MachXO2 デバイス メモリ使用ガイド

2013 年 7 月 テクニカルノート TN1201

はじめに

本テクニカルノートでは、ラティス MachXO2 ™ PLD ファミリのメモリ使用法について説明します。本書は設計エンジニアが ispLEVER® で、これらのデバイスに EBR 及び PFU ベースのメモリを統合する際のガイドとして使用すること意図しています[注1]。

これらデバイスのアーキテクチャは、メモリを多用するアプリケーション用のリソースを提供します。sysMEM ™ EBR（Embedded Block RAM、組み込みブロック RAM）は、PFU ベースの分散メモリを補完するものです。EBR を使用してシングルポート RAM、デュアルポート RAM、擬似デュアルポート RAM、FIFO、及び ROM メモリが構成可能です。LUT（Look Up Table）及び PFU（Programmable Function Unit）は分散シングルポート RAM、分散デュアルポート RAM、及び分散 ROM を実装できます。

EBR ブロック RAM 及び PFU RAM の機能はプリミティブと呼ばれ、本書で後述します。設計者はメモリプリミティブを次の 2 通りの方法で利用できます。

- **IPexpress** ™ の使用 ～ IPexpress の GUI を使用すると、ユーザは必要なメモリタイプとサイズを指定できます。IPexpress はこの指定を受け取り、所定のメモリを実装するために 1 つ以上のメモリプリミティブを使用することでネットリストを構成します。
- **PMI** (Parameterizable Module Instantiation、パラメータ化されたモジュール・インスタンス化）の使用 ～ PMI を使用すると、経験豊富なユーザは GUI を使用せずに、ispLEVER のプロジェクト・ナビゲータ（Project Navigator）から構成可能なメモリモジュールをオンザフライで活用できます。Verilog または VHDL で必要なパラメータ及び制御信号が設定可能です。論理合成時に自動的にブラックボックスを生成できるように、最上位デザインでパラメータを定義し、そして信号を宣言します。

一般的な EBR 及び PFU RAM プリミティブに加えて、規模が MachXO2-640 以上のデバイスでは新しい UFM（User Flash Memory、ユーザフラッシュメモリ）ブロックも集積されています。このブロックは、一部のコンフィグレーション・イメージの格納、EBR データの格納と初期化、PROM データの格納または汎用の不揮発性ユーザフラッシュメモリとしての格納など、多種多様な用途に使用可能です。UFM ブロックは、組み込み機能ブロックの WISHBONE インターフェイスによりデバイスコアに接続します。設計者はデバイスの JTAG、I^2C 及び SPI インターフェイスにより UFM ブロックにアクセスすることもできます。UFM ブロックには次の特長があります。

- 最大 256K ビットの不揮発性ストレージ
- リードアクセスにはバイトアドレス指定が可能。ライトアクセスは 128 バイトのページ単位で実行
- プログラム、消去、及びビジーの各信号
- 自動インクリメント・アドレッシング
- WISHBONE インターフェイス
- 外部アクセスは JTAG、I^2C 及び SPI インターフェイス経由

UFMの詳細については”TN1205, Using User Flash Memory and Hardened Control Functions in MachXO2 Devices（MachXO2 デバイスのユーザフラッシュメモリとハードマクロ制御機能の使用法）″を参照してください。

1. 日本語注：Lattice Diamond を使用する場合も全く同様に適用されます。

© 2011 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

（日本語版は日本語による理解のため一助として提供しています。その作成にあたっては可能な限り正確を期しておりますが、原文英語版との不一致や不適切な訳文がある場合を含み、英語版が正（有効）です。特に電気的特性・仕様値係わる事項については最新版の英語版を必ず参照するようにお願い致します。）

本書ではこれらの各手法や IPexpress の使用法、PMI インターフェイス、メモリモジュール、およびメモリプリミティブについて説明します。

MachXO2 デバイスのメモリ

全ての MachXO2 デバイスには、PIC（Programmable I/O Cell、プログラマブル I/O セル）で囲まれた PFU と呼ばれる複数の論理ブロックが含まれています。また最小の MachXO2 デバイスである MachXO2-256 以外には、sysMEM EBR ブロックがあります。これを図 12-1 と 12-2 に示します。

図 12-1. MachXO2-1200 デバイスの上面図

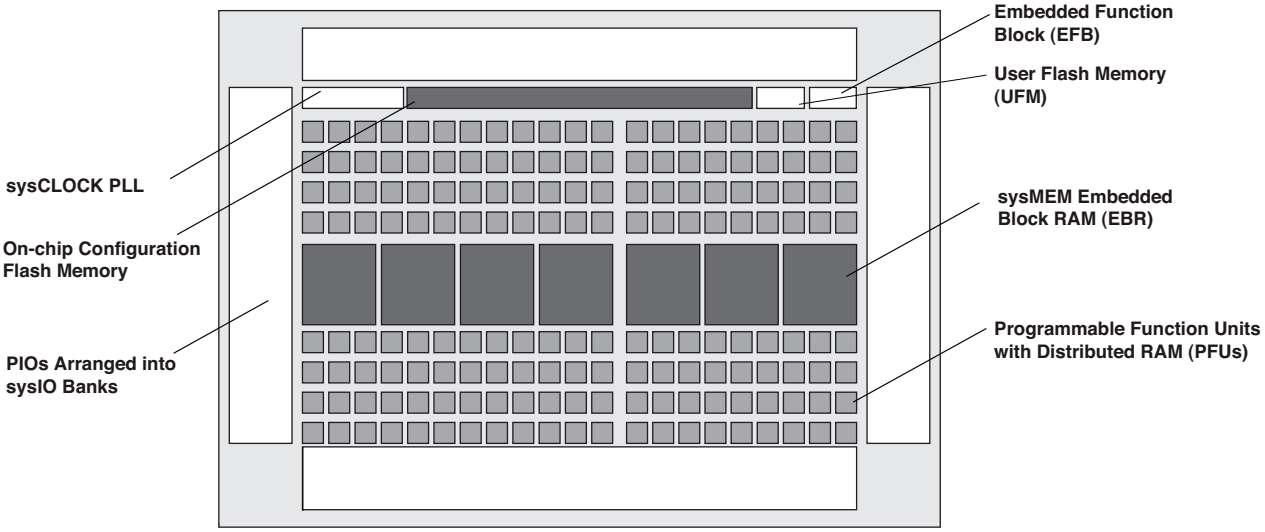
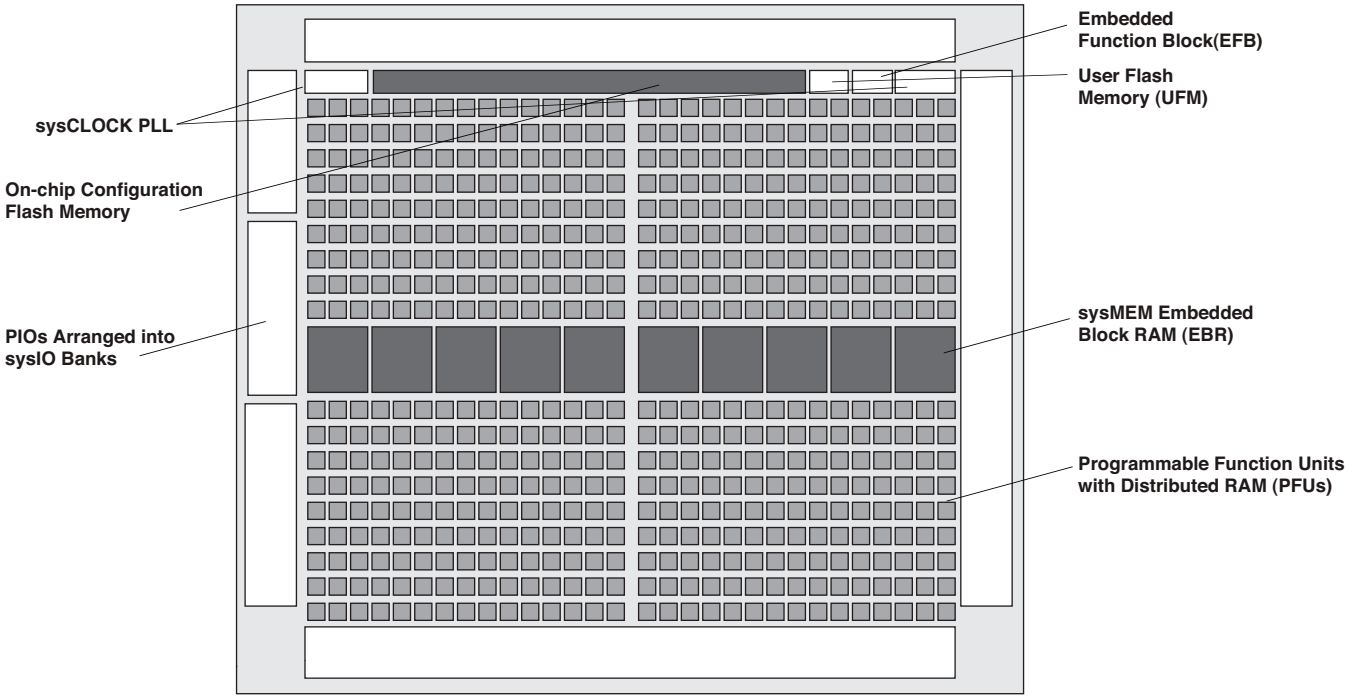


図 12-2. MachXO2-4000 デバイスの上面図



PFU は論理用、及び分散 RAM と ROM 用のビルディング・ブロックを含みます。いくつかの PFU は、分散 RAM 用でない論理用ビルディング・ブロックを提供します。本書では、PFU の EBR と分散 RAM の両方につ

いて、メモリの使用と実装を説明します。EBR及び分散RAMのハードウェア実装の詳細については″MachXO2 ファミリ・データシート ″を参照してください。

IPexpress の利用

設計者はIPexpressを利用すると、様々なデザインにおけるメモリの指定が容易にできます。これらのモジュールは、必要に応じて一般配線や LUT と共に 1 つ以上のメモリプリミティブを用いて構成できます。使用可能なプリミティブは次のとおりです。

- シングルポート RAM（RAM_DQ）～ EBR ベース
- 真の（True）デュアルポート RAM（RAM_DP_TRUE）～ EBR ベース
- 擬似（Pseudo）デュアルポート RAM（RAM_DP）～ EBR ベース
- リード専用メモリ （ROM）～ EBR ベース
- FIFO メモリ （FIFO_DC）～ EBR ベース
- 分散シングルポート RAM（Distributed_SPRAM）～ PFU ベース
- 分散デュアルポート RAM（Distributed_DPRAM）～ PFU ベース
- 分散 ROM（Distributed_ROM）～ PFU ベース
- RAM ベースのシフトレジスタ ～ PFU ベース

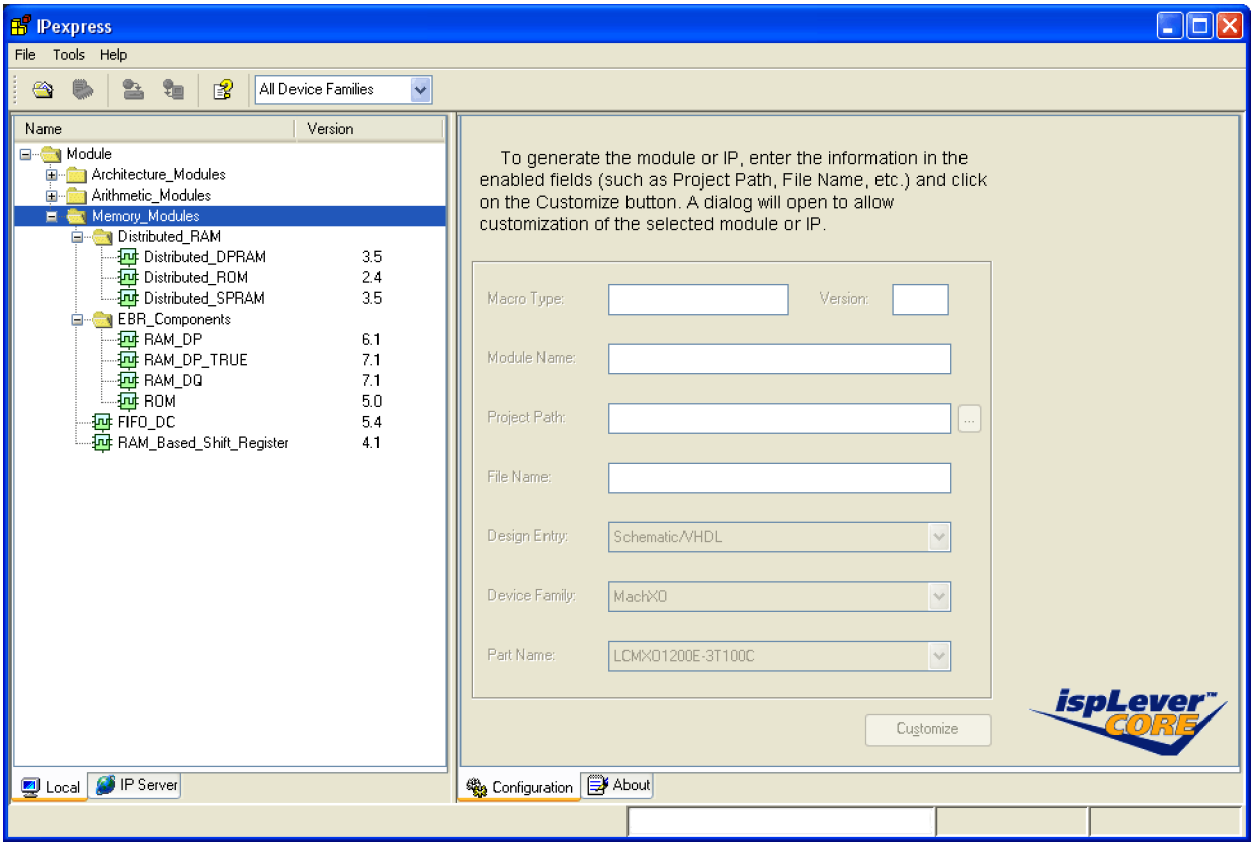
IPexpress フロー

これらのメモリのいずれかを生成するには、MachXO2 デバイスのプロジェクトを作成し（開き）ます。

プロジェクト・ナビゲータから、**Tools > IPexpress** の順に選択します。または、MachXO2 デバイスがプロジェクトの対象になったときに、ツールバーの IPexpress ボタンをクリックする方法もあります。

すると図 12-3 に示すような IPexpress ウィンドウが開きます。

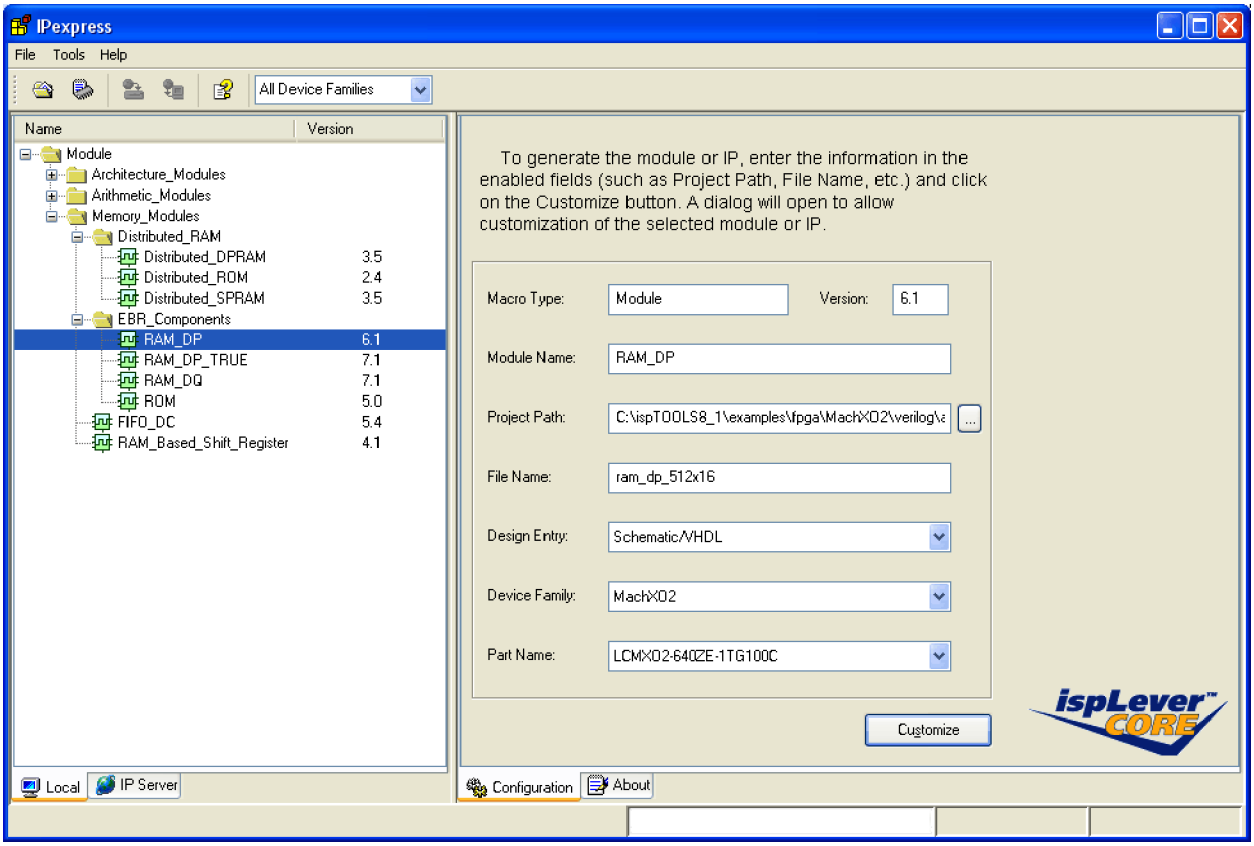
図 12-3. IPexpress ～ メインウィンドウ



このウィンドウの左枠にモジュールツリーがあります。図 12-3 に示すように、EBR ベースのメモリモジュールは **EBR_Components** に、PFU ベースの分散メモリモジュールは **Distributed_RAM** にそれぞれあります。

例として EBR ベースの擬似デュアルポート（サイズ 512×16）の生成を検討してみます。EBR_Components で RAM_DP を選択します。右枠内は図 12-4 に示すように変わります。

図 12-4. IPexpress を使用した擬似デュアルポート RAM（RAM_DP）の生成例



この右枠内において、**Macro Type** や **Module_Name** のようなオプションはデバイスと選択されたモジュールによって決まります。これらを IPexpress で変更することはできません。

Project Path の **Browse** ボタンをクリックすることによって、ユーザは生成されるモジュールが置かれるディレクトリを変更できます。

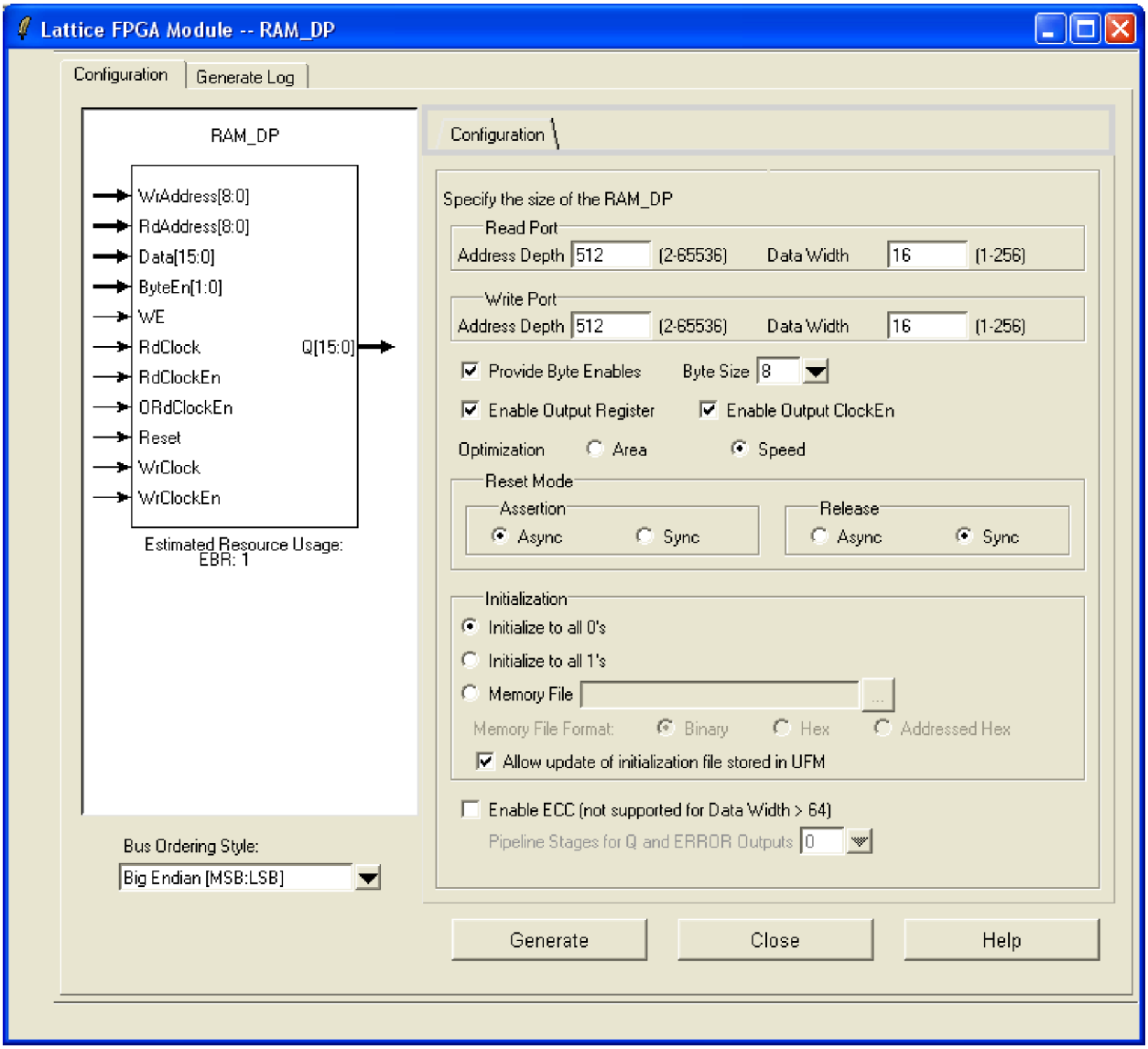
File Name テキストボックスには、ユーザは生成しようとするモジュールのエンティティ名を指定する必要があります。

Design entry（Verilog または VHDL）はデフォルトではプロジェクトタイプと同じです。プロジェクトが VHDL プロジェクトならば選択する設計入力オプションは VHDL となり、プロジェクトタイプが Verilog-HDL ならば Verilog-HDL となります。両方の HDL タイプでスケマティック対応も選択できます。

プロジェクト・ナビゲータからの起動時には、**Device Family** と **Part Name** のプルダウンメニューにはデータがデフォルトで入り、それらをユーザが変更することはできません。ただし IPexpress をスタンドアロン・アプリケーションとして起動した場合は、これらのメニューを使用してユーザはデバイスファミリ内（この例では MachXO2）の異なるデバイスを選択できます。

完了したら **Customize** ボタンをクリックします。これでユーザによる RAM のカスタマイズが可能な別ウィンドウが開きます（図 12-5）。

図 12-5. 擬似デュアルポート RAM（RAM_DP）モジュールのカスタマイズ生成の例



このウィンドウの左枠内はモジュールのブロック図を示します。右枠内には Configuration タブがあり、アドレスポート・サイズやデータ幅を指定するなど、ユーザはオプションを選択して RAM_DP をカスタマイズできます

ユーザは **Read Port** と **Write Port** にあるテキストボックスで、アドレスの深さとデータ幅を指定できます。この例ではサイズ 512×16 の擬似デュアルポートを生成します。ユーザは擬似デュアルポート RAM や真のデュアルポート RAM として、異なるポート幅で作成することもできます。

ハードウェアがサポートするのは、EBR ベース RAM ではクロック同期ライト動作のみであるため、入力データとアドレス制御は常にレジスタでサンプルされて取り込まれます。出力レジスタは EBR ベース RAM ではオプションであるため、チェックボックス **Enable Output Register**（出力レジスタのイネーブル）によって出力レジスタがリードデータポートに挿入されます。

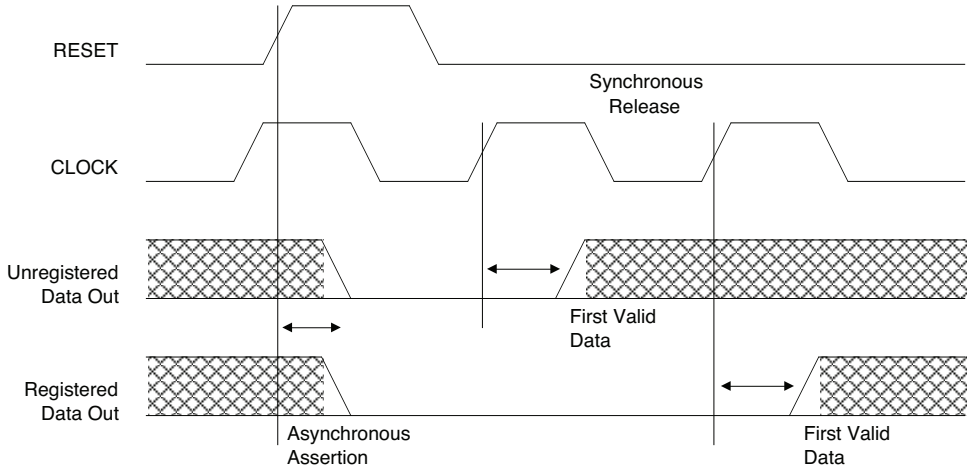
入力データとアドレスの信号にはクロックイネーブルのための制御が常に提供されます。出力レジスタがイネーブルされた場合は、別に **Output Clock Enables**（出力クロックイネーブル）を選択できます。

ユーザは Byte Enables（バイトイネーブル）の使用を指定できます。これを使用すると、メモリの特定バイトのみが上書きされるように入力データをマスクできます。マスクされたバイトは以前にライトされたデータのままになります。

ユーザはメモリの Reset Mode （リセットモード）をアサート及びネゲートの両方について指定できます。同期リセットにはクロックが必要で、リセット信号はセットアップ / ホールド時間要件をアサート及びネゲートの両エッジについて満たす必要があります。同期リセット時は CS レジスタもリセットされるため、ライト機能は自動的にディセーブルされますが、非同期リセット動作中はディセーブルされません。

非同期リセットは同期的にネゲートされるようにプログラム可能です。図 12-6 に示すように、同期ネゲートする場合、リセット後の最初のクロックエッジは非同期リセットを持つ全てのレジスタ、即ちデータ出力レジスタ、FIFO カウンタ及び FIFO フラグレジスタに対して内部リセットを解放します。

図 12-6. 同期ネゲートの非同期リセット



メモリはコンフィグレーション時にオール 1 やオール 0 に初期化できます。（使用できる）UFM ビット数を最大にするためには、EBR をオール 0 のパターンで初期化します。この場合は UFM ビットを使い切ることにはなりません。ユーザは **Memory File**（メモリファイル、”.mem”）で指定した内容でメモリを初期化することもできます。このファイルを RAM に用意することは任意ですが、ROM の場合、メモリ初期化ファイルは必須です。これらのファイルに可能なフォーマットはバイナリ、16 進またはアドレス付き 16 進 (Addressed Hex) です。その詳細については、本書の初期化ファイルのセクションで説明します。

従来のメモリ初期化ファイルは固定で、デバイスのコンフィグレーション・ビットストリームに組み込まれます。一方 MachXO2 アーキテクチャでは、メモリ初期化データを、ユーザによるアクセスや動的な変更が可能な UFM に格納できます。この機能を有効にするには、**Allow Update of initialization file stored in UFM** (UFM に格納した初期化ファイルの更新を可能にする）を選択します。この機能の詳細については ”TN1205, Using User Flash Memory and Hardened Control Functions in MachXO2 Devices（MachXO2 デバイスのユーザフラッシュメモリとハードマクロ制御機能の使用法）” を参照してください。

ここでユーザは **Generate** ボタンをクリックし、カスタマイズされたモジュールを生成できます。次に VHDL または Verilog ネットリストが生成され、指定した場所に配置されます。ユーザはこのネットリストをデザインに取り込むことができます。また同時に生成されるのはインスタンス化テンプレートファイル（*_tmpl.v または .vhd）、ラティスパラメータ・ファイル（*.lpc）、テストベンチ・テンプレートファイル（tb_*_tmpl.v または .vhd）、及び 2 つのログファイル（*_generate.log、*.srp）です。最後に、設計入力タイプが Schematic/VHDL または Schematic/Verilog であれば、スケマティック・シンボルファイル（*.sym）も作成されます。

モジュールが生成されたら、ユーザは *.lpc または Verilog-HDL/VHDL ファイルのいずれかをデザインの最上位モジュールでインスタンス化できます。

メモリモジュールの ECC

IPexpress を使用すると、ユーザは ECC（Error Check Code、エラーチェックコード）を EBR ベースのメモリモジュールに実装できます。モジュールの Configuration タブに ECC を有効にするためのチェックボックスがあります。

ECC の使用を選択した場合、2 ビットエラー信号とエラーコードは次のようになります。

- Error[1:0] = “00” ～ エラーがないことを示す
- Error[1:0] = “01” ～訂正された 1 ビットエラーが検出されたことを示す
- Error[1:0] = “10” ～訂正ができない 2 ビットエラーが検出されたことを示す
- Error[1:0] = “11” ～ 未使用

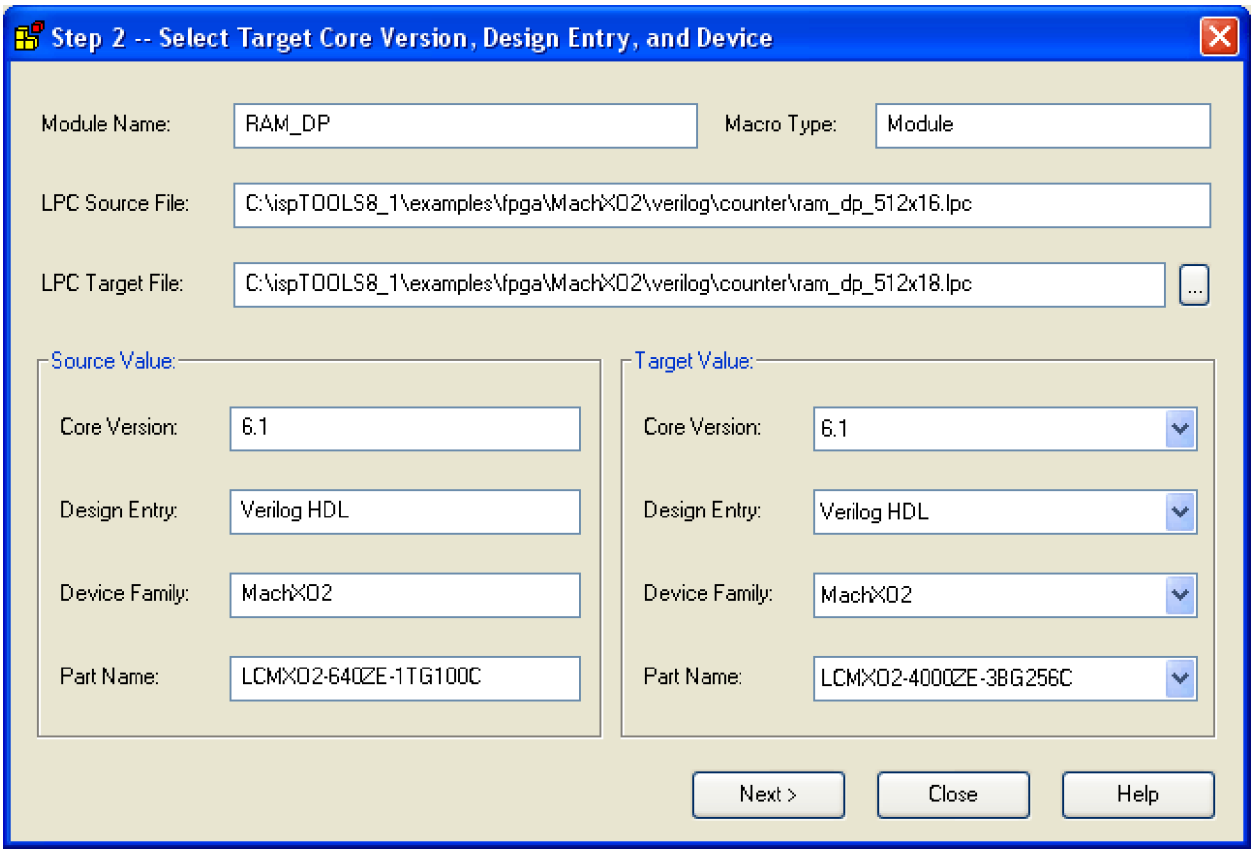
IP 再生成・変更

以前に生成したモジュールを再生成または変更すると便利なことがあります。カスタマイズしたモジュールまたは IP を再生成することによって、デバイスタイプ、設計入力方法、及びモジュール固有のオプションなどの設定のいくつかを変更できます。また古いモジュールまたは IP を最新バージョンに更新することもできます。IPexpress のメインウィンドウから、**Tools > Regenerate IP/Module** の順に選択します。

Select a Parameter File パラメータファイルを選択）ダイアログボックスで、生成したいモジュールまたは IP の *.lpc（Lattice Parameter Configuration）ファイルを選択して、**Open** をクリックします。

これで図 12-7 に示すようなダイアログボックスが開きます。

図 12-7. IP の再生成 / 変更の例



Select Target Core Version, Design Entry, and Device （ターゲット・コアバージョン、デザインエントリ、及びデバイスを選択）ダイアログボックスには、**Source Value**（ソースの値）ボックスに現在のモジュールまたは IP の設定が表示されます。新しい設定は **Target Value**（ターゲットの値）ボックスで確認します。

新しいファイルセットを新しい場所に生成する場合は、その場所を LPC **Target File** ボックスで設定します。.lpc ファイル名の基部が、全ての新規ファイル名の基部になります。LPC Target File に指定するファイルは、末尾が拡張子 .lpc になっている必要があります。

Next をクリックし、前回と同様にモジュールのカスタマイズに進みます。

以降のセクションでは、各種のメモリモジュール（EBR と分散メモリの両方）について詳細に説明します。

PMI の利用

PMI（Parameterizable Module Instantiation）を使用すると、経験豊富なユーザは GUI を使用せずに、ispLEVER プロジェクト・ナビゲータから構成可能なメモリモジュールをオンザフライで利用できます。

必要なパラメータ及び制御信号は Verilog または VHDL のいずれかで設定可能です。トップレベル・デザインは定義されたメモリパラメータと宣言された信号を含みます。その後でインターフェイスは論理合成でブラックボックスを自動的に生成でき、ispLEVER はネットリストをオンザフライで生成できます。ラティスメモリは業界標準メモリと同一なため、各モジュールのパラメータは、オンラインヘルプ・システムを通して入手できる一般のメモリ関連ガイドからも得られます。

PMI モジュールは他のモジュールが HDL で行われるのと同じ方法でインスタンス化されます。そのプロセスは IPexpress プロセスと同様ですが、モジュールをカスタマイズするためにパラメータを設定することが追加されています。ispLEVER ソフトウェアでは、カスタマイズされたモジュールのポート及びパラメータを指定する Verilog または VHDL のインスタンス化コマンド用にテンプレートが用意されています。詳細については、ispLEVER オンラインヘルプのセクション ″Instantiating a PMI Module (PMI モジュールのインスタンス化) ″ を参照してください。

メモリモジュールの推論

最後に、メモリは Verilog または VHDL モジュール内から推論（inferencing）によって使用することも可能です。メモリ推論のための HDL 構造は論理合成ベンダによって異なります。適正な推論構造と属性設定については、論理合成エンジンのベンダから提供される資料を参照してください。

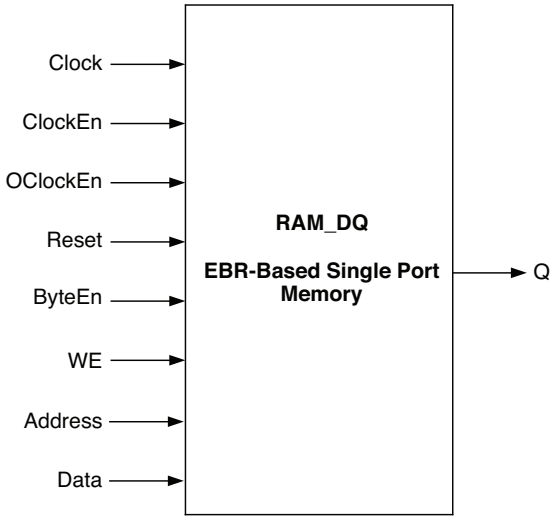
IPexpress メモリモジュール

シングルポート RAM（RAM_DQ）～ EBR ベース

MachXO2 デバイスの EBR ブロックは、シングルポート RAM（RAM_DQ）として構成可能です。ユーザは設計要件に従ったメモリサイズの Verilog-HDL または VHDL ネットリストを IPexpress で生成できます。

IPexpress は図 12-8 に示すようなメモリモジュールを生成します。

図 12-8. IPexpress によって生成されるシングルポート・メモリモジュール



デバイスには多数の EBR ブロックがあるため、生成されるモジュールはこれらの EBR ブロックつまりプリミティブを活用し、それらをカスケード接続して IPexpress の GUI でユーザが指定したメモリサイズを作成します。1 つの EBR ブロックよりも小さいメモリサイズの場合、モジュールは 1 つの EBR ブロックで作成されます。1 つの EBR ブロックよりも大きいメモリサイズの場合は、そのサイズの作成に必要な深さまたは幅で、複数の EBR ブロックをカスケード接続します。

シングルポート RAM モードでは、そのポートの入力データ及びアドレスはメモリアレイの入力でレジスタに取り込まれます。メモリの出力データは必要に応じて出力でレジスタを介して出力されます。

シングルポートメモリの各種ポート及びその定義を表 12-1 に示します。

表 12-1. EBR ベース・シングルポートメモリのポート定義

生成されたモジュールのポート名	記述	アクティブステート
Clock	クロック	クロック立ち上がりエッジ
ClockEn[1]	クロックイネーブル	アクティブ High
*OClockEn[2]	出力クロックイネーブル	アクティブ High
Reset[3]	リセット	アクティブ High
*ByteEn[4]	バイトイネーブル	アクティブ High
WE	ライトイネーブル	アクティブ High
Address	アドレスバス	–
Data	データ入力	–
Q	データ出力	–
*ERROR	ECC	アクティブ High

* 印はオプションのポートを示す。
1. ClockEn は全入力レジスタのクロックイネーブルとして使用
2. OClockEn は出力レジスタのクロックイネーブルとして使用できる。これにより最終ワードを含み、データ出力のフルパイプライン化が可能
3. Reset は RAM のオプションの出力レジスタをリセットするのみ。入力レジスタやメモリの内容をリセットはしない
4. ByteEn は特定のバイトのみを書き換えるために入力データをマスクできる

シングルポート RAM（RAM_DQ）は **NORMAL**、**READ BEFORE WRITE**（リードビフォーライト）、**WRITE THROUGH**（ライトスルー）の各モードで構成可能です。これらのモードによって、同一メモリロケーションへのライト動作とその後に続くリード動作時に、メモリの出力ポート Q からどのようなデータが出力されるかが変化します。

IPexpress は適切に構成された DP8KC プリミティブを使用して MachXO2 シングルポート RAM を実装します。

図 12-9 から 12-14 にシングルポート RAM の内部タイミング波形を示します。

図 12-9. シングルポート RAM のタイミング波形 ～ NORMAL モード、出力レジスタなし

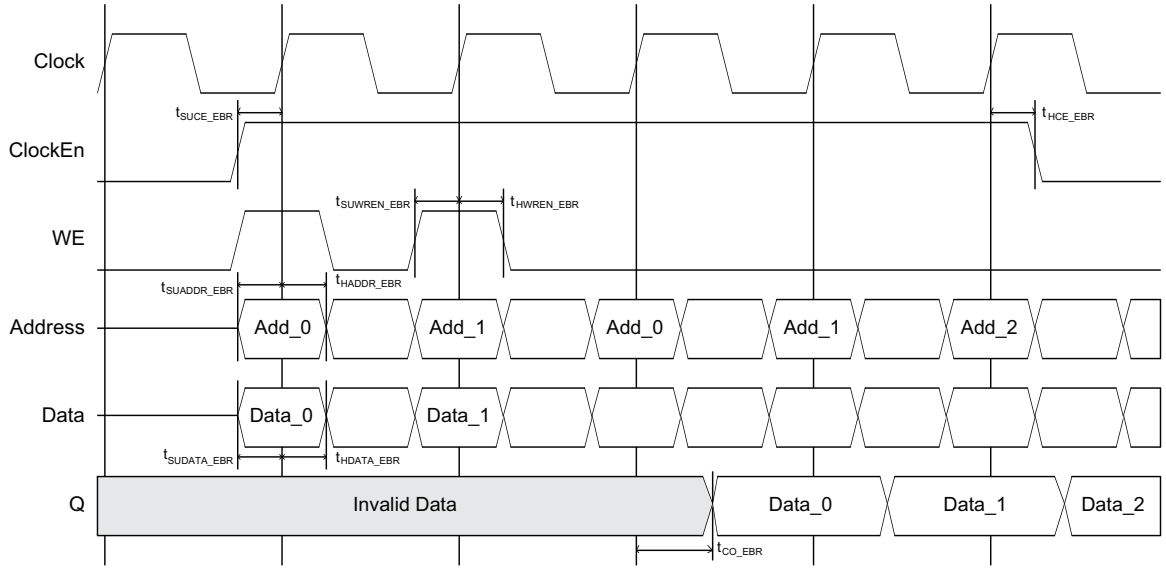


図 12-10. シングルポート RAM のタイミング波形 ～ NORMAL モード、出力レジスタあり

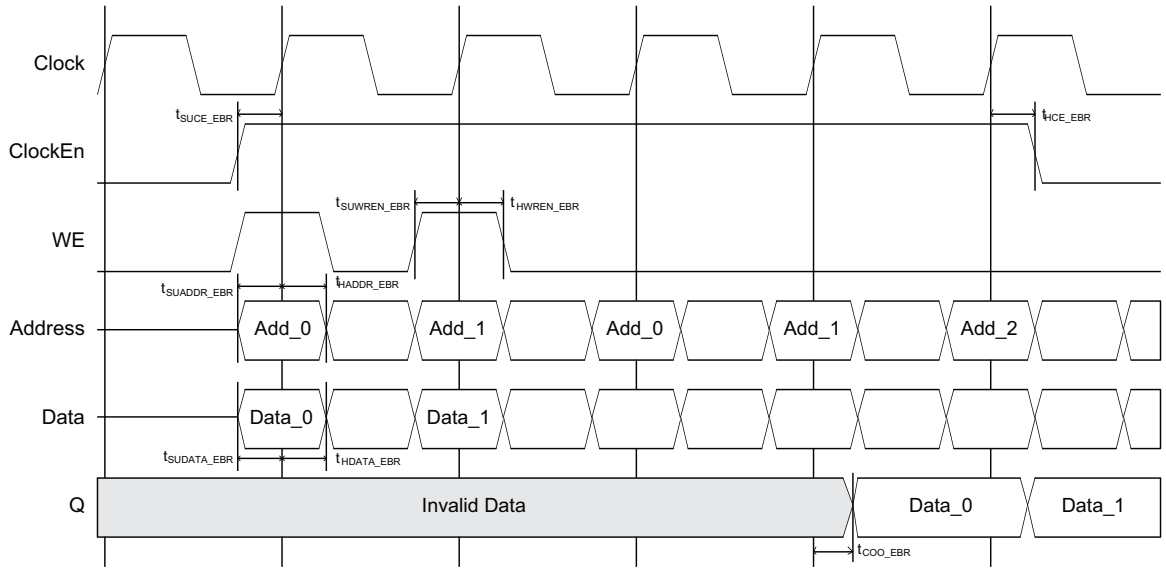


図 12-11. シングルポート RAM のタイミング波形 ～ READ BEFORE WRITE モード、出力レジスタなし

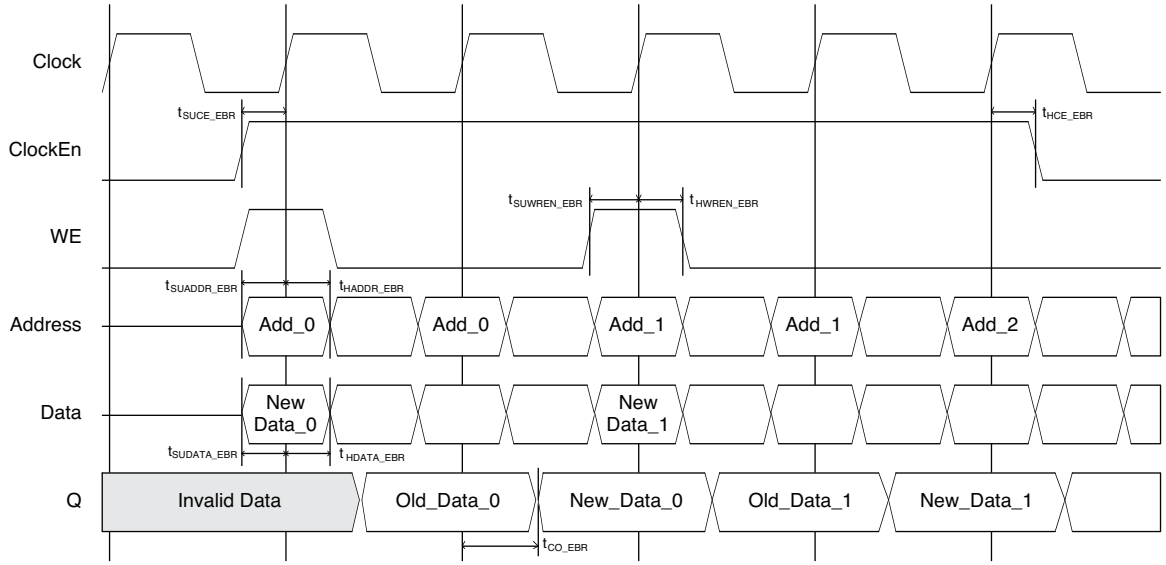


図 12-12. シングルポート RAM のタイミング波形 ～ READ BEFORE WRITE モード、出力レジスタあり

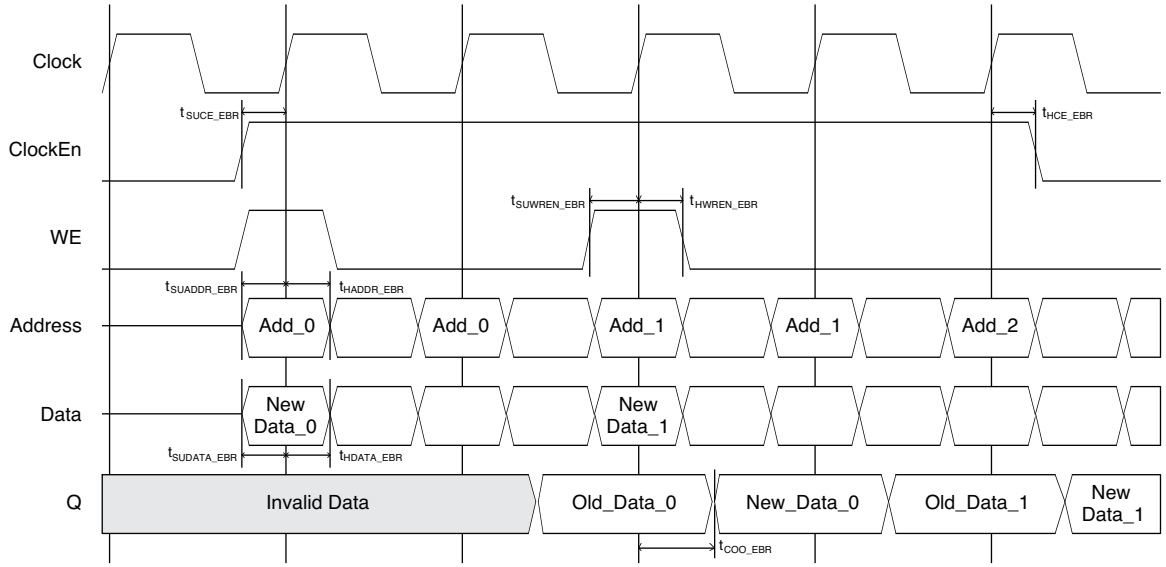


図 12-13. シングルポート RAM のタイミング波形 ～ WRITE THROUGH モード、出力レジスタなし

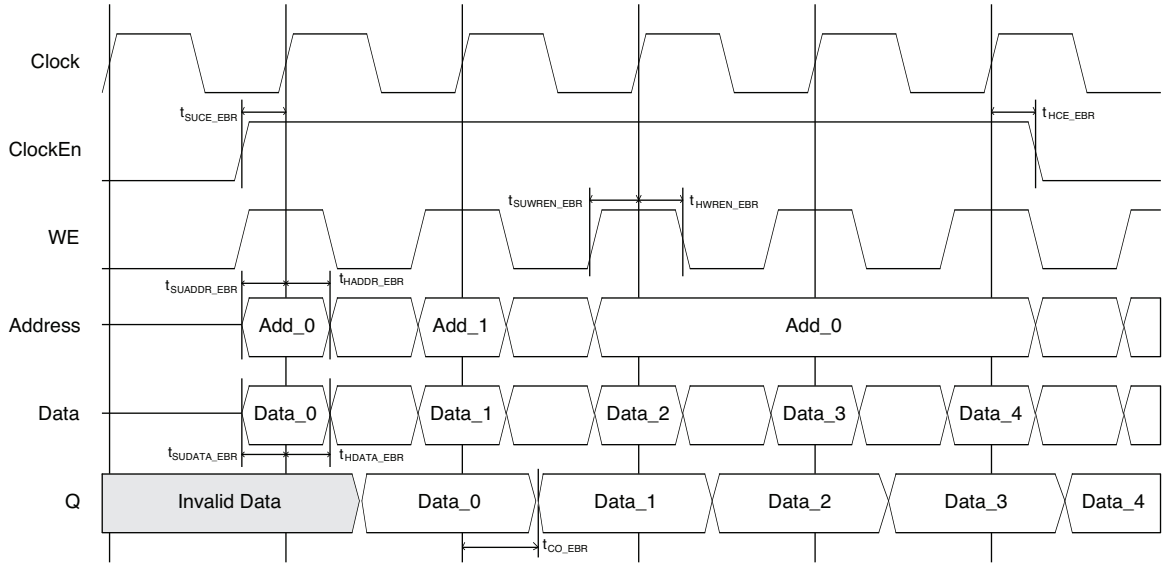
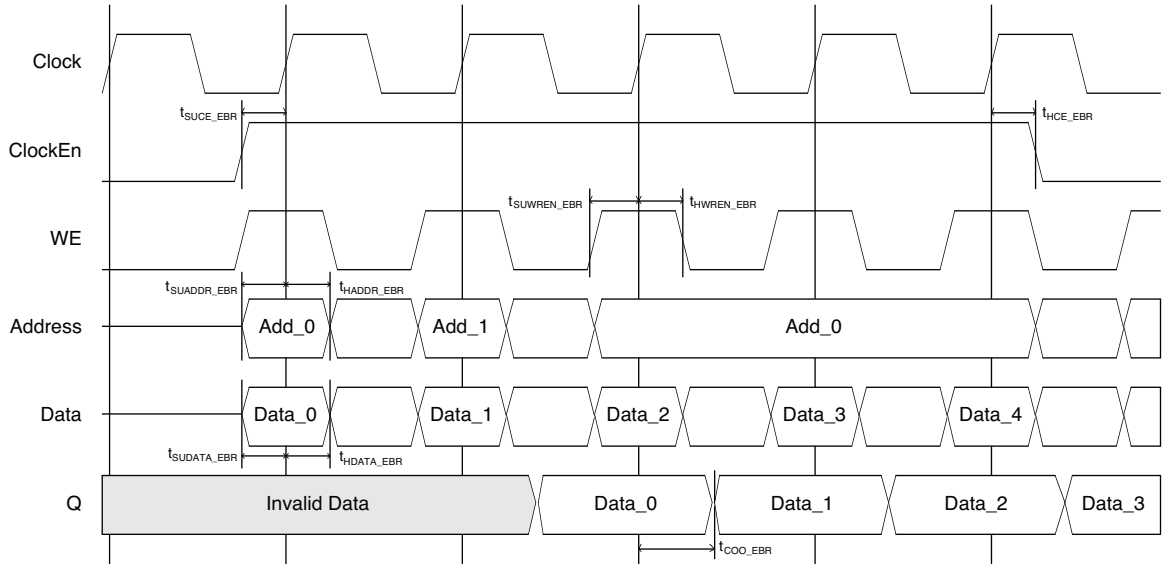


図 12-14. シングルポート RAM のタイミング波形 ～ WRITE THROUGH モード、出力レジスタあり

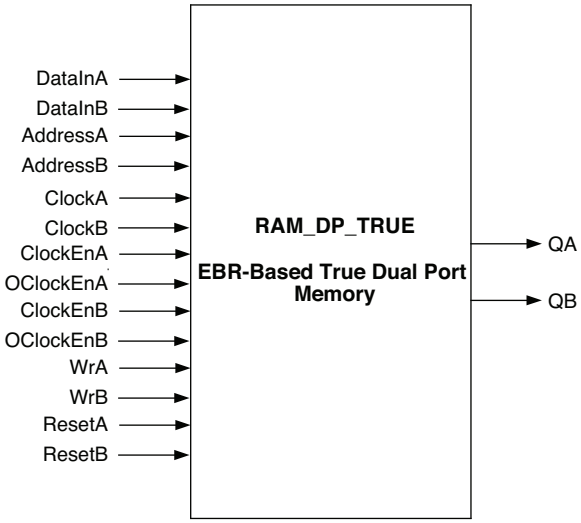


デュアルポート RAM（RAM_DP_TRUE）～ EBR ベース

MachXO2 デバイスの EBR ブロックは、真のデュアルポート RAM（RAM_DP_TRUE）として構成可能です。ユーザは設計要件に応じた各種メモリサイズの Verilog-HDL または VHDL ネットリストを IPexpress で生成できます。

IPexpress は図 12-15 に示すようなメモリモジュールを生成します。

図 12-15. IPexpress によって生成される真のデュアルポート・メモリモジュール



デバイスには多数の EBR ブロックがあるため、生成されるモジュールはこれらの EBR ブロックつまりプリミティブを活用し、それらをカスケード接続して IPexpress GUI でユーザが指定したメモリサイズを生成します。1 つの EBR ブロックよりも小さいメモリサイズの場合、モジュールは 1 つの EBR ブロックで作成されます。1 つの EBR ブロックよりも大きいメモリサイズの場合は、そのサイズの作成に必要な深さまたは幅で、複数の EBR ブロックをカスケード接続します。

真のデュアルポート RAM モードでは、そのポートの入力データ及びアドレスはメモリアレイの入力でレジスタに取り込まれます。メモリの出力データは必要に応じて出力レジスタを介して出力されます。

真のデュアルポートメモリの各種ポート及びその定義を表 12-2 に示します。

表 12-2. EBR ベース真のデュアルポートメモリのポート定義

生成されたモジュールのポート名	記述	アクティブステート
DataInA, DataInB	入力データ、ポート A およびポート B	–
AddressA, AddressB	アドレスバス、ポート A およびポート B	–
ClockA, ClockB	クロック、ポート A およびポート B	クロック立ち上がりエッジ
ClockEnA, ClockEnB[1]	クロックイネーブル、ポート A およびポート B	アクティブ High
*OClockEnA, *OClockEnB[2]	出力クロックイネーブル、ポート A およびポート B	アクティブ High
WrA, WrB	ライトイネーブル、ポート A およびポート B	アクティブ High
ResetA, ResetB[3]	リセット、ポート A およびポート B	アクティブ High
QA, QB	出力データ、ポート A およびポート B	–
*ByteEnA, *ByteEnB[4]	バイトイネーブル、ポート A およびポート B	アクティブ High
*ERROR	ECC	アクティブ High

* 印はオプションのポートを示す
1. ClockEnA/B は全入力レジスタのクロックイネーブルとして使用
2. OClockEnA/B は出力レジスタのクロックイネーブルとして使用できる。これにより最終ワードを含み、データ出力のフルパイプライン化が可能
3. Reset は RAM のオプションの出力レジスタをリセットするのみ。入力レジスタやメモリの内容をリセットはしない
4. ByteEnA/B は特定のバイトのみを書き換えるために入力データをマスクできる

真のデュアルポート RAM（RAM_DP_TRUE）は NORMAL、READ BEFORE WRITE、WRITE THROUGH モードで構成可能です。これらのモードによって、同一メモリロケーションへのライト動作とその後に続くリード動作時に、メモリのポート Q からどのようなデータが出力されるかが変化します。

IPexpress は DP8KC プリミティブを使用して MachXO2 デュアルポート RAM を実装します。

図 12-16 から 12-21 にデュアルポート RAM の内部タイミング波形を示します。

図 12-16. 真のデュアルポート RAM のタイミング波形 ～ NORMAL モード、出力レジスタなし

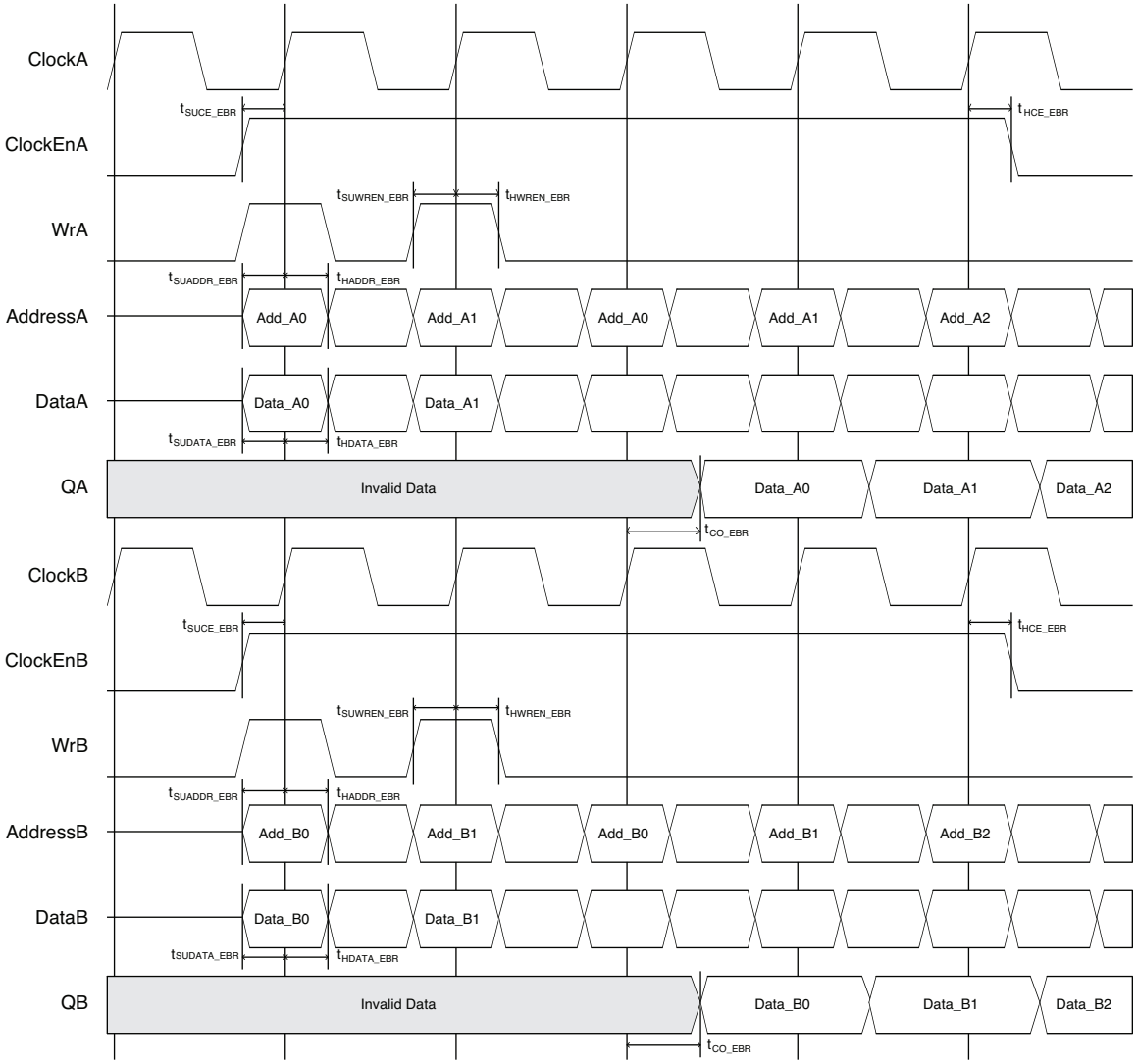


図 12–17. 真のデュアルポート RAM のタイミング波形 ～ NORMAL モード、出力レジスタあり

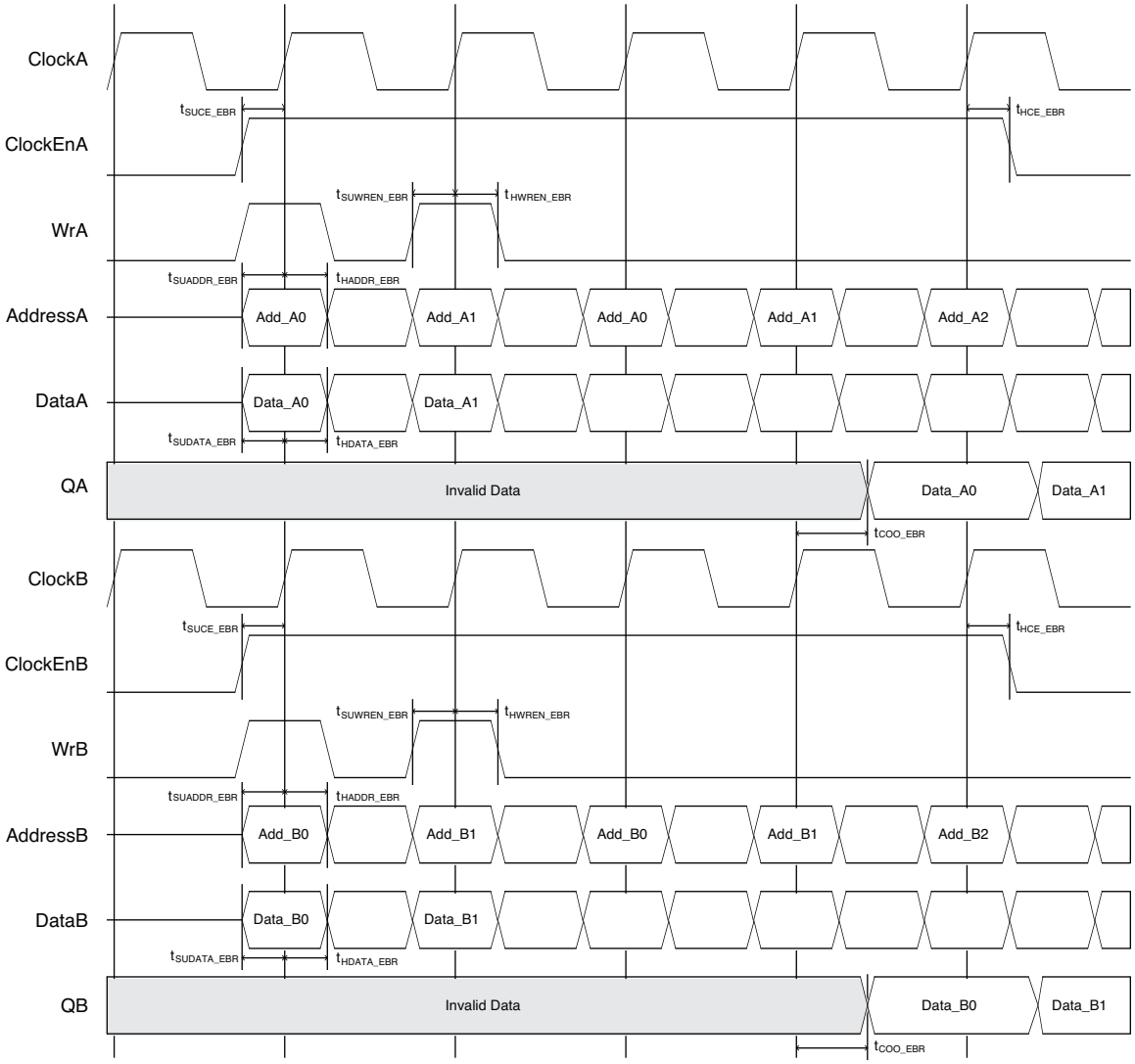


図 12–18. 真のデュアルポート RAM のタイミング波形 ～ READ BEFORE WRITE モード、出力レジスタなし

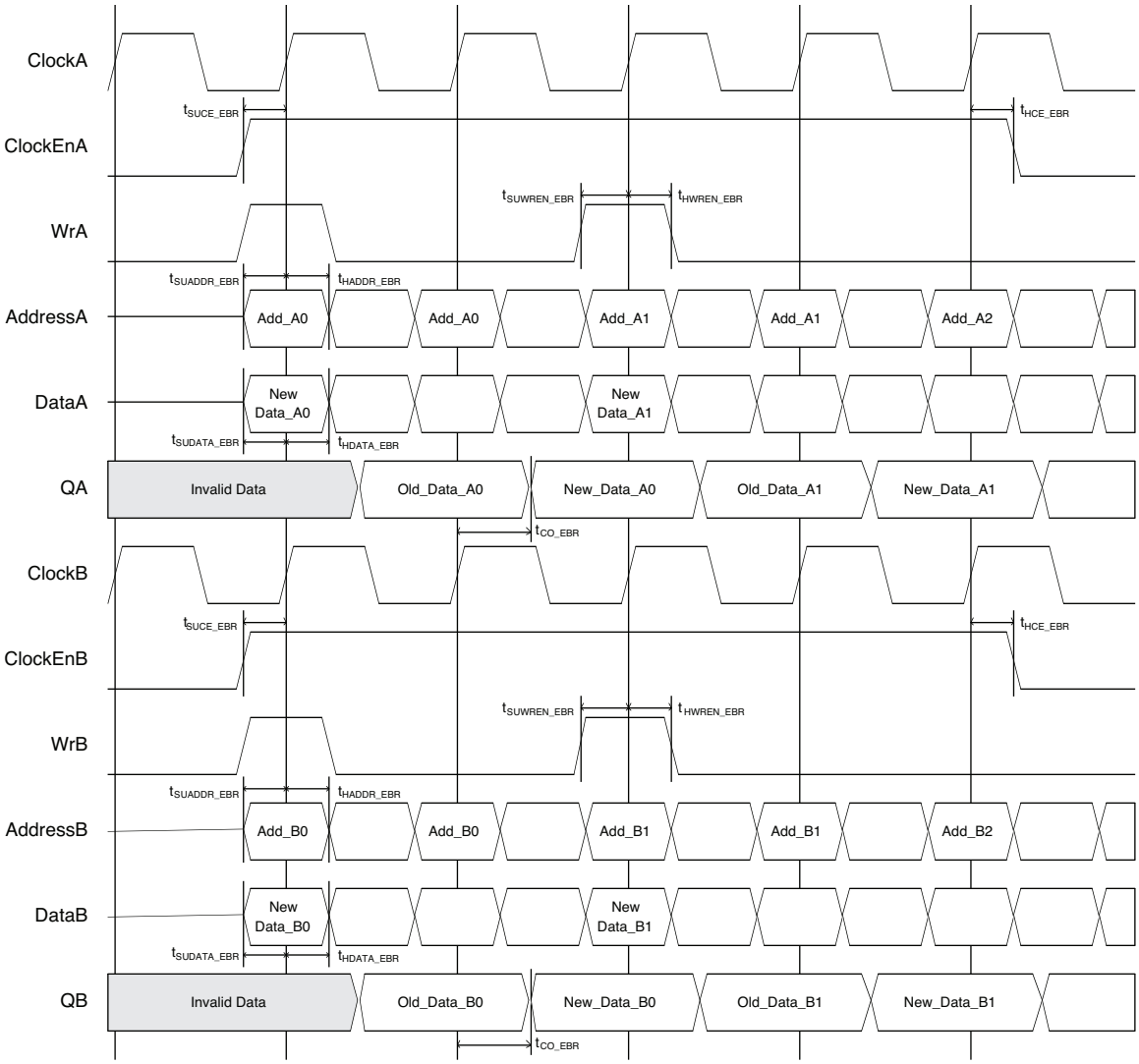


図 12–19. 真のデュアルポート RAM のタイミング波形 ～ READ BEFORE WRITE モード、出力レジスタあり

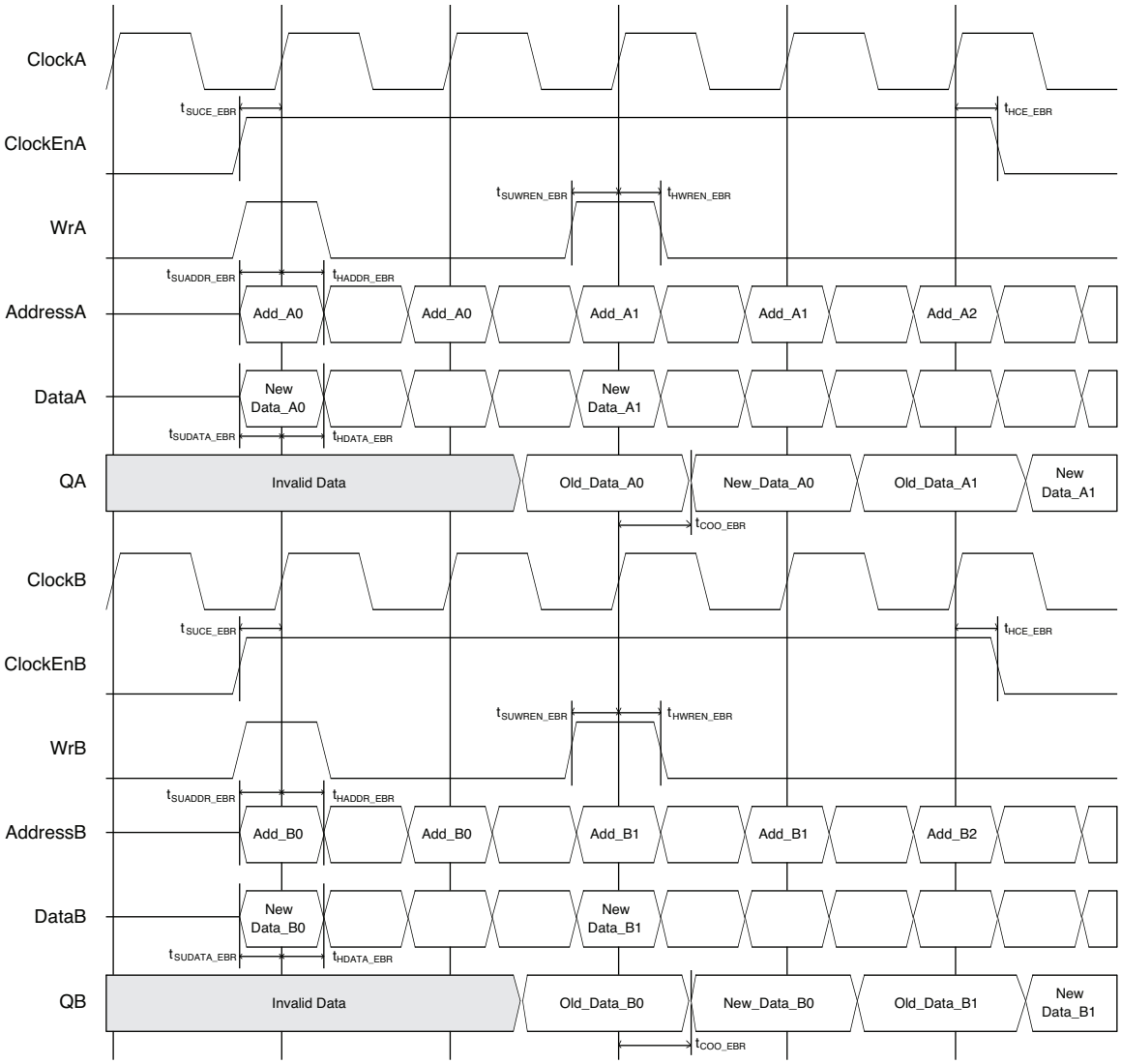


図 12-20. 真のデュアルポート RAM のタイミング波形 ～ WRITE THROUGH モード、出力レジスタなし

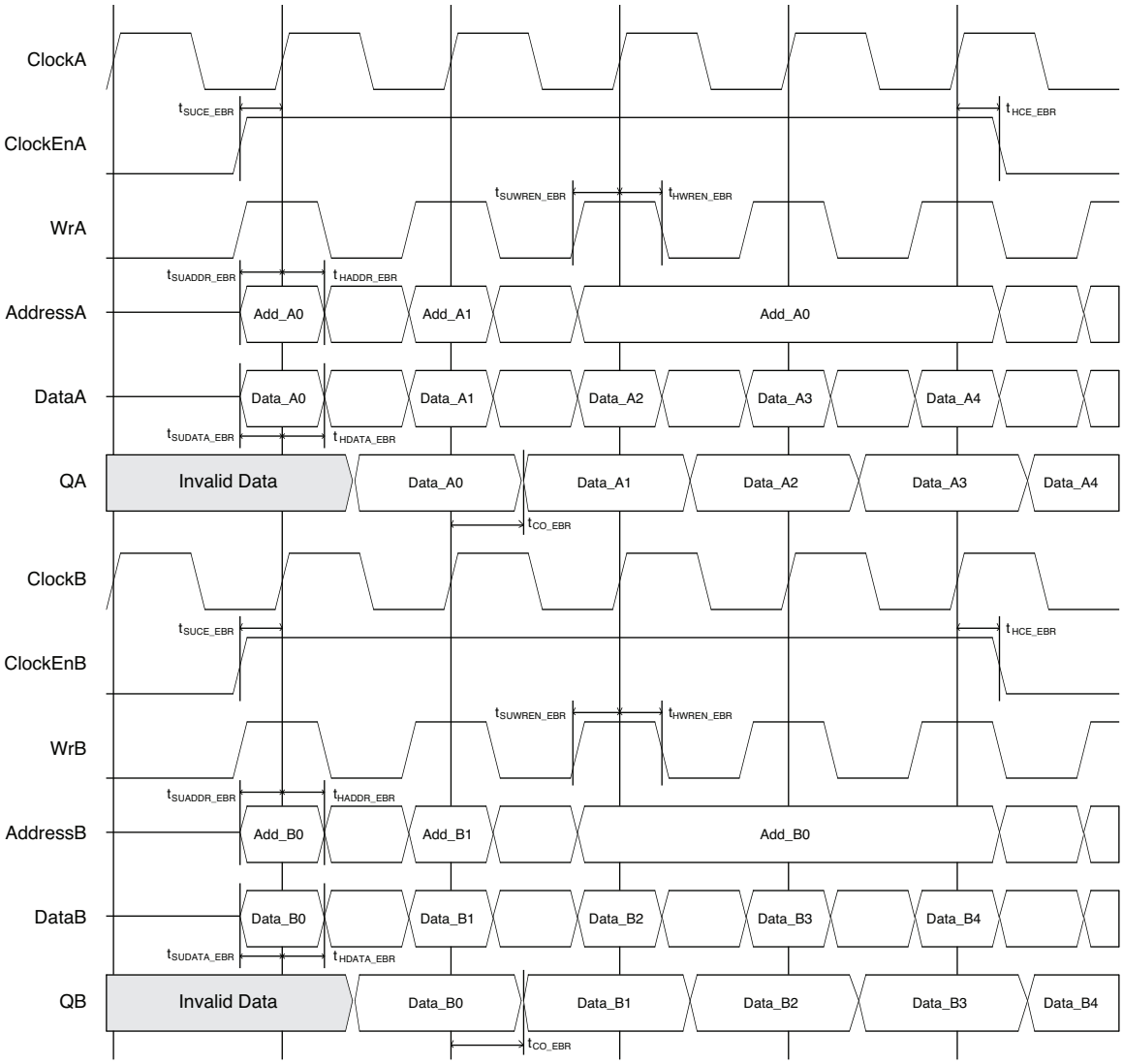
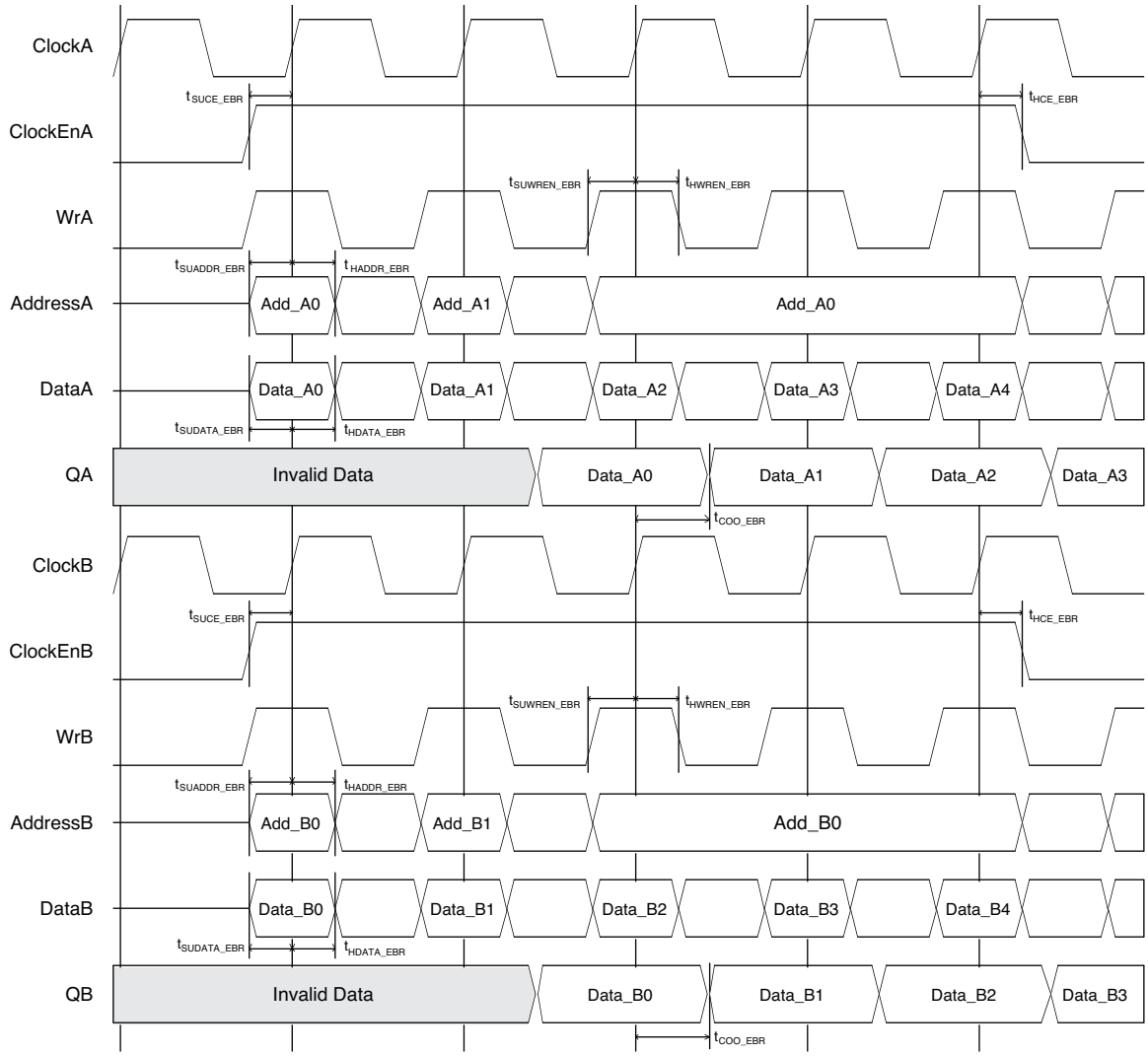


図 12-21. 真のデュアルポート RAM のタイミング波形 ～ WRITE THROUGH モード、出力レジスタあり

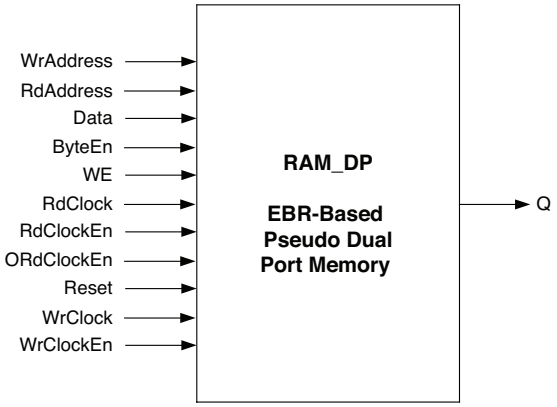


擬似デュアルポート RAM（RAM_DP）～ EBR ベース

MachXO2 デバイスの EBR ブロックは、擬似デュアルポート RAM（RAM_DP）として構成可能です。ユーザは設計要件に応じて、各種メモリサイズの Verilog-HDL または VHDL ネットリストを IPexpress で生成できます。

IPexpress は図 12-22 に示すようなメモリモジュールを生成します。

図 12-22. IPexpress によって生成される擬似デュアルポート・メモリモジュール



デバイスには多数の EBR ブロックがあるため、生成されるモジュールは、これらの EBR ブロックつまりプリミティブを活用し、それらをカスケード接続して IPexpress の GUI でユーザが指定したメモリサイズを作成します。1 つの EBR ブロックよりも小さいメモリサイズの場合、モジュールは 1 つの EBR ブロックで作成されます。1 つの EBR ブロックよりも大きいメモリサイズの場合は、そのサイズの作成に必要な深さまたは幅で、複数の EBR ブロックをカスケード接続します。

擬似デュアルポート RAM モードでは、そのポートの入力データ及びアドレスはメモリアレイの入力でレジスタに取り込まれます。メモリの出力データは必要に応じて出力でレジスタを介して出力されます。

擬似デュアルポートメモリの各種ポート及びその定義を表 12-3 に示します。

表 12-3. EBR ベースの擬似デュアルポートメモリのポート定義

生成されたモジュールのポート名	記述	アクティブステート
WrAddress	ライトアドレス	–
RdAddress	リードアドレス	–
Data	ライトデータ	–
*ByteEn[1]	バイトイネーブル	アクティブ High
WE	ライトイネーブル	アクティブ High
RdClock	リードクロック	立ち上がりエッジ
RdClockEn[2]	リードクロック・イネーブル	アクティブ High
*ORdClockEn[3]	リード出力クロックイネーブル	アクティブ High
Reset[4]	リセット	アクティブ High
WrClock	ライトクロック	立ち上がりエッジ
WrClockEn[2]	ライトクロック・イネーブル	アクティブ High
Q	リードデータ	–
*ERROR	ECC	アクティブ High

* 印はオプションのポートを示す
1. ByteEn は特定のバイトのみを書き換えるために入力データをマスクできる
2. RdClockEn/WrClockEn は全入力レジスタのクロックイネーブルとして使用
3. ORdClockEn は出力レジスタのクロックイネーブルとして使用できる。これにより最終ワードを含み、データ出力のフルパイプライン化が可能
4. Reset は RAM のオプションの出力レジスタをリセットするのみ。入力レジスタやメモリの内容をリセットはしない

IPexpress は PDPW8KC プリミティブ、または DP8KC プリミティブをビット幅の狭いデータポート（9 ビット以下）構成で使用して、MachXO2 擬似デュアルポート RAM（RAM_DP）を実装します。

図 12-23 と 12-24 に擬似デュアルポート RAM の内部タイミング波形を示します。

図 12-23. 擬似デュアルポート RAM のタイミング波形 ～ 出力レジスタなし

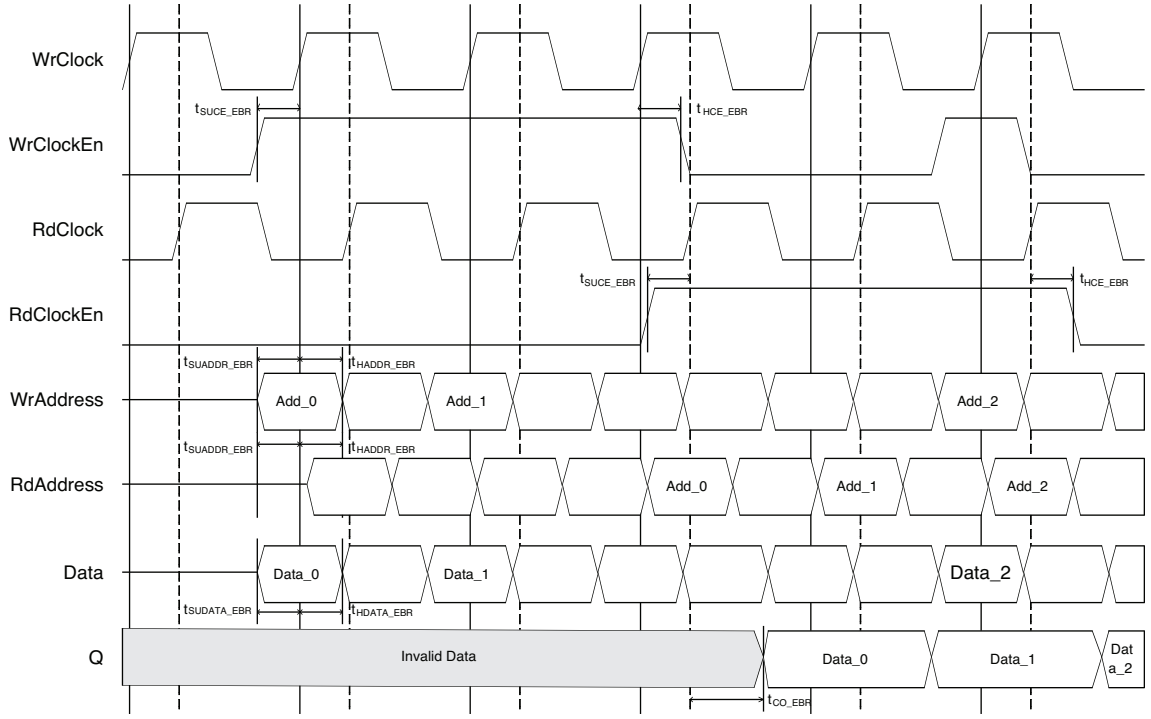
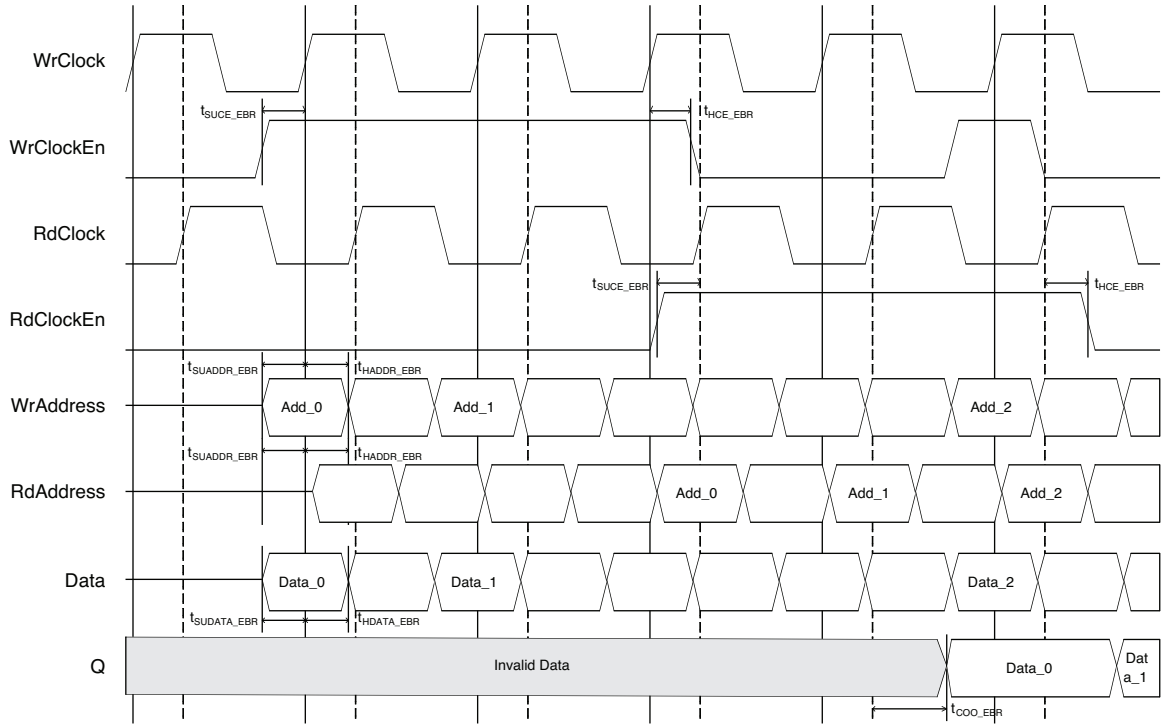


図 12-24. 擬似デュアルポート RAM のタイミング波形 ～ 出力レジスタあり

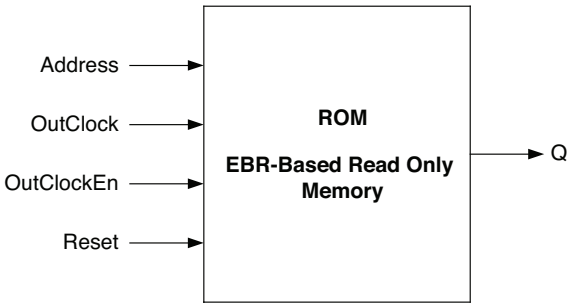


ROM ～ EBR ベース

MachXO2 デバイスの EBR ブロックは ROM（Read Only Memory）として構成可能です。ユーザは設計要件に応じた各種メモリサイズの Verilog-HDL または VHDL ネットリストを IPexpress で生成できます。ユーザは、ROM メモリの内容を初期化ファイルの形で指定する必要があります。

IPexpress は図 12-25 に示すようなメモリモジュールを生成します。

図 12-25. IPexpress によって生成される ROM モジュール



デバイスには多数の EBR ブロックがあるため、生成されるモジュールはこれらの EBR ブロック（プリミティブ）を活用し、それらをカスケード接続して IPexpress の GUI でユーザが指定したメモリサイズを作成します。1 つの EBR ブロックよりも小さいメモリサイズの場合、モジュールは 1 つの EBR ブロックで作成されます。1 つの EBR ブロックよりも大きいメモリサイズの場合は、そのサイズの作成に必要な深さまたは幅で、複数の EBR ブロックをカスケード接続できます。

ROM の各種ポート及びその定義を表 12-4 に示します。

表 12-4. EBR ベース ROM のポート定義

生成されたモジュールのポート名	記述	アクティブステート
Address	リードアドレス	–
OutClock	クロック	クロック立ち上がりエッジ
OutClockEn[1]	クロックイネーブル	アクティブ High
Reset[2]	リセット	アクティブ High
Q	リードデータ	–
*ERROR	ECC	アクティブ High

* 印はオプションのポートを示す
1. **OutClockEn** はオプションの出力レジスタのクロックイネーブルとして使用できる。
2. **Reset** は RAM のオプションの出力レジスタをリセットするのみ。入力レジスタやメモリの内容をリセットはしない

IPexpress を使用して ROM を生成するとき、ユーザは ROM の内容を事前に初期化するために初期化ファイルを指定する必要があります。これらのファイルは *.mem ファイルで、可能なフォーマットはバイナリ、16 進またはアドレス付き 16 進です。初期化ファイルについては、本書のメモリ初期化のセクションで詳しく説明します。

IPexpress は適切に構成された DP8KC プリミティブを使用して、MachXO2 の ROM を実装します。

図 12-26 と 12-27 に ROM の内部タイミング波形を示します。

図 12-26. ROM タイミング波形 ～ 出力レジスタなし

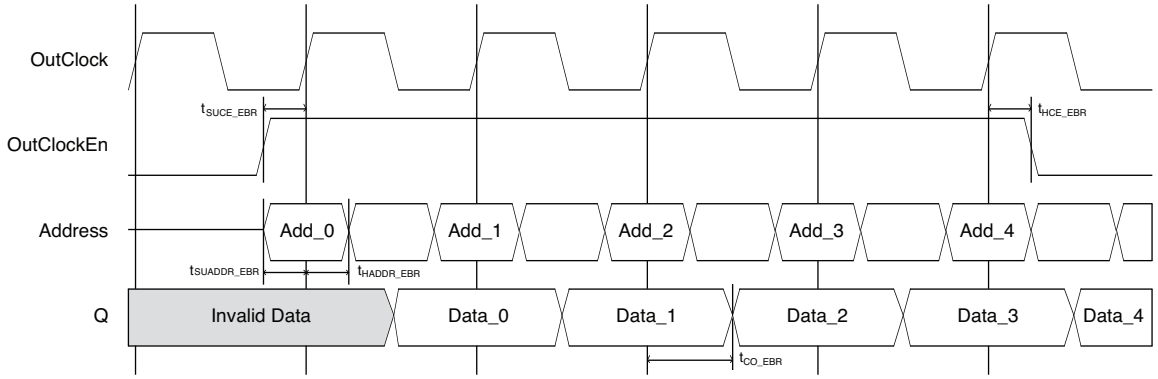
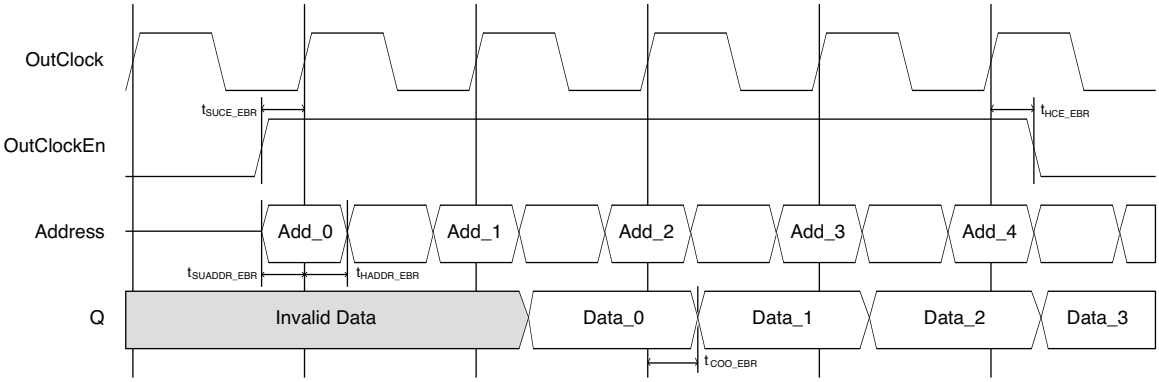


図 12-27. ROM タイミング波形 ～ 出力レジスタあり

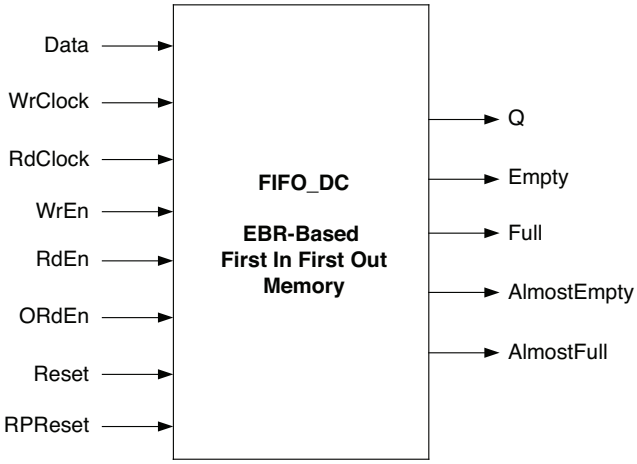


FIFO（FIFO_DC）～ EBR ベース

MachXO2 デバイスの EBR ブロックは、デュアルクロック・ファーストイン・ファーストアウト・メモリ（FIFO_DC）として構成可能です。ユーザは設計要件に応じた各種メモリサイズの Verilog-HDL または VHDL ネットリストを IPexpress で生成できます。

IPexpress は図 12-28 に示すような FIFO_DC メモリモジュールを生成します。

図 12-28. IPexpress によって生成される FIFO モジュール



デバイスには多数の EBR ブロックがあるため、生成されるモジュールは、これらの EBR ブロックつまりプリミティブを活用し、それらをカスケード接続して IPexpress の GUI でユーザが指定したメモリサイズを作成します。1 つの EBR ブロックよりも小さいメモリサイズの場合、モジュールは 1 つの EBR ブロックで作成されます。1 つの EBR ブロックよりも大きいメモリサイズの場合は、そのサイズの作成に必要な深さまたは幅で、複数の EBR ブロックをカスケード接続します。

FIFO_DC モードでは、入力データはメモリアレイの入力でレジスタに取り込まれます。メモリの出力データは、必要に応じて出力でレジスタを介して出力されます。

FIFO_DC の各種ポート及びその定義を表 12-5 に示します。

表 12-5. EBR ベース FIFO_DC メモリのポート定義

ベースの

生成されたモジュールのポート名	記述	アクティブステート
WrClock	ライトポート・クロック	クロック立ち上がりエッジ
RdClock	リードポート・クロック	クロック立ち上がりエッジ
WrEn	ライトイネーブル	アクティブ High
RdEn	リードイネーブル	アクティブ High
*ORdEn[1]	出力リードイネーブル	アクティブ High
Reset[2]	リセット	アクティブ High
RPReset[3]	リードポインタ・リセット	アクティブ High
Q	データ出力	–
Empty	Empty（空）フラグ	アクティブ High
Full	Full（フル）フラグ	アクティブ High
AlmostEmpty	Almost Empty（ほぼ空）フラグ	アクティブ High
AlmostFull	Almost Full（ほぼフル）フラグ	アクティブ High
*ERROR	ECC	アクティブ High

* 印はオプションのポートを示す

1. **ORdEn** はオプションの出力レジスタのクロックイネーブルとして使用できる。これにより最終ワードを含み、データ出力のフルパイプライン化が可能
2. **Reset** は FIFO のオプションの出力レジスタをリセットするのみ。入力レジスタやメモリの内容をリセットはしない
3. **RPReset** はリードポインタのみをリセットする。本文記述を参照

IPexpress は FIFO8KB プリミティブを使用して MachXO2 デュアルクロック・ファーストイン・ファーストアウト・メモリを実装します。

FIFO_DC フラグ

FIFO_DC では Empty（エンプティ）、Almost Empty（ほぼエンプティ）、Almost Full（ほぼフル）、Full（フル）の 4 本のフラグが使用可能です。フラグ Almost Empty 及び Almost Full はプログラマブルな範囲を持ちます。

4 本の FIFO_DC フラグのプログラム範囲を表 12-6 に示します。

表 12-6. FIFO_DC フラグの設定

モジュールフラグ名	記述	プログラミング範囲
Full	Full フラグ設定	1 to (2^N – 1)
AlmostFull	Almost full フラグ設定	1 to (FULL –1)
AlmostEmpty	Almost empty フラグ設定	1 to (FULL –1)
Empty	Empty フラグ設定	0

Empty 値は 0 に固定されています。リセットが解除されると、アクティブ High フラグの Empty 及び Almost Empty は真であるため High にセットされます。

ユーザは、フラグ Almost Empty 及び Almost Full が真になるアドレスの絶対値を指定する必要があります。例えば深さ 512 の FIFO のアドレス位置 500 でフラグ Almost Full が真になる必要がある場合、ユーザは値 500 を IPexpress で指定する必要があります。

フラグ Empty 及び Almost Empty は常にリードクロックに同期して出力され、フラグ Full 及び Almost Full は常にライトクロックに同期して出力されます。

リセット時、リードカウンタとライトカウンタは両方ともアドレス 0 を指しています。リセットのネゲート後、ライトイネーブルがアサートされている時にライトクロックの立ち上がりエッジで、ライトカウンタが指すアドレスに対して FIFO_DC へのデータライトが可能です。

同様に FIFO_DC では、リードイネーブルのアサートされている時にリードクロックの立ち上がりエッジで、リードカウンタが指すアドレスからデータリードが可能です。

RPReset（Read Pointer Reset、リードポインタリセット）は再送動作の効率化に使用され、″パケット化″通信でよく使用されます。RPReset のアサートによって内部リードポインタはゼロにリセットされます。これは一般に、リードポインタ及びライトポインタの両方をゼロにリセットする、新規パケットの前のリセットのアサートとそれぞれ組み合わせて使用されます。この用途では、ユーザはパケットのリード / ライトが FIFO_DC との間でいつ行われるかを注意深く追跡する必要があります。メモリ内容を無効化する可能性を避けるために、RPReset は前のリードサイクルが完了する（1 クロック期間長の RdEn がネゲートされる）までアサートされるべきではありません。RPReset がネゲートされると、フラグ Empty 及び Almost Empty は 1 リードサイクル後に正しい状態であると見なされます。これはバウンダリサイクル遅延（boundary cycle latency）と呼ばれる通常の条件です。

FIFO_DC のデータ出力は IPexpress での選択によって、レジスタ出力とすることも、レジスタなしにすることも可能です。出力レジスタはリードイネーブルによってイネーブルされます。

FIFO_DC デュアル閾値と動的閾値オプション

オプションの Almost Full 及び Almost Empty フラグ閾値は、シングル（= デフォルト）またはデュアル閾値動作として個別に設定できます。さらに、閾値はコンフィグレーション時に固定にすることも（= デフォルト）、オプションポートから動的に設定することもできます。デュアルまたは動的閾値の実装では、LUT ベースのサポートロジックも自動的に生成されます。

表 12-7. EBR ベース FIFO_DC のオプション動的閾値のポート定義

生成されたモジュールのポート名	記述
AmEmptyThresh	Almost Empty シングル閾値
AmFullThresh	Almost Full シングル閾値
AmEmptySetThresh	Almost Empty セット閾値
AmEmptyClrThresh	Almost Empty クリア閾値
AmFullSetThresh	Almost Full セット閾値
AmFullClrThresh	Almost Full クリア閾値

FIFO_DC 動作

出力レジスタがイネーブルされていない場合は、最初のワードの読み出しに 2 クロックサイクル必要です。フラグロジック用のレジスタによって、この追加のクロック遅延が生じます。エミュレート FIFO_DC のアーキテクチャでは、データの読み出し用の内部リードイネーブルは、ユーザが提供するリードイネーブルだけでなく、Empty フラグによっても制御されます。FIFO へのデータ書き込み時に、内部 Empty フラグは、ラ

イトイネーブル（WrEn）によってイネーブルされるライトクロックを使用します。ライトクロックからリードクロックへのクロックドメイン転送が要因で、さらに一周期のクロック遅延が、追加されます。このレジスタはリードイネーブルでイネーブルになるリードクロックを用います。

内部的には、このレジスタ出力は反転された後、ユーザが与えるリードイネーブルと AND されて、FIFO_DC のコアある RAM_DP に対する内部リードイネーブルになります。

したがって、最初のデータリードが伝播するのに 2 クロック周期必要です。最初のデータ出力時に、リードイネーブルが 1 クロック周期 High になり、Empty フラグはネゲートされますが、これはリードイネーブルによってイネーブルされる第 2 のレジスタに伝播されません。最初のクロック周期によって Empty が Low になり、第 2 クロック周期によって内部リードイネーブルが High（RdEn & !EF）になってから、データがその第 2 のクロックサイクルで読み出されます。同様に、Full フラグ後の最初のデータライトには同様の遅延があります。

ユーザが出力レジスタをイネーブルした場合、その出力レジスタによって、最初のデータリード中に追加のクロック遅延が生じます。それらのレジスタはリードクロックを用い、リードイネーブルによってイネーブルされているためです。

1. 最初の RdEn 及びクロックサイクルで、EF を内部的に伝播する
2. 2 番目の RdEn 及びクロックサイクルで、内部リードイネーブルを DPRAM に生成する
3. 3 番目の RdEn 及びクロックサイクルで、データを出力レジスタから読み出す

図 12-29 と 12-30 にデュアルクロック FIFO（FIFO_DC）の内部タイミング波形を示します。

図 12-29. FIFO_DC のタイミング波形 ～ 出力レジスタなし（パイプラインなし）

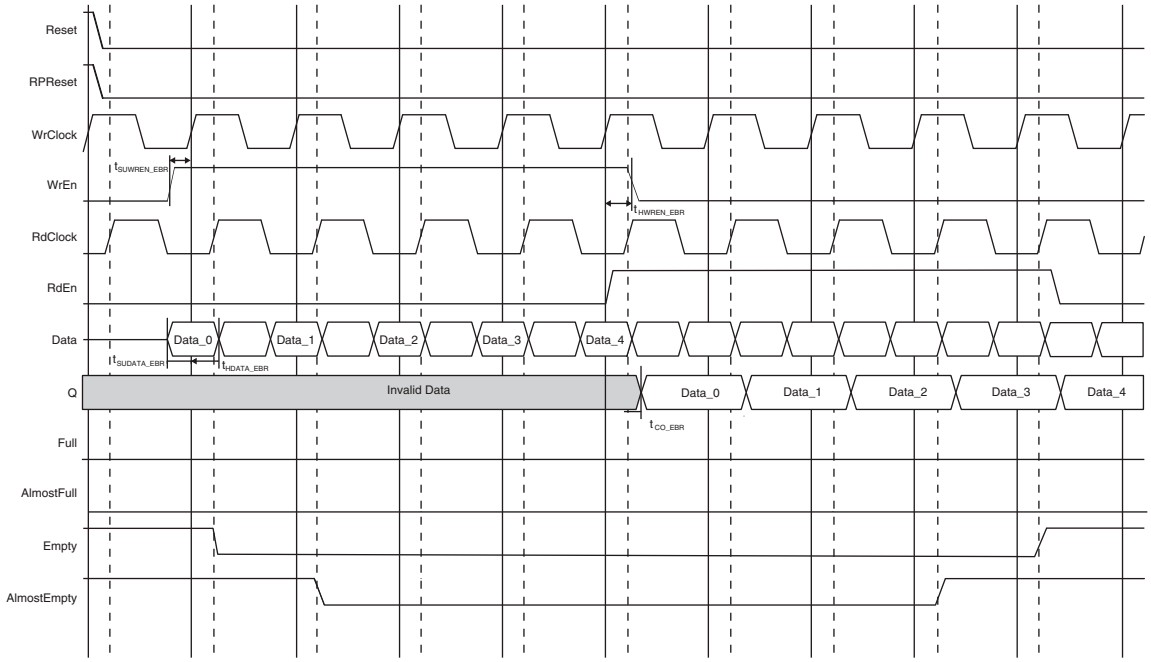
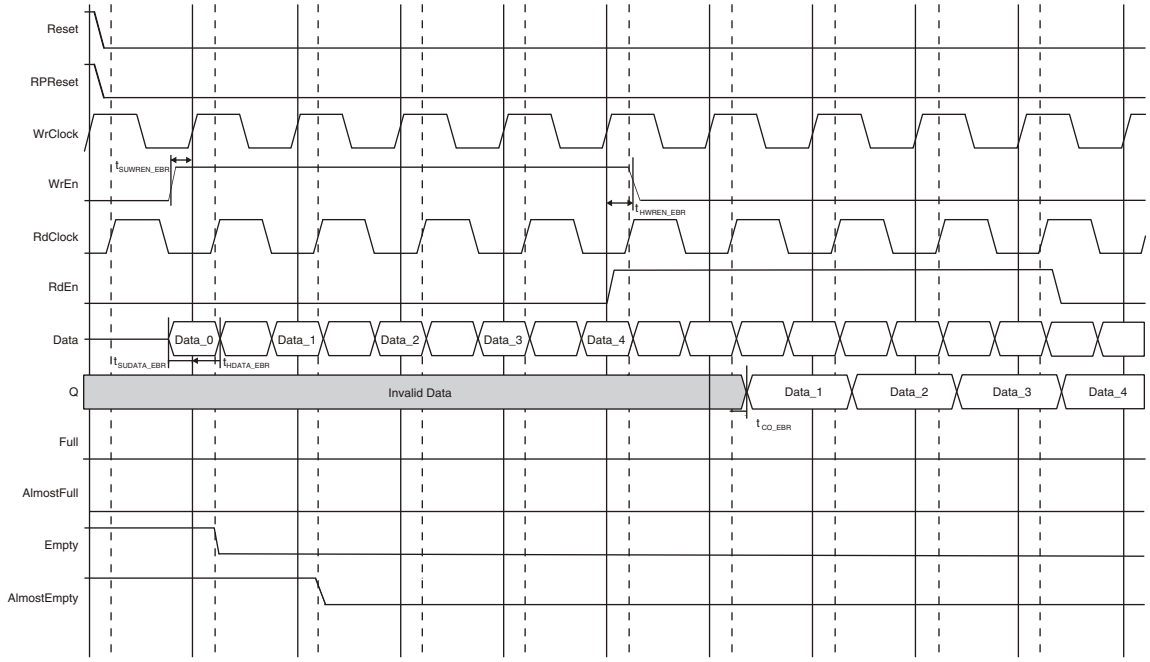


図 12-30. FIFO_DC のタイミング波形 ～ 出力レジスタあり（パイプラインあり）

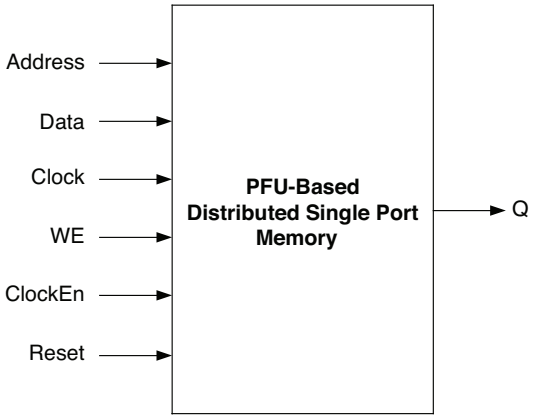


分散シングルポート RAM（Distributed_SPRAM）～ PFU ベース

PFU ベースの分散シングルポート RAM は、PFU 内にある 4 入力 LUT を使用して作成されます。これらの LUT をカスケード接続して、より大きい分散メモリサイズを作成できます。

図 12-31 に IPexpress によって生成される分散シングルポート RAM モジュールを示します。

図 12-31. IPexpress によって生成される分散シングルポート RAM モジュール



生成されるモジュールは、PFU 内にある 4 入力 LUT を活用します。やはり PFU にあるリソースを利用することによって、追加のデコードロジックが生成されます。

このメモリの各種ポート及びその定義を表 12-8 に示します。

表 12-8. PFU ベース分散シングルポート RAM のポート定義

生成されたモジュールのポート名	記述	アクティブステート
Address	アドレス	–
Data	データ入力	–
Clock	クロック	クロック立ち上がりエッジ
WE	ライトイネーブル	アクティブ High
ClockEn	クロックイネーブル	アクティブ High
Reset[1]	リセット	アクティブ High
Q	データ出力	–

1. Reset は出力レジスタがイネーブルされている場合にのみ使用できる

図 12-32 と 12-33 に分散シングルポート RAM の内部タイミング波形を示します。

図 12-32. PFU ベース分散シングルポート RAM のタイミング波形 ～ 出力レジスタなし

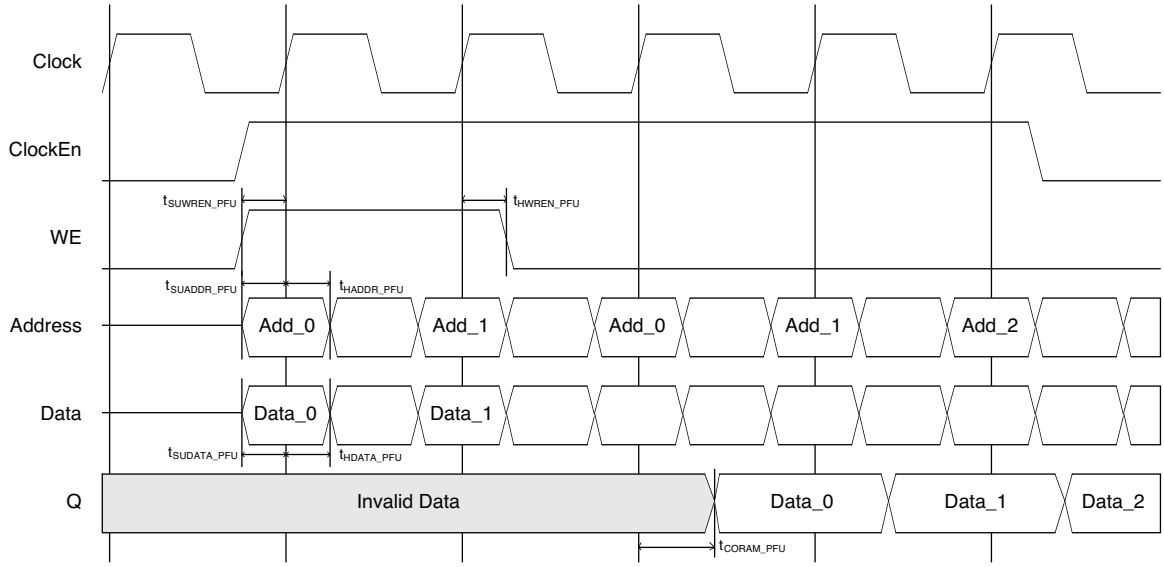
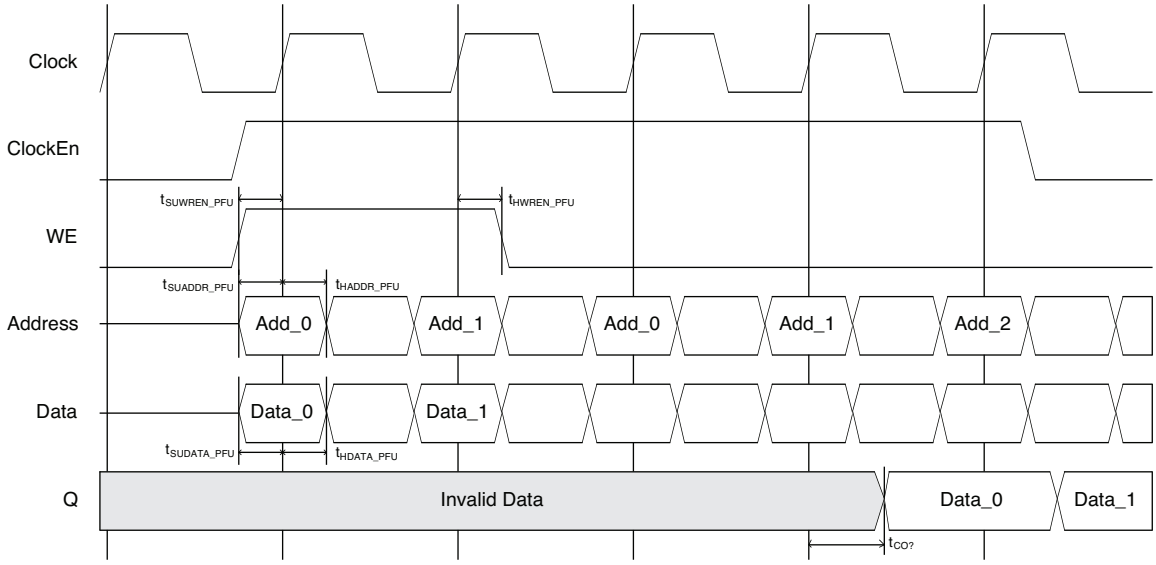


図 12-33. PFU ベース分散シングルポート RAM のタイミング波形 ～ 出力レジスタあり

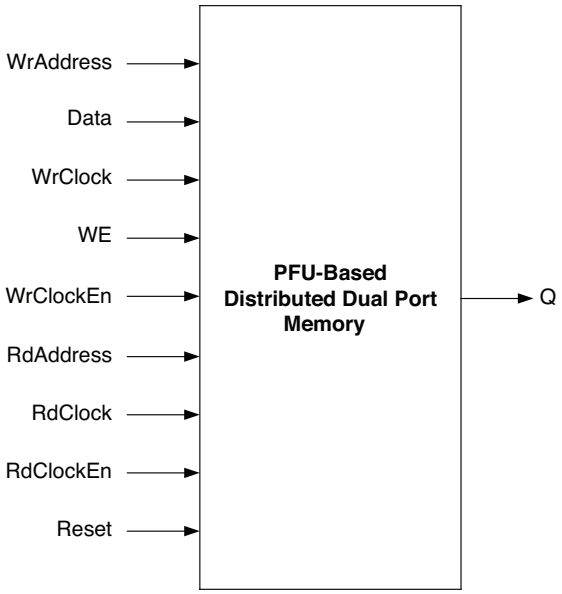


分散デュアルポート RAM（Distributed_DPRAM）～ PFU ベース

PFU ベースの分散デュアルポート RAM は、PFU 内にある 4 入力 LUT を使用して作成されます。これらの LUT をカスケード接続してより大きい分散メモリサイズを作成できます。

図 12-34 に IPexpress によって生成される分散デュアルポート RAM モジュールを示します。

図 12-34. IPexpress によって生成される分散デュアルポート RAM モジュール



生成されるモジュールは、PFU 内にある 4 入力 LUT を活用します。また PFU にあるリソースを利用することによって、追加のデコードロジックが生成されます。

各種ポート及びその定義を表 12-9 に示します。

表 12-9. PFU ベース分散デュアルポート RAM のポート定義

生成されたモジュールのポート名	記述	アクティブステート
WrAddress	ライトアドレス	–
Data	データ入力	–
WrClock	ライトクロック	クロック立ち上がりエッジ
WE	ライトイネーブル	アクティブ High
WrClockEn	ライトクロック・イネーブル	アクティブ High
RdAddress	リードアドレス	–
*RdClock	リードクロック	クロック立ち上がりエッジ
*RdClockEn	リードクロックイネーブル	アクティブ High
Reset*	リセット	アクティブ High
Q	データ出力	–

* 印はオプションのポートを示す

オプションポートのリードクロック（RdClock）及びリードクロックイネーブル（RdClockEn）は、ハードウェア・プリミティブにはありません。これらは、ユーザが IPexpress で出力レジスタのイネーブルを望む場合に、IPexpress によって生成されます。

図 12-35 と 12-36 に分散デュアルポート RAM の内部タイミング波形を示します。

図 12-35. PFU ベース分散デュアルポート RAM のタイミング波形 ～ 出力レジスタなし

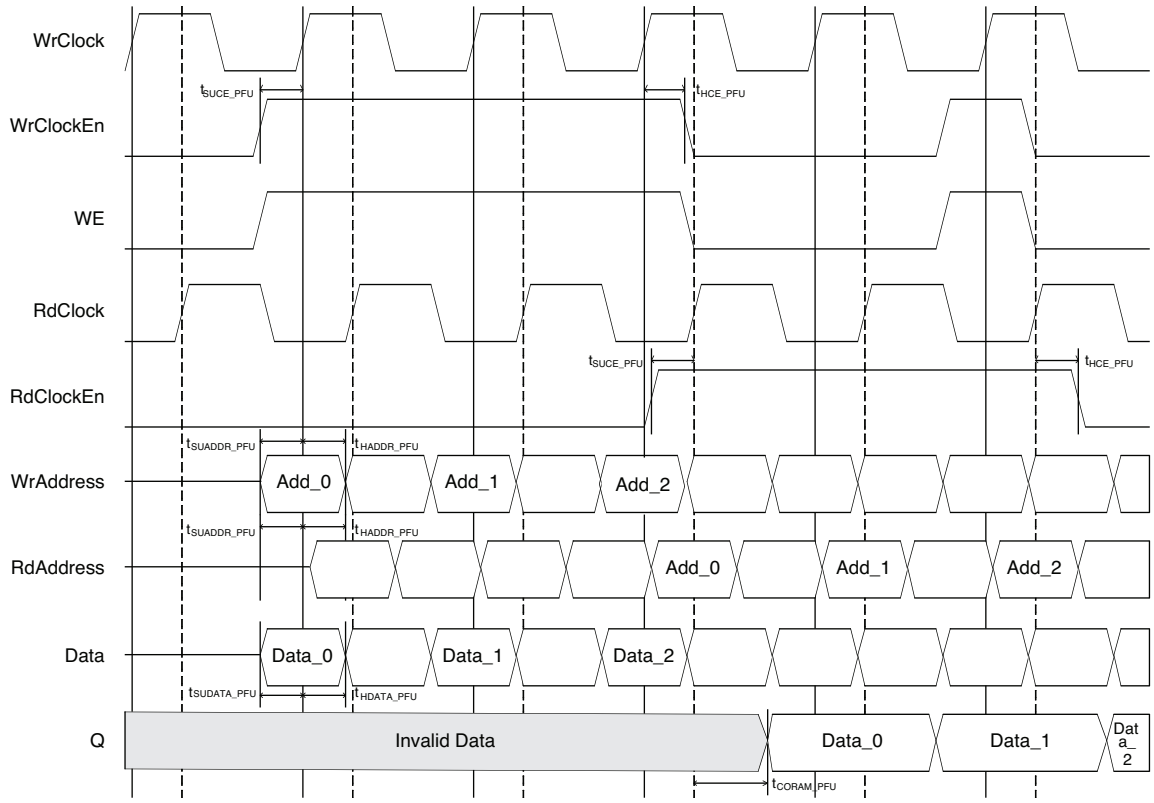
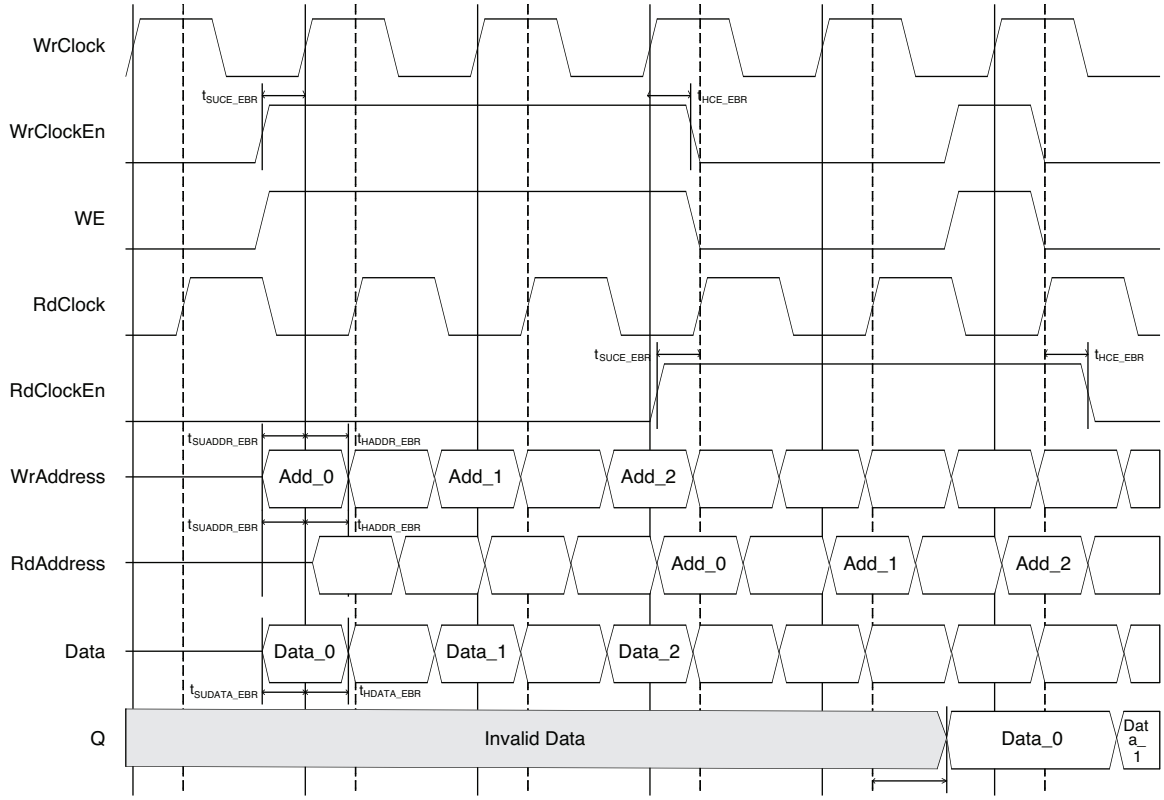


図 12-36. PFU ベース分散デュアルポート RAM のタイミング波形 ～ 出力レジスタあり

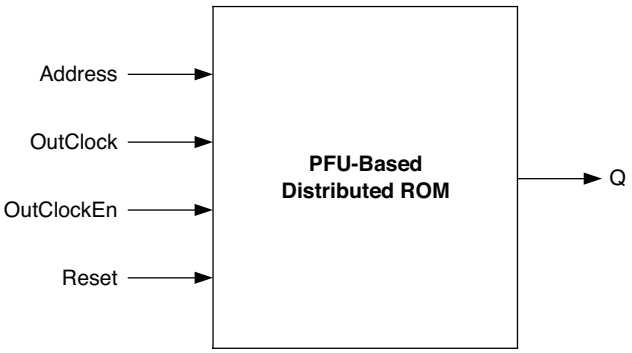


分散 ROM（Distributed_ROM）～ PFU ベース

PFU ベースの分散 ROM は、PFU 内にある 4 入力 LUT を使用して作成されます。これらの LUT をカスケード接続して、より大きい分散メモリサイズを作成できます。

図 12-37 に IPexpress によって生成される分散 ROM モジュールを示します。

図 12-37. IPexpress によって生成される分散 ROM



生成されるモジュールは、PFU 内にある 4 入力 LUT を活用します。また PFU にあるリソースを利用することによって、追加のデコードロジックが生成されます。

各種ポート及びその定義を表 12-10 に示します。

表 12-10. PFU ベース分散 ROM のポート定義

生成されたモジュールのポート名	記述	アクティブステート
Address	アドレス	–
OutClock*	出力クロック	クロック立ち上がりエッジ
OutClockEn*	出力クロックイネーブル	アクティブ High
Reset*	リセット	アクティブ High
Q	データ出力	–

* 印はオプションのポートを示す

オプションポートのアウトクロック（OutClock）及びアウトクロックイネーブル（OutClockEn）は、ハードウェア・プリミティブにはありません。これらは、ユーザが IPexpress で出力レジスタのイネーブルを望む場合に、IPexpress によって生成されます。

図 12-38 と 12-39 に分散 ROM の内部タイミング波形を示します。

図 12-38. PFU ベース ROM のタイミング波形 ～ 出力レジスタなし

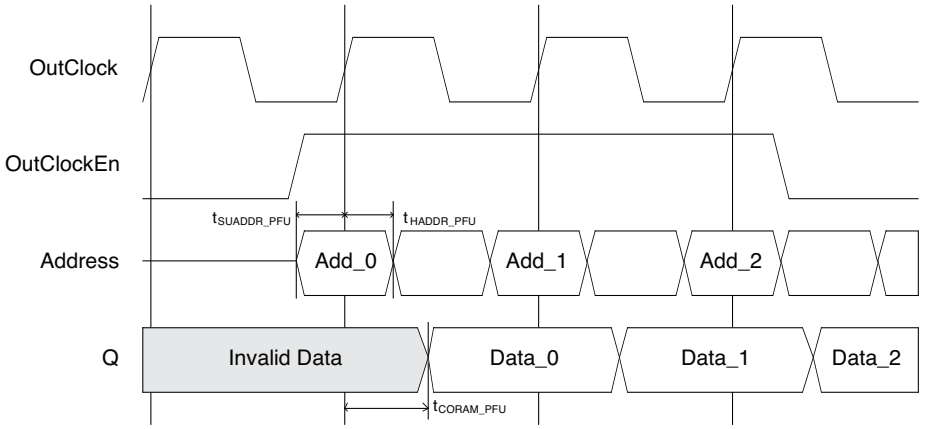
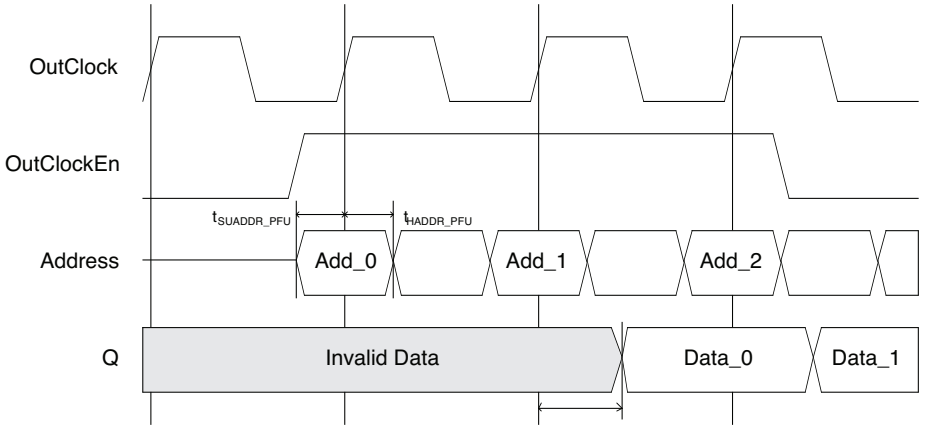


図 12-39. PFU ベース ROM のタイミング波形 ～ 出力レジスタあり

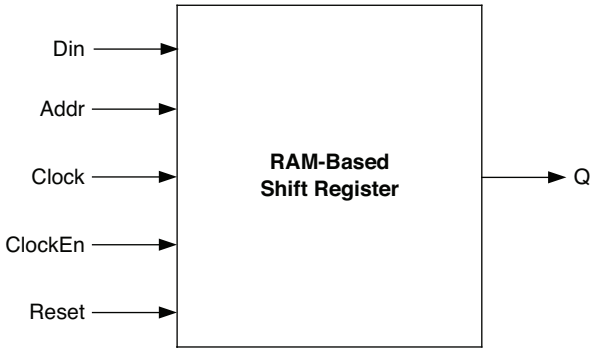


RAM ベースのシフトレジスタ

MachXO2 デバイスの分散 SPRAM ブロックは、LUT ベースのロジックとの組み合わせで、RAM ベースのシフトレジスタとして構成可能です。IPexpress ではユーザは設計要件に従ったシフトレジスタ長の Verilog-HDL または VHDL ネットリストを生成できます。

IPexpress は図 12-40 に示すようなシフトレジスタ・モジュールを生成します。

図 12-40. IPexpress によって生成される RAM ベース・シフトレジスタ



生成されるモジュールは、PFU 内にある 4 入力 LUT を活用します。また PFU にあるリソースを利用することによって、追加のロジックが生成されます。

各種ポート及びその定義を表 12-11 に示します。

表 12-11. RAM ベース・シフトレジスタのポート定義

生成されたモジュールのポート名	記述	アクティブステート
Din	データ入力	–
*Addr	アドレス	–
Clock	クロック	クロック立ち上がりエッジ
ClockEn	クロックイネーブル	アクティブ High
Reset	リセット	アクティブ High
Q	データ出力	–

* 印はオプションのポートを示す

オプションポートの Addr は可変長（Variable Length）タイプが選択されている場合にのみ使用可能で、IPexpress によって生成されます。図 12-41 と 12-42 に RAM ベースのシフトレジスタの内部タイミング波形を示します。

図 12-41. RAM ベース・シフトレジスタのタイミング波形 ～ 出力レジスタなし（Shift = 2）

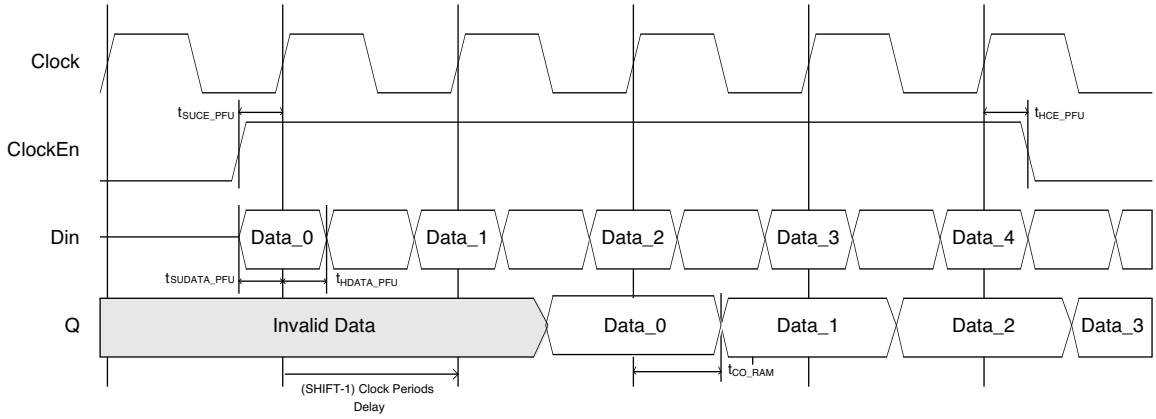
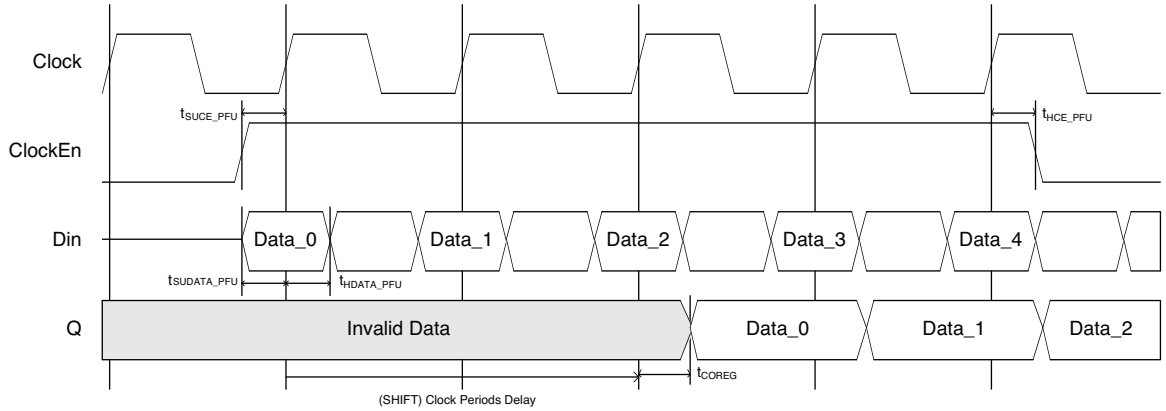


図 12-42. RAM ベース・シフトレジスタのタイミング波形 ～ 出力レジスタあり（Shift = 2）



MachXO2 プリミティブ

シングルポート RAM（SP8KC）～ EBR ベース

シングルポート RAM プリミティブを以下に示します。

図 12-43. シングルポート RAM (SP8KC)

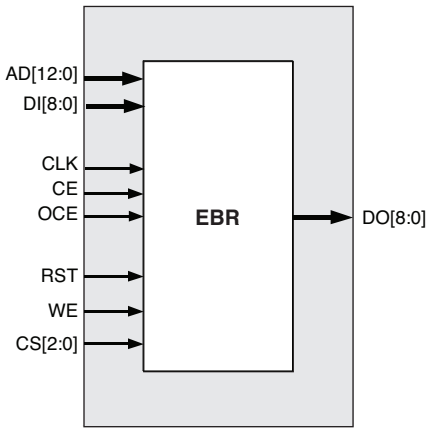


表 12-12. EBR ベースのシングルポートメモリのポート定義

EBR ブロック・プリミティブ（SP8KC）のポート名	記述	アクティブステート
AD	アドレスバス	–
DI	データ入力	–
CLK	クロック	クロック立ち上がりエッジ
CE	クロックイネーブル	アクティブ High
OCE	出力クロックイネーブル	アクティブ High
RST	リセット	アクティブ High
WE	ライトイネーブル	アクティブ High
CS[2:0]	チップセレクト	–
DO	データ出力	–

それぞれの SP8KC プリミティブは 9,216 ビットの RAM で構成されています。SP8KC プリミティブのアドレス深さ及びデータ幅で可能な値を表 12-13 に示します。

表 12-13. MachXO2 シングルポートメモリの 9K メモリサイズ

シングルポート・メモリサイズ	入力データ	出力データ	アドレス [MSB:LSB]
8K x 1	DI	DO	AD[12:0]
4K x 2	DI[1:0]	DO[1:0]	AD[12:1]
2K x 4	DI[3:0]	DO[3:0]	AD[12:2]
1K x 9	DI[8:0]	DO[8:0]	AD[12:3]

表 12-14 に SP8KC に使用可能な各種の属性を示します。これら属性のいくつかはユーザが IPexpress の GUI で選択可能です。属性定義の詳細については付録 A を参照してください。

表 12-14. MachXO2 のシングルポート RAM 属性（SP8KC）

属性（Attribute）	記述	値	デフォルト値	ユーザが IPexpress で選択可能
DATA_WIDTH	データワード幅	1, 2, 4, 9	9	Yes
REGMODE	レジスタモード（パイプライン化）	NOREG, OUTREG	NOREG	Yes
RESETMODE	リセットタイプ選択	ASYNC, SYNC	SYNC	Yes
CSDECODE	チップセレクト・デコード	0b000, 0b001, 0b010, 0b011, 0b100, 0b101, 0b110, 0b111 0b000	0b000	No
WRITEMODE	リード / ライト振る舞い	NORMAL, WRITETHROUGH, READBEFOREWRITE	NORMAL	Yes
GSR	グローバル・セットリセットをイネーブル	ENABLED, DISABLED	DISABLED	No
INITVAL_00 .. INITVAL_1F	初期化値	0x00000000000000000000000000000000000000000000000000000000000000000000 .... 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF (80 キャラクタの 16 進数)	0x00000000000000000000000000000000000000000000000000000000000000000000	No
ASYNC_RESET_RELEASE	リセット解除	ASYNC, SYNC	SYNC	Yes
INIT_DATA	初期値ステータス	STATIC, DYNAMIC	STATIC	Yes

真のデュアルポート RAM（DP8KC）～ EBR ベース

真のデュアルポート RAM プリミティブを以下に示します。

図 12-44. 真のデュアルポート RAM (DP8KC)

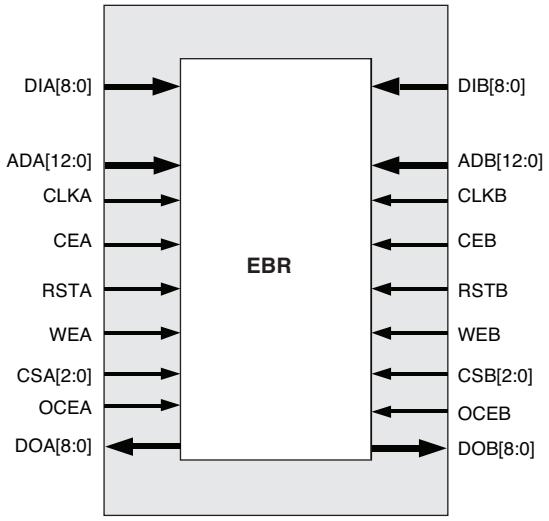


表 12-15. EBR ベースの真のデュアルポート・メモリのポート定義

EBR ブロック・プリミティブ（DP8KC）のポート名	記述	アクティブステート
DIA, DIB	入力データ、ポート A および B	–
ADA, ADB	アドレスバス、ポート A および B	–
CLKA, CLKB	クロック、ポート A および B	クロック立ち上がりエッジ
CEA, CEB	クロックイネーブル、ポート A および B	アクティブ High
RSTA, RSTB	リセット、ポート A および B	アクティブ High
WEA, WEB	ライトイネーブル、ポート A および B	アクティブ High
CSA[2:0], CSB[2:0]	チップセレクト、各ポート	–
OCEA, OCEB	出力クロックイネーブル、ポート A および B	アクティブ High
DOA, DOB	出力データポート、ポート A および B	–

各 DP8KC プリミティブは 9,216 ビットの RAM で構成されています。DP8KC プリミティブのアドレス深さ及びデータ幅で可能な値を 表 12-16 に示します。

表 12-16. MachXO2 真のデュアルポートメモリの 9K メモリサイズ

デュアルポートメモリサイズ	入力データポート A	入力データポート B	出力データポート A	出力データポート B	アドレスポート A[MSB:LSB]	アドレスポート B[MSB:LSB]
8K x 1	DIA	DIB	DOA	DOB	ADA[12:0]	ADB[12:0]
4K x 2	DIA[1:0]	DIB[1:0]	DOA[1:0]	DOB[1:0]	ADA[12:1]	ADB[12:1]
2K x 4	DIA[3:0]	DIB[3:0]	DOA[3:0]	DOB[3:0]	ADA[12:2]	ADB[12:2]
1K x 9	DIA[8:0]	DIB[8:0]	DOA[8:0]	DOB[8:0]	ADA[12:3]	ADB[12:3]

表 12-17 に真のデュアルポートメモリ（RAM_DP_TRUE）に使用可能な各種の属性を示します。これら属性のいくつかはユーザが IPexpress の GUI で選択可能です。属性定義の詳細については付録 A を参照してください。

表 12-17. MachXO2 の真のデュアルポート RAM 属性（DP8KC）

属性（Attribute）	記述	値	デフォルト値	ユーザが IPexpress で選択可能
DATA_WIDTH_A	データワード幅、ポート A	1, 2, 4, 9	9	Yes
DATA_WIDTH_B	データワード幅、ポート B	1, 2, 4, 9	9	Yes
REGMODE_A	レジスタモード（パイプライン化）、ポート A	NOREG, OUTREG	NOREG	Yes
REGMODE_B	レジスタモード（パイプライン化）、ポート B	NOREG, OUTREG	NOREG	Yes
RESETMODE	リセットタイプ選択	ASYNC, SYNC	SYNC	Yes
CSDECODE_A	チップセレクト・デコード、ポート A	0b000, 0b001, 0b010, 0b011, 0b100, 0b101, 0b110, 0b111	0b000	No
CSDECODE_B	チップセレクト・デコード、ポート B	0b000, 0b001, 0b010, 0b011, 0b100, 0b101, 0b110, 0b111	0b000	No
WRITEMODE_A	リード / ライトモード、ポート A	NORMAL, WRITETHROUGH, READBEFOREWRITE	NORMAL	Yes
WRITEMODE_B	リード / ライトモード、ポート B	NORMAL, WRITETHROUGH, READBEFOREWRITE	NORMAL	Yes
GSR	グローバル・セットリセットをイネーブル	ENABLE, DISABLE	DISABLED	No
INITVAL_00 .. INITVAL_1F	初期化値	0x000000000000000000000000000000000000000000000000000000000000000000000000000000 .... 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF (80 キャラクタの 16 進数)	0x00000000000000000000000000000000000000000000000000000000000000000000000000000000	No
ASYNC_RESET_RELEASE	リセット解除	ASYNC, SYNC	SYNC	Yes
INIT_DATA	初期値ステータス	STATIC, DYNAMIC	STATIC	Yes

擬似デュアルポート RAM（PDPW8KC）～ EBR ベース

擬似デュアルポート RAM プリミティブを次に示します。

図 12-45. 擬似デュアルポート RAM（PDPW8KC）

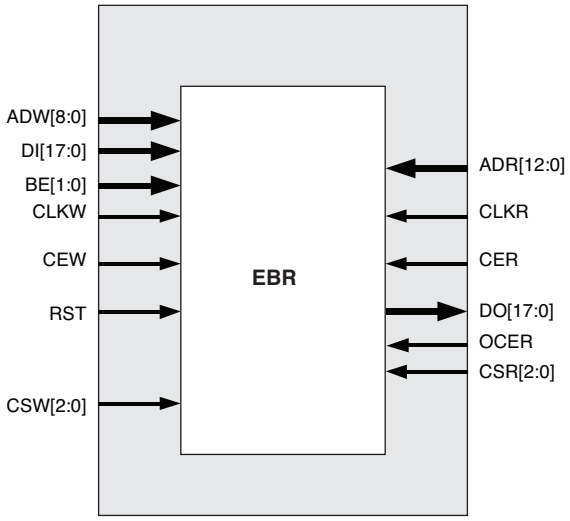


表 12-18. EBR ベースの擬似デュアルポートメモリのポート定義

EBR ブロック・プリミティブ（PDPW8KC）のポート名	記述	アクティブステート
ADW	ライトアドレス	–
DI	ライトデータ	–
BE	バイトイネーブル	アクティブ High
CLKW	ライトクロック	立ち上がりエッジ
CEW	ライトクロック・イネーブル	アクティブ High
RST	リセット	アクティブ High
CSW	ライト・チップセレクト	–
ADR	リードアドレス	–
CLKR	リードクロック	立ち上がりエッジ
CER	リードクロック・イネーブル	アクティブ High
DO	リードデータ	–
OCER	リード出力クロックイネーブル	アクティブ High
CSR	リード・チップセレクト	–

各 PDPW8KC プリミティブは 9,216 ビットの RAM で構成されています。DPW8KC プリミティブのアドレス深さ及びデータ幅で可能な値を 表 12-19 に示します。

表 12-19. MachXO2 擬似デュアルポートメモリの 9K メモリサイズ

疑似デュアル、リードポートメモリサイズ	ライトデータポート	リードデータポート	リードアドレスポート [MSB:LSB]	ライトアドレスポート [MSB:LSB]
8K x 1	DI[17:0]	DO	ADR[12:0]	ADW[8:0]
4K x 2	DI[17:0]	DO[1:0]	ADR[12:1]	ADW[8:0]
2K x 4	DI[17:0]	DO[3:0]	ADR[12:2]	ADW[8:0]
1K x 9	DI[17:0]	DO[8:0]	ADR[12:3]	ADW[8:0]
512 x 18	DI[17:0]	DO[17:0]*	ADR[12:4]	ADW[8:0]

注 : 上位バイトと下位バイトは入力 DI ワードに対して入れ替わる

表 12-20 に擬似デュアルポートメモリ（RAM_DP）に使用可能な各種の属性を示します。これら属性のいくつかはユーザが IPexpress の GUI で選択可能です。属性定義の詳細については付録 A を参照してください。

表 12-20. MachXO2 の擬似デュアルポート RAM 属性（PDPW8KC）

属性（Attribute）	記述	値	デフォルト値	ユーザが IPexpress で選択可能
DATA_WIDTH_W	ライトデータワード幅	18	18	Yes
DATA_WIDTH_R	リードデータワード幅	1, 2, 4, 9, 18	9	Yes
REGMODE	レジスタモード（パイプライン化）	NOREG, OUTREG	NOREG	Yes
RESETMODE	リセットタイプ選択	ASYNC, SYNC	SYNC	Yes
CSDECODE_W	チップセレクト・デコード、ライト	0b000, 0b001, 0b010, 0b011, 0b100, 0b101, 0b110, 0b111	0b000	No
CSDECODE_R	チップセレクト・デコード、リード	0b000, 0b001, 0b010, 0b011, 0b100, 0b101, 0b110, 0b111	0b000	No
GSR	グローバル・セットリセット選択	ENABLE, DISABLE	DISABLED	No
INITVAL_00 .. INITVAL_1F	初期化値	0x00000000000000000000000000000000000000000000000000000000000000000000 .... 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF FFFFFFF (80- character hex strings)	0x00000000000000000000000000000000000000000000000000000000000000000000	No
ASYNC_RESET_RELEASE	リセット解除	ASYNC, SYNC	SYNC	Yes
INIT_DATA	初期値ステータス	STATIC, DYNAMIC	STATIC	Yes

デュアルクロック FIFO（FIFO8KB）～ EBR ベース

デュアルクロック FIFO RAM プリミティブを以下に示します。

図 12-46. FIFO_DC プリミティブ (FIFO8KB)

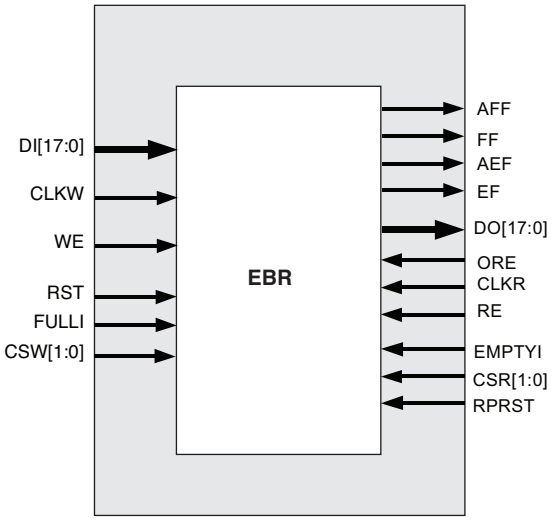


表 12-21. EBR ベース FIFO_DC メモリのポート定義

プリミティブ（FIFO8KB）のポート名	記述	アクティブステート
DI	データ入力	–
CLKW	ライトポート・クロック	クロック立ち上がりエッジ
WE	ライトイネーブル	アクティブ High
FULLI	ライト禁止	アクティブ High
CSW	ライト・チップセレクト	アクティブ High
AFF	Almost Full フラグ	アクティブ High
FF	Full フラグ	アクティブ High
AEF	Almost Empty フラグ	アクティブ High
EF	Empty フラグ	アクティブ High
DO	データ出力	–
ORE	出力データイネーブル	アクティブ High
CLKR	リードポート・クロック	クロック立ち上がりエッジ
RE	リードイネーブル	アクティブ High
EMPTYI	リード禁止	アクティブ High
CSR	リード・チップセレクト	アクティブ High
RPRST	リードポインタ・リセット	アクティブ High

各 FIFO8KB プリミティブは 9,216 ビットの RAM で構成されています。FIFO8KB プリミティブのアドレス深さ及びデータ幅で可能な値を 表 12-22 に示します。

表 12-22. MachXO2 FIFO_DC のデータ幅サイズ

FIFO サイズ	入力データ	出力データ
8K x 1	DI	DO
4K x 2	DI[1:0]	DO[1:0]
2K x 4	DI[3:0]	DO[3:0]
1K x 9	DI[8:0]	DO[8:0]
512 x 18	DI[17:0]	DO[17:0]

表 12-23 に FIFO_DC に使用可能な各種の属性を示します。これら属性のいくつかはユーザが IPexpress の GUI で選択可能です。属性定義の詳細については付録 A を参照してください。

表 12-23. FMachXO2 の FIFO_DC 属性 (FIFO8KB)

属性 (Attribute)	記述	値	デフォルト値	ユーザが IPexpress で選択可能
DATA_WIDTH_W	データ幅、ライトモード	1, 2, 4, 9, 18	18	YES
DATA_WIDTH_R	データ幅、リードモード	1, 2, 4, 9, 18	18	YES
REGMODE	レジスタモード	NOREG, OUTREG	NOREG	YES
RESETMODE	リセットタイプ選択	ASYNC, SYNC	ASYNC	YES
CSDECODE_W	チップセレクト、ライトモード	0b00, 0b01, 0b10, 0b11	0b00	NO
CSDECODE_R	チップセレクト、リードモード	0b00, 0b01, 0b10, 0b11	0b00	NO
GSR	グローバル・セットリセット選択	ENABLED, DISABLED	DISABLED	NO
AEPOINTER	Almost Empty ポインタ	0b00000000000000, ..., 0b01111111111111	–	YES
AFPOINTER	Almost Full ポインタ	0b00000000000000, ..., 0b01111111111111	–	YES
FULLPOINTER	Full ポインタ	0b00000000000000, ..., 0b10000000000000	–	YES
FULLPOINTER1	Full ポインタ -1	0b00000000000000, ..., 0b01111111111111	–	NO
AFPOINTER1	Almost Full ポインタ -1	0b00000000000000, ..., 0b01111111111110	–	NO
AEPOINTER1	Almost Empty ポインタ +1	0b00000000000000, ..., 0b10000000000000	–	NO
ASYNC_RESET_RELEASE	リセット解除	ASYNC, SYNC	SYNC	Yes

FIFO_DC フラグ

FIFO_DC では Empty、Almost Empty、Almost Full、Full の 4 本のフラグが使用可能です。フラグ Almost Empty 及び Almost Full はプログラマブルな範囲を持ちます。

4 つの FIFO_DC フラグのプログラム範囲を表 12-24 に示します。

表 12-24. FIFO_DC フラグ設定

モジュールフラグ名	FIFO アトリビュート名	記述	プログラミング範囲	プログラムビット
Full	FULLPOINTER	Full 設定	1 ～ (2N – 1)	14
	FULLPOINTER1	Full - 1	1 ～ (FULL-1)	14
AlmostFull	AFPOINTER	Almost full 設定	1 ～ (FULL -1)	14
	AFPOINTER1	Almost full - 1	1 ～ (FULL -1)	14
	AEPOINTER1	Almost empty + 1	1 ～ (FULL -1)	14
AlmostEmpty	AEPOINTER	Almost empty 設定	1 ～ (FULL -1)	14
Empty	–	Empty 設定	0	–

Empty 値は 0 に固定されています。リセットが解除されると、アクティブ High フラグの Empty 及び Almost Empty は真であるため High にセットされます。

望む振る舞いとなるようにポインタ属性を設定するためには、細心の注意が必要です。″付録 B. FIFO_DC ポインタ属性の設定 ″を参照してください。

フラグ Empty 及び Almost Empty のレジスタは常にリードクロックを用い、フラグ Full 及び Almost Full レジスタは常にライトクロックを用います。

分散 SPRAM（SPR16X4C）～ PFU ベース

PFU ベースの分散シングルポート RAM プリミティブを以下に示します。

図 12-47. Distributed_SPRAM プリミティブ（SPR16X4C）

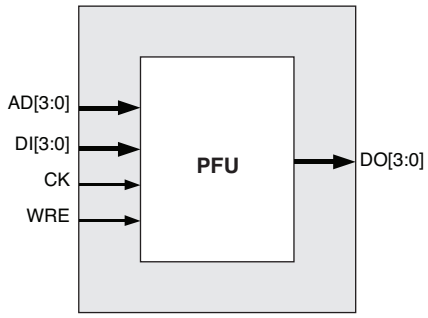


表 12-25. PFU ベースの分散シングルポート RAM のポート定義

PFU プリミティブのポート名	記述	アクティブステート
AD[3:0]	アドレス	–
DI[3:0]	データ入力	–
CK	クロック	クロック立ち上がりエッジ
WRE	ライトイネーブル	アクティブ High
DO[3:0]	データ出力	–

分散 DPRAM（DPR16X4C）～ PFU ベース

PFU ベースの分散擬似デュアルポート RAM プリミティブを以下に示します。

図 12-48. 分散 DPRAM プリミティブ（DPR16X4C）

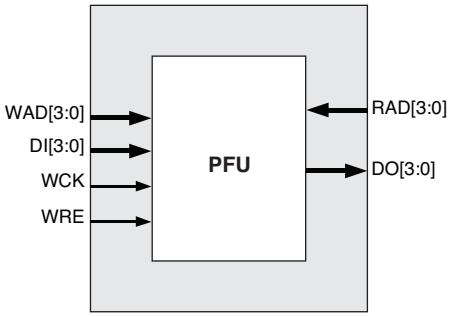


表 12-26. PFU ベースの分散デュアルポート RAM のポート定義

EBR ブロック・プリミティブのポート名	記述	アクティブステート
WAD[3:0]	ライトアドレス	–
DI[3:0]	データ入力	–
WCK	ライトクロック	クロック立ち上がりエッジ
WRE	ライトイネーブル	アクティブ High
RAD[3:0]	リードアドレス	–
DO[3:0]	データ出力	–

分散 ROM（ROMnnnX1A）～ PFU ベース

PFU ベースの分散 ROM プリミティブを以下に示します。

図 12-49. Distributed_ROM プリミティブ（ROM16X1A）

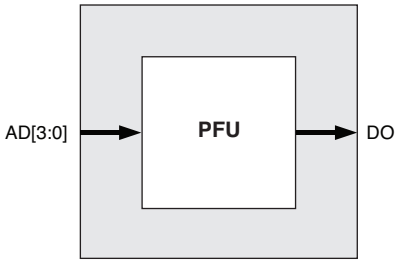


図 12-50. Distributed_ROM プリミティブ（ROM32X1A）

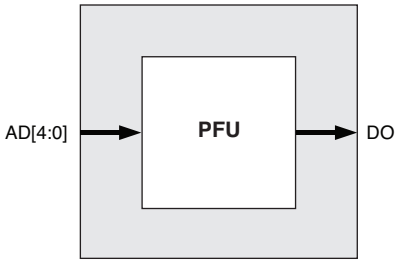


図 12-51. Distributed_ROM プリミティブ (ROM64X1A)

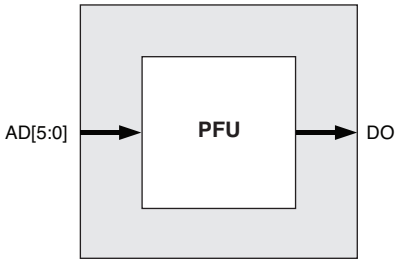


図 12-52. Distributed_ROM プリミティブ (ROM128X1A)

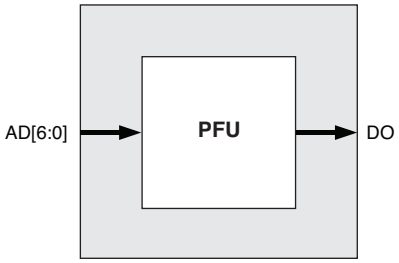


図 12-53. Distributed_ROM プリミティブ (ROM256X1A)

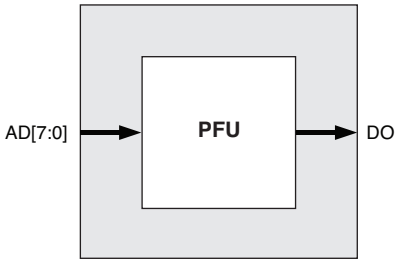


表 12-27. PFU ベース分散 ROM のポート定義

PFU ブロックプリミティブのポート名	記述
AD[n:0]	アドレス
DO	データ出力

メモリ初期化

EBR ベースの ROM または RAM メモリモード、及び PFU ベースの ROM メモリモードではメモリアレイ各ビットのパワーオン状態を指定可能で、値 0 または値 1 のいずれかを持つことができます。

初期化ファイル・フォーマット

初期化ファイルは ASCII 形式で、任意の ASCII エディタを使用して作成または編集できます。IPexpress は次の 3 種類のメモリファイル・フォーマットに対応します。

1. バイナリファイル
2. 16 進ファイル
3. アドレス付き 16 進（Addressed Hex）

メモリ初期化ファイルのファイル名は "*.mem (< ファイル名 >.mem) " です。各行は特定のメモリロケーションに格納される値を表し、文字数（または列数）は各アドレスビット数（またはメモリモジュールの幅）を表します。

初期化ファイルは主に ROM の構成に使用されます。RAM の EBR も初期化ファイルを使用してメモリ内容をプリロードできます。

バイナリファイル

0 と 1 によるテキストファイルです。行数はワード数を示し、列数はメモリの幅を示します。

メモリサイズ 20x32 の例
00100000010000000010000001000000
00000001000000010000000100000001
00000010000000100000001000000010
00000011000000110000001100000011
00000100000001000000010000000100
00000101000001010000010100000101
00000110000001100000011000000110
00000111000001110000011100000111
00001000001001000000100001001000
00001001010010010000100101001001
00001010010010100000101001001010
00001011010010110000101101001011
00001100000011000000110000001100
00001101001011010000110100101101
00001110001111100000111000111110
00001111001111110000111100111111
00010000000100000001000000010000
00010001000100010001000100010001
00010010000100100001001000010010
00010011000100110001001100010011

16 進ファイル

16 進ファイルは同様に行と列の配列で並べた 16 進キャラクタによるテキストファイルです。このファイルの行数はアドレス・ロケーション数と同じで、各行はそのメモリ・ロケーションの内容を示しています。

メモリサイズ 8x16 の例
A001
0B03
1004
CE06
0007
040A
0017
02A4

アドレス付き 16 進

アドレス付き 16 進はアドレスとデータの行で構成されます。各行の先頭はアドレスで、コロンと任意数のデータが続きます。memfile フォーマットは address: data data data data ... で、address と data は 16 進数です。

-A0 : 03 F3 3E 4F
-B2 : 3B 9F

この例では、最初の行は 03 をアドレス A0、F3 をアドレス A1、3E をアドレス A2、4F をアドレス A3 にそれぞれ格納します。2 行目は 3B をアドレス B2、9F をアドレス B3 にそれぞれ格納します。

アドレスとデータの値に制限はありません。値の範囲は addr_width と data_width の値に基づいて自動的にチェックされます。アドレスまたはデータの値に誤りがある場合は、エラーメッセージが出力されます。ユーザが全てのアドレス位置のデータを指定する必要はありません。特定のアドレスでデータが指定されなかった場合、その位置のデータは 0 に初期化されます。IPexpress は論理合成とシミュレーションのフローで、共にメモリ初期化を可能にします。

テクニカルサポート

e-mail: techsupport@latticesemi.com
インターネット：www.latticesemi.com

日本語版改訂履歴

リリース	バージョン	新該当ページ	改訂内容
2010 年 11 月	01.0		初版
2011 年 1 月	01.1		次の図の脚注を削除：図 12-1 MachXO2-1200 上面図、図 12-2 MachXO2-4000 上面図
2013 年 5 月	01.2	-	ロゴ更新、ステータス更新（アドバンスト -> 正規）
2013 年 7 月	01.3	12-27, -28	図 12-29、-30、FIFO_DC タイミング図を更新
		12-47	テクニカルサポート、ホットライン削除

付録 A. 属性（Attribute）定義

DATA_WIDTH

データ幅は RAM 及び FIFO エレメントに関連しています。DATA_WIDTH 属性は各ワードのビット数を定義します。各メモリモジュールの RAM サイズテーブルで定義された値になります。

REGMODE

REGMODE（Register mode）属性は、メモリのパイプラインをイネーブルするために使用されます。この属性は RAM 及び FIFO エレメントに関連しています。REGMODE 属性がとり得る値は、出力パイプライン・レジスタをイネーブルまたはディセーブルする NOREG または OUTREG モードパラメータです。

RESETMODE

RESETMODE 属性を使用すると、ユーザはメモリのリセットモードを選択できます。この属性はブロック RAM エレメントに関連しています。RESETMODE がとり得る値は 2 つのパラメータ、SYNC と ASYNC です。SYNC ではメモリリセットがクロックと同期します。ASYNC では、メモリリセットがクロックと非同期になります。

CSDECODE

CSDECODE（Chip Select Decode）属性はブロック RAM エレメントに関連しています。CS（Chip Select）は、複数のカスケード接続 EBR ブロックがメモリで必要な場合に便利なポートです。CS 信号は、複数の EBR ブロックがカスケード接続されている場合にアドレスの MSB を形成します。CS は 3 ビットのバスであるため、8 つのメモリを容易にカスケード接続できます。CSDECODE がとり得る値はパラメータ ″000″、″001″、″010″、″011″、″100″、″101″、″110″、″111″ です。CSDECODE の値によって、CS[2:0] のデコード値が決まります。CSDECODE_W は書き込み用チップセレクト・デコードで、CSDECODE_R は擬似デュアルポート RAM のリード用チップセレクト・デコードです。CSDECODE_A と CSDECODE_B はデュアルポート RAM エレメントで使用され、A 及び B ポートを指します。

WRITEMODE

属性はブロック RAM エレメントに関連しています。とり得る値は NORMAL、WRITETHROUGH、及び READBEFOREWRITE の各モードパラメータです。NORMAL モードでは、出力データはライト動作中に変化も更新もされません。このモードは全てのデータ幅に対応します。

WRITETHROUGH モードでは、出力データはライトサイクル中に入力データで更新されます。このモードは全てのデータ幅に対応します。

READBEFOREWRITE モードでは、出力データポートはライトサイクル中にライトアドレスに格納されている既存のデータで更新されます。このモードは x9 及び x18 データ幅に対応します。

WRITEMODE_A と WRITEMODE_B はデュアルポート RAM エレメントに使用され、真のデュアルポート RAM の場合に A 及び B ポートを指します。

デュアルポート・モジュールの全モードについて、一方のポートからのリードアクセスと、もう一方のポートからの同一メモリアドレスへのライトアクセスを同時に行うことは推奨されません。この状況ではリードデータが不定になる場合があります。また、両方のポートから同一アドレスへの同時ライトアクセスも推奨されません。これが起きた場合、そのアドレスに格納されるデータは、例えば一方のポートから ″H″、他方から ″L″ をライトしようとしたときに不定になります。

このような状態が発生する場合、ユーザはそれを識別する制御ロジックを実装し、さらに次のいずれかを行うことが推奨されます。

1. リードデータが無効と思われる場合に通知するステータス信号を実装する
2. 両ポートからの同時アクセスを防止する制御ロジックを実装する

GSR

GSR（Global Set/Reset）属性は、RAM エレメントのグローバルセット / リセットをイネーブルまたはディセーブルするために使用されます。

ASYNC_RESET_RELEASE

RESETMODE が ASYNC に設定されている場合、ASYNC_RESET_RELEASE 属性を使用すると、ユーザはリセットをどのようにネゲート・解放するかを選択できます。SYNC に設定した場合、リセットはクロックに同期してネゲートされます。ASYNC に設定した場合、メモリリセットは非同期で（クロックに関係なく）解放されます。

INIT_DATA

INIT_DATA 属性を使用すると、ユーザは EBR 初期化値をどのように格納しその値にアクセスするかを指定できます。STATIC に設定した場合、EBR 初期化値はソフトウェアによって圧縮され、UFM の可変ロケーションに格納されます。DYNAMIC に設定した場合、初期化値は圧縮されず、ユーザがアクセス可能な UFM の固定ロケーションに格納されます。

付録 B. FIFO_DC ポインタ属性の設定

FIFO_DC はポインタ属性を使用して、Full、Almost Full、Almost Empty の各フラグを制御します。

ポインタ属性の値は次の表の式に従って設定されます。

表 12-28. ポインタ属性設定の数式

Flag	Trip Value	Attribute	Port Width	Equation
Full	ff	FULLPOINTER	wrw[1]	[(ff - 1) * wrw] + 1
		FULLPOINTER1		[(ff - 2) * wrw] + 1
Almost Full	aff	AFPOINTER	wrw[1]	[(aff - 1) * wrw] + 1
		AFPOINTER1		[(aff - 2) * wrw] + 1
Almost Empty	aef	AEPOINTER	rdw[1]	[(aef) * rdw] + rdw - 1
		AEFOINTER1		[(aef + 1) * rdw] + rdw - 1

1. ライトポート幅（wrw）およびリードポート幅（rdw）は表 12-29 に従って設定する

表 12-29. ポート幅の値

属性 : Data_width_w, Data_width_r	ポート幅 : wrw, rdw
1	1
2	2
4	4
9	8
18	16

ユーザはフラグ Almost Empty 及び Almost Full が真になるアドレスの絶対値を指定する必要があります。例えば深さ 512 の FIFO のアドレス位置 500 でフラグ Almost Full が真になる必要がある場合、ユーザは aff = 500 を指定する必要があります。

実例：

書き込みデータ幅：18 => wrw=16 を使用

読み取りデータ幅：: 4 => rdw = 4 を使用

Full: ff =16 ((16) 18 ビットワード)

Almost Full: aff = 14

Almost Empty: aef = 8 ((8) 4 ビットワードが （2） 16 ビット書き込みに対応)

Empty: 0 (常時)

計算値：

FULLPOINTER = [(ff – 1) * wrw] + 1
= [(16 - 1) * 16] + 1
= (15 * 16) + 1
= 241
=> 14’b00_0000_1111_0001

FULLPOINTER1 = [(ff – 2) * wrw] + 1
= [(16 - 2) * 16] + 1
= (14 * 16) + 1
= 225
=> 14’b00_0000_1110_0001

AFPOINTER = [(aff – 1) * wrw] + 1
= [(14 - 1) * 16] + 1
= (13 * 16) + 1
= 209
=> 14’b00_0000_1101_0001

AFPOINTER1 = [(aff – 2) * wrw] + 1
= [(14 - 2) * 16] + 1
= (12 * 16) + 1
= 193
=> 14’b00_0000_1100_0001

AEPOINTER = [(aef) * rdw] + rdw – 1
= (8 * 4) + 4 – 1
= (8 * 4) + 3
= 35
=> 14’b00_0000_0010_0011

AEFOINTER1 = [(aef + 1) * rdw] + rdw – 1
= [(8+1) * 4] + 4 – 1
= (9 * 4) + 3
= 39
=> 14’b00_0000_0010_0111