ・デバイスに基づいています。
- (8) QVLD ピンはカウントに含まれています。



FPGA DQS/DQ ピン数については、該当するデバイス・ファミリー・ハンドブックの外部メモリ・インタフェースの章を参照してください。



Stratix II、Stratix II GX、Arria GX、HardCopy II の各デバイスに対して ALTMEMPHY を使用する場合は、デバイスのトップとボトムのみをターゲットとすることができます。



Stratix III デバイスは、各 I/O バンクで外部メモリ・インタフェースをサポートしています。ただし、サイド I/O バンクでサポートしている最大 I/O トグル・レートまたは最大外部メモリ・インタフェース・クロック・レートは、トップ/ボトム・バンク内より低くなります。詳細は、「Stratix III ハンドブック Volume 2」の「Stratix III デバイスの DC およびスイッチング特性」の章を参照してください。

DLL

1 つの DLL は、同じ周波数で動作し、同じ DLL 周波数モードを使う任意数のインタフェース（ピン数、PLL 数、クロック・ネットワーク数、ロジック・エレメント・リソース数で制限されます）をサポートすることができます。各 FPGA ファミリの DLL については、次の項を参照して、周波数とメモリ・データ幅の条件に基づいて複数のメモリ・コントローラに対して DLL の共用が必要か否かを決定してください。

Stratix III デバイス

Stratix III DLL はデバイスの角に配置されているため、最寄りの側にある DQS ピンをその DLL へシフトすることができます。Stratix III デバイスは、合計 4 個の DLL があるので、個別の周波数を持つメモリ・インタフェースは最大 4 個になります。

Stratix II, Stratix II GX, Arria GX, HardCopy II の各デバイス

Stratix II, Stratix II GX, Arria GX, HardCopy II デバイスでは、各 FPGA のトップとボトムに DLL がそれぞれ 1 個あります。各 DLL は、DLL と同じ側にある DQS ピンだけをシフトすることができます。DLL は 2 個しかないので、Stratix II では個別の周波数を持つ最大 2 個のメモリ・インタフェースが可能です。

Cyclone III デバイス

Cyclone III デバイスでは、メモリ・デバイスとのインタフェースに DLL を使用しません。

PLL およびクロック・ネットワーク・リソース

ALTMEMPHY メガファンクションは、ALTMEMPHY MegaWizard® Plug-In Manager 内でインスタンス化された PLL を使います。この PLL は、メモリ・インタフェース・データ・パスとコントローラ内で必要とされる様々なクロックを発生する役割を持ちます。

表 2 に、ALTMEMPHY メガファンクションを使用するクロックの一覧を示します。すべてのクロックは I/O 周波数と同じ周波数で動作します。ただし、コントローラがハーフレート・モードであり、かつコントローラがフルレート・モードのメモリ・インタフェース周波数と同じ周波数で動作する場合、phy_clk_1x はインタフェース周波数の 1/2 で動作します。resynch_clk_2x クロックと measure_clk_2x クロックでは、PLL リコンフィギュレーション機能を使って、電圧と温度 (VT) の変動を調整するためにランタイムで位相シフトが変更されます。

表 2. ALTMEMPHY メガファンクションで使用されるクロック 注 (1) (1 / 2)

PLL 出力 カウンタ	ALTMEMPHY メガファンクション・ クロック名	用途
C0	phy_clk_1x	Static system clock for the data path and controller. アドレス信号とコマンド信号の発生にも使用。
C1	mem_clk_2x	DQS 信号、CK/CK# 信号、DLL に対する入力基準クロックの発生に使用されるスタティック DQS 出力クロック。
C2	write_clk_2x	DQS 信号より 90° 進んだ DQ 信号の発生に使用されるスタティック DQ 出力クロック。
C3	mem_clk_ext_2x	CK/CK# 信号に対して専用クロック出力を使用する場合に使用されるオプションのスタティック・クロック。

PLL 出力 カウンタ	ALTMEMPHY メガファンクション・ クロック名	用途
C4	resynch_clk_2x	再同期化とポストアンプル・パスに使用されるダイナミック・フェーズ・クロック。現在、このクロックは複数インタフェース間で共用できません。
C5	measure_clk_2x	VT トラッキングに使用されるダイナミック・フェーズ・クロック。現在、このクロックは複数インタフェース間で共用できません。
C6 (Stratix IIIのみ)	ac_clk_2x	アドレス信号とコマンド信号の専用スタティック・クロック。

表 2 の注:

- (1) 詳細は、該当する FPGA ファミリのハンドブックのクロック・ネットワークおよび PLL の章を参照してください。

表 3 に、Stratix III、Stratix II、Stratix II GX、Arria GX、HardCopy II、Cyclone III の各デバイスで使用可能な PLL 数と専用クロック出力数の比較を示します。

デバイス・ファミリ	fast PLL (2)	enhanced PLL
Stratix III	N/A	4-12 (3)
Stratix II	4-8	2-4
Stratix II GX	2-4	2-4
Arria GX	2-4	2-4
HardCopy II	2-8	2-4
Cyclone III	N/A	2-4 (3)

表 3 の注:

- (1) 詳細は、該当する FPGA ファミリのハンドブックのクロック・ネットワークおよび PLL の章を参照してください。
- (2) Stratix II および Stratix II GX の fast PLL は、ALTMEMPHY メガファンクションで必要とされるクロック出力数を持っていないため、ALTMEMPHY メガファンクション内で使用することはできません。
- (3) Stratix III および Cyclone III の PLL は Stratix II の enhanced PLL とは異なりますが、Stratix II PLL と比べた場合、fast PLL より enhanced PLL に似ています。

デバイス・ファミリ	enhanced PLL クロック出力数	専用クロック出力数
Stratix III	レフト/ライト: 7 本のクロック出力 トップ/ボトム: 10 本のクロック出力	レフト/ライト: 2 本のシングル・エンドまたは 1 差動ペア トップ/ボトム: 6 本のシングル・エンドまたは 4 本のシングル・エンドと 1 差動ペア
Stratix II	各 6 本のクロック出力	3 本の差動または 6 本のシングル・エンド (トップおよびボトム I/O バンクのみ)
Stratix II GX	各 6 本のクロック出力	3 本の差動または 6 本のシングル・エンド (トップおよびボトム I/O バンクのみ)
Arria GX	各 6 本のクロック出力	3 本の差動または 6 本のシングル・エンド (トップおよびボトム I/O バンクのみ)
HardCopy II	各 6 本のクロック出力	3 本の差動または 6 本のシングル・エンド (トップおよびボトム I/O バンクのみ)
Cyclone III	各 5 本のクロック出力	1 本のシングル・エンドまたは 1 差動ペア

表 5 に、アルテラ・デバイス・ファミリで使用できるクロック・ネットワーク数を示します。



PLL とクロック・ネットワーク・リソースについて詳しくは、該当するデバイス・ファミリのハンドブックのクロック・ネットワークおよび PLL の章を参照してください。

デバイス・ファミリ	グローバル・クロック・ネットワーク	リージョナル・クロック・ネットワーク
Stratix III	16	64-88
Stratix II	16	32
Stratix II GX	16	32

表 5. アルテラ・デバイス・ファミリのクロック・ネットワーク数
注 (1) (2 / 2)

デバイス・ファミリ	グローバル・クロック・ネットワーク	リージョナル・クロック・ネットワーク
Arria GX	16	32
HardCopy II	16	32
Cyclone III	10-20	N/A

表 5 の注:

- (1) ユーザのインタフェースで使用可能なクロック・ネットワーク数をより理解するためには、該当するデバイス・ファミリ・ハンドブックのクロック・ネットワークおよび PLL の章を参照して、デバイス・エリアごとのクロック・ネットワーク・リソース数を確認してください。

複数のインタフェース間でクロック・ネットワーク、PLL クロック出力、または PLL の共用が必要か否かを決めてください。

その他のリソース

TriMatrix™ エンベデッド・メモリ・ブロックのような、外部メモリ・インタフェースで使用されるその他のデバイス・リソースに注意してください。デザイン内の他のモジュールに対して多くの TriMatrix エンベデッド・メモリ・ブロックが必要な場合は、メモリ・インタフェースに対して十分であることを確認してください。

プロジェクト回路の生成

デバイス・リソースの制約を理解したら、複数のコントローラ・デザインで共用するリソースを決定します。これを Quartus II デザインの制約に対する枠組みとして使用することができます。例えば、ALTMEMPHY メガファンクション・データ・パスとインタフェースする Stratix II 外部メモリは、デバイスのトップ側とボトム側でのみサポートされます。また、デザイン内に PCI インタフェースがある場合には、PCI インタフェース用に少なくとも 1 バンクを残しておく必要があります (デバイスのトップ側またはボトム側)。このデザイン例では、1 個の DLL を共用する必要があります。

PHY および コントローラ の生成

各インタフェースは、メモリ・インタフェースを複数回インスタンス化する代わりに、個別に生成する必要があります。



ALTMEMPHY メガファンクションの使用方法については、「[ALTMEMPHY メガファンクション・ユーザガイド](#)」および「[DDR および DDR2 SDRAM 高性能コントローラ・ユーザガイド](#)」を参照してください。



高速なタイミング・クロージャを保証するには、複数のメモリ・インタフェースを持つトップ・レベル・デザインを生成する前に、各コントローラでタイミングを閉じ、LogicLock™ ブロック内に配置しておくことができます。

各高性能コントローラまたは ALTMEMPHY メガファンクションごとにシミュレーション・ファイルを生成できます。次に、各メモリ・インタフェースを個別にシミュレーションするか、あるいはデザイン内のすべてのメモリ・インタフェースをまとめるテストベンチを生成することができます。

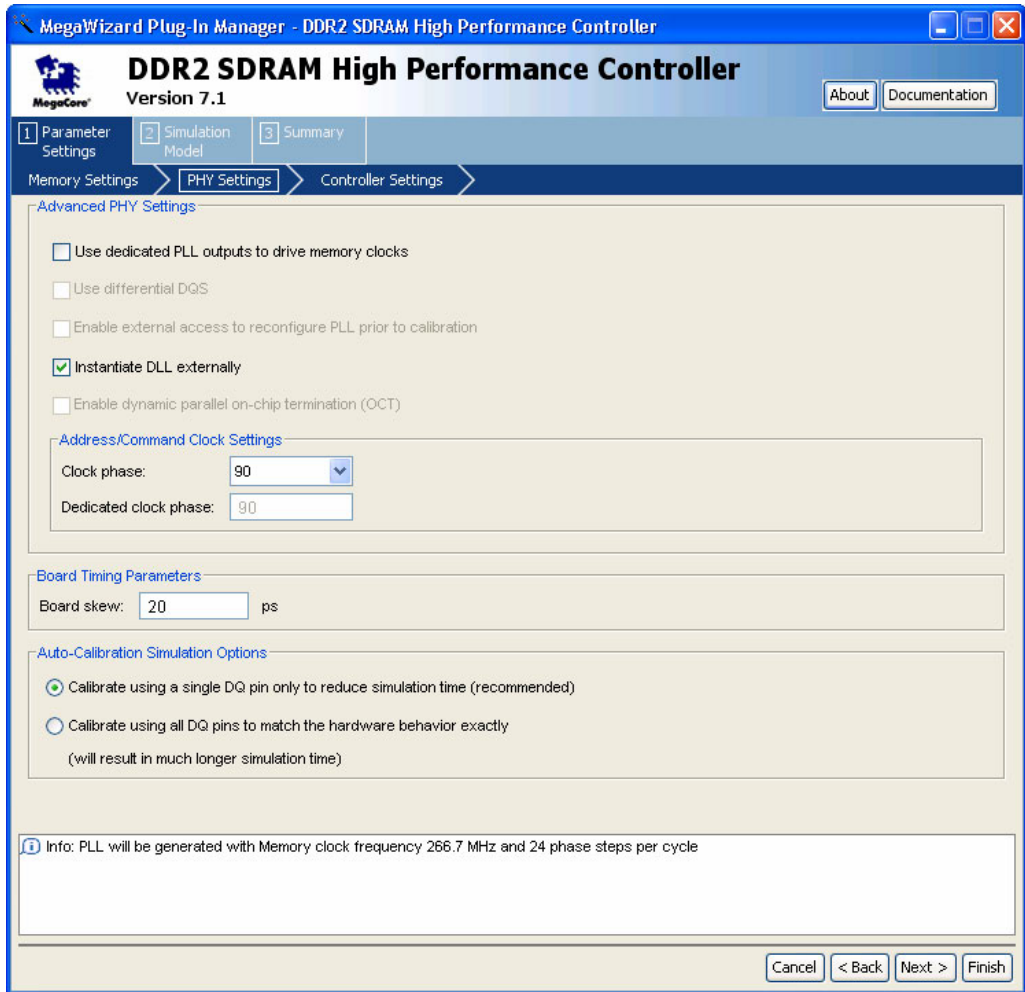


メモリ・インタフェースのシミュレーションについては、「[DDR および DDR2 SDRAM 高性能コントローラ・ユーザガイド](#)」の「[デザイン例シミュレーション](#)」の項または「[ALTMEMPHYメガファンクション・ユーザガイド](#)」の「[ALTMEMPHYシミュレーション](#)」の項を参照してください。

DLL の外部からのインスタンス化

コントローラが DLL を共有する場合には、[図 3](#) に示すように、DDR2 SDRAM High Performance Controller MegaWizard Plug-In Manager の **PHY Settings** ページで **Instantiate DLL externally** オプションがオンになっていることを確認してください。このオプションは、DLL を共有する各インタフェースに対してチェックする必要があります。

図 3. コントローラの DLL 共有を可能にするため DLL を外部からインスタンス化



コントローラのインスタンス化については、「[DDR および DDR2 SDRAM 高性能コントローラ・ユーザガイド](#)」を参照してください。

トップ・レベル・ファイルで、新しい `.v/.vhd` ファイル内で DLL をインスタンス化する必要があります。図 4 に、267 MHz 動作の `.v` ファイル内の例を示します。DLL は、MegaWizard Plug-In Manager から使用することはできません。

図 4. 267 MHz 動作 DLL のインスタンス化

```

module alt_mem_phy_stratixii_dll (
    dll_ref_clk,
    dqs_delay_ctrl
);

input dll_ref_clk;
output wire [5:0] dqs_delay_ctrl;
parameter DLL_DELAY_BUFFER_MODE = "HIGH";
parameter DLL_DELAY_CHAIN_LENGTH = 10;
parameter MEM_IF_CLK_PS_STR = "3750 ps";

stratixii_dll # (
    .delay_buffer_mode (DLL_DELAY_BUFFER_MODE),
    .delay_chain_length(DLL_DELAY_CHAIN_LENGTH),
    .delayctrlout_mode ("normal"),
    .input_frequency (MEM_IF_CLK_PS_STR),
    .jitter_reduction ("false"),
    .offsetctrlout_mode ("static"),
    .sim_loop_delay_increment (144),
    .sim_loop_intrinsic_delay (3600),
    .sim_valid_lock (1),
    .sim_valid_lockcount (27),
    .static_offset ("0"),
    .use_upndnin ("false"),
    .use_upndninclkena ("false")
) dll (
    .clk (dll_ref_clk),
    .aload (1'b0),
    .delayctrlout (dqs_delay_ctrl),
    .offsetctrlout (),
    .dqsupdate (),
    .upndnout ()
);

```

Stratix II デバイスで DLL 周波数モード 1、2、または 3 で動作させる場合、または Stratix III デバイスで DLL 周波数モード 4、5、または 6 で動作させる場合、DLL_DELAY_BUFFER_MODE を **HIGH** に設定します。その他の場合には、**LOW** に設定します。



メモリ・インタフェースのクロック・レートとして使う特定の周波数モードに対する DLL_DELAY_CHAIN_LENGTH の値については、Stratix II、Stratix II GX、Stratix III、Arria GX のハンドブックの「DC およびスイッチング特性」の章を確認してください。MEM_IF_CLK_PS_STR の値としては、メモリ・インタフェースの周期（単位 ps）を設定します。

DLL インスタンスをメモリ・インタフェースへ接続する際、両インタフェースで DLL の `dqs_delay_ctrl` 出力信号を `dqs_delay_ctrl_import` 信号へ接続します。DLL にも基準クロックが必要です。一般に、基準クロックは、一方のメモリ・インタフェースの `mem_clk_2x` 信号に接続されます。このクロックは、メモリ・クロック・レートと同じ周波数である必要があります。DLL を外部からインスタンス化するためにコアを生成する際、DDR および DDR2 SDRAM 高性能コントローラ MegaWizard Plug-In Manager は、`dll_reference_clock` と呼ばれる出力信号を生成します。この信号を DLL モジュールの DLL 入力クロック `dll_ref_clk` ピンへ接続することができます。



その他のクロック・ネットワークまたは PLL を共有する必要がない場合には、19 ページの「[デザインに対する制約の追加](#)」項へ進むことができます。

PLL クロック出力またはクロック・ネットワークの共用

PLL クロック出力を共用する場合には、デザインをさらに変更する必要があります。これらの変更は、PLL の共用方法に依存します。



ALTMEMPHY メガファンクション・ファイルを変更した後は、MegaWizard Plug-In Manager GUI を再度開くことはできません。これは、変更箇所が MegaWizard Plug-In Manager GUI を開くことによつて上書きされてしまうからです。

使用できる PLL クロック出力共用オプションは複数あります。これらの説明を次に示します。

■ 複数の再同期化クロックで PLL を共用

このオプションは、デザイン内の PLL 数を最小にしたい場合で、かつメモリ・インタフェースに必要とされるクロック数を発生するために十分な出力数を PLL が持つ場合に使用されます。模倣パスは、使用可能なクロック・ネットワーク数と PLL 出力数に応じて、共用することも、独立に使用することもできます。このオプションは、Stratix II デバイスと Stratix III デバイスで使用することができます。[図 5](#) に、この手法に対する DLL と PLL 間の接続図を示します。

図 5. 2 個のコントローラが PLL およびスタティック・クロックを別々の再同期化クロックと共用

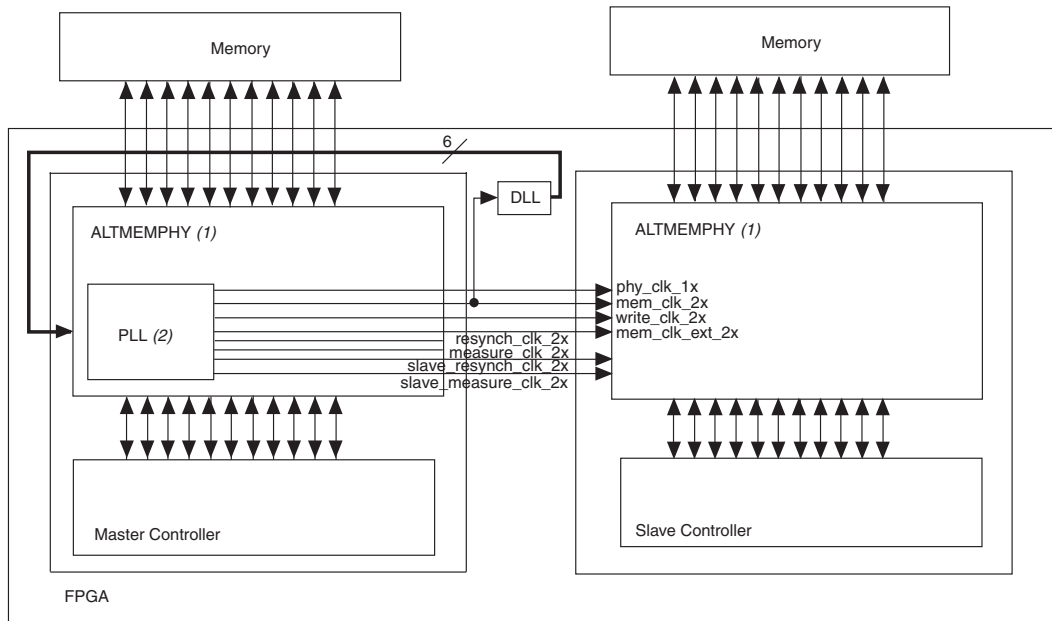



図 5 の注:

- (1) 図のように PLL クロックを共有するように、マスタ・コントローラの ALTMEMPHY メガファンクション・ファイルを変更します。スレーブ・コントローラの ALTMEMPHY メガファンクション・ファイル内で PLL をディセーブルします。
- (2) マスタ・コントローラの PLL を変更して、スレーブ・コントローラの再同期化と模倣バス用に 2 出力を追加します。出力クロックをスレーブ・コントローラへ配線できるようにマスタ・コントローラ PLL を変更します。


この手法で共有できる最大インタフェース数は、使用可能な PLL 出力数とデバイスのピン数によって制限されます。表 2 で説明されているように、ALTMEMPHY メガファンクションには 5 個のクロックが必要です。CK/CK# 信号に専用クロック出力を使用する場合は 6 個必要です。一方、Stratix II PLL は最大 6 個の出力を持つことができます。これは、専用クロック出力を使用する場合、個別の再同期化クロックまたは測定クロックを持たないことを意味します。CK/CK# 信号を発生するために専用クロック出力を使わない場合には、2 つのコントローラ (Stratix II でのこの方式に対する最大数) 間で異なる再同期化クロックを持つことができますが、測定クロックは共有する必要があります。


HardCopy II デバイスへ移行する際には、CK/CK# 信号の発生に専用クロック出力の使用が必要であるため、このオプションを使用することはできません。

トップとボトム of PLL が最大 10 個のクロック出力を持つことができる Stratix III デバイスでは、同じ PLL を個別の再同期化クロックおよび測定クロックと共用するコントローラを 3 個まで持つことができます。あるいは、同じ PLL および測定クロックを個別の再同期化クロックと共用するコントローラを 5 個まで持つことができます。

 同じ測定クロックを共用する場合、異なるコントローラからアクセスされるメモリ・デバイスが同じ配線パターン長を持つように、ボードがレイアウトされていることを確認する必要があります。

同じ PLL で各コントローラに対して異なる再同期化クロックを発生させるときは、マスタ・コントローラの ALTMEMPHY メガファンクションにステート・マシンを追加して、同じ PLL が接続されている各コントローラのキャリブレーションをスケジュールする必要があります。測定クロックも共用されている場合は、キャリブレートするコントローラを決定するアービトレーション・ロジックを追加する必要があります。

 同じ PLL で発生した再同期化クロックに対して、同じ PLL リコンフィギュレーション・ブロックを使用することができます。

 クロック・ネットワークの共用は、ALTMEMPHY メガファンクションのスタティック・クロックに対してのみ使用可能です。

これは、最大 4 個のクロック・ネットワークを最小化する実装し安いオプションです。これらのクロック・ネットワークは、メモリ・インタフェース・ロジックでコントローラの駆動、Stratix II デバイスと Stratix III デバイスで使用可能な DQ、DQS、CK/CK#、アドレス、コマンドの各信号の発生に使用されます。このオプションについては、デザイン例の項で詳しく説明します。図 6 に、この手法に対する DLL と PLL 間の接続図を示します。

図 6. 異なる PLL から出力される別々の再同期化クロックを使用した 2 本のコントローラ・スタティック・クロック

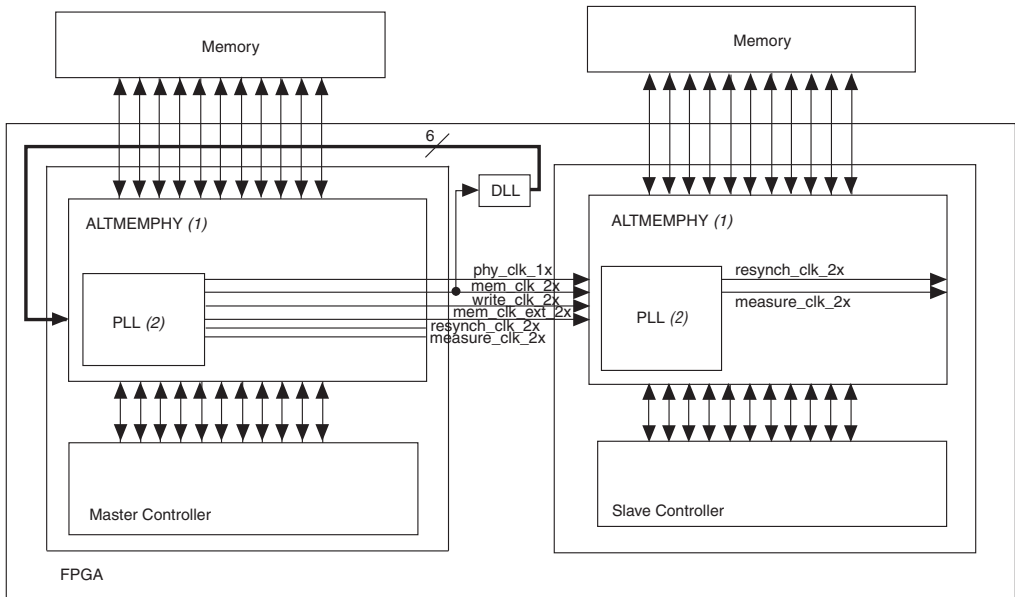


図 6 の注:

- (1) 図のように PLL クロックを共有するように、マスタ・コントローラの ALTMEMPHY メガファンクション・ファイルを変更します。
- (2) 再同期化および模倣バス用に 2 個の出力を持つようにスレーブ・コントローラの PLL を変更します。出力クロックをスレーブ・コントローラへ配線できるようにマスタ・コントローラ PLL を変更します。

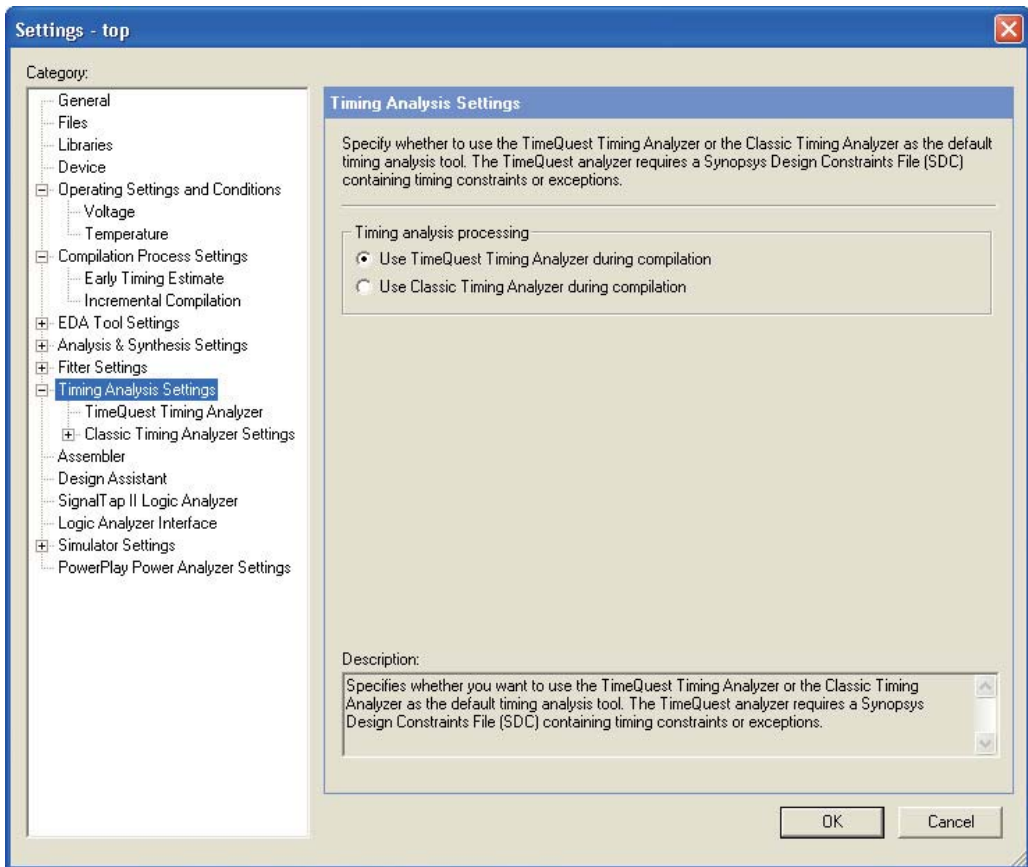
この方式で共有できる最大インターフェース数は、使用可能な PLL 出力数とデバイスのピン数によって制限されます。異なるコントローラからアクセスされるメモリ・デバイスが同じ配線パターン長を持つように、ボードがレイアウトされていることを確認する必要があります。

デザインに対する制約の追加

MegaWizard Plug-In Manager は、タイミング制約用の .sdc ファイル、I/O 規格用の .tcl スクリプト、各コントローラの DQS/DQ グルーピング制約を生成します。TimeQuest™ は、Stratix III and Cyclone III デバイス・ファミリーに対するデフォルトのタイミング・アナライザになっています。Quartus II コンパイラによりタイミングを最適化する場合、および MegaWizard Plug-In Manager を使って手動で生成したタイミング・レポート・スクリプトを使う場合には、TimeQuest タイミング・アナライザをイネーブルする必要があります。TimeQuest タイミング・アナライザをイネーブルするときは、次のステップを実行します。

1. Assignments メニューで **Settings** をクリックします。**Settings** ダイアログ・ボックスが表示されます。
2. **Category** リストで、**Timing Analysis Settings** を選択します。**Timing Analysis Settings** ページが表示されます。
3. **Timing analysis processing** で、**Use TimeQuest Timing Analyzer during compilation** を選択します(図 7に**Settings**ダイアログ・ボックスを示します)。
4. **OK** をクリックします。

図 7. TimeQuest タイミング・アナライザのイネーブル

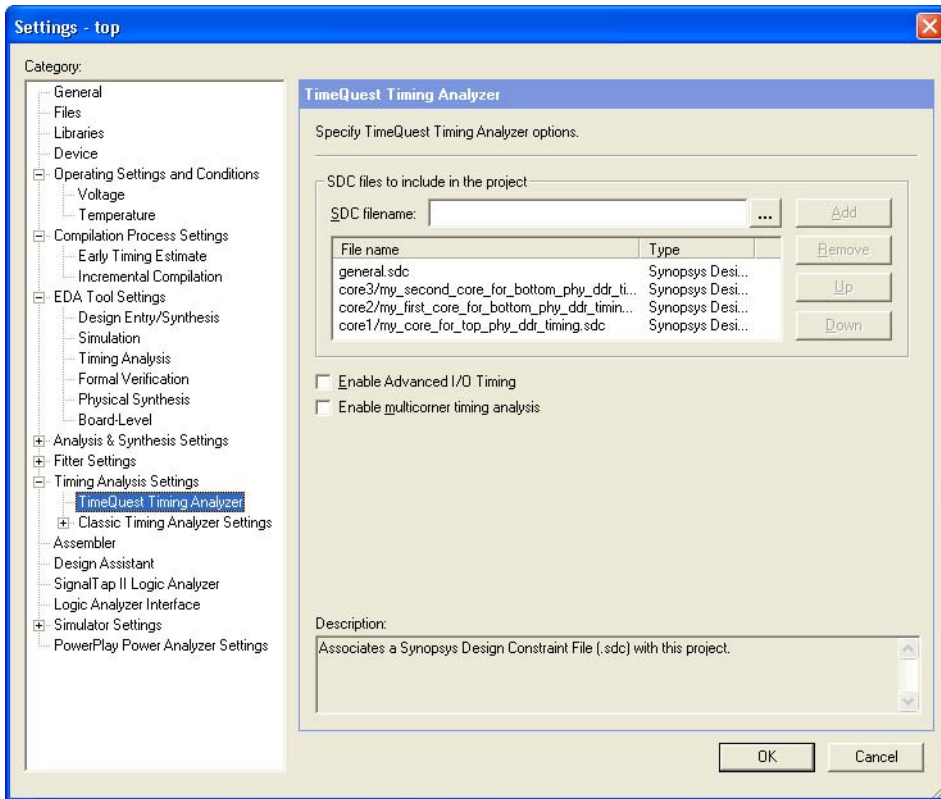


入力周波数クロックの周波数を指定するトップ・レベル .sdc を生成する必要があります。各 PLL 入力クロックは、トップ・レベル .sdc ファイル内に次の行を持つ必要があります。

```
create_clock -period <period_in_ns> <input_PLL_name>
```

トップ・レベル .sdc ファイルの他に、各コントローラの .sdc ファイルを追加する必要があります (図 8)。

図 8. デザイン内の各コントローラに対する .sdc ファイルの追加



ピン・アサインメントを追加するときは、Stratix II、Stratix II GX、Arria GX、HardCopy II、または Cyclone III の各デバイスをターゲットとする場合、<variation_name>_pin_assignments.tcl ファイルを実行します。Stratix III デバイスの場合は、<variation_name>_pin_assignments.tcl を実行して I/O 規格設定を追加し、さらに <variation_name>_phy_assign_dq_groups.tcl を実行して、Quartus II フィットによる正しい配置のために DQ および DQS ピン・グループを相互に関連付けます。

これらのピン・アサインメントでは、各コントローラに対して共通のデフォルトのピン名を使用します。各スクリプトを実行した後に、各コントローラのピン名が独自であるように変更する必要があります。代わりに、**Pin Planner** 内で I/O 規格アサインメントを割り当てることもできます。



Quartus II Text Editor 内で **Find and Replace** 機能を使用して、ピン名のプリフィックスを追加することができます。

Pin Planner を開いて、ピンの実際の位置を割り当てます。

Assignment へ移動して **Setting** をクリックすることにより、**Device and Pin** オプションの **Voltage** セクションで、デザインにデフォルトの I/O 規格を設定することもできます。



Quartus II が正常にデザインをコンパイルできるように、I/O バンク当たりの使用可能なピン数に注意する必要があります。

デザインの コンパイル

デザインをコンパイルする前に、MegaWizard Plug-In Manager で生成された **.sdc** ファイルを修正する必要があります。**.sdc** ファイル内で、各クロックは特定の PLL 出力に関連付けられます。共用されるクロックがあるため、別のコントローラからスタティック・クロックを受け取るダウンストリームのコントローラに対するレポート・タイミング・スクリプト内で、クロックの関連付けを変更する必要があります。

デザインの タイミング 検証

すべての `<variation_name>_phy_report_timing.tcl` ファイルの元になる新しい **.tcl** ファイルを生成して、Quartus II ソフトウェアがデザイン内のすべてのメモリ・インタフェースに対するタイミングをレポートできるようにすることが可能です。他のすべての **.tcl** スクリプトを呼び出す新しい **.tcl** ファイルを用意することができます。



Stratix III デバイスと Cyclone III デバイスでのメモリ・インタフェースのタイミング解析については、「[AN 438: Constraining and Analyzing Timing for External Memory Interfaces in Stratix III and Cyclone III Devices](#)」を参照してください。

Stratix II 複数コントローラ・デザインでは、リード・キャプチャがタイミング・エラーを起こすことがあります。リード・キャプチャに関するタイミング違反が発生した場合、デザイン内で使用する遅延チェーンを解析する必要があります(コンパイルに成功した後、**Fitter report summary** の **Resource** セクションに報告されます)。ホールド・タイムが負の場合、DQS パス遅延チェーンが 0 に設定されていることを確認してください。そうでない場合は、TCL コンソールから次のアサインメントを追加してください。

```
set_instance_assignment -name DQSOUT_DELAY_CHAIN 0 -to
<dqs_pin>
```

同様に、セットアップ・タイムが負の場合、**Assignment Editor** 内で **Input Delay from Pin to Input Register** を 0 に設定するか、または現在の設定より小さい値に設定することにより、DQ ピンで使われている遅延チェーンを削減する必要があります。

ライトおよびアドレス / コマンド・タイミング・エラーが発生した場合、次の設定により解決することができます。

```
set_instance_assignment -name CLOCK_TO_OUTPUT_DELAY 0 -to
<dq/dqs/address/command/CK/CK# pin>
```

再同期化パスがタイミングを満足しない場合、IOE の近くへ再同期化レジスタを移動します。ただし、レジスタを近づけ過ぎると、ホールド・タイム違反が発生することがあります。



デザイン内でタイミング・エラーを発生するその他の問題については、Quartus II リリース・ノート of the latest version をご覧ください。

デザインがタイミング条件を満たすようになると、ゲート・レベル・シミュレーションの実行またはボード・デザイン制約の決定からデザイン・フローを続けることができます。

デザイン例

上述のフローを説明するため、Stratix EP2S90F1508C3 デバイスは次のメモリ条件を持つものとします。

- 1 個のコントローラが 267 MHz で動作する ×72 DDR2 SDRAM DIMM デバイスとインタフェース
- 2 個のコントローラが 267 MHz で動作する ×8 DDR2 SDRAM デバイスとインタフェース

3 個のすべてのコントローラは、ハーフ・レート・モードでアルテラの DDR2 SDRAM 高性能コントローラを使用します。



ハーフ・レート・モードとは、メモリ・コントローラのクロック周波数がメモリ・デバイスの実際のクロック・レートの 1/2 の周波数であることを意味します。ハーフ・レート・モードでは、ALTMEMPHY がメモリ・コントローラとメモリ・デバイスの間でデータをマルチプレクス / デマルチプレクスして、2 つの周波数の間でデータを転送します。

I/O ピン数条件の比較

表 1 から、DDR2 SDRAM DIMM インタフェースは約 124 ピンを必要とし、×8 DDR2 SDRAM インタフェースは約 33 ピンを必要とすることが分かります。「Stratix II デバイス・ハンドブック」に記載されているように、Stratix II は最大 9 グループの ×8 DQS/DQ グループを持ち、DIMM インタフェースは FPGA の片側を必要とします。さらに、Quartus II ピン・プランナは、I/O バンク 3 に 94 ピンおよび I/O バンク 4 に 102 ピンあるため、DIMM インタフェース全体がトップ・バンクにフィットできることを示しています。

I/O バンク 7 と 8 には、それぞれ 100 本と 95 本のユーザ I/O ピンがあるため、2 個の ×8 DDR2 SDRAM インタフェースにより 1 個の I/O バンクを共用することができます。ただし、この場合、×8 DDR2 SDRAM インタフェースが 2 個の I/O バンクに配置されるため、2 個のインタフェースの間の分離は明確です。

結局、DIMM インタフェースはデバイスのトップ (I/O バンク 3 と 4) に配置されることになり、一方 1 個の ×8 DDR2 SDRAM コントローラは I/O バンク 7 に、他方は I/O バンク 8 に、それぞれ配置されることになります。

DLL 条件の決定

2 個のインタフェースがデザインのボトム側にあるため、これらのインタフェースは 1 つの DLL を共用する必要があります。

この例の場合、デバイスのボトムにある 2 個のインタフェースはスタティック・クロックを共有します (19 ページの図 6)。

デザイン例の PHY と コントローラ の生成

DIMM インタフェースはリソースを共用していないため特別な処理 (デザイン例では `my_core_for_top.v`) をしないで、1 つのインタフェースとして生成されます。

DLL を外部からインスタンス化するオプションを使って、2 個のボトムのインタフェースを生成します。デザイン例での `my_first_core_for_bottom.v` は、両ボトム・インタフェースに対してスタティック・クロックを出力するコントローラになります。他のコントローラには `my_second_core_for_bottom.v` の名前が付いています。



コントローラのインスタンス化については、「DDR および DDR2 SDRAM 高性能コントローラ・ユーザガイド」を参照してください。



ファイル構造を分かりやすくするため、トップ・プロジェクトのサブフォルダとして各コントローラを生成します。

DLL のインスタンス化

「DLL の外部からのインスタンス化」の項に従って、デバイスのボトムにある 2 個のコントローラの間で共用する DLL を生成します。

PLL クロック出力の共用

デバイスのボトムにある 2 個のメモリ・インタフェース間でスタティック・クロックを共用するときは、両コントローラの RTL コードのいくつかを変更する必要があります。共用されるスタティック・クロックとしては次の 4 個があります。

- phy_clk_1x
- mem_clk_2x
- write_clk_2x
- mem_clk_ext_2x



デザイン例の.vファイルの中ですべての変更を探すときは、**Find whole words only** オプションをチェックして「///」の検索を行います。

mem_clk_ext_2x クロック信号は、DDIO を使用しない場合に CK/CK# 信号を発生するためのオプション・クロックです。DDIO を使ってこれらのクロック信号を発生する場合は、この信号への接続を使用または変更する必要はありません。デザイン例では、DDIO を使って CK/CK# 信号を発生しています。ただし、下の説明には、mem_clk_ext_2x クロック信号に必要な RTL の変更が含まれています。



phy_clk_1x 信号は、トップ・レベルへ出力としてすでにエクスポートされていることに注意してください。他のコントローラへスタティック・クロックを出力しているコントローラ内では、この信号について変更することはありません。別の PLL からスタティック・クロックを入力しているコントローラでは、下に説明するように、この信号の方向を出力から入力へ変更する必要があります。

verilog を使っているデザインの場合、スタティック・クロックを出力しているコントローラ（デザイン例では **my_first_core_for_bottom.v** と呼ばれます）内で次の内容を実行してください。



各ステップは、.v ファイルに対して明示的に記述されています。.vhd ファイルに対するステップに従うことができますが、VHDL 言語の制約のために追加した入力ポートまたは出力ポートを持つ内部信号を追加する必要があることがあります。さらに、シミュレーション動作を変えないようにするため、追加した内部信号へ信号を割り当てる際に、デザインに余分なデルタ遅延が加わらないようにする必要があります。

1. `<variation_name>_phy_alt_mem_phy_sii.v/.vhd` で、次を実行します。

- a. Replace the wire declaration for the `mem_clk_ext_2x` signal in the `<variation_name>_phy_alt_mem_phy_clk_reset_sii.v` module with an output signal declaration.

```
///
//wire mem_clk_ext_2x;
output mem_clk_ext_2x;
///
```

- b. `<variation_name>_phy_alt_mem_phy_clk_reset_sii.v` モジュール内で出力ボード信号 `mem_clk_ext_2x` を追加します。
 - c. `<variation_name>_phy_alt_mem_phy_clk_reset_sii` モジュール・インスタンス内で `mem_clk_ext_2x` 信号のポート・マッピングを追加します。
 - d. `<variation_name>_phy_alt_mem_phy_clk_reset_sii.v` モジュールのポート宣言リスト内で、`mem_clk_2x` 信号と `write_clk_2x` 信号の配線宣言を出力信号宣言で置き換えます。
2. これらの出力を次のファイルへ送ります。
 - `<variation_name>_phy.v/.vhd`
 - `<variation_name>_controller_phy.v/.vhd`
 - `<variation_name>.v/.vhd`

他のコントローラからクロックを受け取るコントローラの中で、次を実行します。

1. `phy_clk` の信号宣言を出力から入力へ変更し、次のファイル内で、入力ポート、宣言、さらに `mem_clk_ext_2x`、`write_clk_2x`、`mem_clk_2x` の各信号に対するマッピングを追加します。
 - `<variation_name>.v/.vhd`
 - `<variation_name>_controller_phy.v/.vhd`
 - `<variation_name>_phy.v/.vhd`
 - `<variation_name>_phy_alt_mem_phy_sii.v/.vhd`

2. さらに、他のコントローラからスタティック・クロックを受け取るコントローラの `<variation_name>_phy_alt_mem_phy_sii.v/.vhd` ファイル内で、次を行う必要があります。

- a. `<variation_name>_phy_alt_mem_phy_sii.v` モジュール内で、`mem_clk_2x` 信号と `write_clk_2x` 信号の配線宣言をコメントにします。

```
// Clocks:
// full-rate memory clock
///
//wire mem_clk_2x;
///
//write_clk_2x is a full-rate write clock.
//It is -90° aligned to the system clock:
///
//wire write_clk_2x;
///
```

- b. コード `assign phy_clk=phy_clk_1x_src` を `assign phy_clk_1x_src=phy_clk` へ変更します。

```
///
//assign phy_clk = phy_clk_1x_src;
assign phy_clk_1x_src = phy_clk;
///
```

- c. `<variation_name>_phy_alt_mem_phy_clk_reset_sii.v` モジュール・インスタンス内で、ポート `mem_ext_clk_2x` を追加します。

- d. `<variation_name>_phy_alt_mem_phy_clk_reset_sii.v` モジュール・ポート・リストおよびポート宣言内で、入力ポート `mem_ext_clk_2x` を追加します。

- e. `my_second_core_for_bottom_phy_alt_mem_phy_clk_reset_sii.v` モジュール内で、`mem_clk_2x`、`phy_clk_1x`、および `write_clk_2x` の各ポート宣言を出力から入力へ変更します。

```
///
//output wire phy_clk;
input wire phy_clk;
///
```

- f. `mem_clk_ext_2x` 信号に対する配線宣言をコメントにします。

- g. PLL インスタンス内で、出力 c0、c1、c2、c3 のクロック名をコメントにするか、または変更します。これらの PLL 出力をデザインの他の部分で使用することができますが、出力クロック信号が `<variation_name>.v/.vhd` までの階層をたどることに注意する必要があります。図 9 に、ダウンストリーム・コントローラに対する PLL のインスタンス化を示します。

図 9. ダウンストリーム・コントローラ上の PLL インスタンス化

```
my_second_core_for_bottom_phy_alt_mem_phy_pll_sii pll
(
    .inclk0 (pll_ref_clk),
    .areset (pll_reset),
    /// .c0 (phy_clk_1x),
    /// .c1 (mem_clk_2x),
    /// .c2 (write_clk_2x),
    /// .c3 (mem_clk_ext_2x),
    .c4 (resync_clk_2x),
    .c5 (measure_clk_2x),
    .scanwrite (pll_scanwrite),
    .scanread (pll_scanread),
    .scanclk (pll_scanclk),
    .scandata (pll_scandata),
    .scandone (pll_scandone),
    .scandataout (pll_scandataout),
    .locked (pll_locked)
);
```

- h. `<variation_name>_phy_alt_mem_phy_clk_reset_sii.v` モジュール内で、コード `assign phy_clk_1x = mem_clk_2x` をコメントにします。

トップ・レベル・ファイルの生成

各コントローラを図 10 のように接続する必要があります。`.bdf` ファイルを使用する場合、これらのコントローラにポートが追加されているため、ボトム・コントローラのシンボル・ファイルを再生成する必要があります。

デザイン例に 対する制約の 追加

TimeQuest をイネーブルし、3 個のコントローラに対してすべての .sdc ファイルを追加します。

各コントローラには固有の <variation_name>_pin_assignments.tcl ファイルが添付されています。これらのピン・アサインメントでは、各コントローラに対して共通のデフォルトのピン名を使用します。各 .tcl ファイル内のピン名を変更して、そのコントローラの実際のトップ・レベル・ピン名 (図 10 の例ではプリフィックス top_、bot1_、bot2_ が付きます) がトップ・レベル・ファイル内のピン名に一致するようにします。図に、デザイン例内のいくつかのピン・アサインメントのスナップショットを示します。

図 11. デザイン例でのピン・アサインメント

From	To ▲	Assignment Name	Value	Enabled
	bot1_mem_dq[4]	Output Enable Group	720039649	Yes
	bot1_mem_dq[4]	Location	IOBANK_7	Yes
	bot1_mem_dq[5]	I/O Standard	SSTL-18 Class II	Yes
	bot1_mem_dq[5]	Output Enable Group	720039649	Yes
	bot2_mem_ras_n	I/O Standard	SSTL-18 Class II	Yes
	bot2_mem_ras_n	Location	IOBANK_8	Yes
	bot2_mem_we_n	I/O Standard	SSTL-18 Class II	Yes
	bot2_mem_we_n	Location	IOBANK_8	Yes
	top_mem_addr[0]	I/O Standard	SSTL-18 Class II	Yes
	top_mem_addr[0]	Location	EDGE_TOP	Yes
	top_mem_addr[1]	I/O Standard	SSTL-18 Class II	Yes
	top_mem_addr[1]	Location	EDGE_TOP	Yes

Pin Planner を開いて、ピンの実際の位置を割り当てます。Assignment へ移動して Setting をクリックすることにより、Device and Pin オプションの Voltage セクションで、デザインにデフォルトの I/O 規格を設定することもできます。

デザイン例の コンパイル

デザインをコンパイルする前に、MegaWizard Plug-In Manager により生成された .sdc ファイルを修正する必要があります。.sdc ファイル内で、各クロックは特定の PLL 出力に関連付けされます。共用されるクロックがあるため、別のコントローラからスタティック・クロックを受け取るダウンストリームのコントローラに対するレポート・タイミング・スクリプト内で、クロックの関連付けを変更する必要があります。

<variation_name>_phy_dds_timing.sdc を開く場合、スクリプトがコア名を階層名の一部としてアサインメントの残りの部分に対応させることに注意してください。ただし、ダウンストリーム・コントローラのスタティック・クロックは実際には別のコントローラから入力されるため、\${corename} 変数をアップストリーム・コントローラの名前で変更する必要があります。例えば、次の変更が必要です。

```
set ck_pll_pattern
*${corename}_alt_mem_phy_sii_inst|clk|pll|altpll_component|pll|clk\[1\]
```

から次へ:

```
set ck_pll_pattern
*my_first_core_for_bottom_phy_alt_mem_phy_sii_inst|clk|pll|altpll_component|pll|clk\[1\]
```

ここで、`my_first_core_for_bottom_phy` は、アップストリーム・コントローラのコア名です。

次を検索します。

```
${corename}_alt_mem_phy_sii_inst|clk|pll|altpll_component|pll|clk\[n\]
```

ここで、`n` は 0、1、2、または 3。\${corename} は .sdc ファイル内の実際のアップストリーム・コントローラ・コア名で置き換えます。

次の 2 項目は変更する必要はありません。

```
${corename}_alt_mem_phy_sii_inst|clk|pll|altpll_component|pll|clk\[4\]
${corename}_alt_mem_phy_sii_inst|clk|pll|altpll_component|pll|clk\[5\]
```

これらは、キャリブレーションと VT トラッキングに必要な再同期化クロックと測定クロックであるため、変更は不要です。



フィッタ設定は **Standard Fit** (最高エフォート) に設定され、最適化技術はデザイン例の **Speed** に設定されます。

入力 PLL ピンの周波数を指定するもう 1 つの .sdc ファイルを追加する必要があります。このデザイン例では、次の行を持つファイル **general.sdc** を使用します。

```
create_clock -period 7.5 top_pll_ref_clk
create_clock -period 7.5 bot1_pll_ref_clk
create_clock -period 7.5 bot2_pll_ref_clk
```

デザイン例の タイミング 検証

デザインがタイミング条件を満たすことを確認します。満たさない場合には、遅延チェーンを追加または削除して、あるいは配置制約を使ってレジスタを近くに移動して解決します。



デザイン内ですべての `<variation_name>_report_timing.tcl` ファイルを呼び出す .tcl ファイルを生成することができます。

図 12 に、デザイン内の 3 個のコントローラの初期タイミング解析結果を示します。

図 12. デザイン例の最終タイミング解析結果

```

i Info: -panel_name "my_core_for_top_phy Mimic (hold)"
i Info:                setup hold
i Info: Write          | 0.258 0.436
i Info: Address Command | 1.245 0.910
i Info: Half Rate Address/Command | 5.020 0.595
i Info: DQS vs CK      | 0.445 0.699
i Info: Read Capture   | 0.122 0.576
i Warning: Read Resync | 0.213 -0.035
i Warning: Core        | 0.014 -0.035
i Info: Core Reset/Removal | 0.267 0.173
i Info: Postamble      | 0.267 0.628
i Info: Mimic          | 0.441 0.000
i Info: Evaluation of Tcl script ./core1/my_core_for_top_phy_report_timing.tcl was successful
i Info: -panel_name "my_first_core_for_bottom_phy Mimic (hold)"
i Info:                setup hold
i Info: Write          | 0.624 0.702
i Info: Address Command | 1.753 0.817
i Info: Half Rate Address/Command | 5.493 0.444
i Info: DQS vs CK      | 0.518 0.593
i Info: Read Capture   | 0.590 0.118
i Info: Read Resync    | 0.188 0.047
i Warning: Core        | 0.014 -0.035
i Info: Core Reset/Removal | 0.267 0.173
i Info: Postamble      | 0.329 0.626
i Info: Mimic          | 0.014 0.000
i Info: Evaluation of Tcl script ./core2/my_first_core_for_bottom_phy_report_timing.tcl was successful
i Info: -panel_name "my_second_core_for_bottom_phy Mimic (hold)"
i Info:                setup hold
i Info: Write          | 0.618 0.705
i Info: Address Command | 1.821 0.911
i Info: Half Rate Address/Command | 5.611 0.603
i Info: DQS vs CK      | 0.519 0.754
i Info: Read Capture   | 0.590 0.118
i Warning: Read Resync | 0.199 -0.028
i Warning: Core        | 0.014 -0.035
i Info: Core Reset/Removal | 0.267 0.173
i Info: Postamble      | 0.298 0.657
i Info: Mimic          | 0.274 0.000
i Info: Evaluation of Tcl script ./core3/my_second_core_for_bottom_phy_report_timing.tcl was successful

```

図 12 に示すように、読み出し再同期化タイミングがトップ・インタフェースのタイミングを満たしていません。再同期化クロックがダイナミックであるため、これは真のタイミング・エラーではありません。ただし、`.sdc` ファイル内で使用している再同期化 / バランス・パラメータを変更して、この違反を解決することができます。これを行うときは、次の行を検索します。

```
set b [expr {$b_plus_d * 0.65 - $fpga_RESYNC_SETUP_ERROR}]
```

次の行は、上記行の下の 4 行にあります。

```
set d [expr {$b_plus_d * 0.35 + $fpga_PA_DQS_SETUP_ERROR}]
```

デフォルトのバランス比は、0.65 および 0.35 です。ただし、これはパッケージ/集積度の各組み合わせに対して正しいバランス比ではありません。特定のインタフェースに対してこれらの数値（合計 1.00 になる条件で）を変更する必要がある場合があります。このデザイン例では、3つのすべての .sdc ファイル内で比を 0.6 および 0.4 へ変更すると、トップ・インタフェースでの再同期化タイミングの問題は解決しますが、他のデザインでは異なる比が必要になります。



コアのタイミング・エラーに対処するときは、レジスタを IOE に近づける（セットアップ違反）か、または IOE から遠ざける（ホールド・タイム違反）ことができます。

図 13 に、デザインの最終タイミング解析結果を示します。

図 13. デザイン例の最終タイミング解析結果

```

Info: -panel_name "my_core_for_top_phy Mimic (hold)"
Info:          setup  hold
Info: Write           | 0.258 0.434
Info: Address Command | 1.124 0.990
Info: Half Rate Address/Command | 4.874 0.593
Info: DQS vs CK       | 0.406 0.697
Info: Read Capture    | 0.122 0.576
Info: Read Resync     | 0.169 0.040
Info: Core            | 0.009 0.018
Info: Core Reset/Removal | 0.139 0.127
Info: Postamble       | 0.461 0.622
Info: Mimic           | 0.443 0.000
Info: Evaluation of Tcl script ./core1/my_core_for_top_phy_report_timing.tcl was successful

Info: -panel_name "my_first_core_for_bottom_phy Mimic (hold)"
Info:          setup  hold
Info: Write           | 0.624 0.683
Info: Address Command | 1.876 0.813
Info: Half Rate Address/Command | 5.593 0.553
Info: DQS vs CK       | 0.528 0.702
Info: Read Capture    | 0.590 0.118
Info: Read Resync     | 0.186 0.043
Info: Core            | 0.009 0.018
Info: Core Reset/Removal | 0.139 0.127
Info: Postamble       | 0.139 0.652
Info: Mimic           | 0.009 0.000
Info: Evaluation of Tcl script ./core2/my_first_core_for_bottom_phy_report_timing.tcl was successful

Info: -panel_name "my_second_core_for_bottom_phy Mimic (hold)"
Info:          setup  hold
Info: Write           | 0.618 0.705
Info: Address Command | 1.909 0.831
Info: Half Rate Address/Command | 5.571 0.588
Info: DQS vs CK       | 0.539 0.742
Info: Read Capture    | 0.590 0.118
Info: Read Resync     | 0.212 0.038
Info: Core            | 0.009 0.018
Info: Core Reset/Removal | 0.139 0.127
Info: Postamble       | 0.484 0.635
Info: Mimic           | 0.262 0.000
Info: Evaluation of Tcl script ./core3/my_second_core_for_bottom_phy_report_timing.tcl was successful

```

デザインがタイミングを満たすようになると、ゲート・レベル・シミュレーションの実行またはボード・デザイン制約の決定からデザイン・フローを続けることができます。

まとめ

ALTMEMPHY メガファンクションを使用して、アルテラのFPGAを高性能外部メモリ・インタフェースヘインタフェースさせることができます。デバイスのアーキテクチャは、さらにFPGAを複数のメモリ・インタフェースヘインタフェースさせることができます。

このアプリケーション・ノートに示すように、ALTMEMPHY メガファンクション・データ・パスを使って複数コントローラ・デザインを生成する方法は多数あります。ALTMEMPHY メガファンクション・データ・パスおよび高性能コントローラとしてユーザのシステムに最適な方法を選択して、それを要求に合わせて修正することができます。

初期デザインで複数メモリ・インタフェースの条件と制約を理解すると、システムのデザインを向上させることができます。

改訂履歴

表 6 に、このアプリケーション・ノートの改訂履歴を示します。

表 6. 改訂履歴		
日付 & ドキュメント・バージョン	変更内容	概要
2007 年 6 月 v1.0	初版	—



101 Innovation Drive
San Jose, CA 95134
www.altera.com
Technical Support:
www.altera.com/support/
Literature Services:
literature@altera.com

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.

