合成可能な FCRAM コントローラ

XAPP266 (1.0) 2002 年 2 月 27 日

概要

このアプリケーション ノートでは、ダブル データ レート (DDR) 高速サイクル RAM (FCRAM) コントローラをインプリメントし、Virtex-II アーキテクチャに与える影響について説明します。

はじめに

一般的な DRAM メモリは、共通のメモリコアとセル アレイで構成されています。このコア技術を使用し、さらにペリフェラル ロジック回路でわずかに変更を行うことで、EDO、SDRAM、DDR SDRAM、Direct Rambus DRAM (RDRAM) などさまざまな高性能メモリを作成できるようになりました。

しかしながら、この従来型の技術では新しいメモリにおいても本来コア アーキテクチャが持つ制限がついてしまいます。そのため、このインターフェイスのかわりに、FCRAM を採用して内部 DRAM コアを再設計し、内部パフォーマンスを向上することができました。この再設計されたコアは消費電力を抑えるだけではなく、メモリ レイテンシも下げることができます。高密度メモリ、広いバンド幅、低消費電力などが要求されるデザインでは、FCRAM を従来のメモリ技術のかわりに使用できます。また、FCRAM はサーバーやハードウェアの機能を高めるデバイスとしてだけではなく、コンピュータ ネットワーク デバイスとしても使用できます。FCRAM™ は、富士通株式会社により商標登録されています。

このアプリケーション ノートでは、Virtex-II デバイスにインプリメントされた FCRAM コントローラ デザインについて説明します。FCRAM について基本的な説明をした後、インプリメントされたコントローラについて詳しく説明します。

DDR FCRAM について

基礎

このセクションでは、FCRAM インターフェイスおよびその動作について簡単に説明します。FCRAM について既に知識のある方は、次のセクション 「FCRAM コントローラ デザイン」へお進みください。

FCRAM デバイスは、SSTL-II I/O とインターフェイスしたコア電圧 2.5V で動作します。このアプリケーション ノートは、スピード グレード -22/-24/-30 で最大クロック周波数 154 MHz の FCRAM を対象にしています。FCRAM には、x8 または x16 （デバイスのデータ (DQ) ピンの数）コンフィギュレーション の 256 Mb があり、これらは富士通株式会社および株式会社東芝の製品です。

FCRAM は、クロックの立ち上がり時および立ち下がり時の両方でデータの読み書きを行う DDR (ダブル データ レート) インターフェイスを採用しています。これによりデータ処理能力が 2 倍になり、一定のクロック周波数を保つことができるため、多くの DRAM で採用されています。立ち上がり（ポジティブ）クロック エッジは、CLK の遷移が High を示し、立ち下がりクロック エッジは、$\overline{\text{CLK}}$ の遷移が Low を示します。

FCRAM は、行（上位アドレス）、列 (下位アドレス)、およびバンク（一般的な FCRAM にはバンクが 4 個含まれる）でアドレス指定されます。メモリ アクセス（読み込み/書き出し）は、バースト転送です。バースト転送とは、指定したバンクおよびアドレスからメモリ アクセスが開始し、プログラムされたシーケンスで指定した時間だけ継続する転送です。

FCRAM コントロール ロジックは、2 つの信号 $\overline{\text{CS}}$ および FN で構成されています。各 FCRAM の動作は、2 つの連続するコマンド入力により決定されます。最初のコマンドでコントローラ ステート マシン

の読み出し (RDA) または書き込み (WRA) が実行されます。RDA コマンドの後には読み出し (LAL) またはモード レジスタ セット (MRS) コマンドが実行されます。WRA コマンドの後には書き込みコマンド (LAL) またはメモリ リフレッシュ (REF) コマンドが実行されます。

図 1 に、FCRAM ステート マシンを示します。この図は、1 つのバンク動作を示しており、破線矢印は自動的に行われるシーケンスを示します。

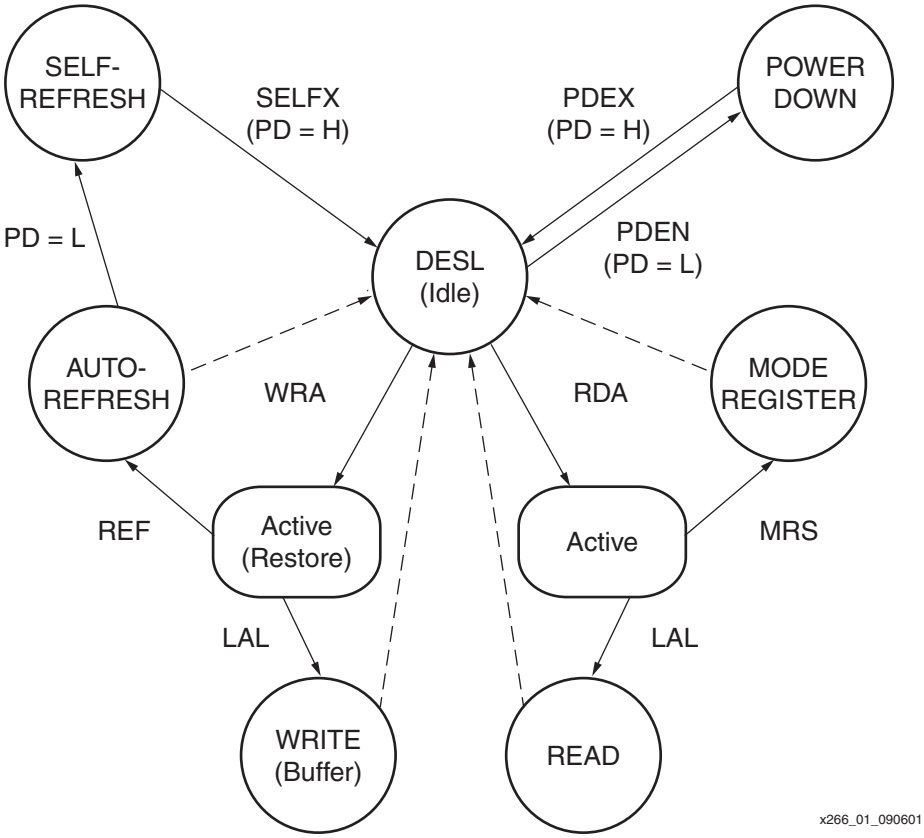

図 1： FCRAM ステート マシンの図

すべての FCRAM アドレスおよびコマンド信号は、クロックの立ち上がりエッジで FCRAM によりラッチされます。従来型の DDR DRAM と同様に、FCRAM も双方向性データ ストローブ信号 (DQS) を使用します。通常、このストローブ信号はクロックとして使用され、読み出しおよび書き込み中のデータを取得します。メモリの読み出しの中、FCRAM からデータにエッジ アラインしたストローブ信号が送信されます。このため、コントローラはデータを取得するために、ストローブ信号に遅延を与える必要があります。メモリの書き込み中、コントローラはデータにセンター アラインしたストローブ信号を送る必要があります。FCRAM は DQ および DQS 間の遅延を内部で一致させ、データを取得します。FCRAM 仕様では、データのすべてのバイト（8 DQ 行）に DQS が 1 つあります。

読み出し動作

FCRAM の読み出しコマンド (図 2) は、RDA コマンドで開始されます。このコマンドは、$\overline{CS}$ Low および FN High のアサートで発行されます。RDA コマンド中、指定したバンクおよび上位アドレスがアクティブになります。次のクロック サイクルで LAL コマンドが発行されます。この LAL コマンドは $\overline{CS}$ High のディアサートで発行されます。LAL コマンド中、下位アドレスがアクティブになります。

データは、読み出しコマンドの発行後にコントローラ CAS レイテンシ (CL) サイクルから取得できます。DQS の立ち上がりおよび立ち下がりエッジは、DQ バスでの有効なデータを示します。DQS はバースト長が終了するまでトグルします。

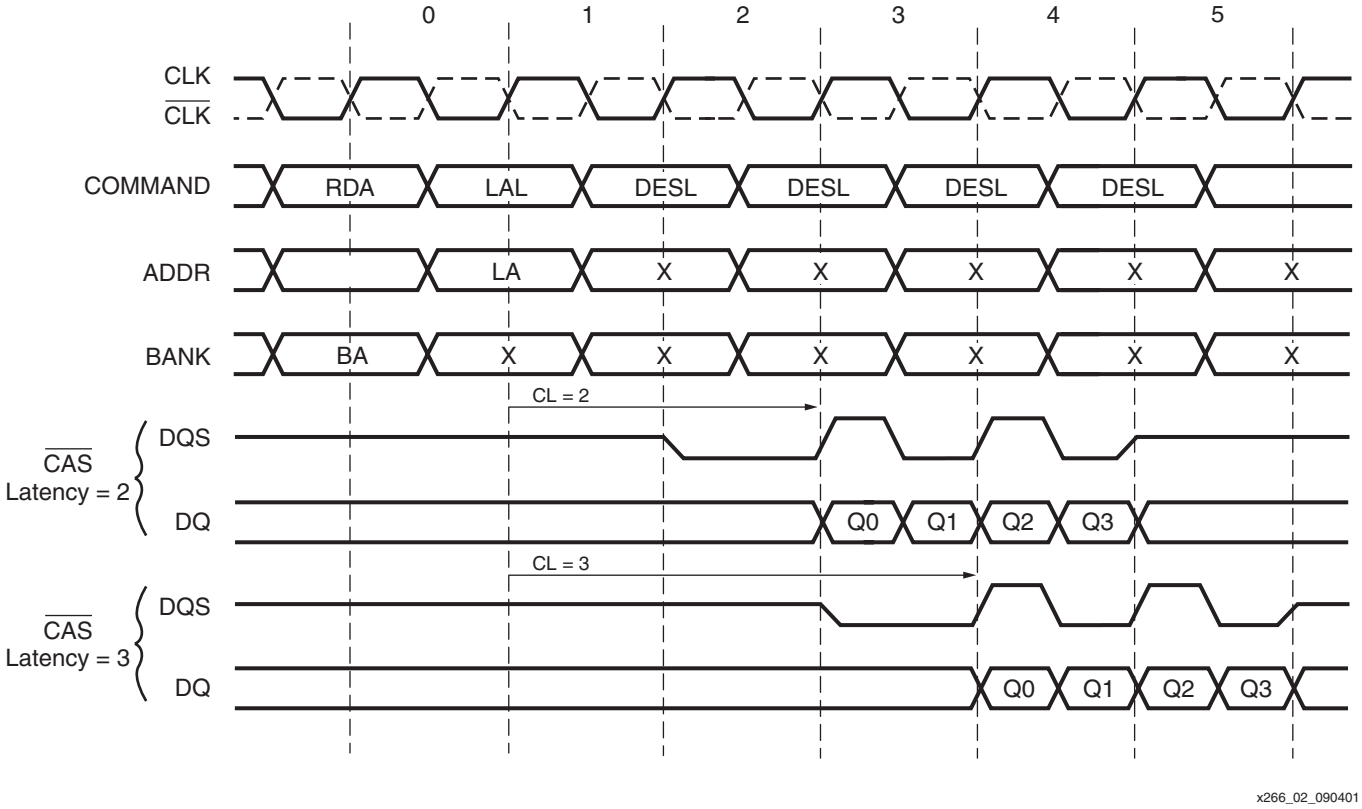

図 2： 読み出し動作のタイミング

書き込み動作

FCRAM の書き込みコマンド (図 3) は、WRA コマンドで開始されます。このコマンドは、$\overline{CS}$ Low のアサートおよび FN Low のディアサートで発行されます。WRA コマンド中、指定したバンクおよび下位アドレスがアクティブになります。次のクロック サイクルで LAL コマンドが発行されます。この LAL コマンドは、$\overline{CS}$ High のディアサートで発行されます。LAL コマンド中、下位アドレスがアクティブになります。

データ ストローブ信号 (DQS) が立ち上がる前の T_{DS} (data-in セットアップタイム) で、必ずデータが出力されます。LAL コマンドが発行された後、通常は最初の DQS 立ち上がりエッジで書き込みデータ レイテンシ (WL) サイクルが開始します。バースト長が終了するまで、DQS の立ち上がりおよび立ち下がりエッジで残りのデータ入力が行われます。

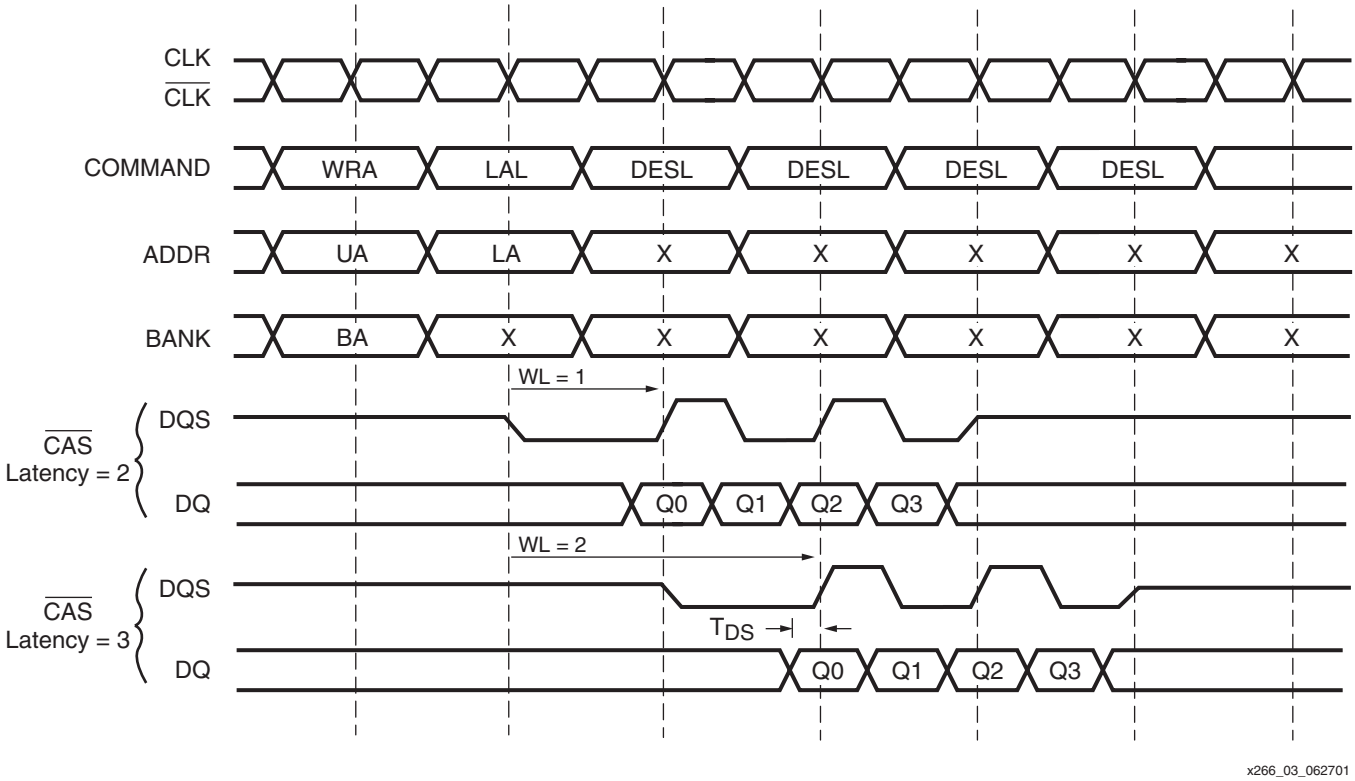

図 3： 書き込み動作のタイミング

モード レジスタ セット (MRS)

モード レジスタは、FCRAM 動作のモードを定義します。電源をオンにした時、モード レジスタは定義されていないためプログラムする必要があります。一度プログラムされると、電源がオフになるまで、または MRS コマンドが発行されて内容が更新されるまでレジスタの内容が保持されます。

FCRAM MRS モードは、RDA コマンドで開始されます。この RDA コマンドは、$\overline{CS}$ Low および FN High のアサートで発行されます。モード レジスタを設定する場合、RDA コマンド中のバンク入力およびアドレス入力は無視されます。MRS コマンドは、次のクロック サイクルで発行されます。$\overline{CS}$ Low のアサートで MRS コマンドが 発行されます。FCRAM のコンフィギュレーション値は、MRS コマンド中にバンクおよびアドレス ピンで発行されます。

通常 FCRAM には、標準モードおよび拡張モードの 2 種類のモード レジスタがあります。この 2 種類のモード レジスタの設定はそれぞれ異なるため、バンク入力に準じてどちらかを選択し、MRS コマンド中に設定を行う必要があります。

MRS コマンド中、必要な FCRAM コンフィギュレーションの情報がアドレス ピンに含まれます。標準モード レジスタ コンフィギュレーションは、バースト長 (A[2:0])、バースト タイプ (A3)、CAS レイテンシ (A[6:4]) およびテスト モード (A7) をプログラムします。

拡張モード レジスタ コンフィギュレーションは、DLL イネーブル (A0) および出力ドライバ インピーダンス コントロール (A1) をプログラムします。

バースト長 (BL)

FCRAM への読み出しおよび書き込みはバースト転送されます。つまり、行および列を選択すると、バースト長分の列に対して読み出しまたは書き込みコマンドが実行されます。このバースト長の設定はプログラム可能です。FCRAM メモリがサポートするバースト長は 2 または 4 です。

バースト タイプ (BT)

バースト タイプは、シーケンシャルまたはインターリーブの 2 タイプの設定が可能です。

CAS レイテンシ (CL)

読み出し動作中、CL は読み出しコマンド (LAL) の発行とデータが有効になる地点間でのクロック サイクルの遅延です。書き込み動作の場合、CL は書き込みコマンド (WRA) の発行と FCRAM へデータが送信される地点間でのクロック サイクルの遅延です。

テスト モード

テスト モードは、製造業者が使用するための動作モードです。通常の動作モードの場合、このビットは必ず 0 に設定してください。

DLL イネーブル

このビットを 0 に設定することにより、DLL がイネーブルになります。DLL をディスエーブルにする機能をサポートしないメモリ ベンダもあります。

出力ドライバ インピーダンス コントロール

このアプリケーション ノートをリリースする段階で、FCRAM 製造業者によりサポートされていないため、必ず 0 に設定してください。

リフレッシュ

FCRAM は、DRAM と同じ従来型のコンデンサを使用しているため、リフレッシュ動作を周期的に行い、セルに書き込まれたデータを保持する必要があります。また FCRAM には、自動リフレッシュおよびセルフ リフレッシュ機能があります。

自動リフレッシュは、WRA コマンドで開始されます (CS Low をアサートおよび FN Low をディアサート)。次に REF コマンドを発行します (CS Low をアサート)。REF コマンドの 2 クロック サイクル内に PD ピンを Low にアサートすると、FCRAM はセルフ リフレッシュ ステートに遷移し、PD がリリースされるまでこの状態を維持します。

FCRAM コントローラ デザイン

このセクションでは、Virtex-II FCRAM コントローラのデザインについて説明します。FCRAM コントローラには、ユーザー インターフェイスおよび FCRAM インターフェイスがあります。このデザインは、Verilog コードで記述されていますが、さまざまなメモリ コンフィギュレーションに対応するように変更できます。

FCRAM コントローラ デザインには、次のような特徴があります。

- バースト長は、2 または 4 をサポート
- CAS レイテンシは、2 または 3 をサポート
- リフレッシュ モード機能をユーザーが任意で行うことができ、またコントローラが自動的に開始することも可能
- 初期化シーケンス
- 下位レベル FCRAM 機能のインプリメンテーション
- FCRAM が、読み出し実行中に DQS を使用してデータを取得
- Virtex-II -5 デバイスにおいて、DDR FCRAM に最大 154 MHz までのインターフェイス

従来型の SDRAM とは異なり、FCRAM にはアクセス後にバンク/列をオープン状態にするというオプションがありません。このオプションのかわりに FCRAM ではアクセスが終了する毎に自動的に列を閉

じバンクの充電を行います。このため、ユーザーは各バースト長サイズのアクセスに対して、新しい読み出しまたは書き込みコマンドを毎回発行する必要があります。

FCRAM は、バースト長が 2 または 4 のみで処理を行うため、FCRAM デバイスの最大処理能力で実行すると、ユーザーの負担が大きくなる場合があります。つまりバースト長は、2 クロック サイクルで完了するため、2 サイクル毎に新しいメモリ アクセス コマンドを発行しなければなりません。これらのコマンドを発行する前に、バンクの衝突、読み出し-書き込みの転換時間、期限切れのリフレッシュカウンタなどの違反を必ず確認してください。違反についての詳細は、「FCRAM コントローラの動作」のセクションを参照してください。

ユーザー インターフェイスで違反を監視し、その違反に応じた処置を行う方法にかわり、FCRAM がこれらの違反を監視することで、ユーザー インターフェイスがシンプルになりました。開始バンク、アドレス ロケーション、完了すべきデータ転送の数などのメモリ アクセス コマンドを発行するだけで、FCRAM コントローラが自動的にすべてのインプリメンテーション作業を行います。

図 4 に、FCRAM コントローラの上位ブロック図を示します。モジュール *fcram_cntrl* が、FCRAM コントローラ ブロックの最上位にあります(図 5)。このモジュールには、クロック生成回路、コントローラ ステート マシン、リフレッシュ カウンタ、アドレス カウンタ、FCRAM へのデータ パスなどのサブモジュールが含まれます。すべての信号リファレンスおよびその説明は、このモジュールに関連しています。モジュール *user_int* は、ユーザー インターフェイスのプレース ホルダーです。この図では、(直接またはパイプラインを介して) システム信号を FCRAM コントローラへ伝搬しています。

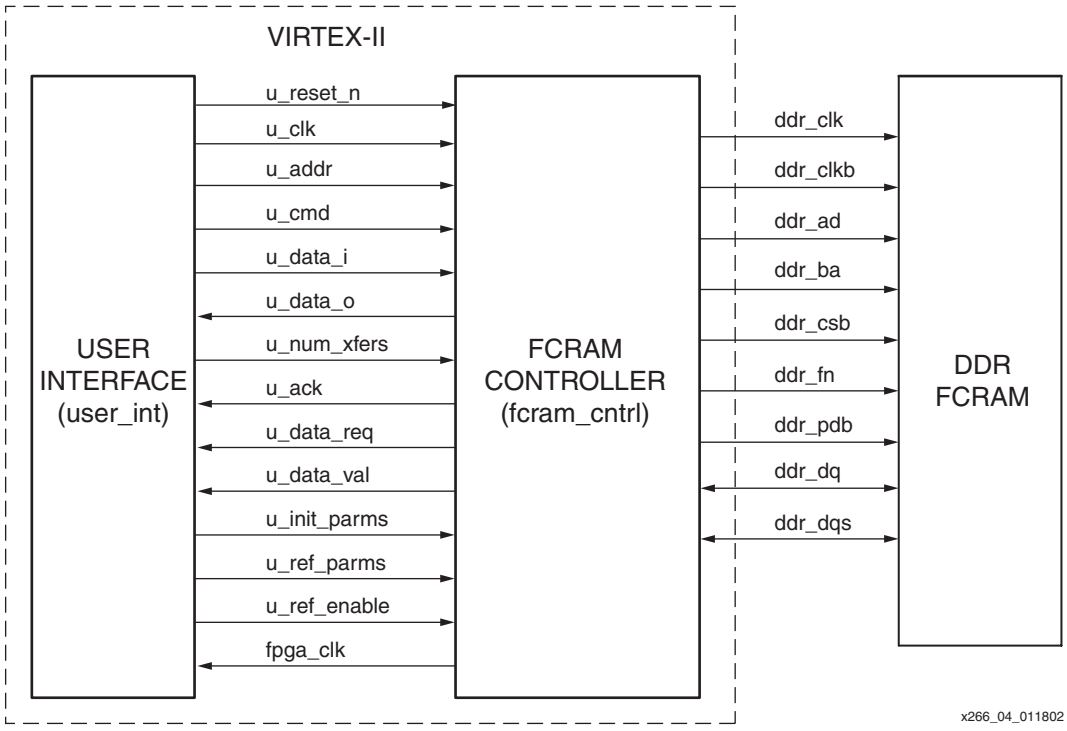

図 4： 最上位ブロック図

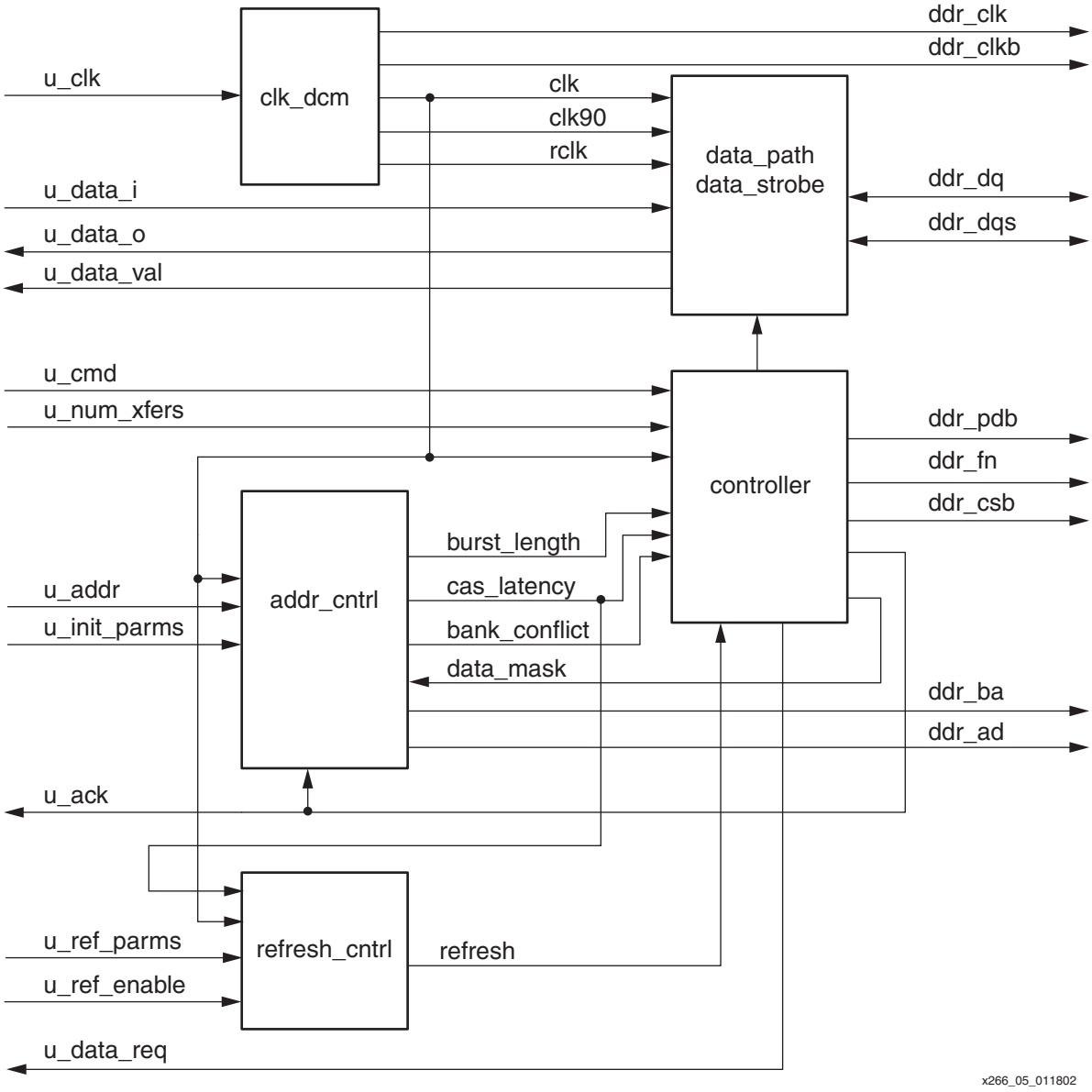

図 5： fcram_cntrl ブロック図

FCRAM コントローラの動作

表 1 に、FCRAM コントローラへのユーザー インターフェイス信号を示します。表 2 には、FCRAM デバイスへのインターフェイス信号を示します。

表 1: FCRAM コントローラへのユーザーインターフェイス

ピン名	方向	幅	説明
u_reset_n	In	1	リセット、アクティブ Low
u_clk	In	1	入力クロック
u_addr	In	27	アドレス : u_addr = {bank(2), row(15), col(10)}

表 1: FCRAM コントローラへのユーザーインターフェイス

u_cmd	In	3	コントローラにより実行されるコマンド [0 x x] NOP [1 0 0] 書き込みリクエスト [1 1 0] 読み出しリクエスト [1 0 1] セルフ フレッシュ リクエスト [1 1 1] 自動リフレッシュ リクエスト
u_data_i	In	32	書き込みデータ
u_data_o	Out	32	読み出しデータ
u_num_xfers	In	4	転送する 32 ビット データ値の数
u_ack	Out	1	コントローラがユーザー インターフェイスにより発行されたコマンドを認識 (実行を保証)
u_data_req	Out	1	ユーザーが書き込みデータ値 (u_data_i) を与える
u_data_val	Out	1	読み出しデータ値 (u_data_o) は有効
u_init_parms	In	10	初期化パラメータ : u_init_parms = {CL(3),BL(3),TE,BT,DE,DIC}
u_ref_parms	In	20	リフレッシュ間隔パラメータ : u_ref_parms = {ref_burst_cnt[3:0], ref_interval_cnt[15:0]}
u_ref_enable	In	1	自動コントローラ リフレッシュをイネーブル
fpga_clk	Out	1	FCRAM コントローラ内部クロック

メモ :

1. **MSB:**このデザインでは、上位ビットが MSB です。例 : u_cmd[2:0] = 100 が書き込みリクエスト。

表 2: FCRAM デバイスへのコントローラ インターフェイス

ピン名	方向	幅	説明
ddr_clk	Out	1	クロック
ddr_clkb	Out	1	反転クロック
ddr_ad	Out	15	アドレス
ddr_ba	Out	2	バンク アドレス
ddr_csb	Out	1	コマンド
ddr_fn	Out	1	コマンド
ddr_pdb	Out	1	コマンド
ddr_dq	In/Out	16	データ
ddr_dqs	In/Out	2	データ ストローブ

データ バス幅

このアプリケーション ノートでは、x16 の FCRAM デバイスを対象にして説明します。一方、さまざまなメモリ コンフィギュレーションや複数 FCRAM デバイスに対応できるように、データ幅はパラメータ設定可能であり、HDL コードを使用して簡単に変更できます。メモリ コンフィギュレーションの変更についての詳細は、付録 A を参照してください。このアプリケーション ノートでは、32 ビット転送としてユーザー インターフェイスでのデータ転送を例にあげて説明します。データ幅を変更する場合は、このアプリケーション ノートに記載されている 32 ビットでの基準値を、変更した値に置き換えてください。

動作なし (IDLE/DESL)

```
Set u_cmd = 0xx
```

このコマンドで、コントローラを IDLE ステートに保持します。

初期化

初期化シーケンスを行うと FCRAM のモード レジスタの設定が可能になります (表 3)。初期化シーケンスは、コントローラがリセットされたとき、または電源をオンにしたときに自動的に行われます。したがって、モード レジスタ セット (MRS) コマンドおよび拡張モード レジスタ セット (EMRS) コマンドなどを発行する必要はありません。

この初期化シーケンスの間、ユーザー インターフェイスが FCRAM コントローラに初期化パラメータを与えます。初期化パラメータは、u_init_parms を使用してユーザー インターフェイスから渡され、次のように示します。

```
u_init_parms[9:0] = {CL(3),BL(3),TE,BT,DE,DIC}
```

表 3: 初期化パラメータの説明

パラメータ名	幅	説明
CL	3	CAS レイテンシ [0 0 x] 予約済み [0 1 0] 2 [0 1 1] 4 [1 x x] 予約済み
BL	3	バースト長 [0 0 0] 予約済み [0 0 1] 2 [0 1 0] 4 [0 1 1] 予約済み [1 x x] 予約済み
TE	1	テスト モード [0] レギュラー モード (デフォルト) [1] テスト モード
BT	1	バースト タイプ [0] シーケンシャル [1] インターリーブ
DE	1	DLL イネーブル [0] DLL イネーブル (デフォルト) [1] DLL ディスエーブル
DIC[1]	1	出力ドライブ インピーダンス コントロール [0] スタンダード [1] 予約済み

メモ :

1. 現在、FCRAM 製造業者では DIC オプションをサポートしていないため、このビットを Low に接続してください [u_init_parms(0)=0]。今後は互換性を持たせる予定です。

リセット信号が解除されると、まずシステムは DCM がロックするまで待機します。ロック完了後、コントローラは u_init_parms ベクタをラッチし、リセット/初期化プロセスを開始します。表 4 に、FCRAM プロセスの詳細を示します。

表 4: 電源投入時の初期化およびリセット条件

コマンド	解説
DESL	12 サイクルまたはそれ以上
MRS	リセット アドレスで MRS コマンドを発行
DESL	4 サイクルまたはそれ以上の間、同じアドレスを保持
DESL	アドレス変更
DESL	4 サイクルまたはそれ以上の間、前のアドレスを保持 - リセットの状態が完了
EMRS	拡張モード レジスタの設定
MRS	モードレジスタの設定
REF	2 つまたはそれ以上の自動リフレッシュ コマンドを発行
*Ilock	EMRS の後、Ilock クロック サイクルを待つ
*WRITE	4 つのバンクすべてに書き込みコマンドを発行

リセット状態が完了した後、EMRS、MRS および REF コマンドは、任意の順序で実行できます。Ilock は EMRS コマンドと関連しているため、このリファレンス デザインでは、表に示す順序でコマンドを発行し、必要な初期化の時間を最低限に抑えます。

このリファレンス デザインは、アスタリスク (*) が付いたコマンド以外すべてを発行します。最後の書き込みコマンドを 4 つ （各バンクに 1 つ）発行する前に Ilock クロック サイクルが生成される必要があります。スタートアップ シーケンスには、スタートアップ時にユーザー インターフェイスで必要なコマンドを発行、また HDL コードを変更する必要があります。コマンドが発行されると初期化シーケンスが完了し、FCRAM デバイスの通常動作ができるようになります。コントローラが初期化プロセス実行中にコマンドが発行されると、FCRAM 仕様に違反します。詳細については、「初期化シーケンス」を参照してください。

リフレッシュ

リフレッシュを実行するには 2 つの方法があります。

ユーザーによるリフレッシュの開始

このモードを使用するには、ユーザーインターフェイスで u_ref_enable = 0 に設定します。このモードで、ユーザーが必要なリフレッシュ コマンドを FCRAM コントローラへ発行します。コマンド発行の設定は、u_cmd = 101 (セルフ リフレッシュの場合) または u_cmd = 111 （自動リフレッシュの場合) です。コントローラがこのコマンドを認識 （u_ack をアサート）すると、要求されたコマンドが FCRAM へ発行され、リフレッシュが開始します。セルフ リフレッシュ モードの場合、セルフ リフレッシュ コマンドが与えられている限り、コントローラはリフレッシュ ステートから遷移しません。

このモードを使用した場合、FCRAM 仕様を満たすために充分な回数のリフレッシュ コマンドが実行されるように、ユーザー インターフェイスで管理する必要があります。

コントローラによるリフレッシュの開始

このモードを使用するにはインターフェイスで u_ref_enable = 1 に設定します。リフレッシュ間隔タイマーが終了すると、コントローラが自動的に FCRAM へ自動リフレッシュ コマンドを発行します。

これらのリフレッシュ コマンドは、限られた範囲内でのみ認識されます。つまり、別のコマンドが実行中または複数のバースト アクセスの途中にリフレッシュ コマンドが挿入されることはありません。リフレッシュ間隔タイマーが終了し現在の動作が完了した場合にリフレッシュが最優先されます。

ユーザーはコントローラへ次のようにパラメータを渡します。

```
u_ref_parms = {ref_burst_cnt(4), ref_interval_cnt(16)}
```

ref_burst_cnt は、1 つの列 (バースト リフレッシュ) で行うリフレッシュの回数を示します。ref_interval_cnt は、リフレッシュを行う間隔を示します。

バースト リフレッシュ

FCRAM 仕様によると、自動リフレッシュ コマンドを発行する前に $T_{REFI-MIN}$(自動リフレッシュ間隔) サイクル間待機する必要があり、この最大間隔は $T_{REFI-MAX}$ で定義されています。ただし 1 行に複数 (最大 8) のリフレッシュを発行した場合は、これらの仕様規定が緩和されることもあります。これがバースト リフレッシュの概念です。

たとえば、タイム 0 で自動リフレッシュ コマンドが 1 つ発行された場合、次の自動リフレッシュが実行されるまで $T_{REFI-MIN}$ サイクル間待機します。また、REFI-MAX サイクル間を超えることはありません。シングル自動リフレッシュのかわりにバースト自動リフレッシュが実行された場合、これらのコマンドはすぐに実行されます (ただし $T_{REFI-MIN}$ サイクルを待つ必要はありません)。次の自動リフレッシュ コマンドを実行するには *n* x ($T_{REFI-MIN}$) サイクルを待つ必要があります。ただし *n* x ($T_{REFI-MAX}$) を超えてはいけません。

つまり、バースト リフレッシュを多く行うほど、次に自動リフレッシュを発行するまでの時間が長くなります。自動リフレッシュが実行されている間、長時間メモリへ接続した状態になります。

リフレッシュ間隔の計算

リフレッシュ間隔の値を正確に計算しない場合、FCRAM 仕様に違反する可能性があります。最初に実行されるリフレッシュの回数を、行 (ref_burst_cnt) にて決定します。次に FCRAM データ シートを参照してクロック周波数および T_{REFI} 最小/最大値を検索して次のように値を算出します。

$$ref\_interval\_cnt_{MIN} = (t_{REFI-MIN}) x (ref\_burst\_cnt) / T_{CK}$$
$$ref\_interval\_cnt_{MAX} = (t_{REFI-MAX}) x (ref\_burst\_cnt) / T_{CK} - (u\_num\_xfers + I_{RC})$$

自動リフレッシュ カウンタのタイマーが終了したとき、読み出しまたは書き込み処理がまだ実行中である可能性があります。そのため、リフレッシュを実行する前に、メモリ アクセス後の待ち時間である IDLE 時間および転送できる最大データ数を考慮して計算する必要があります。これらの値は上記の計算式 $ref\_interval\_cnt_{MAX}$ に含まれています。

ref_interval_cnt および ref_burst_cnt には、将来に向けた機能拡大のため、余分なビットが含まれています。

メモリ アクセス

このセクションでは、FCRAM コントローラへの読み出しまたは書き込み要求が正常に実行されるために必要なコマンドおよび信号について説明します。

一般的なメモリ アクセスのフローを次に示します。

- ユーザーが、要求するメモリおよびメモリ アクセスのバンク ロケーションを指定。
- ユーザーが、メモリ アクセスのための転送数を指定。
- ユーザーが、コントローラへ読み出しおよび書き込みコマンドを発行。
- FCRAM コントローラがコマンド (u_ack = 1) を認識します。認識された後にユーザーがメモリ アドレス、バンク ロケーション、転送数およびメモリ アクセス コマンドを解除できます。この時すでにユーザーは次のコマンドを発行できるため、確実にコントローラのパイプラインをフル状態に維持できます。
- ユーザーは、書き込み中にデータの供給または読み出し中にデータの受信を行う必要がある。

バースト転送

このセクションでは、バースト メモリ アクセスのコントローラ インプリメンテーションについて説明します。

1 個の FCRAM メモリ アクセスの場合、データ値と FCRAM データ バスの幅が同じであるため、データ値のバースト長 (BL) により転送数に制限ができます。連続したメモリ ロケーションへの複数 FCRAM メモリ アクセスを必要とする場合、FCRAM コントローラは、自動的にこれらのメモリを繋ぎ合わせてバースト メモリ アクセスを行うことができます。

これは u_num_xfers を使用してユーザー インターフェイスで実行できます。この値は、データ バス u_data_i または u_data_o を介してコントローラから、またはコントローラへデータが転送されるクロック サイクルの数になります。

たとえば、1 クロック サイクル間 u_data_i の書き込みリクエスト データを u_num_xfers = 1 に設定します。または、u_data_req を 1 クロック サイクル間 High にします。同じようにして、1 クロック サイクル間、u_data_o の読み出しリクエスト データを u_num_xfers = 1 に設定します。または、u_data_val を 1 クロック サイクル間 High にします。この場合、システム バスでの 1 転送が、FCRAM(DDR) バスでは 2 転送になることを確認してください。つまりフル バースト転送の場合、BL = 2 ではu_num_xfers = 1 に設定します。また、BL = 4 では u_num_xfers = 2 に設定します。

同様にして、FCRAM バスで 16 の連続するデータ転送を発行するには、ユーザー インターフェイスで u_num_xfers = 8 に設定し、1 つのコマンドでインプリメントできます。u_num_xfers は 4 ビットであるため、ユーザー インターフェイスには最大 16 連続メモリ アクセス、つまり FCRAM バスで 32 データ転送までを実行できるオプションがあります。

アドレス変換

メモリ アクセスの開始ポイントは、u_addr で示されます。このバスは FCRAM のバンク、行、列アドレスを次のようにマップします。

```
u_addr[26:0]   = {ba, row, col}

u_addr[26:25] = bank[1:0]
u_addr[24:10] = row[14:0]
u_addr[9:0]    = col[9:0]
```

メモ：

1. 10 ビット列の値により、将来の FCRAM コントローラ機能が拡張できます。一方、選択した FCRAM デバイスがサポートしない列アドレスへのアクセスは絶対にしないでください。たとえば、x16 デバイスは 7 ビットの列アドレスを使用するため、u_addr[9:7] を 0 に設定します。その他のメモリ コンフィギュレーションについては、FCRAM データ シートを参照してください。

コマンド (u_ack = 1) が認識されるとすぐにコントローラは u_addr に与えられた値をラッチします。これらの値はデコードされ、最初のコマンド (WRA/RDA) でコントローラは上位（行）アドレスおよびバンク アドレスを FCRAM へ出力します。次のコマンドで、コントローラは下位（列）アドレスを FCRAM へ出力します。

与えられた要求の範囲内 （u_num_xfers 転送数がまだ完了していない）で各読み出しまたは書き込み動作 （WRA/RDA および LAL の組み合わせ）が正常に行われている場合、コントローラは自動的にバンク アドレスを 1 つインクリメントします。前のセクション 「モード レジスタ セット (MRS)」でバースト長について述べたとおり、行および列を選択すると読み出しおよび書き込みコマンドがバースト長分だけバースト転送されます。したがって、バンク アドレスがオーバー フローする場合、（例 ：3 から 0 への移行）現在のアドレスは、プログラムされたバースト長 (BL) の分だけインクリメントされます。複数のバースト アクセスの場合、メモリへのアクセスはバンク、行、そして列へと順番にアクセスします。

アクセスの規則

FCRAM 仕様では、一度バンク アクセスを行い、再び同じバンクへアクセスするには I_{RC} サイクル（読み出し/書き込みサイクル時間）待機する必要があります。このため、複数の読み出し/書き込みまたはこの 2 つの組み合わせコマンドを IRC サイクル内で同じバンクに発行した場合は、バンク衝突エラーが生じます。

さらに、読み出しコマンドに続いて別のバンクに書き込みコマンドを発行する場合、I_{RWD} クロック サイクル (書き込み-読み出しの転換時間) の後に書き込みコマンドを実行する必要があります。この仕様を考慮しないと、読み出し-書き込みの転換違反になります。

書き込み-読み出し転換 (I_{WRD}) 時間は 1 クロック サイクルであるため、書き込み-読み出しの転換違反にはなりません。

ユーザー インターフェイスで違反を監視するかわりに、FCRAM コントローラがバンク衝突や読み出し-書き込みの転換違反を監視します。要求されたコマンドが FCRAM 仕様に違反する場合は、パラメータが一致するまで FCRAM コントローラが IDLE ステートに遷移するなど、FCRAM コントローラが問題を処理します。

読み出し要求

読み出し要求を実行するには、ユーザー インターフェイスで次のように設定する必要があります。

```
u_addr[26:0] = {ba, row, col}
u_num_xfers  = 転送される 32 ビット データ値の数
u_cmd        = 110
```

このコマンドが FCRAM コントローラに認識 (u_ack = 1) されるまで、これらの値を維持する必要があります。認識されるとユーザー インターフェイスではこれらの値を解除し、次のコマンドを発行します。u_data_val が High になり、u_data_o に有効な読み出しデータが含まれていることを示します。

書き込み要求

書き込み要求を実行するには、ユーザー インターフェイスで次のように設定する必要があります。

```
u_addr[26:0] = {ba, row, col}
u_num_xfers  = 転送される 32 ビット データ値の数
u_data_i     = 最初の 32 ビット データ値
u_cmd        = 100
```

このコマンドが FCRAM コントローラに認識 (u_ack = 1) されるまで、これらの値を維持する必要があります。認識されると、ユーザー インターフェイスでは u_addr, u_num_xfers および u_cmd を解除できます。コントローラが u_data_req で実行されるデータを要求するまで u_data_i の最初のデータ片は維持される必要があります。u_data_req が High にアサートされた後の最初の立ち上がりクロック エッジで、現在の 32 ビット データ値が有効になり、次のクロックで次の 32 ビット データ値が有効になります。

データ マスク

データ マスク (DM) を使用すると、書き込みコマンド中にデータ片をマスク オフできます。これは使用するデバイスによって異なりますが、データ マスクを指定する方法は 2 つあります （ボンディング オプション）。

1. 従来型の外付け DM ピンを使用する方法
2. LAL コマンド中のアドレス ピン A14-A11 を介してエンコードされたマスクを渡す方法

エンコードされたマスクを使用する方法は、周波数の影響を受けません。また後者の FCRAM インプリメンテーションは、エンベデッド データ マスクです。

このコントローラにインプリメントされたデータ マスク機能は、BL = 4 の場合、また処理回数が奇数の場合のみ使用できます。データ マスクの機能については次のとおりです。

ユーザー インターフェイスでコントローラに 32 ビット データ転送を行う回数を指定します。32 ビット データ転送を奇数回 (例 : `u_num_xfers = 3`) に指定すると 、1.5 のフル バースト転送となります。FCRAM は、残りの 0.5 を処理するために、書き込みコマンドの最後のクロック サイクルをマスク アウトします。

この処理は、データ マスク機能により実行されます。コントローラは自動的に `u_num_xfers` から適当なデータ マスクを抽出し、LAL サイクル中にアドレス ピンを介して FCRAM へこの値を渡します。このようなデザインでは、ユーザー インターフェイスからデータ マスクを手動で指定できません。

下位アドレス アクセス中、すべてのメモリ書き込みにデータ マスクが与えられます。つまり、最後のメモリ転送以外のすべての偶数転送および奇数転送には 「write all words」と設定されたマスク値があります。また、BL = 4 の場合で、奇数メモリ転送の最後の転送には、「write first two words」と設定されたマスク値があります。

FCRAM コントローラの詳細

デジタル クロック マネージャ (DCM) インプリメンテーション

このセクションでは、*clk_dcm* ブロックについて説明します。このアプリケーション ノートのリファレンス デザインのクロック設計には、Virtex-II DCM、グローバル クロック ネットワーク、および IOB DDR レジスタを使用します。図 6 に、クロック構造を示します。最初の DCM である `DCM_CLK` には 2 つのクロック出力があります。1 つ目の出力 (`clk`) は、ユーザー入力クロック (`u_clk`) へ直接接続します。2 つ目の出力 (`clk90`) は、`u_clk` が 90 度位相シフトしています。この `clk` 出力は、FCRAM クロック (`ddr_clk` および `ddr_clkb`) を生成するための IOB DDR フリップフロップも駆動します。

2 つ目の DCM である `DCM_RCLK` には出力が 1 つあります。このクロック (`rclk`) は、ユーザー入力クロック (`u_clk`) を位相シフトしたクロックです。メモリ読み出し中に DQS ドメインからデータを再び取得するために、このクロックを使用します。`rclk` クロック ドメインにデータを取得すると、このリファレンス デザインではメイン システム クロック ドメイン (`clk`) へ読み出しデータを転送します。位相シフト値はシステムにより異なるため、それぞれプログラムする必要があります。クロック設計手法

についての詳細は、「リード リキャプチャ タイミング解析」の「読み出しデータ パス」を参照してください。

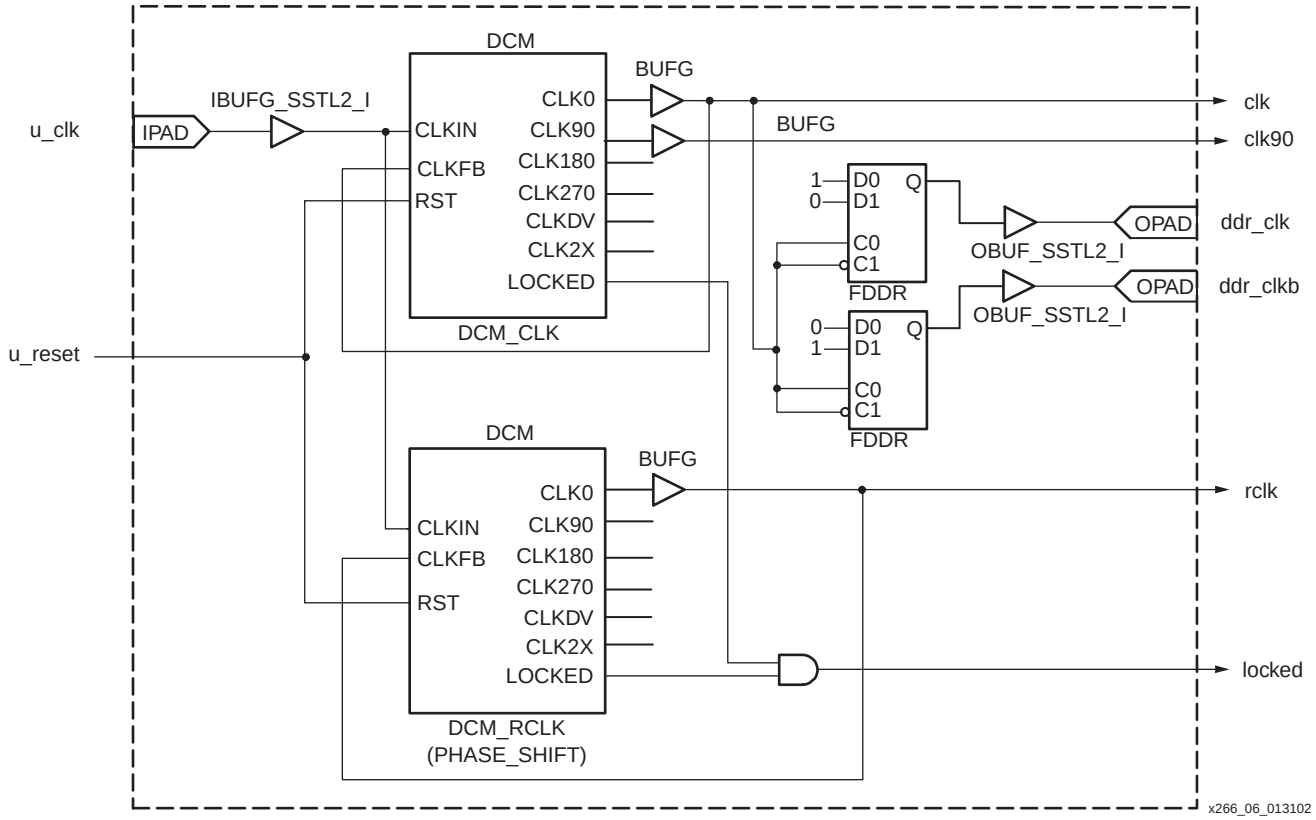

図 6： clk_dcm ブロックの DCM インプリメンテーション

データ パス

Virtex-II デバイスの IOB は、DDR 機能を直接インプリメンテーションできるように機能が充実しています。このアプリケーション ノートでは、この機能を活用して DDR を IOB にインプリメントします。さらに、DDR FCRAM インターフェイスへのすべての入力および出力を IOB に格納して、clock-to-out 遅延を最低限に抑えることもできます。図 7 に、Virtex-II デバイスで、1 つの IOB に標準 DDR をインプリメントした例を示します。

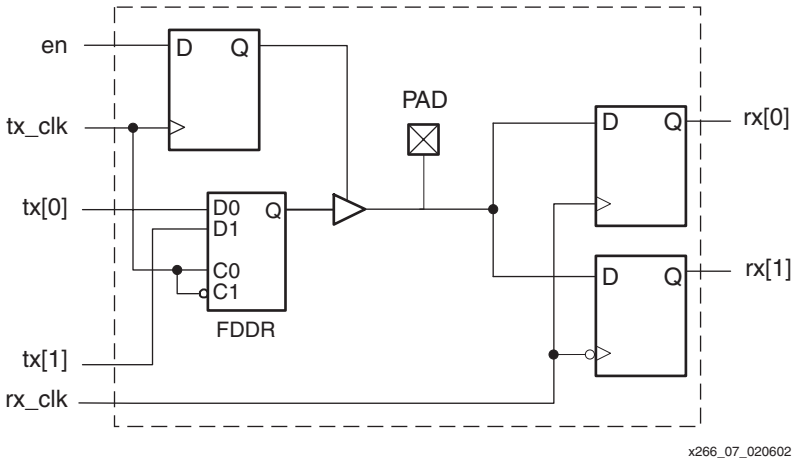

図 7 : DDR IOB インプリメンテーションの例

図 8 に、データ パスおよびデータ ストローブ生成ロジックの回路図を示します。データ パスを表示するため、この図では HDL 階層バウンダリを省略しています。詳細は、*data_path* および *data_strobe* HDL ファイルを参照してください。

複数ステージのパイプライン遅延を示すため、すべての入力信号に SRL ラベルを使用しています。これらの遅延により、データ (`ddr_dq`) およびデータストローブ (`ddr_dqs`) 信号が FCRAM 制御信号とアラインします。ユーザー データ バス (`u_data_i` および `u_data_o`) は SDR、FCRAM データ バスは DDR であるため、ユーザー データ バスは、FCRAM データ バス幅の 2 倍になります。また、図 8 では示していませんが、Virtex-II の IOB には `ddr_dqs`トライ ステートおよび出力フリップ フロップ、また、`ddr_dq` トライステートおよび出力／入力フリップ フロップがインプリメントされています。

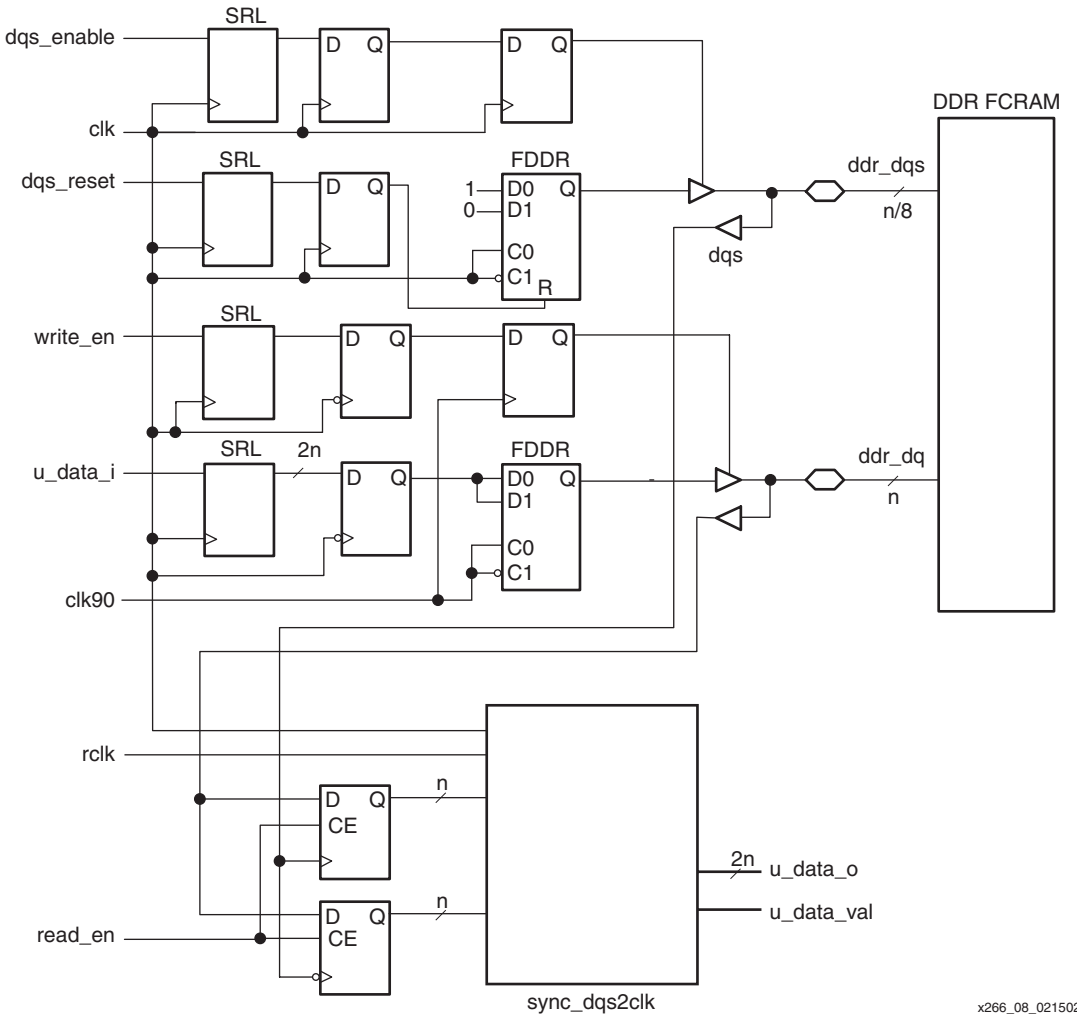

図 8： データ パス

書き込みデータ パス

メモリ書き込み中、コントローラはデータにセンター アラインしたストローブ信号を送る必要があります。FCRAM 仕様により、FCRAM ピンで CLK および DQS が関係付けられます。また、仕様ではこの 2 つの信号間の位相に多少のスキューが生じることが確認されていますが、通常ほとんど一致します。このスキューを最低限に抑えるために、clk および $\overline{\texttt{clk}}$ を採用した DDR フリップフロップから CLK 信号および DQS 信号が送信されます。

controller ブロックで生成される dqs_enable 信号は、dqs_reset 信号が DQS フリップフロップをリセット状態に維持している間、トライ ステート出力を制御します。この 2 つの信号により、DQS タイミング パラメータ （DQS プリアンブル セットアップ時間など）が一致します。dqs_reset 信号が解除されると、DDR フリップフロップ入力は H /L のトグル動作をする DQS に接続します。

DQS は clk から生成されるため、DQ 信号は、clk90 の DDR フリップフロップから出力されます。これにより自動的にデータ ストローブは、データにセンターアラインします。write_en 信号は *controller* ブロックで生成され、データ パスのトライ ステート出力を制御します。u_data_i は、ユーザ データ入力です。これらの信号は clk ドメインと同期しているため、最初に $\overline{\texttt{clk}}$ に転送され、次に clk90 ドメインに転送されます。これにより、クロック ドメイン転送のタイミング要件が緩和されます。

読み出しデータ パス

メモリの読み出し中、FCRAM は FPGA に DQ 信号および DQS 信号を与えます。このリファレンス デザインは、DQS 信号をクロックとして使用して読み出しデータ DQ を取得します。「ローカル クロック分配のためのピン配置制約」で説明されているとおり、DQS は専用のローカル クロック リソースに分配されます。DQS はストロービングするため、DQS ドメインで取得したデータをすぐに再取得する必要があります。

データを再度取得するには、DQS ドメインとシステム クロック ドメインの関係を確認することが必要です。メモリ読み出し中のデータ到達時間は、ボード レイアウトなどのシステム依存要因により異なります。これらの変数に対応するため、このリファレンス デザインでは DCM を使用して、システム クロックを位相シフトしたクロック (rclk) を生成します。これにより再取得したクロックを DQS クロック ドメインと一致できます。詳細は、「リード リキャプチャ タイミング解析」を参照してください。

DQS ドメインのデータは、`rclk` で直接デュアル ポート LUT RAM へ書き込まれます。システム クロックはデュアル ポート LUT RAM からデータを読み出します。再取得したクロックは内部システム クロックに非同期であるため、クロック ドメイン間で転送されたデータは 2 度取得されます。このため、セットアップ、ホールド、またはメタスタビリティの問題はありません。この再取得したロジックおよび同期化ロジックは、*sync_dqs2clk* モジュールで処理されます。図 8 で示すとおり、このモジュールには、読み出しデータ、リキャプチャ クロック、システム クロックおよびイネーブル信号 （これは図に表示されていない）の入力があります。また、`u_data_val` および `u_data_o` 信号を出力してユーザー インターフェイスとシステム クロック ドメインを同期化します。

コントローラ ステート マシン

図 9 に主要コントローラ ステート マシンの略図を示します。このステート マシンは、ワンホット ステート マシンでコード化されており、ステートの複製を含むことにより、各レベルでのデコード作業が省略されます。図 9 は略図であるため、複製されたステートの大部分は表示していません。詳細は、コントローラ HDL ファイルのステート マシンについての記述を参照してください。

電源投入時、コントローラはIDLE ステートです。リセットが解除され、DCM がロックすると、コントローラは自動的に初期化プロセスを開始します。このシーケンスが完了すると、コントローラは読み出し、書き込みおよびリフレッシュ コマンドが実行できるメインの IDLE ステートへと遷移します。

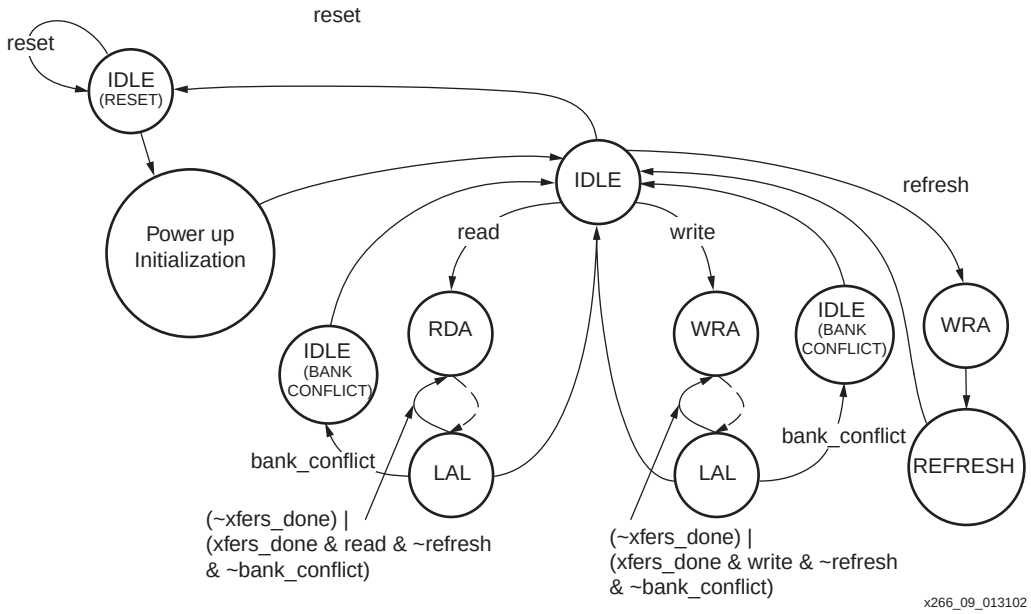

図 9： ステート マシンの図

読み出しコマンド

u_cmd が読み出しコマンドに設定されると、コントローラは RDA ステートへ遷移し、次に LAL ステートへと遷移します。コントローラは、指定した送信数が完了するまで、これらのステート遷移を繰り返します。u_num_xfers が完了すると、xfers_done = 1 がアサートされ、ステート遷移が終了します。

xfers_done がアサートされると、コントローラは別のコマンドを発行できるようになります。別の読み出しコマンドを発行する場合は、リフレッシュ カウンタが有効（refresh = 0 と表示）であり、指定したアドレスのバンク競合（bank_conflict = 1 と表示）がない状態のとき、読み出しコマンドがすぐに実行されます。これは、コントローラが別の読み出しコマンドを開始する RDA ステートへ戻る動作で確認できます。

このように発行した読み出しコマンドがバンク競合を起こした場合、コントローラは IDLE (BANK_CONFLICT) ステートへ遷移します。同様に、このコマンドが書き込みコマンドの場合でも、コントローラが FCRAM の読み出し-書き込み転換時間を違反していないことを確認した後に、IDLE (BANK_CONFLICT)ステートへと遷移します。このようにコントローラは要求されたバンクへ再度アクセスできるようになるまで、IDLE ステートに遷移できるようになり、アクセス違反が回避されます。

また、xfers_done がアサートされ、リフレッシュ カウンタが無効になった場合 (refresh = 1 と表示) または発行されたコマンドが読み出し/書き込みコマンドでない場合もコントローラは IDLE ステートへ遷移します。refresh がアサートされるとコントローラは自動的に WRA ステートへと遷移し、次に REFRESH ステートへと遷移して自動リフレッシュが実行されます。refresh がアサートされない場合は、コントローラは次の有効なコマンドが発行されるまで IDLE ステートから遷移しません。

書き込みコマンド

u_cmd が書き込みコマンドに設定されると、コントローラは、WRA ステートへ遷移し、次に LAL ステートへ遷移します。u_num_xfers が完了して xfers_done = 1 がアサートされるまで、コントローラはこれらのステート遷移を繰り返します。

xfers_doneがアサートされると、コントローラは別のコマンドを発行できるようになります。別の書き込みコマンドを発行する場合、リフレッシュ カウンタが有効（refresh = 0）であり、指定したアドレスでバンク競合がない状態のとき、書き込みコマンドがすぐに実行されます。これは、コントローラが別の書き込みコマンドを開始する WRA ステートへ戻る動作で確認できます。

このように発行された書き込みコマンドによりバンク競合が生じた場合、コントローラはIDLE (BANK_CONFLICT) ステートへ遷移します。このようにコントローラは要求されたバンクへ再度アクセスできるようになるまで、IDLE ステートに遷移できるようになり、アクセス違反が回避されます。

また、xfers_done がアサートされ、発行されたコマンドが書き込みコマンドでない場合、またはリフレッシュ カウンタが無効の場合でも、コントローラは IDLE ステートへ遷移します refresh がアサートされ、コントローラが自動的に WRA ステートへ遷移し、次に REFRESH ステートへと遷移して自動リフレッシュが実行されます。refresh がアサートされない場合は、コントローラは次の有効なコマンドが発行されるまで IDLE ステートから遷移しません。

タイミング図

初期化シーケンス

図 10 は、初期化シーケンスを示します。最初にシステムはリセット状態を維持し、初期化データ (u_init_parms および u_ref_parms) をユーザー インターフェイスへ送信する必要があります。このリファレンス デザインでは、ユーザー リセットおよび DCM LOKED 信号の組み合わせがシステム リセットになります。リセットが解除 (u_reset_n = 1) されるとシステムは DCM がロックするまで待機します。DCM がロックすると、コントローラ ステート マシンはリセット状態から解除され、自動的に 「電源投入時の初期化およびリセット条件」を開始します。

FCRAM 仕様では、EMRS コマンド中、FCRAM DLL はイネーブルになります。このため、ユーザーはコマンドを発行する前に、必ず FCRAM DLL がロックされたことを（EMRS コマンドが発行された後に ILOCK サイクルが始まる）確認する必要があります。FCRAM DLL がロックされた後に、4 つの

書き込みコマンド （各バンクに 1 つ）を発行する必要があります。図 10 で示すとおり、電源投入初期化コマンドから EMRS コマンドまでを発行するには INIT TIME クロック サイクルが必要です。このリファレンス デザインでは、INIT TIME は CL により異なり、CL = 2 の場合は INIT TIME = 29 クロック サイクルとなり、CL = 3 の場合は、INIT TIME = 32 クロック サイクルとなります。したがって、一度 DCM がロックされると INIT TIME および ILOCK を加算したクロック サイクルが終わるまで、ユーザー インターフェイスは 4 つの書き込みコマンドを発行できません。これらのコマンドが発行されると初期化シーケンスは完了し、システムの通常動作が可能になります。

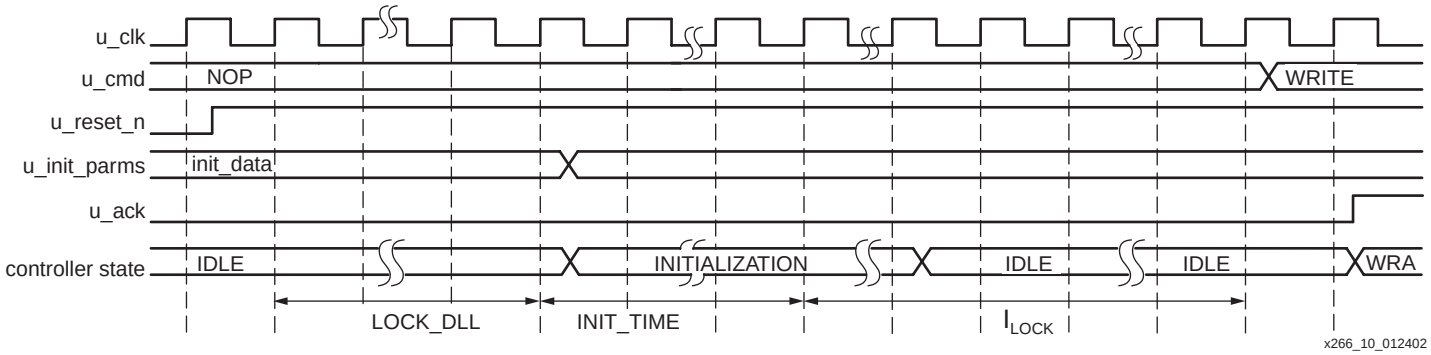

図 10： 初期化タイミング図

書き込みサイクル

図 11 に、BL = 4 および CL = 2 の連続する書き込みコマンドのタイミング図を示します。この例では、読み出しおよび書き込みの両方のメモリ転送において u_num_xfers が 2 に設定されています。これは、u_data_req で示すように、ユーザー インターフェースから 4 クロック サイクル間、u_data_i にデータを与える必要があります。

サイクル T_2 では書き込みコマンドが u_cmd に発行されます。このときコントローラは IDLE ステートなので、コマンドをすぐに受け取ることができます。サイクル T_3 では WRA ステートへ遷移し、u_ack 信号がアサートされると要求が受け入れられたことを示します。

コントローラは、書き込み要求が出されたとき、この要求を行うのに必要なデータがあることを予測するため、書き込みの最初の 2 つのデータが u_data_i に含まれています。サイクル T_4 で、コントローラにより u_data_req が High になり、次の立ち上がりクロック エッジ (T_5) で、コントローラがこの 2 つのデータを取得します。このようにして、次のクロック サイクルで、別のデータが与えられます。u_num_xfers が 2 に設定されていると、2 つの 32 ビット値がユーザー インターフェイスから与えられて転送されます。この例ではバースト長が 4 にプログラムされており、FCRAM の完全なバースト転送が行われていることを確認できます。

最初のコマンドが認識されるとすぐに 2 番目のメモリ動作が開始されます。T4 で u_ack がアサートされると、すぐにユーザー インターフェイスから 2 番目の書き込みコマンド、アドレス、バンクおよび転送数が発行されます。この場合、最初の書き込みリクエストの u_num_xfers は、2 であるため、この 2 番目のコマンドが認識される最も早いサイクルは、2 クロック サイクル後の T_5 になります。

書き込みは連続して行われ、バンクの競合は生じないため、FCRAM のバンド幅はすべて使用されます。

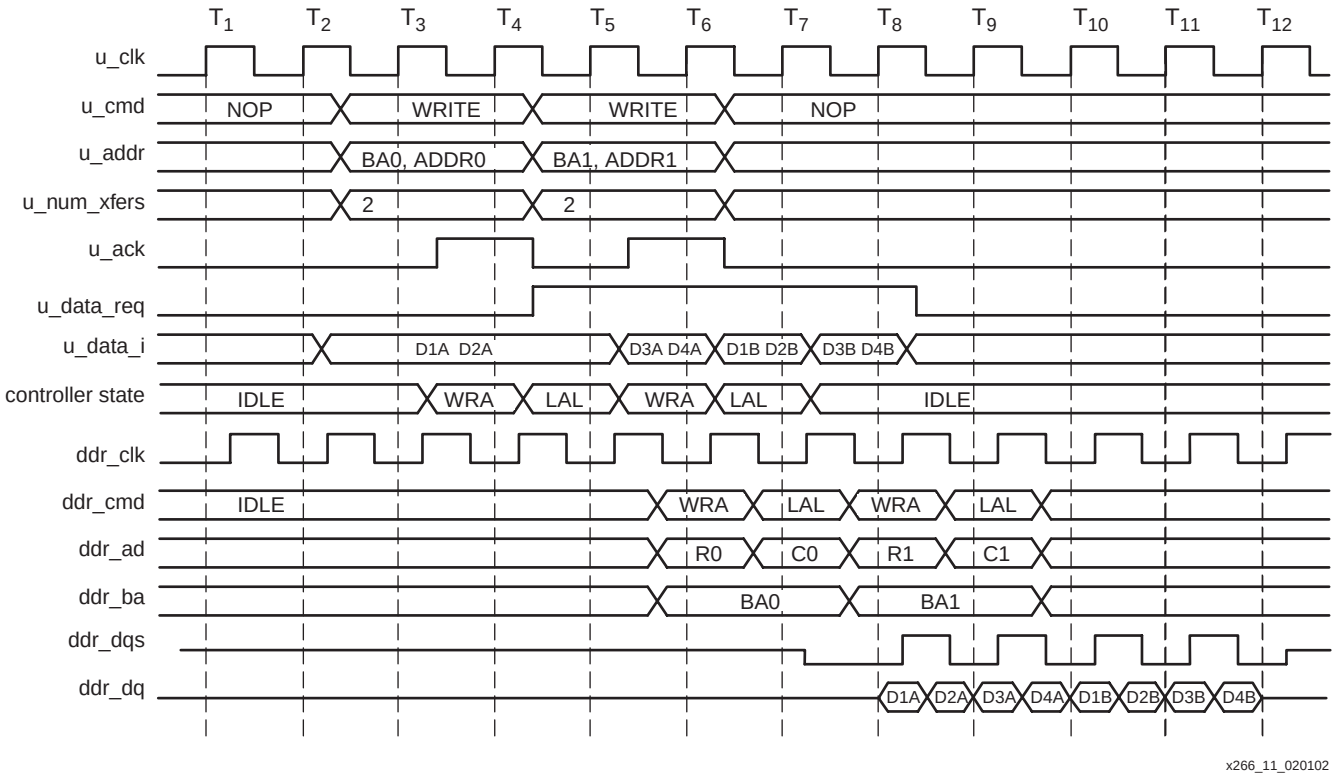

図 11： 書き込みタイミング図

図 12 に、BL = 4 および CL = 2 の場合の書き込みタイミング図 （2）を示します。この例では、u_num_xfers が 5 に設定されています。そのため、ユーザー インターフェイスから 5 クロック サイクル間、u_data_i にデータを与える必要があります。

バンク 2 (BA2) から開始し、開始アドレスは、R0 および C0 と示されます。バンク アドレスは、連続する書き込みコマンドが FCRAM に発行されると、自動的にインクリメントされることを確認できます。また、バースト長が 4 の場合、u_num_xfers を 5 に設定すると、2 つのフル バースト書き込みおよび 3 番目のバースト書き込みは半分行われます。このため WRA コマンド中、最初の 2 つの書き込みコマンドのデータ マスクは、「write all words」と設定されます。FCRAM コントローラは、最後の奇数転送を認識し、LAL コマンド中にデータマスクを 「write first two words」と設定します。この動作は T_{11} で確認できます。

バンク値が T_{10} でオーバーフローすると、列アドレスは自動的にバースト長分インクリメントされます。BL = 4 および開始列アドレスが C0 の場合は、最初のコマンドは C0、C1、C2、C3 と列に書き込みます。このため、バンク アドレスが T_{10} でオーバーフローした場合、ターゲット アドレスは C4 へ自動的にインクリメントされます。これは T_{11} で確認できます。

u_num_xfers = 5 に設定した場合、データ転送は T_{12} で完了していることを確認できます。ただし FCRAM 仕様により、データ マスク コマンドが発行されていても DQS 入力はバースト長が終了するまで連続する必要があります。このため、DQS は T_{13} サイクルまで連続します。

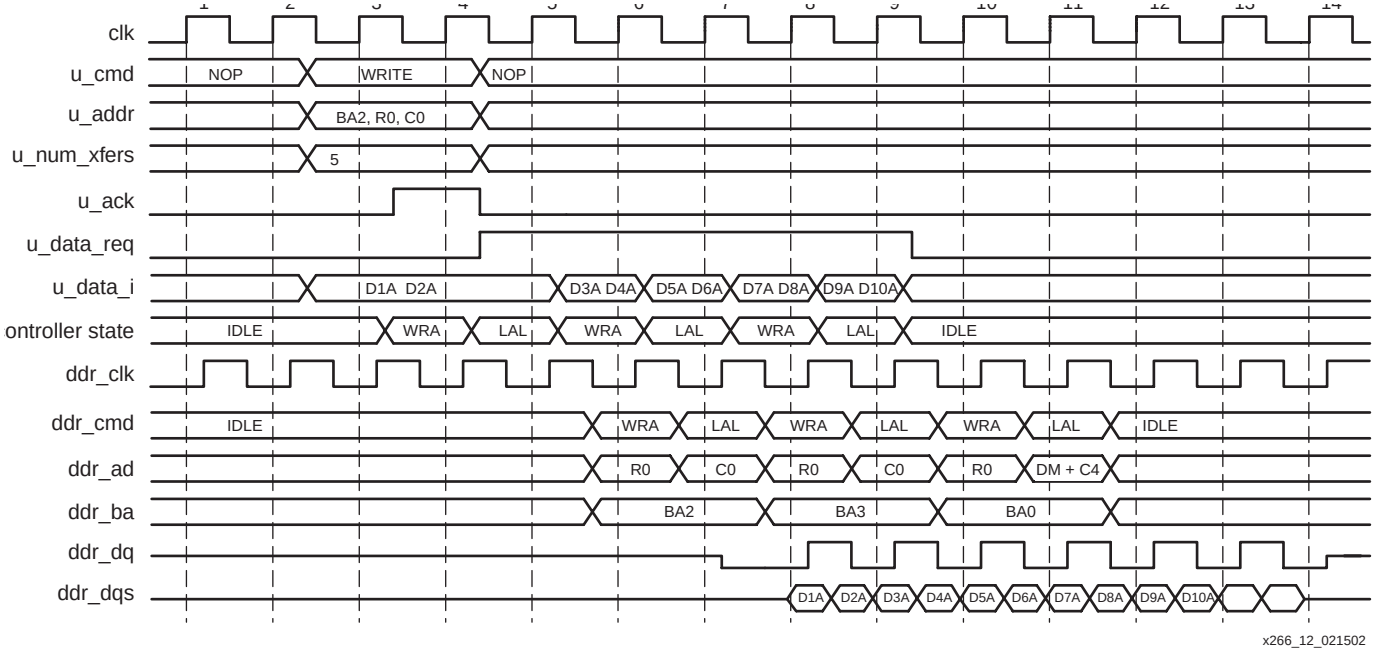

図 12： 書き込みタイミング図 （2）

読み出しサイクル

図 13 に、BL = 4 および CL = 2 の場合の連続する読み出しタイミング図を示します。

サイクル T_2 で u_cmd に読み出しコマンドが発行されます。コントローラは IDLE ステートなので、すぐにコマンドを受けることができます。サイクル T_3 で、コントローラは RDA ステートへと遷移し、u_ack 信号をアサートして要求が受け入れられたことを示します。

u_num_xfers = 2 に設定されているため、コントローラは 32 ビット データ値を u_data_o にメモリから取り出します。信号 u_data_val は、u_data_o にある現在の 32 ビット値が読み出しリクエストからの有効データであること示します。

最初の読み出しリクエストに対して u_ack が確認されると、2 番目の読み出しリクエストが発行されます。u_num_xfers は、両方の読み出しリクエストに対して 2 に設定されているため、u_data_val で示すようにコントローラは 2 クロック サイクルの間、ユーザー インターフェイスへデータを与えます。

読み出しリクエストは連続して行われ、バンクの競合は生じないため、FCRAM のバンド幅はすべて使用されます。

図 14 に、BL = 4 および CL = 2 の場合の連続する読み出しタイミング図 (2) を示します。

この例では、u_num_xfers が 3 に設定されています。バースト長は 4 であるため、最初のメモリ読み出しをフル バースト転送し、2 番目の読み出しを半分バースト転送します。FCRAM コントローラは自動的にこれらの読み出しコマンドを発行し、T_8 で 2 番目のコマンドのバンク アドレスをインクリメントします。FCRAM 仕様では、読み出しコマンドはデータ マスクを使用しません。このため、読み出しコマンドで 2 つ読み出しデータをフル バースト転送しますが、残りの奇数転送に対するマスクは FCRAM が行います。u_data_val 信号がサイクル T_{16} で Low に遷移すると、3 つの u_num_xfers が完了したことを示します。

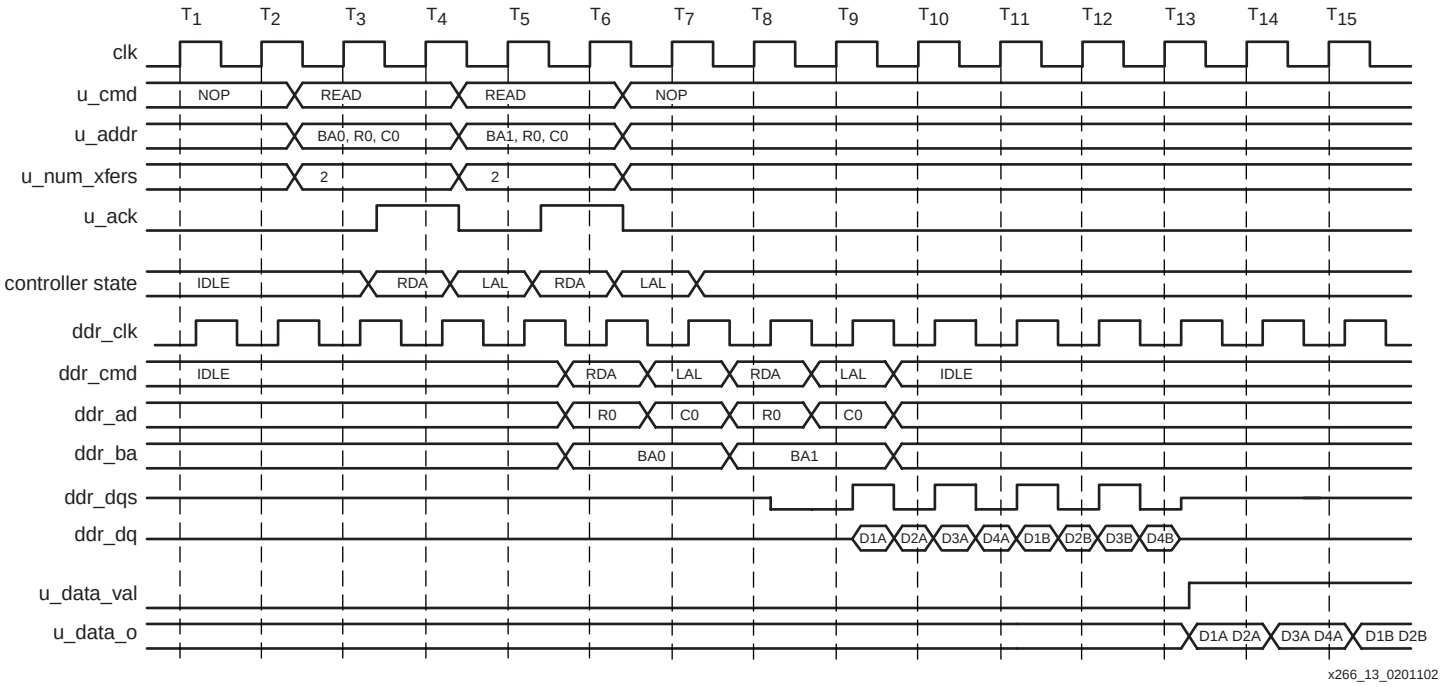

図 13： 読み出しタイミング図

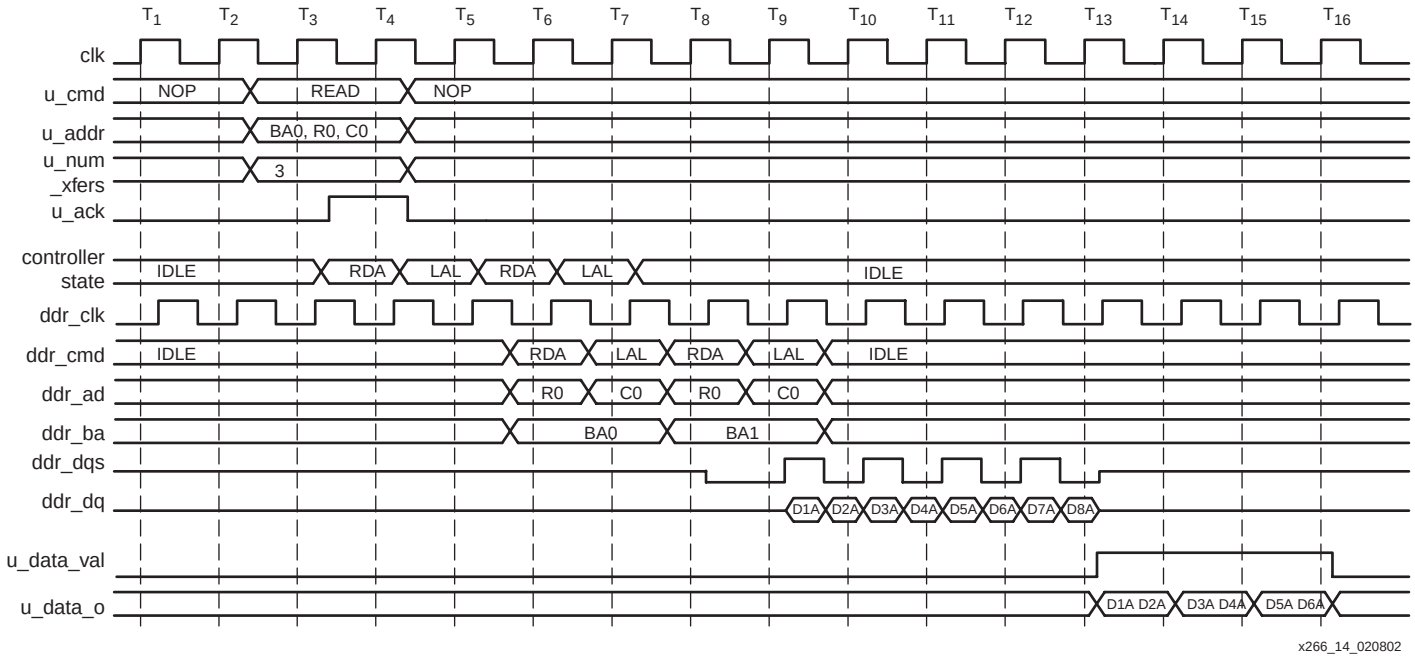

図 14： 読み出しタイミング図 （2）

I/O タイミング解析

完全に同期したシステムの最大データ率は、送信デバイスの clock-to-out、信号のフライト時間、および受信デバイスのセットアップ 時間により異なります。SDR システムを使用した場合、ビット レートは単にクロック周波数の逆数になります (100 MHz SDR = 100 Mb/s = 10 ns ビット レート)。また、DDR システムを使用した場合、ビット レートは次のように減少します (100 MHzDDR = 200 Mb/s = 5 ns ビット レート) 。

クロック周波数が増加するにつれ、システム パフォーマンスは制限されます。DRAM ベンダーは、このシステム パフォーマンスを向上させるため、双方向データ ストローブを使用したソース同期クロック設計手法を提供しています。

このセクションでは、リファレンス デザインのタイミング解析の例を示します。この解析では、-5 スピード グレードの Virtex-II デバイス、および -22 スピード グレードの FCRAM デバイスを使用しています。表 5 および 表 6 に、パラメータを示します。実際に解析を行う場合は、最新のデータシートからパラメータ値を取得してください。この解析サンプルの値は、ザイリンクスの『Virtex-II データシート』バージョン 1.6 (1) から取得しています。

表 5: -22 スピード グレードの FCRAM パラメータ

パラメータ	説明	最小	最大	単位
t_{CK}	クロック サイクル時間	6.5	10	ns
t_{QSQV}	DQS からのデータ出力有効時間	0.4 x t_{CK} - 0.4	-	ns
t_{QSQ}	DQS からのデータ出力スキュー	-0.52	0.52	ns
t_{DS}	DQS からのデータ入力セットアップ時間	0.6	-	ns
t_{DH}	DQS からのデータ入力ホールド時間	0.6	-	ns
t_{DSPREH}	DQS 入力プリアンブル ホールド時間	0.25 x t_{CK}	-	ns
t_{CKQS}	クロックからの DQS アクセス タイム	-0.85	0.85	ns
t_{DQSS}	DQS Low から High へのセットアップ時間	0.75 x t_{CK}	1.25 x t_{CK}	ns
t_{IS}	入力セット アップ タイム （DQS およびデータは除く）	1.0	-	ns
t_{IH}	入力ホールド時間 （DQS およびデータを除く）	1.0	-	ns

表 6: Virtex-II デバイスのパラメータ

パラメータ	説明
T_{IOPI}	入力パッド遅延 (SSTL2)
T_{IOPICK}	入力セット アップ、遅延なし (SSTL2)
T_{IOICKP}	入力ホールド時間、遅延なし (SSTL2)
$T_{ICKOFDCM}$	DCM を使用したクロック入力からデータ出力まで
T_{OSSTL2_I}	出力スイッチ調整 (SSTL2-I)
T_{OSSTL2_II}	出力スイッチ調整 (SSTL2-II)

読み出しタイミング解析

メモリ読み出し中、FCRAM デバイスは FPGA へ入力する DQ および DQS 信号を生成します。図 15 に、FCRAM 仕様に従った 3 つの信号タイミング関係を示します。

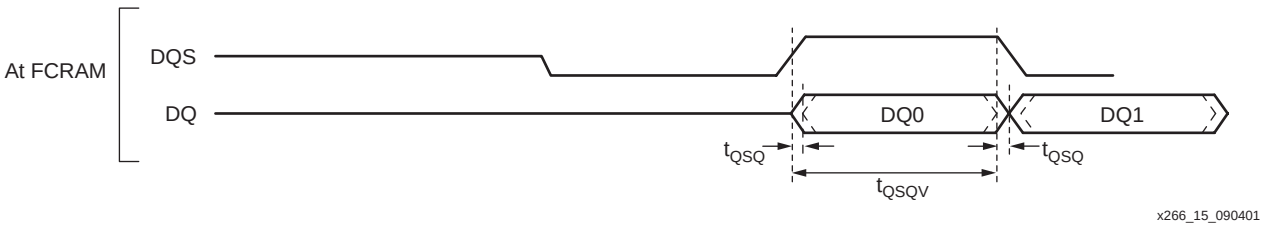

図 15 : DQS および DQ の読み出しモードの AC タイミング

FCRAM 仕様では (図 15)、「t_{QSQV} = DQS からのデータ出力有効時間」が保証されています。「t_{QSQ} = DQS からのデータ出力スキュー」が t_{QSQV} から削除されると、FCRAM からワースト ケース データ出力ウインドウが表示されます。このウインドウで、読み出しサイクルのセットアップ時間およびホールド時間が Virtex-II IOB で一致しているかを確認する必要があります。

図 15 に、DQS および DQ がエッジ アラインしていることを示します。このため、コントローラが DQS に遅延を与えることで、セットアップおよびホールド時間が Virtex-II IOB の 8 つすべての DQ 入力に対して一致します。

FPGA ではリソースが固定されているため、DQS に遅延を与えるために配線を検出することが困難です。しかしながら、FPGA 内で DQS ラインの配線遅延の決定し、ボード上で遅延を追加して DQ データ有効ウインドウ内で DQS を配置することができます。

図 16 に、FCRAM のピン上、FPGA のピン上、および IOB フリップフロップでの DQ および DQS のタイミング関係を示します。図 16 に、DQ ラインの配線遅延値を t_{DQ} で示し、DQS の配線遅延値を t_{DQS} で示します。

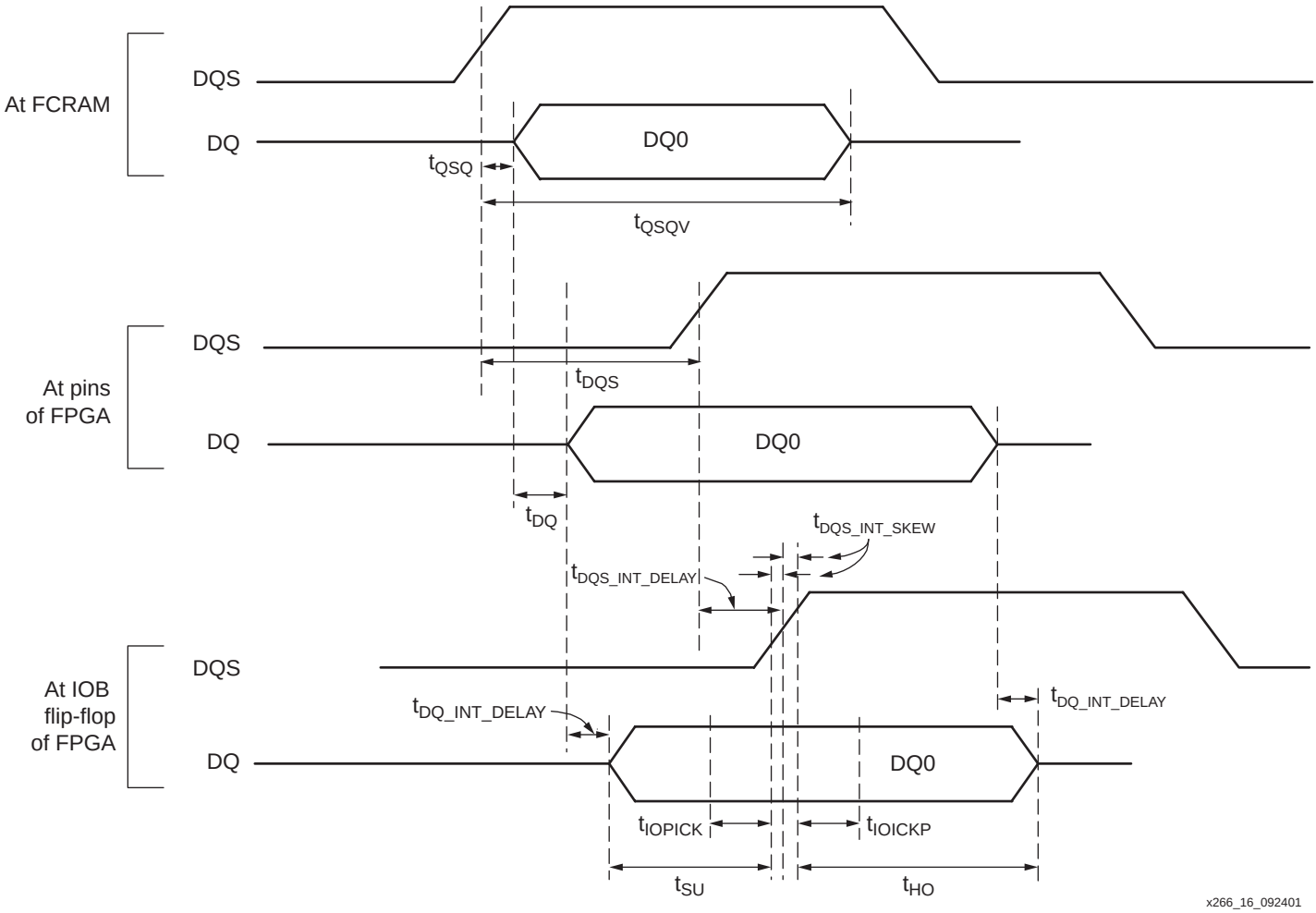

図 16： 読み出しサイクルのセットアップおよびホールド タイミング図

DQS が FPGA のピンに到達すると、IOB に入力し、8 つのデータ (DQ) IOB へ接続されます。そのため、DQS の内部 FPGA 遅延は、SSTL2 パッドを通過するときの遅延に、DQ ロードの配線遅延を加算した値になります。図 16 に、DQS の内部 FPGA 遅延を $t_{DQS\_INT\_DELAY}$ と示し、8 つの DQ ロード配線遅延を $t_{DQS\_INT\_SKEW}$ と示します。データ (DQ) 信号は、IOB にラッチされるため、配線遅延はありません。

このデザインは、ローカルクロック設計手法を使用して DQS を分配しているため、一定の配置制約が必要になります。この制約についての詳細は、「ローカル クロック分配のためのピン配置制約」を参照してください。制約規準を満たしている場合は、表 7 に示す配線遅延の例を参照してください。

表 7: サンプル内部データ パス (DQ) およびデータ ストローブ (DQS) 配線パラメータ

パラメータ	説明	最小	最大	単位
$t_{DQ\_INT\_DELAY}$	入力パッケージ遅延	0.000	0.000	ns
$t_{DQS\_ROUTE\_DELAY}$ (Note 1)	パッド入力から DQ ロードまでの配線遅延		0.320	ns
$t_{DQS\_INT\_DELAY}$	入力クロック遅延 (t_{IOPI_SSTL2}) + $t_{DQS\_ROUTE\_DELAY}$		1.50	ns
$t_{DQS\_INT\_SKEW}$ (Note 1)	配線バリエーション	-0.050	0.050	ns

メモ：

1. これらの値はインプリメンテーション結果に基づいているため、実際の値とは異なります。正しい配置を保持するには、手動で IOB をロックする必要があります。

データ (DQ) が最大遅延およびクロック (DQS) が最小遅延の場合、ワースト ケース セットアップが生じます。また、最大クロック (DQS) 遅延および最小データ (DQ) 遅延の場合、ワースト ケース ホールドが生じます。t_{SU} (セットアップ) および t_{IOPICK} の差異はセットアップのスラックであり、t_{HO} (ホールド) および t_{IOICKP} の差異はホールドのスラックであることを 図 16 で確認できます。

$$t_{SU} = (t_{DQS} + t_{DQSINTDELAY} + t_{DQSINTSKEWMIN}) - (t_{QSQ} + t_{DQ} + t_{DQINTDELAY}) > t_{IOPICK}$$

$$t_{HO} = (t_{QSQV} - t_{QSQ}) - (t_{DQS} - t_{QSQ} - t_{DQ}) + t_{DQINTDELAY} - (t_{DQSINTDELAY} + t_{DQSINTSKEWMAX}) > t_{IOICKP}$$

これらの計算式では、DQS と DQ の配線の差 (t_{DQS} - t_{DQ}) が算出されます。この値は、DQ のデータ有効ウインドウで DQS をセンター アラインするために必要な遅延となります。(t_{DQS} - t_{DQ}) の計算式の結果は次のとおりです。

$$t_{IOPICK} - (t_{DQSINTDELAY} + t_{DQSINTSKEWMIN}) + (t_{QSQ} + t_{DQINTDELAY}) < t_{DQS} - t_{DQ}$$
$$\ldots < -t_{IOICKP} + t_{QSQV} + t_{DQINTDELAY} - (t_{DQSINTDELAY} + t_{DQSINTSKEWMAX}) \quad \text{(EQ 1)}$$

『Virtex-II データ シート』バージョン 1.6 (参考資料 1) の値を使用した場合、次のような値が算出されます。

$$1.38 - (1.50 - 0.05) + (0.52 + 0.0) < t_{DQS} - t_{DQ} < -(-0.81) + ((0.4 \times 6.5) - 0.4) + 0.0 - (1.50 + 0.050)$$
$$0.450ns < t_{DQS} - t_{DQ} < 1.460ns$$

t_{DQS} - t_{DQ} の値は、1 ns より少し長い値となりました。この解析は、FPGA のタイミングが最大であることを仮定して行われます。ベストおよびワースト ケースでの正常動作を保証するには、DQS の内部配線遅延を考慮して最小タイミング解析を行う必要もあります。

この解析では最小タイミング値を使用するかわりに、表 8 で示すプロレイティング ファクタ (PF) を使用しています。

表 8: Virtex-II 最大タイミング プロレイティング

パラメータ	説明	値
PF	プロレイティング ファクタ (最大値の百分率)	0.25

最大電圧および最低温度で動作している場合、最良プロセス段階で最小タイミングが生じます。この影響はダイ (FPGA) 全体に及ぼすため、隣接する IOB でベスト ケース タイミングおよび IOB 内のワースト ケース タイミングが生じることはありません。つまり、この解析はすべての Virtex-II パラメータを平等に比例分配します。

EQ 1 は、最小タイミングを算出する計算式です。

$$(t_{IOPICK} \times PF) - (((t_{DQSINTDELAY} + t_{DQSINTSKEWMIN}) \times PF) + (t_{QSQ} + (t_{DQINTDELAY} \times PF)) < t_{DQS} - t_{DQ} \quad \text{(EQ 2)}$$
$$\ldots < (-t_{IOICKP} \times PF) + t_{QSQV} + (t_{DQINTDELAY} \times PF) - PF(t_{DQSINTDELAY} + t_{DQSINTSKEWMAX})$$

$$(1.38 \times 0.25) - (1.50 \pm 0.05)0.25 + (0.52 + (0.00 \times 0.25)) < t_{DQS} - t_{DQ}$$
$$\ldots < -((-0.81) \times 0.25) + ((0.4 \times 6.5) - 0.4) + (0.00 \times 0.25) - 0.25(1.50 + 0.050)$$
$$0.503ns < t_{DQS} - t_{DQ} < 2.015ns$$

(t_{DQS} - t_{DQ}) の範囲は、$0.503ns < t_{DQS} - t_{DQ} < 1.460ns$ となります。

書き込みタイミング解析

書き込みサイクルのクリティカル タイミングとは、DQS 信号に近い DQ 信号のセットアップおよびホールドです。メモリの書き込み中、FCRAM コントローラは DQ にセンター アラインした DQS を生成します。これにより、FPGA の出力に 4 分の 1 サイクルのセットアップおよびホールド時間を生成します。ただし 「読み出しタイミング解析」で説明したとおり、配線遅延により DQ に対する DQS をオフ セットする必要があります。図 17 に CL = 2 の書き込みコマンド タイミング図を示します。これらの配線遅延により、セットアップ時間が長くなり、ホールド時間が短くなっていることを確認できます。.

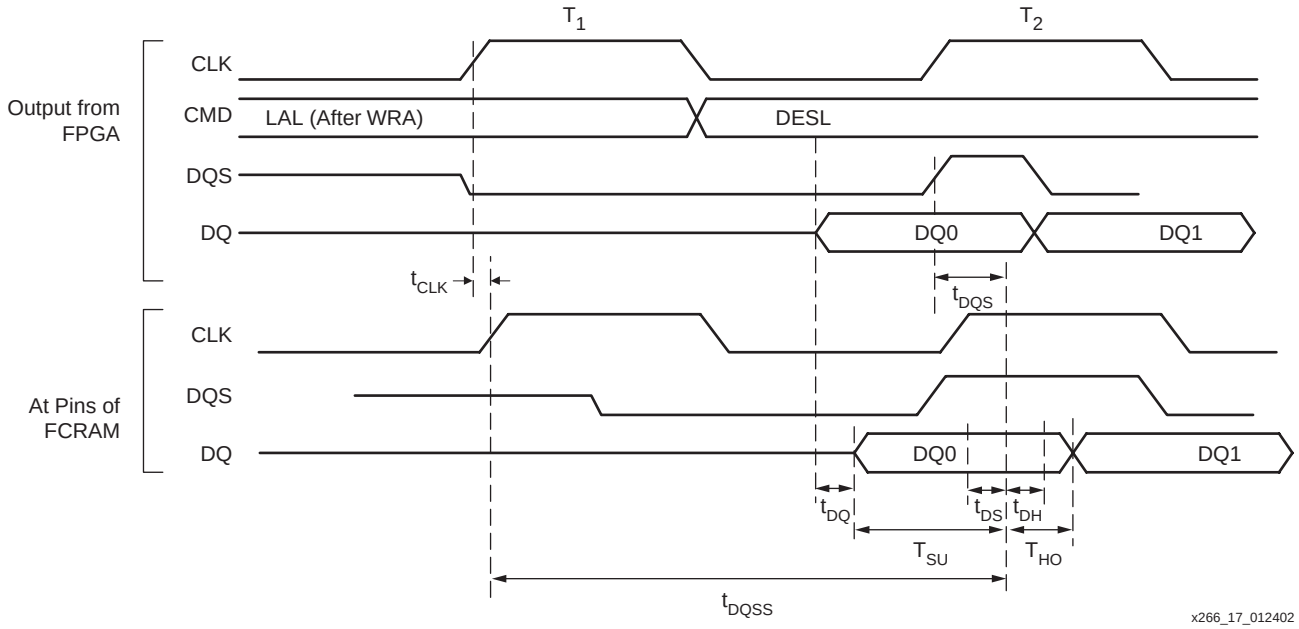

図 17 : 書き込みタイミング図

DQ の配線遅延は t_{DQ}、DQS の配線遅延は t_{DQS}、また CLK の配線遅延は t_{CLK} で示しています。表 3 の FCRAM セット アップ (t_{DS}) および ホールド (t_{DH}) 値を使用し、これらの 3 つの遅延の関係を次の計算式で示します。T_{SU} (セットアップ) と T_{DS} の差異はセットアップのスラックであり、T_{HO} (ホールド) と T_{DH} の差異はホールドのスラックです。

$$t_{SU} = \frac{t_{CK}}{4} + t_{DQS} - t_{DQ} > t_{DS}$$

$$t_{HO} = \frac{t_{CK}}{4} + t_{DQ} - t_{DQS} > t_{DH}$$

この計算式により (t_{DQS} -t_{DQ}) の値は次のようになります。

$$t_{DS} - \frac{t_{CK}}{4} < t_{DQS} - t_{DQ} < \frac{t_{CK}}{4} - t_{DH} \quad \text{(EQ 3)}$$

$$-1.025ns < t_{DQS} - t_{DQ} < 1.025ns$$

書き込みサイクル中におけるその他のタイミング要件は、DQS と CLK の関係です。このタイミング要件は、DQS の Low から High へのセットアップ時間 (t_{DQSS}) として指定され、図 17 に示すとおり CLK 立ち上がりエッジから DQS の立ち上がりエッジまでの時間です。FCRAM 仕様には、このパラメータに対して最小値および最大値の両方あります (表 4)。DQS および CLK は、同じクロックを使用する DDR フリップフロップを介して生成されるため、1 クロック サイクル (t_{CK}) のセットアップ時間があります。ただし、CLK および DQS 両方の配線長は、この値 (t_{CK}) を調整し、また出力標準の調整（図 17 には示していませんが、FCRAM の推奨によると、CLK は SSTL2_I、DQS は SSTL2_II ）を行います。EQ4 は、DQS と CLK の関係を示します。

$$t_{DQSS(MIN)} < t_{CK} + t_{DQS} + T_{OSSTL2(II)} - t_{CLK} - T_{OSSTL2(I)} < t_{DQSS(MAX)} \quad \text{(EQ 4)}$$

表 4 の値を使用した場合、次の値が算出されます。

$$-1.015ns < t_{DQS} - t_{CLK} < 2.235ns$$

リード リキャプチャ タイミング解析

メモリ読み出し中、データは DQS 信号で IOB フリップフロップに取り込まれます。DQS はストローブ信号であるため、データを IOB から次のデータ パスへと送信するためのクロック エッジを確実に生成する保証がありません。このため、DQS から別のクロック ドメインへデータを再度取り込む必要があります。

このリファレンス デザインは、データの再取り込みを行うためにユーザー クロックを位相シフトさせたクロックを使用しています。この方法を使用するには、正確な位相シフトを計算する必要があります。これらのクロックについては、図 18 を参照してください。IOB DDR フリップフロップを介して内部ユーザー クロックを送信すると、FCRAM クロック (`ddr_clk`) が生成されます。このクロックは、FPGA から FCRAM (t_{CLK}) へ入力します。

FCRAM 仕様に記載されているとおり、クロックが入力されている間、メモリは DQS 信号を $\pm t_{CKQS}$ 範囲内で出力します。DQS 信号は FCRAM から出力され、IOB フリップフロップに接続している FPGA (t_{DQS}) へ入力します。

DCM の位相シフト機能を使用して、リキャプチャ クロックと FPGA へ入力する DQS 信号をアラインします。DQS ドメインからリキャプチャ クロック ドメインへデータが転送されるため、リキャプチャ クロックは DQS が到達する最も早いところに位置する必要があります。これにより、クロック ドメイン転送の理想的なタイミングが保証されます。EQ 5 は、リキャプチャ クロックの位相シフト値を算出する計算式です。

$$Target\ Phase\ Shift = (T_{ICKOFDCM} \times PF) + t_{CLK(MIN)} + t_{DQS(MIN)} + t_{DQSINTDELAY(MIN)} \quad \text{(EQ 5)}$$

DQS とリキャプチャ クロック間のタイミング関係には制約が必要です。ベスト ケースでは、DQS ドメインからリキャプチャ クロック ドメインへデータを転送するには、1 クロック必要になります。一方、ワースト ケースでは 1 クロック周期から最大と最小のパス タイミングの差を減算します。表 8 に示すプロレイティング値を使用して、次の計算式で位相シフト値を算出します。

$$DQS\ to\ rclk = t_{CK} - Phase\ Shift\ (Max) - Phase\ Shift\ (Min) \quad \text{(EQ 6)}$$

$$Phase\ Shift\ (Max) = T_{ICKOFDCM} + t_{CLK(MAX)} + t_{CKQS(MAX)} + t_{DQS(MAX)} + t_{DQSINTDELAY(MAX)}$$

$$Phase\ Shift\ (Min) = Target\ Phase\ Shift$$

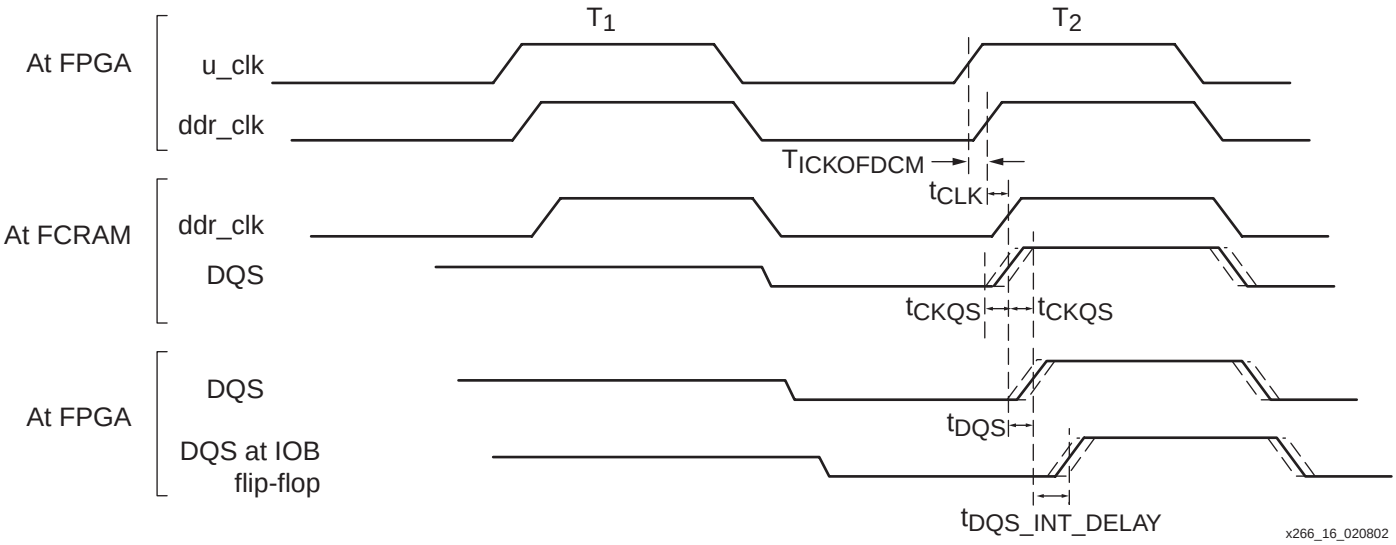

図 18： クロックから DQS への出力遅延時間

制御信号のタイミング解析

すべてのアドレスおよび制御信号は、コントローラにより clk の立ち下りエッジで生成されます。これにより自動的に 1/2 サイクル セットアップおよびホールド時間が生成されます。これらの値はクロック (t_{CLK}) の配線遅延およびコマンド/アドレス信号 (t_{CMD}) の配線遅延によりオフセットされます。コマンド、アドレス セットアップ (t_{IS}) および ホールド (t_{IH}) については 表 3 を参照してください。

$$T_{SU} = \frac{t_{CK}}{2} + t_{CLK} - t_{CMD} > t_{IS}$$

$$T_{HO} = \frac{t_{CK}}{2} + t_{CMD} - t_{CLK} > t_{IH}$$

(t_{CLK} - t_{CMD}) の値は、次のとおりです。

$$t_{IS} - \frac{t_{CK}}{2} < t_{CLK} - t_{CMD} < \frac{t_{CK}}{2} - t_{IH} \quad \text{(EQ 7)}$$

$$-2.25\ ns < t_{CLK} - t_{CMD} < 2.25\ ns$$

タイミング解析について

このセクションでは、リファレンス デザインのクリティカル I/O タイミングのサンプル解析を示します。また、クロック、データ、データ ストローブ、および FPGA と FCRAM 間のアドレス/コントロール 配線ラインなど、これらの関係を成立させるには計算式が必要です。この関係により、FPGA および FCRAM 両方に対して必要な I/O タイミングが保証されます。この解析は、ユーザー指定デザインに合うようカスタマイズする必要があります。

EQ 1 から EQ 3 では、データおよびデータ ストローブ配線との間に制約を与えています。

$$t_{IOPICK} - (t_{DQSINTDELAY} + t_{DQSINTSKEWMIN}) + (t_{QSQ} + t_{DQINTDELAY}) < t_{DQS} - t_{DQ} \quad \text{(EQ 1)}$$
$$\ldots < -t_{IOICKP} + t_{QSQV} + t_{DQINTDELAY} - (t_{DQSINTDELAY} + t_{DQSINTSKEWMAX})$$

$$(t_{IOPICK} \times PF) - ((t_{DQSINTDELAY} + t_{DQSINTSKEWMIN}) \times PF) + (t_{QSQ} + (t_{DQINTDELAY} \times PF)) < t_{DQS} - t_{DQ}$$
$$\ldots < (-t_{IOICKP} \times PF) + t_{QSQV} + (t_{DQINTDELAY} \times PF) - PF(t_{DQSINTDELAY} + t_{DQSINTSKEWMAX}) \quad \text{(EQ 2)}$$

$$t_{DS} - \frac{t_{CK}}{4} < t_{DQS} - t_{DQ} < \frac{t_{CK}}{4} - t_{DH} \quad \text{(EQ 3)}$$

EQ 4 は、DQS のプリアンブル タイミングを特定します。これにより DQS とクロック配線配線の間の関係が成立します。

$$t_{DQSS(MIN)} < t_{CK} + t_{DQS} + T_{OSSTL2(II)} - t_{CLK} - T_{OSSTL2(I)} < t_{DQSS(MAX)} \quad \text{(EQ 4)}$$

EQ 7 は、ドレス/コントロール信号および FPGA/FCRAM 間のクロック配線の関係に制約を与えています。

$$t_{IS} - \frac{t_{CK}}{2} < t_{CLK} - t_{CMD} < \frac{t_{CK}}{2} - t_{IH} \quad \text{(EQ 7)}$$

サンプル タイミング解析で使用した値を使用すると、次のような関係が成立します。

$$0.503 \ ns < t_{DQS} - t_{DQ} < 1.025 \ ns$$
$$-1.015 \ ns < t_{DQS} - t_{CLK} < 2.235 \ ns$$
$$-2.25 \ ns < t_{CLK} - t_{CMD} < 2.25 \ ns$$

EQ 5 は、特定したシステムにリキャプチャ クロックの場所を適合するためのサンプル計算式です。EQ 6 では、データ ストローブからリキャプチャ クロックまでのパスに正しく制約が与えられているかを確認できます。

$$Target\ Phase\ Shift = (T_{ICKOFDCM} \times PF) + t_{CLK(MIN)} + t_{DQS(MIN)} + t_{DQSINTDELAY(MIN)} \quad \text{(EQ 5)}$$

$$DQS\ to\ rclk = t_{CK} - Phase\ Shift\ (Max) - Phase\ Shift\ (Min) \quad \text{(EQ 6)}$$

ローカル クロック分配のためのピン配置制約

ピン配置を行う前に、パッド入力からデータ ロードのクロック ピンまでに DQS ラインを配線するために最適なリソースが必要になります。グローバル クロック ツリーを使用して DQS ラインを分配する方法がありますが、各 DQS ラインは、8 つのデータしか駆動しないため、この方法は有効なクロック リソースを非効率的に使用してしまう結果となります。

Virtex-II には、デバイスの左右エッジに沿ってローカル クロック分配ネットワークがあります。このネットワークにより、信号は IOB へ入力し、固定数の IOB クロック ピンと直接接続している高速かつロー スキューのローカル配線リソースへ信号が接続されます。このセクションでは、これらのリソースについての概要、およびリソースを有効活用して DQS クロック ラインを分配する方法を説明します。

『Virtex-II データ シート』(参考資料 1) に記載されているとおり、各入力/出力 タイルには 1 つのスイッチ マトリックスを共有する 4 つの IOB があります。IOB PAD4 が最上部、IOB PAD1 が最下部に位置しています。DQS 信号はローカル クロック ラインにアクセスするため、DQS パッドは IOB PAD4 (一番下の IOB) に位置する必要があります。指定したパッケージに IOB PAD4 がない場合は、入力/出力タイルを DQS 信号に使用できない場合があります。

DQS パッドを IOB PAD4 に配置すると、ローカル クロック ラインへ直接アクセスできます。このローカル クロックは、選択した DQS 入力/出力タイルより 5 行上へ伸びる HEX ラインです (選択した DQS

入力/出力タイルも駆動）。なお選択した DQS 入力/出力タイルより 6 行下へも伸びています。データ (DQ) パッドは、これらの 12 行内に配置する必要があります。

FPGA Editor のサンプル図 (図 19) は、DQS パッドから 5 列上および 6 列下の DQ パッドを駆動する DQS パッド （最上部から 6 列目）を示しています。このデザイン ファイルにはピン配置の例があります。

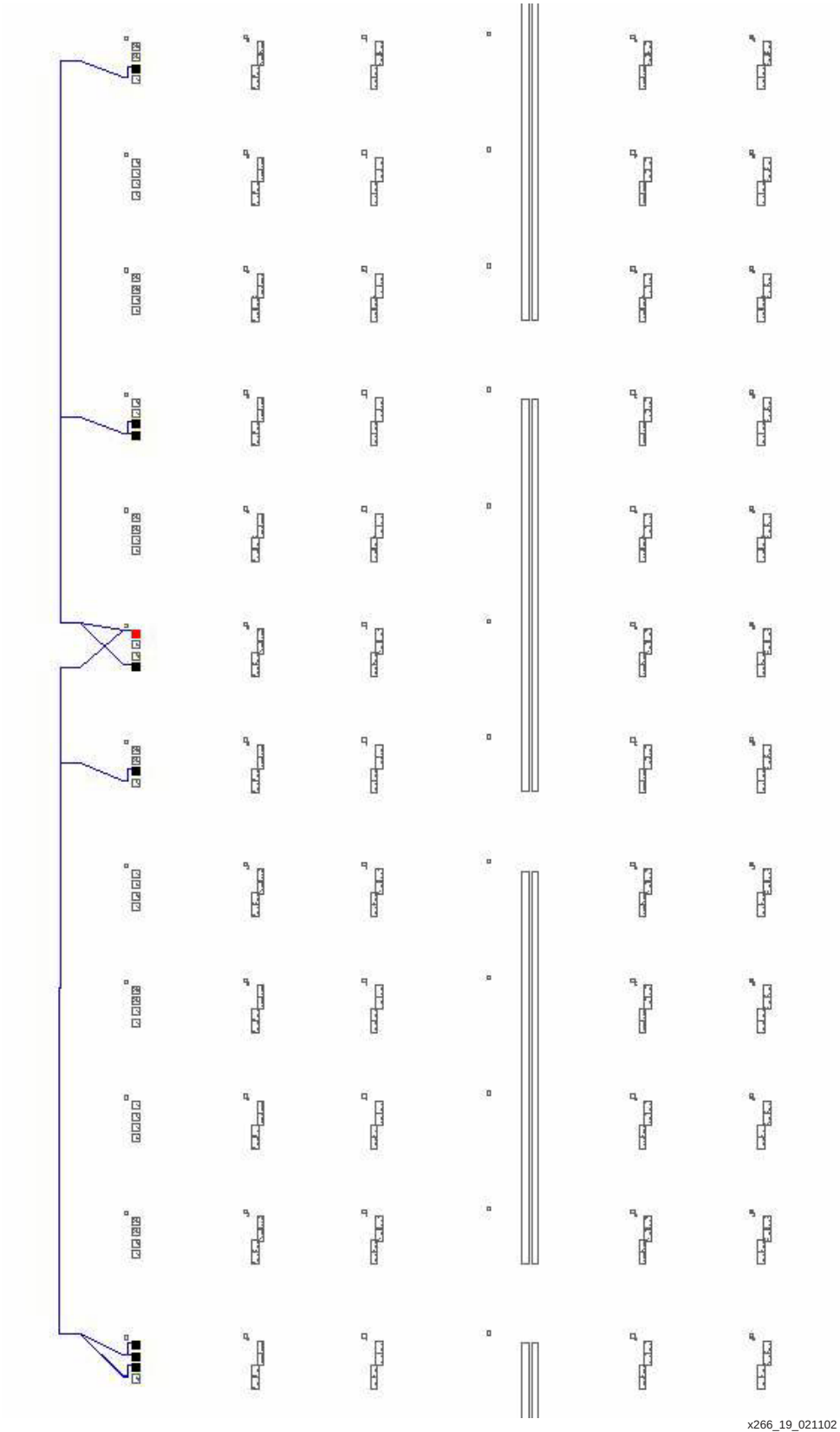

図 19： サンプル DQS 配線

デザイン インプリメンテーション

このリファレンス デザインには、Verilog ソース コード、制約ファイル、およびサンプル インプリメンテーション スクリプトが含まれています。このデザインは、3 つのグローバル クロック バッファと 2 つの DCM を使用しています。16 ビット データ バスには、約 600 個のスライスが必要です。このリファレンス デザインは、次のザイリンクス FTP サイトから入手できます。
ftp://ftp.xilinx.com/pub/applications/xapp/xapp266.zip

参考資料

その他の詳細については、次の資料を参照してください。

1. ザイリンクス社、データシート 「Virtex-II 1.5v フィールド プログラマブル ゲート アレイ」2002 年 www.xilinx.com.
2. Xilinx Inc., XAPP200, Synthesizable 1.6 GBytes/s DDR SDRAM Controller, Application Note, 2000
3. Toshiba Inc., DDR FCRAM, Data sheet, 2001
4. Fujitsu Inc., DDR FCRAM, Data sheet, 2001

まとめ

高性能かつ低電力な FCRAM メモリは、メモリ集積度が高く、広いバンド幅を必要とするアプリケーションに最適です。このアプリケーション ノートは、FCRAM 技術の説明およびザイリンクス Virtex-II ファミリに FCRAM コントローラをインプリメントし、その影響ついて説明しています。

FCRAM デバイスは、ソース同期インターフェイスを採用しています。このインターフェイスは、データと共に送信され、データを取得するクロックとしても使用される双方向データ ストローブを使用します。このシステムを使用したタイミング解析は、従来型の完全同期システムとは大きく異なります。そのため、このアプリケーション ノートでは、設計者がタイミングを容易に検証できるようにサンプル タイミング解析、タイミング図および計算式を用いて説明しています。

タイミング クロージャを完了する前に、ボード レベルでのタイミング解析が必要です。ザイリンクスでは、タイミング解析および正常なシグナル インテグリティを保証する IBIS シミュレーションなどのボード レベル デザイン ツールの使用を推奨しています。また、レース スタックアップのシミュレーション、配線長、ピン キャパシタンスを解析に含むことにより、FCRAM デバイスの読み出し、またすべての信号に対して正しく終端されているかを検証します。さらに、Virtex-II ユーザー ガイド に記載されている同時スイッチ出力 (SSO) ガイドラインにも従ってください。

リファレンス デザインは、シングル x16 FCRAM デバイスを対象にしていますが、付録 A で説明しているとおり、Verilog コードを変換し、異なるメモリ コンフィギュレーションを使用することもできます。

付録 A

DDR FCRAM メモリ バス幅を変更するには、次のように Verilog HDL ソースを変更してください。

- `define.v`

必要なメモリ バス幅の値を入力します。

```
'define DDR_DATA_WIDTH <desired width>
```

HDL コードでインスタンシエートされるコンポーネントがある場合、さまざまなメモリ バス幅に対応するため、ユーザーはコンポーネントに変更を行う必要があります。

- `data_path.v`

data_path モジュールは、DQ バスの DDR 入力および出力フリップ フロップをインスタンシエートします。8 DQ ビット (1 バイト) のインスタンシエーションは、`data_path.v` HDL ファイルの中の *v2_ddr_iob* モジュールに含まれます。必要な外部メモリ バス幅と一致するように、インスタンシエートの数も変更する必要があります。たとえば、インターフェイスが、x16 メモリ 1 つだけの場合、*v2_ddr_iob* モジュールのインスタンシエーションは 2 回になります。インターフェイスが、x72 ビット

バスを生成する複数 FCRAM デバイスの 場合は、このモジュールに対して 9 回インスタンシエーションが必要になります。HDL ファイル内で 「DDR IOB Instantiations」セクションを検出し、バス幅と一致するようにインスタンシエーション数を変更してください。

- `data_strobe.v`

data_strobe モジュールは、DQS 信号に DDR 出力フリップ フロップをインスタンシエートします。シングル DQS ビットのインスタンシエートは、`data_strobe.v` HDL ファイルの中にある *v2_dqs_iob* モジュールの中に含まれます。外部メモリ コンフィギュレーションと一致するようにインスタンシエート数を変更する必要があります。たとえば、1 バイトに 1 データ ストローブを含むシングル x16 メモリ インターフェイスの場合、*v2_dqs_iob* モジュールのインスタンシエートが 2 回必要です。x72 ビット バスを生成する複数 FCRAM インターフェイスの場合、このモジュールのインスタンシエートは 9 回必要です。HDL ファイル内で 「DQS I/O Block Instantiations」セクションを検出し、必要なメモリ コンフィギュレーションと一致するようにインスタンシエート数を変更してください。

このリファレンス デザインが、複数の FCRAM デバイスを制御するように設定されている場合は、信号の読み出しを確認してください。クロック信号、アドレス信号、制御信号は、すべてのメモリ デバイスで共有されています。デバイスが追加されると、これらの信号のパフォーマンスが低下します。このため、最適な読み出しおよびこれらの信号の配置を決定するには IBS およびその他のボード レベル シミュレーションを実行する必要があります。通常、推奨される読み込みについては、メモリ ベンダにより詳細情報が与えられます。複製ドライバが必要な場合は、HDL コードを変更する必要があります。

改訂ヒストリ

次の表は、このアプリケーション ノートの改訂ヒストリを示します。

日付	バージョン	履歴
02/27/02	1.0	初版リリース