XAPP1321 (v1.0) 2017 年 10 月 23 日

DDR4 メモリ インターフェイスの高速キャリブレーションとデイジーチェーン機能

著者: John Schmitz、Parth Joshi、Santosh P

概要

このアプリケーション ノートでは、ザイリンクス UltraScale™ および UltraScale+™ デバイスにおける、DDR4 キャリブレーション時間の大幅な短縮 (10 倍の高速化) を実現する方法と、パーシャル リコンフィギュレーションまたはフル リコンフィギュレーション中に DDR4 メモリのコンテンツを維持し、デイジー チェーン機能を有効にする方法について説明します。パラレル インターフェイスのデータ レートの高速化が進むにつれて、キャリブレーション スキームと所要時間は急激に増加します。キャリブレーション時間が長くなると、メモリが動作可能になるまで貴重な時間を浪費しなければなりません。デイジー チェーン機能とは、FPGA にビット ファイルを読み込み、DDR4 メモリにメモリ コンテンツを読み込んで、次のビット ファイルがそのデータを使用できるようになるまで格納できる機能です。このアプリケーション ノートでは、DDR4 メモリ コントローラーにキャリブレーション データを迅速にロードする方法と、FPGA がリコンフィギュレーションされた場合でもメモリ内のコンテンツを有効に維持する方法を示します。また、リコンフィギュレーション中に外部 DRAM コンテンツを維持する機能をシステム レベルで実装する場合の考慮事項についても説明します。

このアプリケーション ノートのリファレンス デザイン ファイルは、ザイリンクスのウェブサイトからダウンロードできます。デザイン ファイルの詳細は、「リファレンス デザイン」を参照してください。

高速トレーニングとコンテンツの維持

DDR4 はビット レートが高速であるため (UltraScale+ デバイスで最大 2,667Mb/s)、最大限のパフォーマンスが得られるようにインターフェイスを最適に調整するには、複雑なトレーニング アルゴリズムを使用する必要があります。一部のアプリケーションでは、この処理に長い時間がかかります。しかし、多くのアプリケーションでは、最初のフル トレーニングを実行した後は、インターフェイスのリセットや再起動ごとに高速なミニ トレーニングを実行できるため、起動時間は大幅に短縮されます。

DDR4 メモリのコンテンツを維持することは、多くのアプリケーションにとって有益です。たとえば、FPGA を完全にパワーダウンして消費電力を最大限削減する場合や、FPGA のリコンフィギュレーション中に DDR4 メモリのデータを維持し、次の新しい機能が使用できるように中間値を格納しておく場合がこれに該当します。後者はしばしばデイジー チェーン機能と呼ばれます。

これらのユース ケースは、いずれもザイリンクス DDR4 メモリ インターフェイスのトレーニング アルゴリズムがメモリ内の値を保存する機能と、保存された値を利用してトレーニングの所要時間を短縮する機能を利用しています。この保存/復元プロセスの詳細は、『UltraScale アーキテクチャ FPGA メモリ IP LogiCORE IP 製品ガイド』(PG150) [参照 1] を参照してください。トレーニング データおよびキャリブレーション データは、パーシャル リコンフィギュレーションを使用する場合に FPGA 内のスタティック領域に保存することも、短時間での起動またはフル リコンフィギュレーションのために FPGA の外部に格納することも可能です。このプロセスでは、ユーザー デザインによって抽出可能なレジスタ内の既知の位置にトレーニング データを置きます。このデータは同じレジスタ内に戻すことができ、フル トレーニング プロセスをスキップするようにトレーニング アルゴリズムに指示するフラグをセットできます。最初のフル トレーニングの完了後に電圧と温度が変化する可能性があるため、トレーニングでは若干の調整が自動的に行われます。これらの調整は、最初のトレーニング プロセスよりもはるかに短時間で完了します。

DRAM は、メモリ コントローラーからのコマンド シーケンスによってセルフ リフレッシュ モードに移行します。このステートでも、ホスト コントローラーとの最小限のやり取りは必要です。クロックはオフにでき、ほとんどのコマンド ピンとアドレス ピンは無視されます。これは DRAM のコンテンツを維持する低消費電力動作モードです。コントローラーは、DRAM コンテンツにアクセスする場合、別のシーケンスを実行して DRAM を通常の動作モードに戻します。通常、セルフ リフレッシュ モードは、DRAM へのアクセスが不要なときの消費電力削減に使用されます。

DRAM のセルフ リフレッシュ機能には、FPGA と組み合わせて使用した場合、ほかにもいくつかの用途があります。特に、この機能により、DRAM のコンテンツを失うことなく、FPGA のフル リコンフィギュレーションとパーシャル リコンフィギュレーションが可能になります。これは一部のアプリケーションで重要です。ソフト ロジック ベースのメモリ コントローラーは、FPGA のフル リコンフィギュレーション プロセス中にその機能を継続できないため、当然セルフ リフレッシュのサポートが必要になります。パーシャル リコンフィギュレーションの場合、サポートの必要性はそれほど明白ではありませんが、やはり非常に重要です。単にメモリ コントローラーをスタティック領域に配置するソリューションは一見簡単に思われますが、デバイス内の配線の問題が解決しにくくなります。したがって、リコンフィギュレーションされる領域にメモリ コントローラーを配置したままで、DRAM コンテンツを維持できるようにする必要があります。ザイリンクス FPGA のパーシャル リコンフィギュレーションについては、『Vivado Design Suite ユーザー ガイド: パーシャル リコンフィギュレーション』(UG909) [参照 2] を参照してください。

DDR4 DRAM コンポーネントのセルフ リフレッシュ モード

DDR4 DRAM のセルフ リフレッシュ機能については、JEDEC 仕様 JESD79-4B とメモリ ベンダーのデータシートに記載されています。簡単にまとめると、SRE コマンドによってセルフ リフレッシュ モードに移行し、SRX コマンドを使用してセルフ リフレッシュ モードを終了します。DRAM がセルフ リフレッシュ モードに移行した後は、CK、ODT、コマンド、アドレスはすべて don't care になります。CKE は Low に維持する必要があります。JEDEC JES79-4B 仕様の図 146 を参照してください。RESET_n は High に維持する必要があります。

DDR4 RDIMM のセルフ リフレッシュ モード

DDR4 RDIMM をセルフ リフレッシュ モードに移行するプロセスは、DDR4 DRAM の場合と類似していますが、フル リコンフィギュレーションまたはパーシャル リコンフィギュレーションと併用する場合の要件が異なります。RDIMM 上のレジスタ クロック ドライバー (RCD) は、クロック入力ピンがフロートする前に RDIMM がクロック ストップ パワーダウン モードになっていることを要求します。さらに、このモードのとき、RCD は DDR4 DRAM コンポーネントへの CKE ピンを自動的に Low に駆動します。クロック ストップ パワーダウン モードに移行するには、セルフ リフレッシュ モードへの移行後にクロックの両方のレッグ (ck_t および ck_c) を Low に駆動します。RCD チップは ck_t および ck_c 上に内部プルダウン抵抗を備えていますが、これらの入力はフロートにする前に Low に駆動されている必要があります。この場合、FPGA オプション PUDC (コンフィギュレーション中のプルアップ) は RCD の弱いプルダウンを無効にし、RCD チップのクロック ストップ パワーダウン モードを終了させることがあるため、このオプションは利用できません。クロック ストップ パワーダウン モードの詳細は、JEDEC DDR4 RCD01 および DDR4 RCD02 仕様を参照してください。

ボードレベルの考慮事項

DRAM に対する外部信号の処理

DDR4 DRAM に対する重要な外部信号は、CKE と RESET_n です。RDIMM では、ck_t および ck_c 信号の特殊な取り扱いも必要になります。これらの信号の扱い方は、DRAM のコンフィギュレーション (コンポーネント、DIMM) とリコンフィギュレーション モード (フル、パーシャル) によって異なります。これらの信号については、すべてのシステムで最適な結果が得られるような処理方法はありません。この文書では、デザインごとに最適な方法を選択できるように、いくつかの選択肢とそれぞれの長所および短所について説明します。

RESET_n

RESET_n は、電源投入中は Low に維持されている必要があります。初期化プロセスが開始されるとコントローラーによって Low に維持され、その後 High 駆動されます。このプロセスの後、RESET_n は High にとどまる必要があります。Low になると、DRAM がリセットされてデータが破損するおそれがあります。FPGA のフル リコンフィギュレーション中と、コントローラーの I/O バンクが配置されている場所のパーシャル リコンフィギュレーション中は、FPGA ピンはフロートします (PUDC を使用した弱いプルアップは可能)。詳細は、『UltraScale アーキテクチャ コンフィギュレーション ユーザー ガイド』(UG570) [参照 3] の「コンフィギュレーション ピンの定義」の表を参照してください。通常は 4.7kΩ 抵抗を使用して RESET_n ピンをプルダウンします。これにより、DRAM は電源投入中にリセット状態に維持されます。残念なことに、特殊な処理を適用せずに FPGA をリコンフィギュレーションした場合、このプルダウン抵抗によって DRAM がリセットされてしまいます。RESET_n は、DRAM クロックに対して非同期の CMOS 信号です。

パーシャル リコンフィギュレーション

この場合の最も簡単なソリューションは、RESET_n 信号を DDR4 メモリ コントローラー コアからスタティック領域に配線し、ピンへ出力することです。スタティック領域に含まれるロジックは、メモリ コントローラーとハンドシェイクし、セルフ リフレッシュに移行したことを認識します。この場合、スタティック ロジックにメモリ コントローラーの RESET_n 出力と FPGA 上の OBUF の間の OR ゲートを組み込んで、メモリ コントローラーの RESET_n 信号を駆動できます。最初のコンフィギュレーションおよび FPGA の動作中は、このオーバーライド信号が `0` にセットされるように注意する必要があります。その後、セルフ リフレッシュ ステートへの移行後に、この信号をセットできます。これにより、RESET_n 出力は強制的に High になり、リコンフィギュレーションの実行時に DRAM のセルフ リフレッシュ動作を妨げません。図 1 にこの手法を示します。

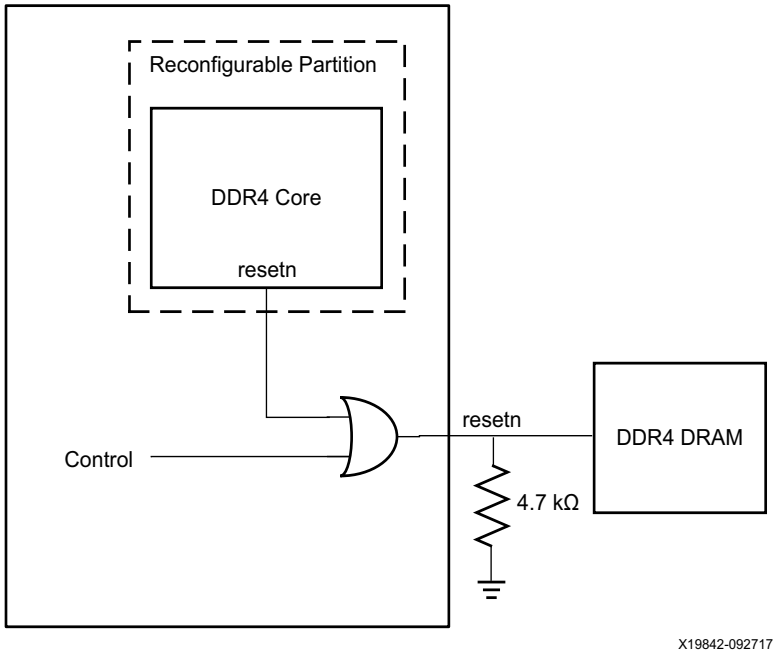

図 1: パーシャル リコンフィギュレーション ソリューション

DDR4 コントローラー コアからの RESET_n 信号をスタティック領域に配線できない場合は、外部ロジックを使用して、セルフ リフレッシュ ステート中に RESET_n を強制的に High にする必要があります。このシステム ロジックは、セルフ リフレッシュがアクティブであることをコントローラー コアが示したときにリセット オーバーライド信号をアクティブにし、セルフ リフレッシュが終了したときにこれを非アクティブにします。リセット オーバーライドを駆動するシステム ロジックは、図 2 に示すように、FPGA の外部に配置することも内部に配置することも可能です。

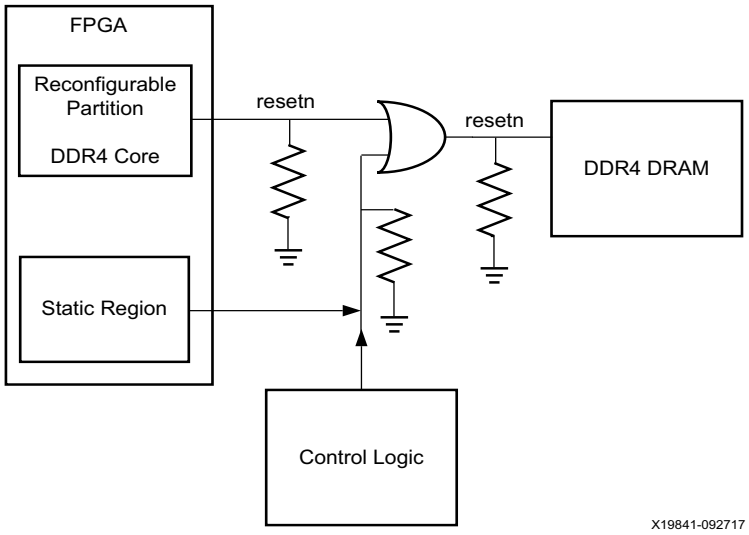

図 2: 外部ロジックによるパーシャル リコンフィギュレーション ソリューション

フル リコンフィギュレーション

フル リコンフィギュレーションの場合、RESET_n オーバーライドは FPGA の外部で発生させる必要があります。システム ロジックは、図 3 に示すように、セルフ リフレッシュがアクティブであることをコントローラー コアが示したときにリセット オーバーライド信号をアクティブにし、セルフ リフレッシュが終了したときにこれを非アクティブにします。

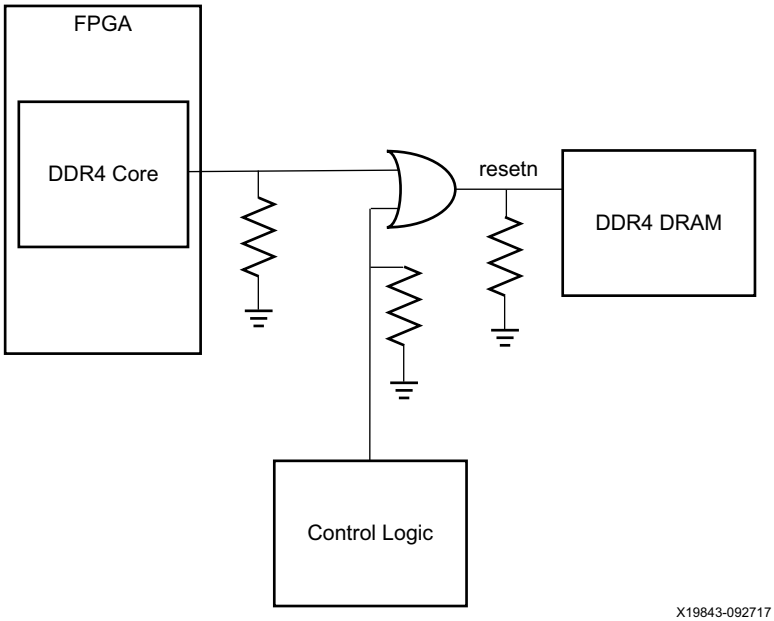

図 3: フル リコンフィギュレーション ソリューション

CKE (DDR4 コンポーネントおよび Unbuffered UDIMM/SODIMM の場合)

CKE 信号の扱いは RESET_n よりも難しくなります。これは CKE 信号が DRAM に対する高速コマンド信号で、クロックを基準とするセットアップ タイムとホールド タイムを満たす必要があるためです。さらに、CKE 信号は、通常は V_{TT} 電源 (DDR4 の場合は 0.6V) に対して比較的低インピーダンス (たとえば 40Ω または 50Ω) で終端されます。コントローラーからのドライバーは、通常は SSTL12 信号用に構成されます。CKE 信号は、ほかのコマンド信号、制御信号、アドレス信号と共に、DDR4 DRAM コンポーネント V_{REFCA} によって基準ピンと比較されます。V_{REFCA} は、V_{DD} 電源 (これも 0.6V) の中点に設定されます。図 4 に CKE の標準的なトポロジを示します。

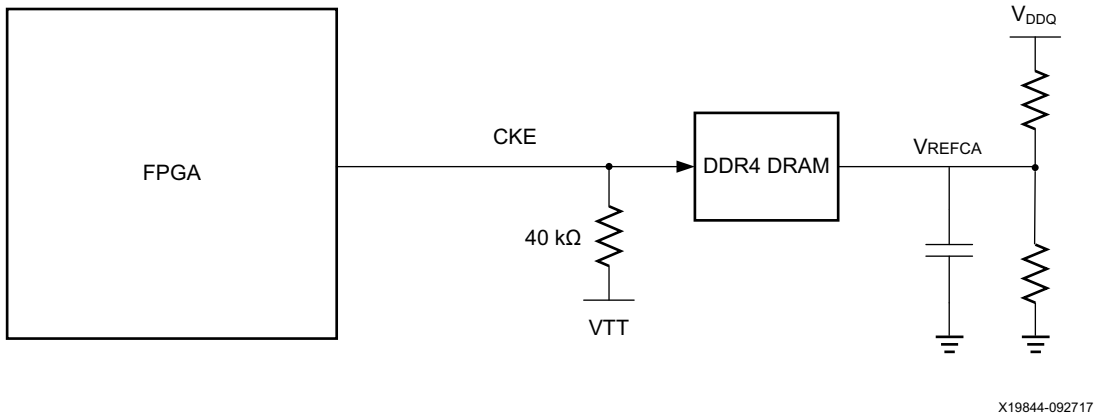

図 4: 標準的な CKE トポロジ

V_{TT} への強い抵抗を用いた終端は V_{REFCA} に等しくなり、FPGA の出力が無効 (high-Z) のときに CKE 信号がロジックしきい値に非常に近くなるため、推奨できません。システム内のわずかなノイズによって CKE が高レベル信号として認識されると、DRAM が誤ってセルフ リフレッシュを終了し、データ破損を引き起こす可能性があります。FPGA のリコンフィギュレーションと組み合わせて使用されるデフォルト コンフィギュレーションにおいて、これは許容されません。

スタティック領域を使用できる RESET_n ソリューションとは異なり、CKE には高速なタイミング要件が課せられます。リコンフィギュレーションされないスタティック バンク内に CKE を配置した場合、バンク間のタイミング スキューが高すぎて、この要件を満たせません。利用可能なシステム コンポーネントとデザインのトレードオフによって、CKE の問題には複数のソリューションがあります。なお、CKE は、RDIMM の場合も特別な扱いは不要です (詳細は「ck_t および ck_c (RDIMM の場合)」を参照)。

アクティブ CKE プルダウンによる制御

1つのソリューションは、CKE を直接プルダウンすることです。この手法は、トランジスタと抵抗を使用して、セルフ リフレッシュへの移行後に CKE 信号をプルダウンします。この方法は CKE 信号の品質に直接影響を与えるため、シグナル インテグリティへの影響ができるだけ小さく抑えられるように注意する必要があります。CKE は、メモリ クロックと同じ周波数で動作する SDR 信号として動作します。図 5 に実装例を示します。

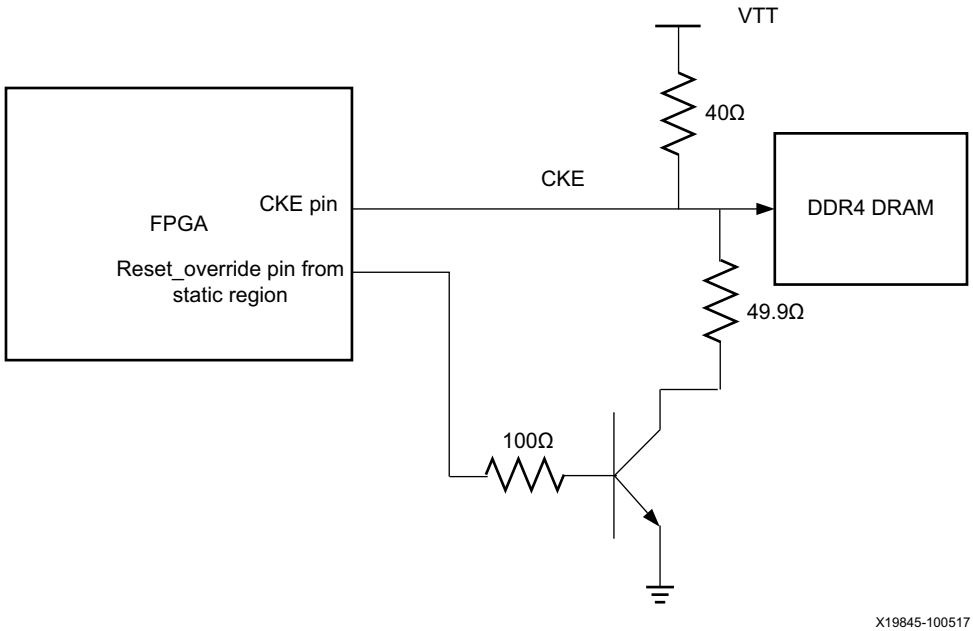

図 5: アクティブ CKE プルダウンによる制御

この手法は、V_{TT} への 40Ω 終端に非常に近い位置に抵抗とトランジスタを配置できるコンポーネント システムでよく使用されます。シミュレーションを実行して、特定のボード トポロジに対するこの手法の適合性を確認する必要があります。Unbuffered DIMM および SODIMM は、DIMM カード上のスタブが長いため、この手法は推奨できません。

V_{TT} 電源による制御

V_{TT} への強い抵抗を用いた終端を使用すると、リコンフィギュレーション中に FPGA 出力が無効 (High-Z) になったときに CKE 信号が不適当な電圧領域まで引き上げられます。V_{TT} 電源がセルフ リフレッシュ中に電圧を下げられるか、またはオフにされれば、この期間中 CKE 信号は適切な低電圧にとどまります。多くの V_{TT} レギュレータはイネーブル ピンを備えており、この目的に利用できます。図 6 に例を示します。この例は、Texas Instruments 社の TPS51200 レギュレータを使用しています。ピン 7 の「EN」ピンがイネーブル ピンです。

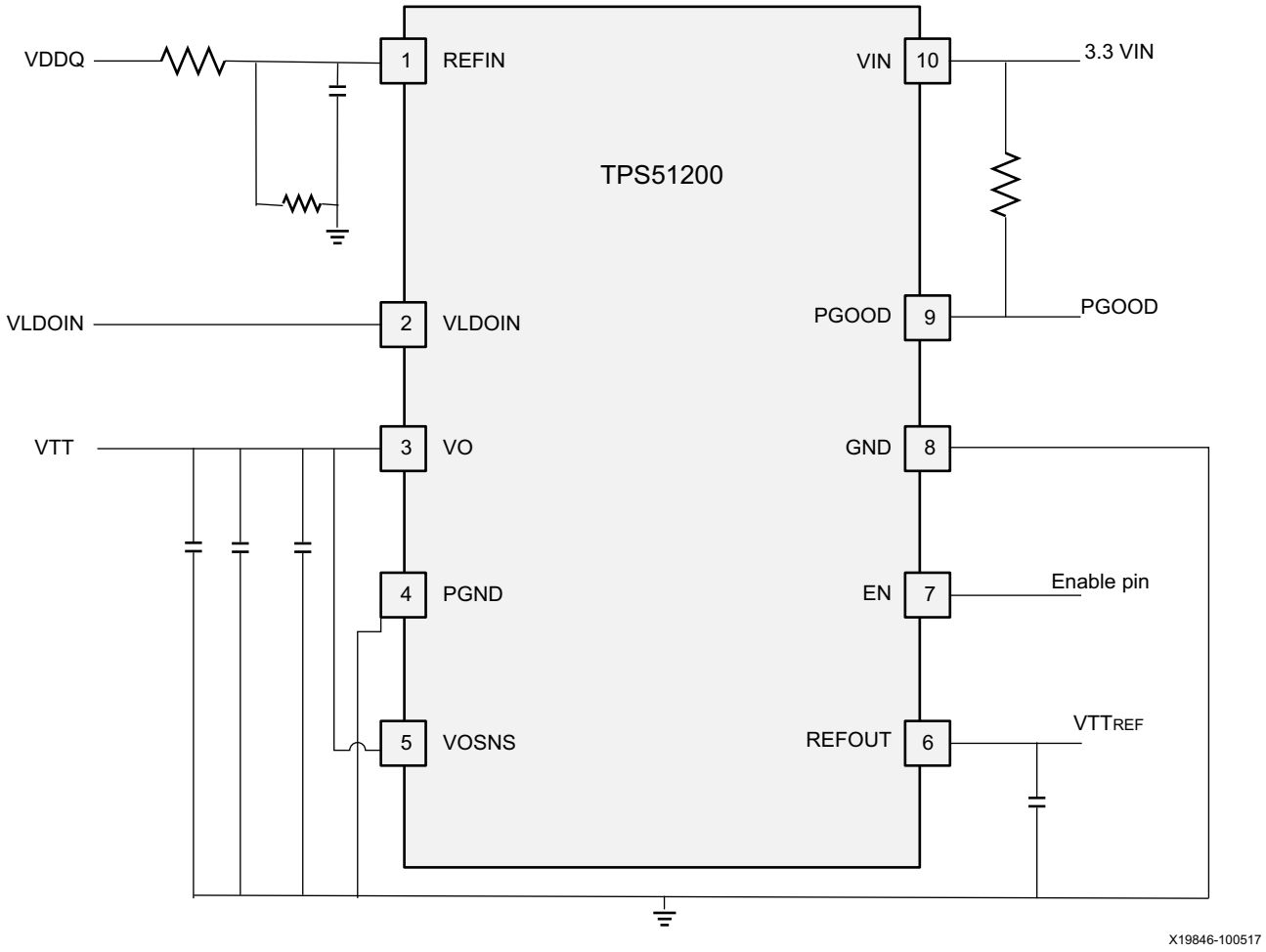

図 6: V_{TT} 電源による制御

セルフ リフレッシュに移行してから FPGA がリコンフィギュレーションされるまでの間、レギュレータがシャットダウンするのにかかる時間を考慮に入れる必要があります。また、リコンフィギュレーションが完了したとき、DRAM コントローラー コアに対してリセット信号をリリースする前に、V_{TT} レギュレータが V_{TT} を復元したことを確認する必要があります。レギュレータは、大きなアンダーシュートやオーバーシュートなしにオンおよびオフにする必要があります。レギュレータの出力は、無効のときは Low に維持される必要があります (外部プルダウン抵抗を使用して実現可能)。Texas Instruments 社の TPS51200 は、オンおよびオフの立ち上がりを制御し、無効のときは出力をプルダウンします。

また、V_{TT} 電源と V_{REFCA} 電源 が同じレギュレータから供給されないようにする必要があります。これは、CKE (およびほかのコマンド/アドレス/制御信号) は V_{REFCA} 信号を基準とするためです。この場合、V_{REFCA} 信号は抵抗分圧回路などのほかのソースから供給する必要があります。図 7 に標準的な抵抗分圧回路を示します。

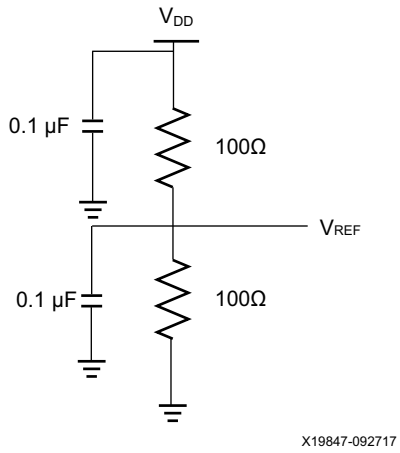

図 7: 標準的な V_{REF} 抵抗分圧回路

ck_t および ck_c (RDIMM の場合)

RDIMM アプリケーションでは、リコンフィギュレーション プロセス中に RCD へのクロック ピンをフロートさせる前に、RDIMM 上の RCD チップがクロック ストップ パワーダウン モードに移行する必要があります。DDR4 メモリ コントローラー IP では、セルフ リフレッシュへの移行後に ck_t と ck_c の両方が Low に駆動されるようにする必要があります。また、リコンフィギュレーションの終了後にクロックを適切に再起動する必要があります。

FPGA デザインで PUDC (コンフィギュレーション中のプルアップ) を設定しないでください。PUDC は、リコンフィギュレーション中にクロック ピンを Low に維持するために必要とされる、RCD デバイス内の弱いプルダウンの妨げになります。

Memory Interface Generator IP デザインを使用した Vivado デザイン ツール フロー

このセクションでは、セルフ リフレッシュ モードへの移行、キャリブレーション データの保存、セルフ リフレッシュ モードの終了、キャリブレーション データの復元のプロセスについて説明します。パーシャル リコンフィギュレーション フローとその利点についても詳しく説明します。

ザイリンクス プログラマブル ロジック DDR4 コントローラーによるセルフ リフレッシュのサポート

ザイリンクス PL DDR4 コントローラーは、キャリブレーションの保存および復元と関連して使用されるセルフ リフレッシュ機能をサポートします。これは、DRAM をセルフ リフレッシュ モードに移行するだけでなく、セルフ リフレッシュを迅速に終了して通常動作へ復帰できるようにキャリブレーション データを保存する、複数段階のプロセスです。セルフ リフレッシュへの移行プロセスは次のとおりです。

1. ユーザー デザインは、メモリ コントローラーへのトラフィックを停止します。
2. ユーザー デザインは、メモリ コントローラーに対するセルフ リフレッシュ要求を app_sref_req ポート上に発行します。
3. コントローラーは、保留中の DRAM トランザクションをフラッシュし、DRAM をセルフ リフレッシュ モードに移行させた後、app_sref_ack 信号を使用して保存要求に肯定応答 (ACK) を返します。
4. ユーザー デザインは、コントローラーからキャリブレーション データをスタティック領域 (パーシャル リコンフィギュレーションの場合) または FPGA の外部 (フル リコンフィギュレーションの場合) にコピーします。

セルフ リフレッシュ モードでは、DRAM の CKE は Low に維持される必要があります。また、RESET_n 信号は High に保たれる必要があります。CKE と RESET_n はこのアプリケーションの重要な外部信号です (「ボード レベルの考慮事項」を参照)。

この段階で、フル リコンフィギュレーションまたはパーシャル リコンフィギュレーションが実行可能です。CKE 信号と RESET_n 信号が DRAM 側で変動しない限り、DRAM はセルフ リフレッシュ モードのままになり、データのインテグリティは維持されます。セルフ リフレッシュ モードを終了して動作を再開するには、次のシーケンスが必要です。

1. ユーザー インターフェイスのリセット信号がディアサートされてからメモリ コントローラー クロックの 50 サイクル以内に、mem_init_skip 信号と app_restore_en 信号がアサートされ、高速キャリブレーション サイクルが完了するまでアサートされたままになる必要があります (キャリブレーションの完了は init_calib_complete 信号によって示される)。mem_init_skip 信号は、通常の DRAM 初期化プロセスをスキップするようにキャリブレーション プロセスに直接指示します。app_restore_en 信号は、フル キャリブレーション プロセスを実行する (DRAM のコンテンツを破壊する) のではなく、格納されたデータを使用するようにコントローラーに指示します。
2. ユーザー デザインは、格納されたキャリブレーション データをスタティック パーティションまたはオフチップ メモリからコントローラーにコピーします。
3. ユーザー デザインは、app_restore_complete 信号をアサートすることで、キャリブレーション データのコピーが完了したことを示します。
4. コントローラーは、DRAM のセルフ リフレッシュを終了し、取得したキャリブレーション データを使用して高速キャリブレーション プロセスを実行します。
5. 高速キャリブレーション プロセスの完了は、init_calib_complete 信号のアサートで示されます。
6. ユーザー デザインは、mem_init_skip 信号と app_restore_en 信号をディアサートします。
7. これで、ユーザー デザインは、維持された以前のデータでメモリ コントローラーにトランザクションを送信できます。

コントローラーのセルフ リフレッシュ機能のザイリンクスによるインプリメンテーションの詳細は、『UltraScale アーキテクチャ FPGA メモリ IP LogiCORE IP 製品ガイド』(PG150) [参照 1] の「DDR3/4 コア アーキテクチャ」を参照してください。

保存/復元機能によるキャリブレーション時間の短縮

保存/復元機能は、トレーニング時間を劇的に短縮するのに非常に有用で、パーシャル リコンフィギュレーション フローを使用する必要もありません。この機能は、標準的な Memory Interface Generator (MIG) IP サンプル デザインでサポートされており、最上位ポートを生成する保存/復元機能を選択することで有効になります。ポートの詳細は、『UltraScale アーキテクチャ FPGA メモリ IP LogiCORE IP 製品ガイド』(PG150) の「保存/復元」を参照してください。キャリブレーション データは、同じ FPGA 内のブロック RAM または FPGA の外部のオンボード メモリに格納できます。この選択はアプリケーションに依存します。このアプリケーション ノートでは、同じ FPGA 上の ブロック RAM にキャリブレーション データを格納する例を示します。付属のリファレンス デザインを参照してください。このリファレンス デザインには、`example_top` モジュール内に保存/復元シーケンスを駆動するステート マシンがあります。保存/復元機能を使用する場合、このステート マシンは DDR4 インターフェイスと同じクロックで実行できます。リファレンス デザインは保存/復元機能とセルフ リフレッシュ機能の両方で使用されるため、スタティック領域からもう 1 つのクロックを動作させています。この機能に必要とされるハードウェアの考慮事項はありません。詳細は、『UltraScale アーキテクチャ FPGA メモリ IP LogiCORE IP 製品ガイド』(PG150) [参照 1] の「保存/復元」を参照してください。

保存/復元プロセス中にメモリの初期化が実行されます。この機能の主な利点は、フル キャリブレーションを実行した場合に比べてキャリブレーション時間が短縮されることです。この機能は、以前にキャリブレーションされたデータを Xilinx System Debugger (XSDB) ブロック RAM に復元し、キャリブレーションの DQS ゲート トラッキング ステージのみを実行します。この機能の欠点としては、追加のストレージが必要になることと、ロジックが若干追加されることです。

リファレンス デザインに付属の TCL スクリプトは保存/復元機能を実行します。このアプリケーション ノートに付属のリファレンス デザインは、Vivado Design Suite バージョン 2016.4 を使用してハードウェアでテストされています。詳細な手順については、関連する `readme` ファイルを参照してください。

パーシャル リコンフィギュレーション

パーシャル リコンフィギュレーションでの DDR4 セルフ リフレッシュ機能の利点は次のとおりです。

- キャリブレーション時間の短縮
  - 保存/復元機能とセルフ リフレッシュ機能を組み合わせると、キャリブレーション時間が約 50ms に短縮されます。
- 消費電力の削減
  - セルフ リフレッシュ機能は、一般に未使用時の DRAM の消費電力の削減に使用されます。

    - セルフ リフレッシュと FPGA のパーシャル リコンフィギュレーションを組み合わせると、DRAM と FPGA の両方の消費電力を削減できます。
- データ インテグリティの維持
- パーシャル リコンフィギュレーションの柔軟性が向上し、パーシャル リコンフィギュレーション フローを使用して追加のロジックを配線できる

パーシャル リコンフィギュレーション デザインの生成

図 8 に、パーシャル リコンフィギュレーション デザイン生成のブロック図を示します。

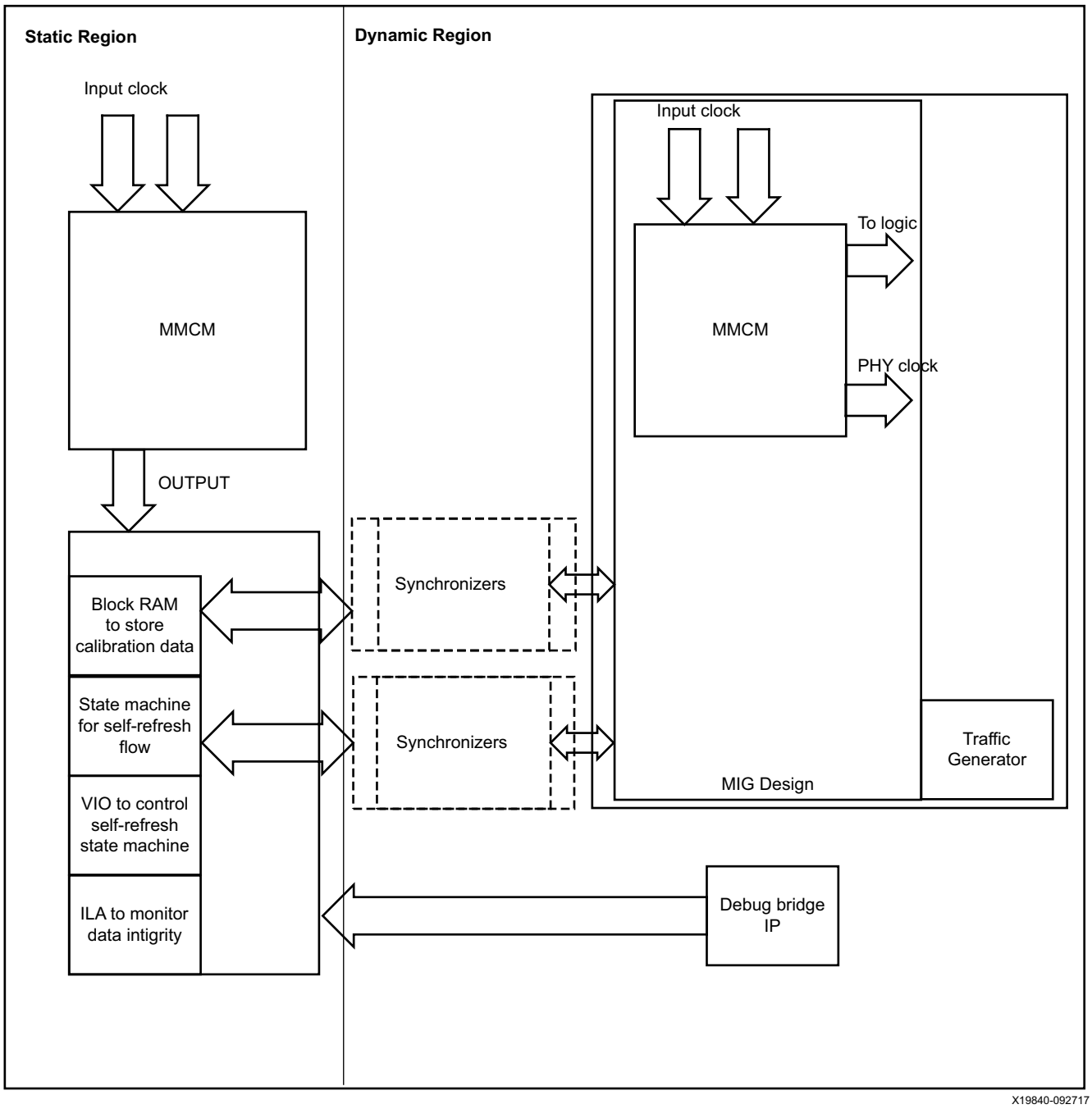

図 8: パーシャル リコンフィギュレーション デザイン生成のブロック図

ブロック図に示すように、このデザインには 2 つの領域があります。スタティック領域には、FPGA の再プログラム中も維持されるロジックがあります。ダイナミック領域は、FPGA のコンフィギュレーション中に再プログラム可能です。アプリケーションとロジックの要件に基づいて、ダイナミック領域とスタティック領域の両方をフロアプランできます。これには Pblock を使用して実行できます。このデザインでは、ダイナミック領域に Pblock があり、デバイスのその他の部分はスタティック領域として使用されます。

注記: Vivado Design Suite は、より小さく、より管理しやすい物理ブロック (Pblock) へとデザインを階層的に分割する機能を備えています。Pblock には、デザイン内の任意の箇所のロジック モジュールとプリミティブ ロジックを含めることができます。

パーシャル リコンフィギュレーション フローにおけるソフトウェアに関する考慮事項

パーシャル リコンフィギュレーション フローにおけるソフトウェアに関する考慮事項は次のとおりです。

- クロックはダイナミック領域からは駆動できません。
- クロックはスタティック領域からダイナミック領域へは駆動できます。
- MIG DDR4 IP を使用する場合のパーシャル リコンフィギュレーション フローのインプリメンテーションには、バッチ モードでは Vivado Design Suite 2016.3 以降、プロジェクト モードでは Vivado Design Suite v2017.1 以降が必要です。

MIG DDR4 IP デザインの生成要件

MIG IP デザインを生成するには、次の手順を実行します。

1. 図 9 に示すように、MIG デザインを作成する際は MIG GUI でセルフ リフレッシュ機能を有効にする必要があります。

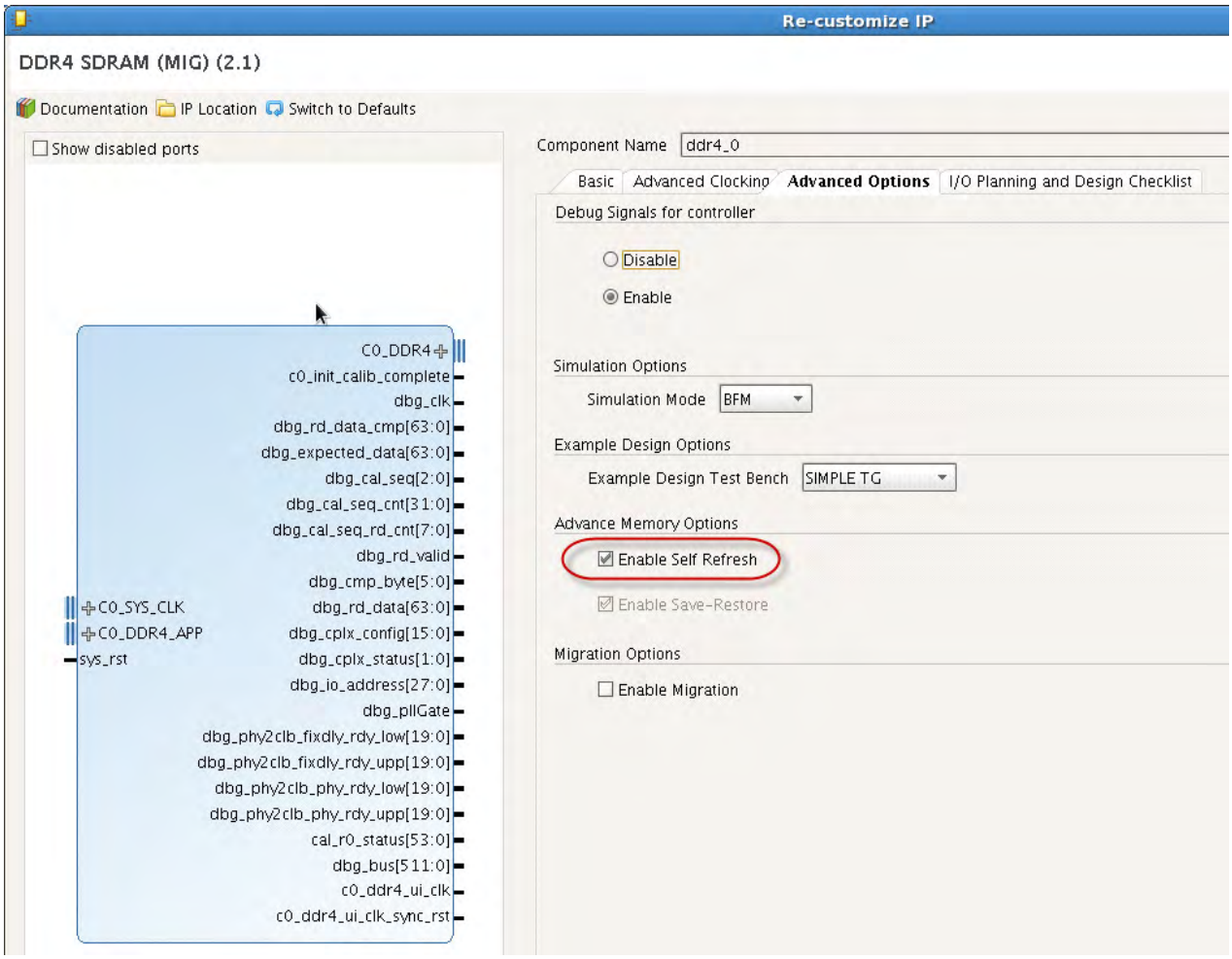
Re-customize IP
DDR4 SDRAM (MIG) (2.1)
Documentation
IP Location
Switch to Defaults
Show disabled ports
C0_DDR4
c0_init_calib_complete
dbg_clk
dbg_rd_data_cmp[63:0]
dbg_expected_data[63:0]
dbg_cal_seq[2:0]
dbg_cal_seq_cnt[31:0]
dbg_cal_seq_rd_cnt[7:0]
dbg_rd_valid
dbg_cmp_byte[5:0]
dbg_rd_data[63:0]
dbg_cplx_config[15:0]
dbg_cplx_status[1:0]
dbg_io_address[27:0]
dbg_pllGate
dbg_phy2clb_fixdly_rdy_low[19:0]
dbg_phy2clb_fixdly_rdy_upp[19:0]
dbg_phy2clb_phy_rdy_low[19:0]
dbg_phy2clb_phy_rdy_upp[19:0]
cal_r0_status[53:0]
dbg_bus[511:0]
c0_ddr4_ui_clk
c0_ddr4_ui_clk_sync_rst
C0_SYS_CLK
C0_DDR4_APP
sys_rst
Component Name
ddr4_0
Basic
Advanced Clocking
Advanced Options
I/O Planning and Design Checklist
Debug Signals for controller
Disable
Enable
Simulation Options
Simulation Mode
BFM
Example Design Options
Example Design Test Bench
SIMPLE TG
Advance Memory Options
Enable Self Refresh
Enable Save-Restore
Migration Options
Enable Migration

図 9: MIG GUI

2. MIG IP は、図 10 に示すように合成する必要があります。

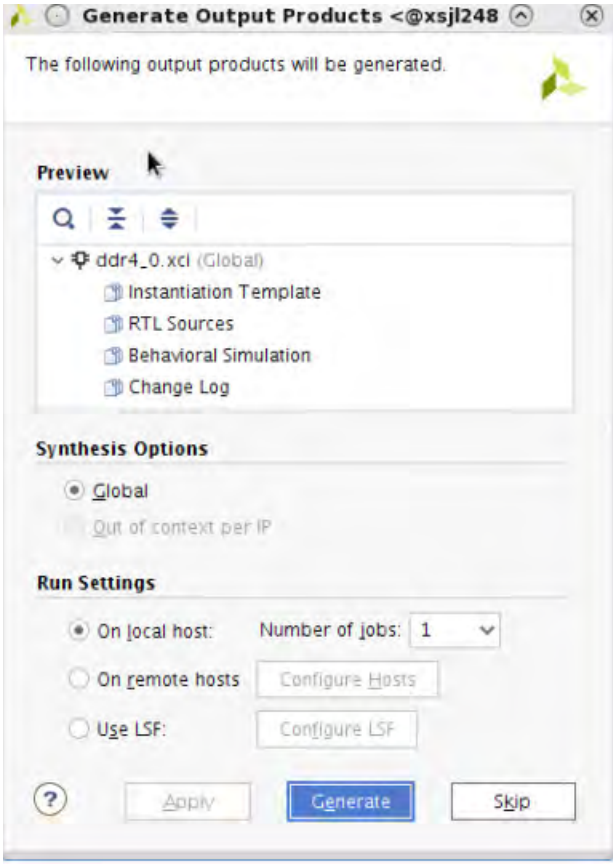

図 10: 合成オプション

3. Vivado Design Suite 2017.1 以降では、Debug Bridge IP のインスタンシエーションは不要です (Vivado Design Suite 2017.1 については、付属のプロジェクト モードのデザインを参照)。ただし、Vivado Design Suite 2016.3 および 2016.4 では、次の手順でダイナミック領域に debug_bridge IP をインスタンシエートする必要があります。
   a. すべてのポートをスタティック領域内の最上位モジュールに配置します。
   b. スタティック領域クロックと debug_bridge IP の clk ポートを接続します。
4. ポートの接続については、付属のリファレンス デザインを参照してください。Vivado Design Suite は、その他の必要な接続をすべて自動的に実行します。

デザイン アーキテクチャ

リファレンス デザインは、これらの手法がどのように実際のシステムで使用されるかの実例を示すものです。サンプル デザインは、パーシャル リコンフィギュレーションの実行中に、メモリ コントローラーを使用してセルフ リフレッシュ プロセスによって DRAM にデータを維持する方法を示します。このリファレンス デザインは、プロジェクト モードでは Vivado Design Suite 2017.1、非プロジェクト モードでは Vivado Design Suite 2017.2 を使用してハードウェアでテストされています。サンプル デザインは、次のイベント シーケンスを実行します。

1. メモリ コントローラーに対してリセットをリリースし、通常の方法でキャリブレーションさせます。
2. メモリにパターンを書き込みます。
3. メモリ コントローラーをセルフ リフレッシュに移行させます。
4. キャリブレーションから得られたデータを格納します。
5. しばらくの間停止します。
6. メモリ コントローラー領域をリコンフィギュレーションします。
7. 保存されたキャリブレーション値をメモリ コントローラーに戻します。

8. メモリ コントローラーのセルフ リフレッシュを終了させて、保存されたキャリブレーション値を使用して DRAM へのアクセスを再び有効にします。
9. DRAM からデータをリードバックして、書き込まれた元のデータが維持されていることを検証します。
10. データ検証の結果 (成功または失敗) を表示します。

リファレンス デザインの最上位モジュールは、下位モジュールをラップおよび制御し、イベント シーケンスを実行します。さらに、このモジュールはメモリ コントローラーを介してデータを作成および格納し、リコンフィギュレーションが完了してセルフ リフレッシュが終了した後、コントローラーから受信したデータをチェックします。サンプル デザインの最上位モジュールは、スタティック領域に配置されます。このモジュールには次のモジュールが含まれます。

- MMCM – スタティック領域のクロックを生成します。このクロックは、MIG IP 内の XSDB ブロック RAM クロックと同じ周波数で動作する必要があります。
- ブロック RAM – ダイナミック領域の XSDB ブロック RAM からのキャリブレーション データを格納します。
- ステート マシン – セルフ リフレッシュへの移行/終了サイクルを制御します。
- VIO – セルフ リフレッシュ ステート マシンとトラフィック ジェネレーターを制御します。
  - Virtual Input/Output (VIO) コアは、内部 FPGA 信号をリアルタイムでモニターおよび駆動できるようにカスタマイズ可能なコアです。
- ILA – データ比較エラーとキャリブレーション完了信号をモニターします。
  - カスタマイズ可能な Integrated Logic Analyzer (ILA) IP コアは、デザインの内部信号のモニターに使用できるロジック アナライザー コアです。
- パフォーマンス カウンター – セルフ リフレッシュ終了後の復元の所要時間を示します。
- 内部コンフィギュレーション アクセス ポート (ICAP) インスタンス ICAPE2 – パーシャル リコンフィギュレーション プロセスをモニターします。

ダイナミック領域には次のモジュールが含まれます。

- シンクロナイザー – スタティック領域とダイナミック領域の間で信号を駆動し、クロック乗せ換え (CDC) の問題を防ぎます。
- MIG DDR4 IP デザイン (IP ごとのアウト オブ コンテキスト (OOC) で合成されたオプション) – 各種の DDR4 SDRAM メモリとインターフェイスするためのソリューションを提供します。
- (MIG IP サンプル デザインからの) トラフィック ジェネレーター (トラフィック ジェネレーターの入力は VIO によって制御)。トラフィック ジェネレーターはサンプル デザイン (`example_top.sv`) にインスタンシエートされ、アプリケーション インターフェイスを介してメモリ デザインを駆動します。トラフィック ジェネレーターとその機能の詳細は、『UltraScale アーキテクチャ FPGA メモリ IP LogiCORE IP 製品ガイド』(PG150) [参照 1] を参照してください。
- Debug Bridge IP (フィールド アップデートおよびパーシャル リコンフィギュレーション ソリューションとの組み合わせ) – ユーザーが選択可能なモード From_BSCAN_to_Debug は、ILA、VIO、メモリ IP、JTAG2AXI などのデバッグ コアに接続されるデバッグ ブリッジ インスタンスを各リコンフィギャラブル モジュールに追加します。

このリファレンス デザインには、`example_top` モジュール (リファレンス デザインを参照) 内にセルフ リフレッシュ機能を駆動するステート マシンがあります。セルフ リフレッシュ プロセス中は、メモリの初期化は省略されます。メモリの初期化を省略することで、セルフ リフレッシュの終了時にデータ インテグリティを維持し、DRAM に対する通常の操作を再開できます。この機能の主な利点は、消費電力の削減とキャリブレーション時間の短縮です。この機能は、以前にキャリブレーションされたデータを XSDB ブロック RAM に復元し、キャリブレーションの DQS ゲート トラッキング ステージのみを実行します。またこの機能は、セルフ リフレッシュ サイクルに移行する前にメモリに書き込まれたデータが、セルフ リフレッシュ サイクルの終了後も保持されるよう、データ インテグリティを維持します。

非プロジェクト モードのフロー

MIG IP デザインを使用したパーシャル リコンフィギュレーション フローは、現在、Vivado Design Suite 2016.3 以降の非プロジェクト モードでサポートされています。MIG IP は、グローバル コアまたは OOC で合成されたコアでなければなりません。詳細なディレクトリ構造は、関連するリファレンス デザインを参照してください。

`*.prj` ファイル拡張子を持つプロジェクト ファイルには、スタティック領域およびダイナミック領域のすべての XCI ファイル、Verilog ファイル、System Verilog ファイルが含まれます。ブラック ボックス Verilog ファイル `*_bb.v` には、最上位ダイナミック領域モジュールのすべてのポートが含まれます。このモジュールは、最上位スタティック モジュールにインスタンシエートされます。

デザインの変更またはデザインへの新しいモジュールの追加が必要になった場合は、プロジェクトの `*.prj` ファイルをアップデートする必要があります。ダイナミック領域に追加された新しいポートがある場合は、ブラック ボックス Verilog ファイル `*_bb.v` をアップデートする必要があります。

フル インプリメンテーションを実行するには、`design.tcl` ファイルを見つけて、次のコマンドを実行します。

```
vivado -mode batch -source design.tcl
```

このコマンドはフロー全体を初期化し、3 つのビット ファイルとそれに関連するクリア ビット ファイルを生成します。

- ビット ファイル – デザイン全体のスタティック モジュールおよびリコンフィギャラブル モジュールと共に使用されます。
- パーシャル クリア ビット ファイル – 次のリコンフィギャラブル モジュール用にリコンフィギュレーション可能な領域を準備します (UltraScale ファミリ デバイスにのみ必要。UltraScale+ ファミリ デバイスでは不要)。
- パーシャル ビット ファイル – リコンフィギャラブル モジュールと共に使用されます。

リファレンス デザインに付属の TCL スクリプトは、DDR4 セルフ リフレッシュへの移行/終了サイクルの検証手順を実行します。詳細な手順は、リファレンス デザインの `readme` ファイルを参照してください。

プロジェクト モードのフロー

MIG IP デザインを使用したパーシャル リコンフィギュレーション フローは、Vivado Design Suite 2017.1 以降のプロジェクト モードでサポートされています。MIG IP は、Vivado Design Suite 2017.2 によってグローバル コアとして合成されている必要があります。リコンフィギャラブル モジュール内の IP の OOC 合成は、Vivado Design Suite 2017.3 以降でサポートされています。詳細なディレクトリ構造は、関連するリファレンス デザインを参照してください。

セルフ リフレッシュへの移行サイクル

次の手順でセルフ リフレッシュ サイクルを開始します。

1. 要求信号の app_sr_Req がアサートされます。
2. 肯定応答 (ACK) 信号の app_sr_ack がアサートされます。
3. XSDB ブロック RAM のデータがスタティック領域のブロック RAM に保存された後、図 11 に示すように、xsdb_bram_save_complete 信号が Low から High に遷移します。

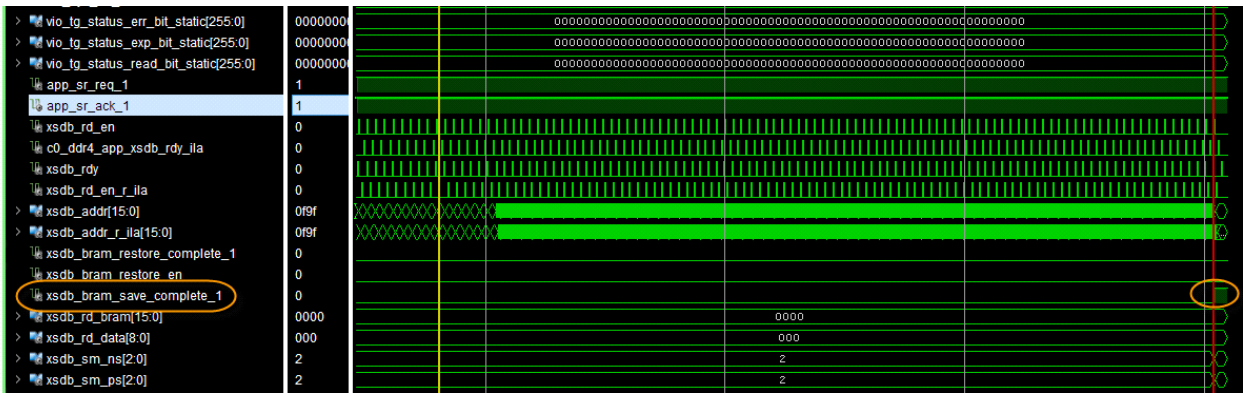

図 11: xsdb_bram_save_complete 信号の遷移

セルフ リフレッシュ終了サイクル

次の手順でセルフ リフレッシュ終了サイクルを開始します。

1. データ インテグリティを維持してセルフ リフレッシュ サイクルを終了するには、メモリの初期化を避ける必要があります。つまり、hw_init_skip_en 信号がアサートされる必要があります。
2. セルフ リフレッシュ終了サイクルでユーザー インターフェイス リセット信号 (ui_clk_sync_rst) がディアサートされてから通常のインターコネクトの 50 サイクル以内で、xsdb_bram_restore_en 信号をアサートします。この信号は、キャリブレーションが完了するまでアサートを保持します。ここで、メモリ インターフェイスの reset_n 信号は 1、cke は 0 になります。
3. スタティック領域のブロック RAM のデータがダイナミック領域の XSDB ブロック RAM に復元された後、図 12 に示すように、ダイナミック領域の xsdb_bram_restore_complete 信号が Low から High に遷移します。

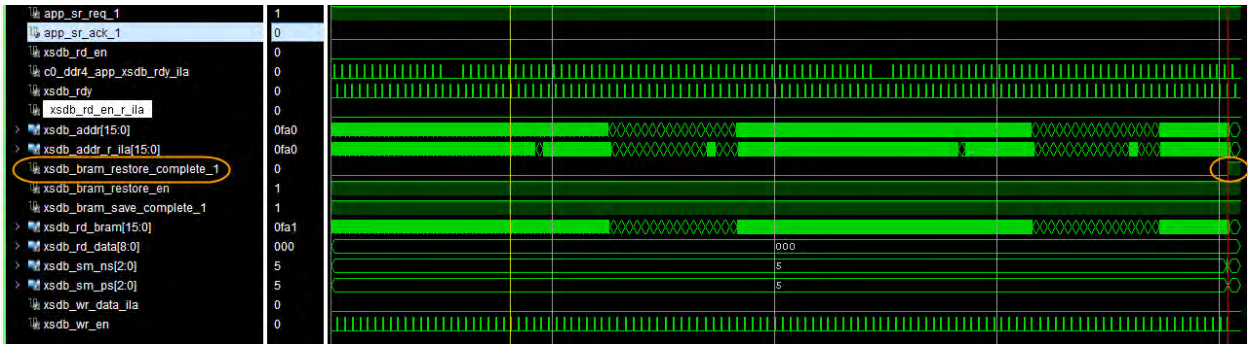

図 12: xsdb_bram_restore_complete 信号の遷移

図 13 に、MIG のキャリブレーション ステージ ステータスの GUI を示します。セルフ リフレッシュ終了サイクルの後、MIG DDR4 インターフェイスはフル キャリブレーションを実行していないことに注意してください。DQS_GATE トラッキング以外のすべてのキャリブレーション ステージはスキップされています。GUI は、DQS_GATE トラッキングのキャリブレーションに合格したことを示しています。

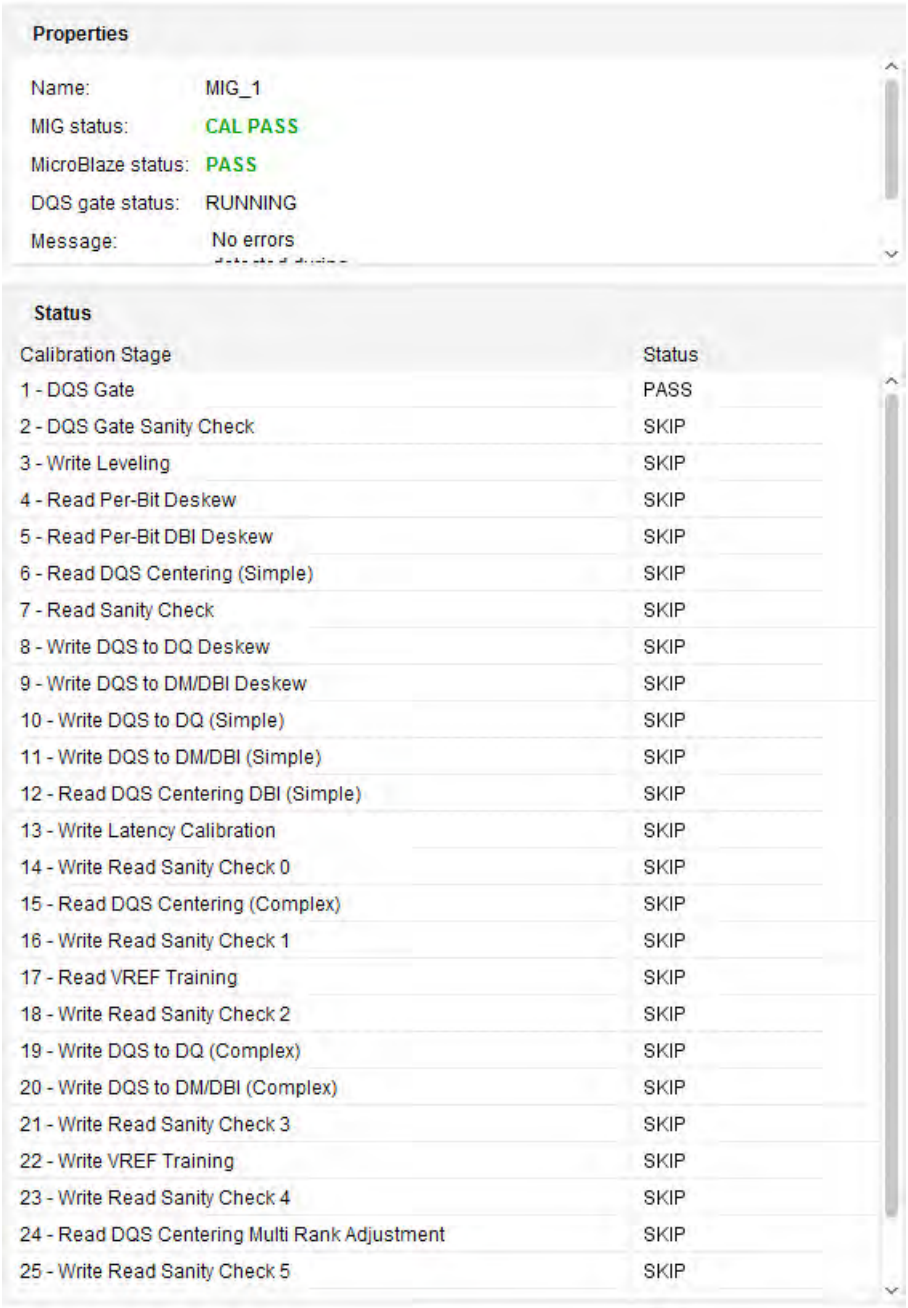

図 13: キャリブレーション ステージ ステータスの GUI

まとめ

このアプリケーション ノートでは、DDR4 IP の保存/復元機能を使用してトレーニング (キャリブレーション) 時間を大幅に短縮する方法について説明しました。また、DDR4 のセルフ リフレッシュ機能を使用して、さまざまなニーズに応じて FPGA のパーシャル リコンフィギュレーションまたはフル リコンフィギュレーションを使用できるようにする方法と、これらの機能の使用を目的として特定の DDR4 DRAM 信号を処理する方法についても説明しました。

リファレンス デザイン

このアプリケーション ノートのリファレンス デザイン ファイルは、ザイリンクスのウェブサイトからダウンロードできます。表 1 に、リファレンス デザインの詳細を示します。

表 1: リファレンス デザインの詳細

パラメーター	説明
全般	
開発者	ザイリンクス
ターゲット デバイス	UltraScale および UltraScale+ デバイス
ソース コードの提供	あり
ソース コードの形式	Verilog
既存のザイリンクス アプリケーション ノート/リファレンス デザイン、またはサードパーティからデザインへのコード/IP の使用	Vivado Design Suite からの MIG DDR4 IP、Clock Wizard、ILA、VIO、Debug Bridge IP
シミュレーション	
論理シミュレーションの実施	なし
タイミング シミュレーションの実施	なし
論理シミュレーションおよびタイミング シミュレーションでのテストベンチの利用	なし
テストベンチの形式	N/A
使用したシミュレータ/バージョン	N/A
SPICE/IBIS シミュレーションの実施	なし
インプリメンテーション	
使用した合成ツール/バージョン	N/A
使用したインプリメンテーション ツール/バージョン	N/A
スタティック タイミング解析の実施	なし
ハードウェア検証	
ハードウェア検証の実施	あり
使用したハードウェア プラットフォーム	ザイリンクスの内部テスト ボード

参考資料

このアプリケーション ノートの参考資料は次のとおりです。

注記: 日本語版のバージョンは、英語版より古い場合があります。

1. 『UltraScale アーキテクチャ FPGA メモリ IP LogiCORE IP 製品ガイド』(PG150: 英語版、日本語版)
2. 『Vivado Design Suite ユーザー ガイド: パーシャル リコンフィギュレーション』(UG909: 英語版、日本語版)
3. 『UltraScale アーキテクチャ コンフィギュレーション ユーザー ガイド』(UG570: 英語版、日本語版)

改訂履歴

次の表に、この文書の改訂履歴を示します。

日付	バージョン	内容
2017 年 10 月 23 日	1.0	初版

お読みください: 重要な法的通知

この資料に関するフィードバックおよびリンクなどの問題につきましては、jpn_trans_feedback@xilinx.com まで、または各ページの右下にある [フィードバック送信] ボタンをクリックすると表示されるフォームからお知らせください。いただきましたご意見を参考に早急に対応させていただきます。なお、このメール アドレスへのお問い合わせは受け付けておりません。あらかじめご了承ください。