

同期 FIFO の理解

関連プロジェクト: なし

関連製品ファミリ: CY7C42x5/CY7C42x1

概要

AN1042 では同期 FIFO が提供する機能と特長を簡単に紹介します。本アプリケーション ノートでは同期 FIFO のデータ幅と深さの拡張についても説明します。

はじめに

同期 FIFO は、動作速度が高いため、高性能システム向けの理想的なオプションになります。また、同期 FIFO は、システム性能を改善し、複雑さを低減するための他の利点もあります。これらは、同期フラグ、ハーフ フル、プログラム可能なオールモストエンプティとオールモストフル フラグというステータス フラグを持っています。これらの FIFO はデータ幅拡張、深さ拡張、および再送信などの機能も含まれています。非同期 FIFO は、読み出しと書き込みパルスが外部クロック リファレンスなしで生成されることを必要とするのに対し、同期 FIFO は内部動作のタイミングを決めるためにフリーランニング クロックを使用しているため、高速モードでは非同期 FIFO より使いやすいです。

説明範囲

本アプリケーション ノートでは、同期 FIFO のアーキテクチャの概要、主な機能、使用ガイドライン、および一般的な応用について説明します。

本アプリケーション ノートは個々のサイプレスの同期 FIFO デバイスの機能を説明しませんが、全体的な概要を提供します。個々のデバイスの詳細については、サイプレスのウェブサイト (www.cypress.com) で関連するデバイス データシートを参照してください。

同期 FIFO のアーキテクチャ

同期 FIFO の基本的なビルディング ブロックは、メモリ アレイ、フラグ論理ブロック、および拡張論理ブロックです。図 1 に同期 FIFO の論理ブロック図を示します。メモリ アレイは、デュアルポート メモリ セルから構成されます。これらのセルは書き込みポートと読み出しポートへの同時アクセスを可能にします。この同時アクセスにより、FIFO は固有の同期特性があります。これらの両ポートのアクセスにはタイミングまたは位相の制限はありません。これは、1 つのポートを介して特定の速度でメモ

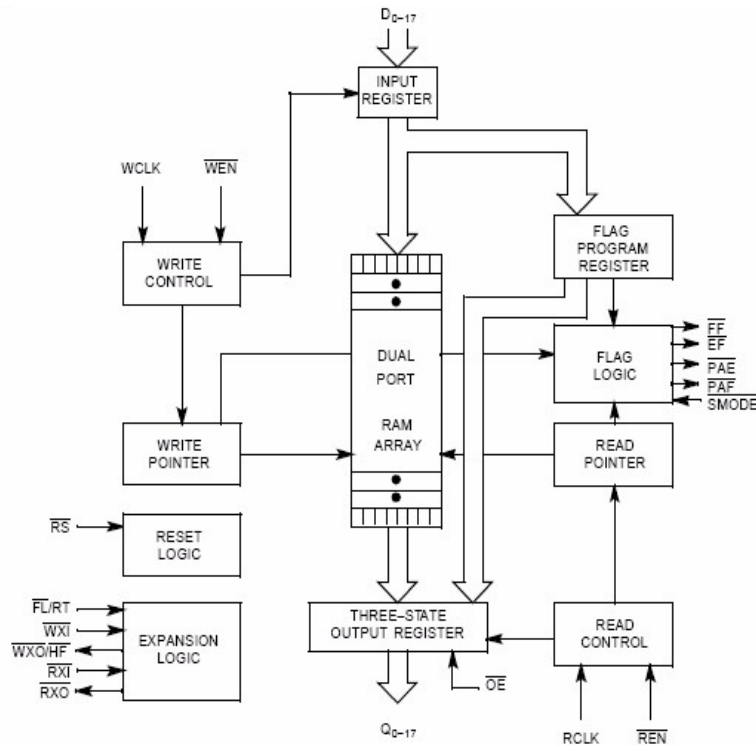
リに書き込んでいる間、他のポートを介して別の速度で読み出すことができるという独立動作を意味します。これは、データがメモリ アレイへ／から書き込まれる／読み出される速度を最適化することを可能にします。サイプレスは x9 ビット幅 CY7C42x5 と x18 ビット幅 CY7C42x5 の同期 FIFO を提供します。両方のデバイスは 66MHz と 100MHz での高速度動作に対応します。

データは読み出しアドレス ポインタと書き込みアドレス ポインタという 2 つのポインタによりメモリ アレイへ／から導かれます。各処理後、アレイ上の次のアドレスへ順次アクセス可能とするためにそれぞれのポインタがインクリメントされます。詳細については、同期 FIFO のチュートリアルを参照してください。

フラグ論理ブロックは 2 つのアドレス ポインタの値を比較します。2 つのポインタの差が 0 の場合、FIFO は空で、エンプティフラグがアサートされます。2 つの値の差がデバイスの深さに等しい場合、FIFO はフルで、フル フラグがアサートされます。ハーフ フル、プログラム可能なオールモストエンプティ フラグとオールモストフル フラグ等の他のフラグも、同じ方法で生成されます。プログラム可能なフラグはオフセット レジスタにプログラムされた値と FIFO のワード数を比較することで生成されます。

最後に、拡張論理は深さ拡大において複数のデバイスをカスケード接続することで論理的により深い FIFO を作るために使用されます。通常の「非深さのカスケード接続」動作では、各アドレス ポインタはその最大値に達するとゼロに戻ります。深さ拡張モードでは、アドレス ポインタが最大値に達すると、拡大ピンにパルスが駆動されます。これにより、トークンが他の FIFO に渡されます。トークンが渡された後、アドレス ポインタはそのトークンが戻るまでインクリメントしません。本質的には、書き込みまたは読み出し動作を処理する責任が他のデバイスに渡されます。深さが拡張されている場合はどんな時でも 1 つの FIFO のみが読み出し動作を処理し、1 つのみが書き込み動作を処理します。トークンが戻ると、アドレス ポインタはゼロにリセットされ、動作は再開します。

図 1. 論理ブロック図 – 同期 FIFO のアーキテクチャ (CY7C42x5)



リセット

電源投入後、FIFO をリセットする必要があります。デバイスをリセットすると、読み書きアドレス ポインタをゼロにセットし、出力データレジスタをクリアし、ステータス フラグがデバイスのエンプティ状態を示すようにフラグをセットします。デバイスは、 \overline{RS} ピンを LOW にアサートしてリセットされます。同期 FIFO は \overline{RS} の立ち下がりエッジを必要とします。これにより、プロセススーパーバイザ チップなどのデバイスは、 \overline{RS} を直接に駆動できます。 V_{CC} が変化している時、これらのデバイスはリセットをアサートし、 V_{CC} とすべてのクロックが安定する最短時間の間リセット信号を LOW に保持します。

\overline{RS} のアサート時、デバイスへの読み書き処理を実行しようとしてははいけません。これは、読み出しと書き込みイネーブル (\overline{REN} 、 \overline{WEN}) をデアサートする、または RCLK と WCLK の両方を LOW 状態にして実現できます。また、 \overline{RS} のデアサート (立ち上がり) エッジ後にリセット復元時間 t_{RSR} が満了するまで読み書き動作を無効にする必要があります。

注: リセットは非同期的動作で、実行のために、WCLK と RCLK の遷移を必要としません。

ステータス フラグ

エンプティ フラグ、プログラム可能なオールモストエンプティ フラグ、ハーフフル フラグ、プログラム可能なオールモストフル フラグ、およびフル フラグ (\overline{EF} 、 \overline{PAE} 、 \overline{HF} 、 \overline{PAF} 、 \overline{FF}) などのステータス フラグは FIFO のステータスを判定するために使用されます。これらのフラグは読み出しと書き込みアドレス ポインタの値を比較することで生成されます。外部制御論理ブロックはその FIFO で読み出し処理や書き込み処理が行えるか判定するためにこれらのフラグを使用します。また、FIFO 内のフラグ論理は空の FIFO からの読み出しとフル FIFO への書き込みを禁止します。空の FIFO を読み出す場合、出力は常にデバイスから最後に読み出された有効なデータを表示します。フルになった FIFO への書き込みは破棄されます。

エンプティ フラグ (\overline{EF}) とフル フラグ (\overline{FF}) は同期フラグで、すなわち、それらはそれぞれのクロックに同期します。エンプティ フラグ (\overline{EF}) は読み出しクロック (RCLK) に同期し、フル フラグ (\overline{FF}) は書き込みクロック (WCLK) に同期します。フラグをそれぞれのクロックに同期すると、外部同期が不要になります。通常、FIFO への書き込みを行う論理ブロックは書き込む前に FIFO がフルになっていないことを確認する必要があります。同様に、読み出し制御ロジックは FIFO から読み出す前にエンプティ フラグ (\overline{EF}) を確認します。プログラム可能なオールモス

トエンpty フラグ (**PAE**) とプログラム可能なオールモストフル フラグ (**PAF**) は CY7C42x1 FIFO では同期になります。**PAE** フラグは RCLK に同期化され、**PAF** フラグは WCLK に同期化されます。CY7C42x5 などの他の FIFO は、**SMODE** 制御信号を使用してプログラム可能なフラグの同期と非同期動作を可能にします。**PAE** と **PAF** フラグのプログラムおよびフラグの動作の詳細については、サイプレスのウェブサイト (www.cypress.com) に掲載しているデバイス データシートを参照してください。

読み出し制御論理ブロックと書き込み制御論理ブロックのどちらがこのフラグを使用するか決まっていなかったため、ハーフフル フラグ (**HF**) は非同期です。

非同期 FIFO としての動作

多くの場合、ユーザーはフリーランニングクロックを同期 FIFO の RCLK と WCLK ピンに接続したくはなく、データがデバイスへ／から入出力する時に、これらのクロックにパルスを入力します。これは同期 FIFO には許可動作モードですが、特別な注意が必要です。同期 FIFO を非同期 FIFO として使用するためには、同期 FIFO の RCLK と WCLK ピンにパルスを入力します。WCLK の各立ち上がりエッジではデバイスへにデータを書き込み、RCLK の各立ち上がりエッジではデータをデバイスから読み出します。読み出しと書き込みイネーブル (**REN**, **WEN**) は、リセットと再送信の場合を除いて、動作のすべての段階の間、LOW に駆動することが可能です。リセットと再送信間に読み書き処理が行われないように注意してください。すなわち、これは、**REN** と **WEN** を HIGH に駆動するか、RCLK や WCLK を LOW にして実現します。RCLK と WCLK を HIGH にすることは、(それぞれのイネーブル信号がアサートされた場合にのみ) 内部クロックパルスを発生させ、許可されません。

この動作モードの他の問題は、フラグ更新サイクルです。フリーランニングのクロックがデバイスに提供されないため、同期フラグは、いつでも自動的に更新されるわけではありません。エンptyな FIFO に何回か書き込まれたことを想定します。FIFO はエンptyではありませんが、「フラグ更新サイクル」がないため、**EF** は、まだアサートされます。ユーザーの立場から見ると、FIFO から最初のワードを読み出すのに 2 つの読み出しサイクルが必要であるように見えます: 最初のサイクルはフラグ更新サイクルで、2 番目のサイクルは、最初の読み出しを実行します。

フラグが更新された後、データが各 RCLK 立ち上がりエッジで FIFO から読み出されます。**EF** がアサートされる度にこの読み出しサイクルを追加することに注意してください。同様に、**FF** は、フルになった時もフラグ更新サイクルを必要とします。FIFO がフルになり、読み出しが実行される度に、次のデータ部分を書き込むために WCLK 立ち上がりエッジが 2 つ必要です。

フラグ更新サイクル

エンpty フラグとフル フラグは同期であるため、最新値に更新するために、それぞれのクロックの 1 つの立ち上がりエッジを必要とします。

境界条件 (フル、エンpty) では、「フラグ更新サイクル」として知られているデッド サイクルがあります。このデッド サイクルでは、フル フラグとエンpty フラグは、フラグ監視用制御ロジックが認識できるように最低 1 クロック サイクルの間アサートされます。

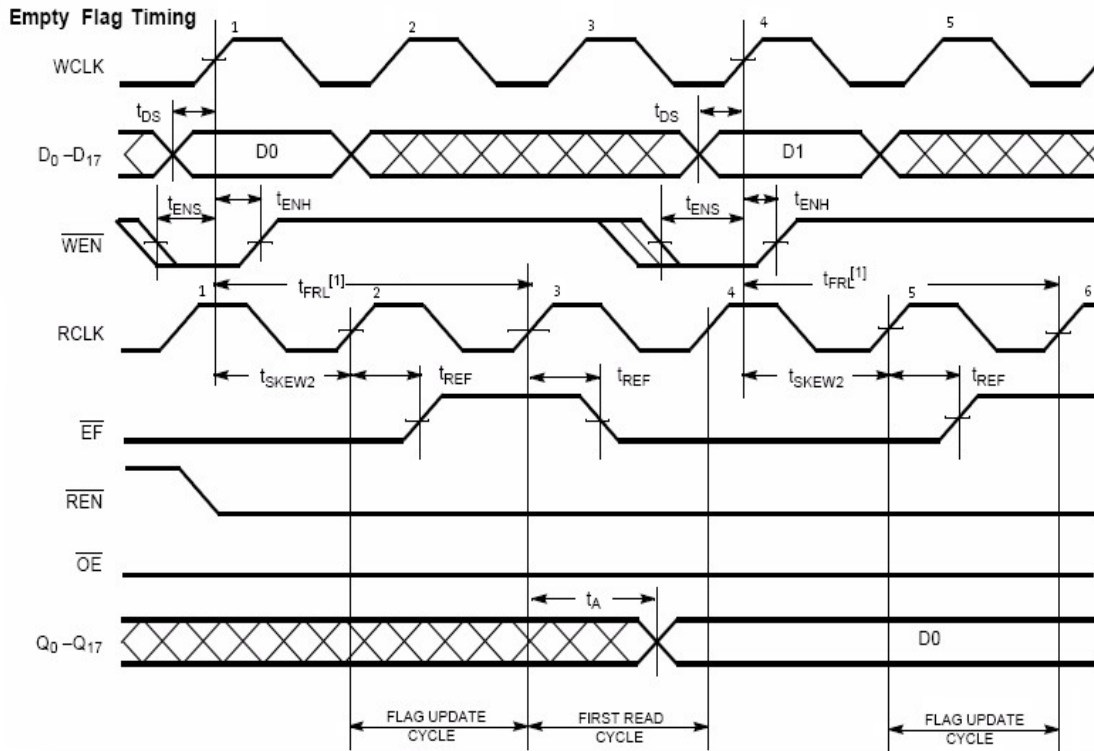
例えば、エンpty FIFO を使用していると仮定します。図 2 に示すように、WCLK のクロック サイクル 1 でこのデバイスへの書き込みは実行されます。よって、この時、このデバイスは、エンptyではありません。エンpty フラグ (**EF**) が読み出しクロックと同期するため、フラグは、デバイスが RCLK 立ち上がりエッジを受信するまでデアサートされません。デバイスへの読み出し処理は、**EF** がデアサートするまで防止されます。最初の RCLK 立ち上がりエッジ (最初の書き込みが完了 - RCLK のクロック サイクル 2) でフラグを更新し、第 2 の RCLK 立ち上がりエッジ (クロック サイクル 3) で最初のワードを読み出します。最初の RCLK サイクルは、デッド サイクルまたはフラグ更新サイクルと呼ばれます。この追加サイクルによって、エンpty フラグ (**EF**) のアサートとデアサートは、最低 1 サイクル行われることが保証されます。

RCLK と WCLK の非同期状態 (FIFO アプリケーションではよくある) の場合、デッド サイクルが無いと、フラグ アサートは非常に短い時間になってしまう場合があります。

RCLK と WCLK が同期したアプリケーションの場合、この「デッド」サイクルまたは「フラグ更新」サイクルは発生しないので、設計上考慮する必要はありません。読み出し制御ロジックはエンpty フラグ (**EF**) がデアサートされるまで、読み出しイネーブル (**REN**) を無視します。そのため、制御信号が RCLK サイクル 2 とサイクル 4 での読み出しのためにセットアップされてもデータは、データラインに表示されません。**EF** がデアサートされた後の RCLK の立ち上がりエッジでは、(RCLK クロック サイクル 6 で **REN** がアサートされる限り) FIFO から最初のワードを読み出します。

フリーランニング クロックを使用しないアプリケーションでは、RCLK は、最初のワードを読み出すために、LOW から HIGH まで 2 回 (1 回目の立ち上がりエッジは「フラグ更新サイクル」のためで、2 回目の立ち上がりエッジは、最初のワードを読み出すのため) 遷移します。

注: フラグ更新サイクルは、エンpty バウンダリとフル バウンダリの両方で発生します。**PAE** と **PAF** フラグに対応する「フラグ更新サイクル」はありません。

図 2. フラグ更新サイクル^[1]


¹ $t_{SKEW2} >$ 仕様内の最小値の場合、 t_{FRL} (最大値) = $t_{CLK} + t_{SKEW2}$ です。 $t_{SKEW2} <$ 仕様内の最小値の場合、 t_{FRL} (最大値) = $(2 \times t_{CLK} + t_{SKEW2})$ または $(t_{CLK} + t_{SKEW2})$ です。レイテンシ タイミングは、エンプティ バウンダリ ($\overline{EF} = \text{LOW}$) にのみ適用されます。

再送信

再送信機能は、FIFO から、以前に読み出されたデータ ブロックを読み出すために使用されます。この機能は、シリアル通信インターフェースによく使用されます。データ送信中にエラーが発生すれば、データ パケットは FIFO から再送信することができます。シリアル メディアを介して再送信することも可能です。

FIFO の再送信ピン (\overline{RT}) にパルスを入力して再送信機能にアクセスすることができます。 \overline{RT} ピンを LOW に駆動することにより、FIFO の読み出しアドレス ポインタは、物理位置 0 にセットされます。図 3 に再送信処理を示します。再送信機能が正しく実行されるために、再送信される可能性があるデータを FIFO に書き込む前に、FIFO をリセットする必要があります。

以下は例です。深さ 1000 ワードのデータ パケットを他の基板に送信したいとします。データを FIFO に書き込み、シリアル メディアを介してデータを送信するシリアル トランシーバに渡すことができます。最初、FIFO は、FIFO 内の読み出しと書き込みアドレス ポインタを 0 位置に設定してリセットされます。そして、1000 データ ワードは FIFO に書き込まれます。その後、 \overline{EF} がデアサートされ、シリアル トランシーバ デバイスは FIFO から読み出し始めます。

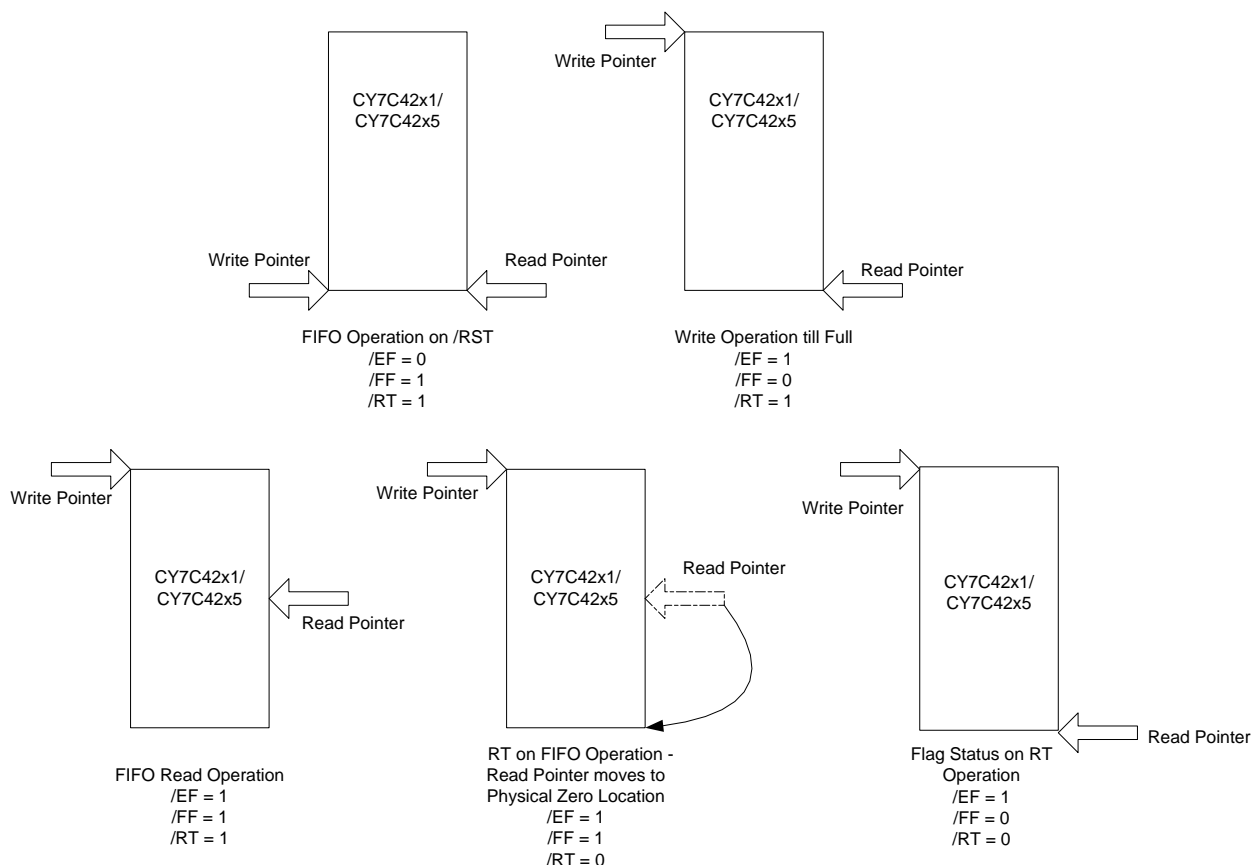
データは FIFO から読み出されるため、読み出しアドレス ポインタは位置 1024 に達するまでインクリメントし、FIFO がエンプティになります。データは、FIFO から読み出されましたが、FIFO から消去されないことに注意してください。読み出しプロセスの間、問題が発生した場合、 \overline{RT} ピンにパルスを入力し、読み出しアドレスポインタを位置 0 に設定して、データ パケットをその送信先に再送信することができます。このプロセスは、無限に繰り返すことができます。

\overline{RT} ピンは、読み出しアドレス ポインタを位置 0 にリセットするだけです。FIFO は、特定データ パケットのデバイス内の格納場所が分かりません。再送信する可能性があるデータ パケットをデバイスへ書き込む前に、デバイスをリセットしなければなりません。

FIFO の通常動作の間、または \overline{RT} ピンがデアサートされた後、FIFO は、読み出しと書き込みトランザクションを通常通り実行することができます。 \overline{RT} のアサート中には、読み出し処理や書き込み処理を実行することはできません。

注: 再送信は、非同期動作で、実行のために RCLK、または、WCLK の遷移を必要としません。

図 3. 再送信時の FIFO 動作



拡張のコンフィギュレーション

幅の拡張

幅の拡張機能は、幅がより広いデータパスを持つ FIFO を設計するために使用されます。2 個の $\times 18$ FIFO で幅を拡張して $\times 36$ FIFO を設計することなどもできます。幅が拡張される FIFO では、読み出し、書き込み、と再送信処理は通常と同様です。実際は、この時の FIFO は、実行されている現時点のモードについて識別もありません。

複数の FIFO の幅を拡張するために、フラグを組み合わせる「複合フラグ」を作成する必要があります。これは、FIFO 間でフラグを外部で AND 処理してできます。

複合フラグは、エンプティフラグとフルフラグのために生成されます。フラグを結合することにより、FIFO が同期された状態のまま(それぞれ同じワード数が入っている)になることを確保できます。図 4 を参照してください。

まず、幅が拡張された FIFO がどのように同期を失うかを理解する必要があります。この問題の根本原因は、クロック、RCLK、と WCLK の非同期関係に依存しています。例えば、2 つの FIFO の幅が拡張され、両方の FIFO が 1 つのワードを持っている(エンプティ状態から 1 つ目の位置)とします。読み出しと書き込みクロック間の非同期な位相関係により、FIFO への読み出し処理と書き込み処理がほぼ同時に行われる可能性があります。読み出し処理が先に実行される場合、このデバイ

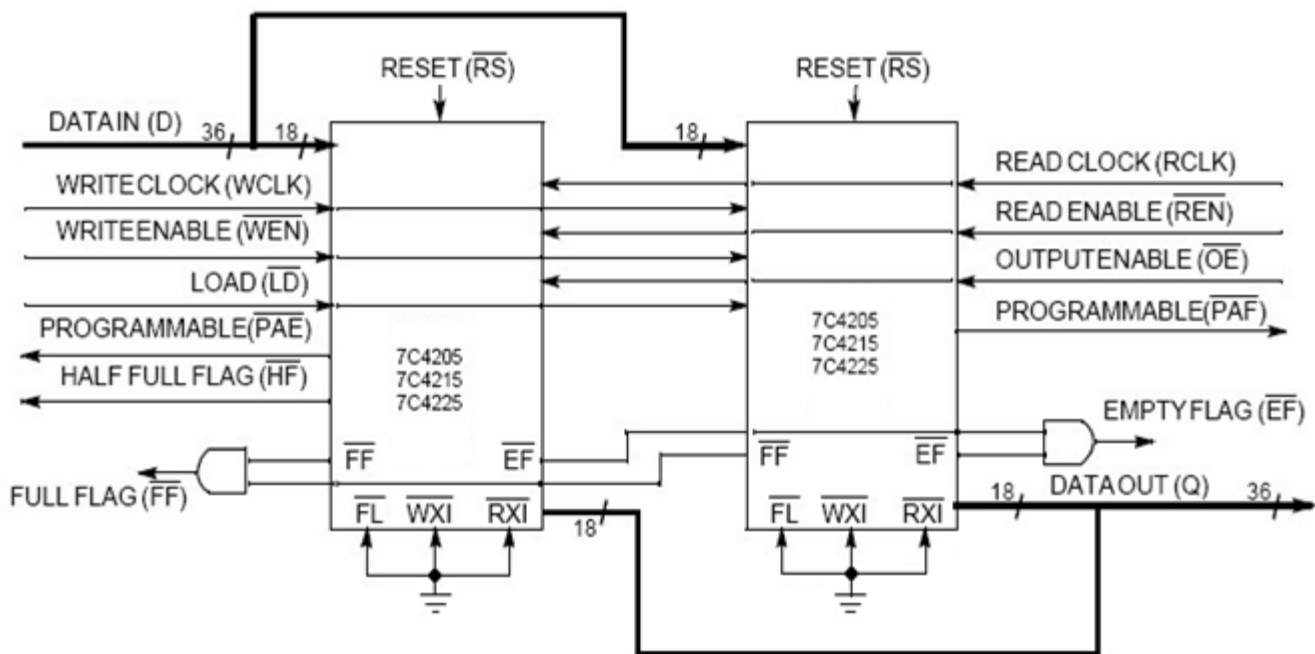
スは、ワードが書き込まれる直前にエンプティになります。この時、デバイスは、フラグ更新サイクルを待つため、基本的にこの 1 つの読み出しサイクルはなくなります。他の場合では、書き込み処理が読み出し処理の直前に行われた場合、デバイスは瞬時だけ 2 ワードを持ち、その後は 1 ワードのみ持つ状態に戻ります。FIFO がエンプティになることがないため、フラグ更新サイクルは不要です。

幅が拡張された FIFO において、このシナリオでは、異なった応答することがあるため、同期を失ってしまうかもしれません。小さなクロックスキューやデバイスプロセスの違いにより、読み出しを最初に行う FIFO もあれば、書き込みを最初に行う FIFO もあります。1 つ以上のデバイスがフラグ更新サイクルを必要とするため、FIFO は同期になりません。

FIFO のエンプティフラグを結合して複合エンプティフラグを作成することにより、いずれかの FIFO がエンプティの時に読み出しイネーブル ($\overline{\text{REN}}$) 信号をデアサートすることもできます。 $\overline{\text{REN}}$ が最低 1 クロックサイクルの間デアサートされる限り、すべてのエンプティ FIFO は、必要とするフラグ更新サイクルを取得し、FIFO が同期のままになります。

フルバウンダリではフルフラグにこの対応を適用することもできます。 $\overline{\text{REN}}$ と $\overline{\text{WEN}}$ を駆動する制御ロジックは、複合フラグのいずれかがアサートされた時にこれらのイネーブルをデアサートする必要があります。

図 4. 同期 FIFO (CY7C42x5) の幅拡張

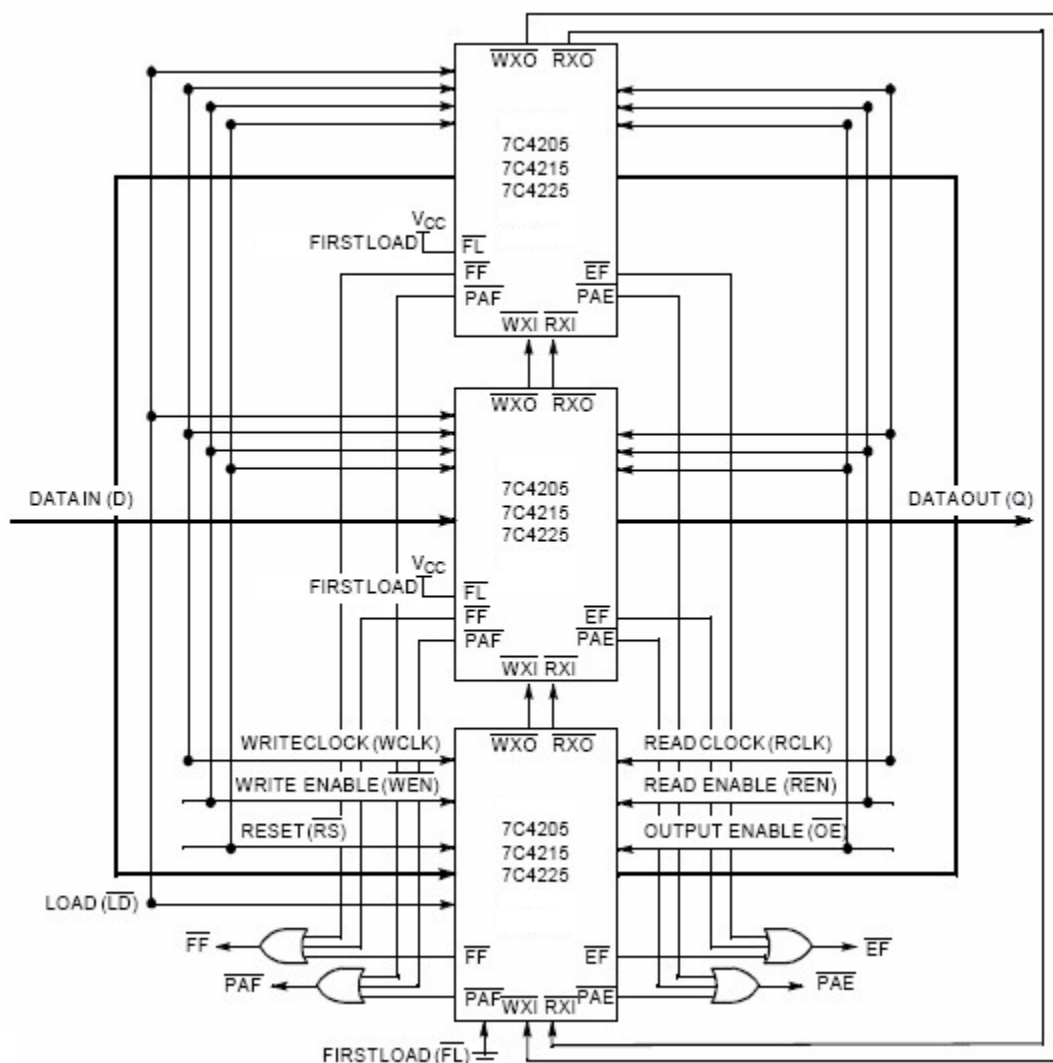


深さの拡張

深さの拡張機能は、複数の FIFO をスケード接続して、論理的により深い FIFO バッファを作成するために使用されます。同期 FIFO は、トークン パッシング方式で深さの拡張を行います。各 FIFO は並列接続され（読み出しと書き込みデータ バスが共有）WEN、REN、RCLK、WCLKなどの制御ラインを共有します。トークン チェーンでは、1 つのデバイスは、ファースト ロードピン (FL) をグランド接続して「一番目」のデバイスとして指定されます。残りのすべてのデバイスのFLは V_{CC} に接続しなけ

ればなりません。1 番目のデバイスの書き込みアドレス ポインタが最大値に達すると、拡張パルスがWXOピンへ駆動されます。1 番目のデバイスのWXOピンを、拡張パルスを認識し、後続の書き込み処理を実行する次のデバイスのWXIピンに接続します。同様に、読み出しアドレス ポインタが最大値に達すると、R XOを介してトークンを次のデバイスのR XIピンに渡します。CY7C42x5 FIFO の深さ拡張コンフィギュレーションについては、図 5 を参照してください。

図 5. 同期 FIFO (CY7C42x5) の深さ拡張



デバイスが正常に動作するために、複合エンpty フラグと複合フル フラグを生成する必要があります。複合フラグは、トークン パス チェーンで FIFO のすべてのフラグを外部で OR 処

理して生成されます。これにより、フル フラグがアサートされる前に、すべての FIFO がフルになることが確保されます。同様に、複合エンpty フラグがアサートされるために、すべての

FIFO がエンプティになる必要があります。**PAE**、**PAF**、および **HF** フラグは、深さ拡張モードに使用されません。これらのフラグは、特定 FIFO 内のワード数を正しく表示しますが、これらのフラグから複合フラグを作成すると、深さが拡張された FIFO バッファの全体の状態に関する情報を提供します。

深さ拡張は、深さ拡張ロジックなしの FIFO で実現することもできます。ピン ポン法は、複数 FIFO からのデータの読み書きを交互に実行するために使用されます。書き込みは、デバイス間で各トランザクションを実行して行われます。2 つの FIFO で深さが拡張される場合では、最初の書き込みは FIFO#1、2 番目の書き込みは FIFO#2、3 番目の書き込みは FIFO#1 へと続きます。読み出しも同じ方法で処理されます。

同期 FIFO のアプリケーション

FIFO は、異なった速度で動作するか、または非同期クロックソースを使用するプロセッサとペリフェラル デバイス間でのデータ転送中に発生する問題解決のための理想的なソリューションです。例えば、最新のプロセッサは、これらのプロセッサに接続されるペリフェラル デバイスより速いです。プロセッサとペリフェラル デバイス間でデータ交換を行う時に処理速度を減速する必要がないように FIFO を使用することができます。ペリフェラルがプロセッサより速い場合、問題を解決するために FIFO を使用することもできます。

コンピュータ ネットワークとデジタル電話回線の場合、データを複数のブロックに分割し、データラインで送信します。ブロックに分割され、データラインで高速で送信されるデータは、データ転送のために FIFO を必要とします。

FIFO は、HDTV/IPTV のセット トップ ボックスで使用されています。MPEG エンコーディング／デコーディングを実行し、オーディオ処理をするメイン CPU と受信信号を取り込む通信プロセッサである CPLD の間でのデータフローを制御します。

バス速度が不均整になる通常のシナリオは、異なったローカル バス周波数で動作する複数基板が他の基板と、またはより

高い速度のシステム バスを持つマスター コントローラーとデータを交換する必要があるデータ取得装置で発生します。FIFO の機能を使ってこの処理を可能にして、2 つの異なったクロックを入力と出力に使用することができます。これにより、データフローをローカル バス周波数と同期化することができます。この機能は、2 つの独立したポートへの制約のない同時アクセスが可能なデュアル ポート メモリ セルの使用により得られたものです。

FIFO は、異なった速度で実行するプロセッサ間で過度の待ち時間なくデータを転送するために、プロセッサ間通信で広く使用されます。FIFO は、テレコミュニケーション システムでビデオ／音声とデータ パケットなどの順序データをバッファリングするために使用されます。

FIFO は、アドレス付け機能を備えていないため、デュアル ポート メモリと区別されます。FIFO は、その名称通りに、順次アクセスのみをサポートするデータ バッファです。そのため、外部アドレス付けが不要になります。これにより、複雑さ、ピン数、基板の面積が削減されます。

要約

同期 FIFO は、異なった速度で動作する、または非同期クロックソースを使用するシステム間でのデータ転送に理想的です。これらの FIFO は、100MHz での高速動作をサポートし、最適な速度を達成するためにクロック入力をフリーランニング クロックに接続することにより同期モードに使用可能です。非同期 FIFO を必要とするデザインに簡単に統合するために、パルス状のクロックを入力し、これらの FIFO を非同期 FIFO として使用することもできます。再送信と同期フラグなどの機能により、これらのデバイスは、使いやすい多用途のものになります。これらの FIFO 同士を幅や深さでカスケード接続して、単一のデバイスには備えられていない FIFO コンフィギュレーションを作成することができます。そのため、サイプレスの同期 FIFO は、高性能のアプリケーションでのデータのバッファリングと同期化に理想的です。

改訂履歴

文書名: 同期 FIFO の理解 – AN1042

文書番号: 001-95839

版	ECN	変更者	発行日	変更内容
**	4722769	TOSI	05/14/2015	これは英語版 001-19979 Rev. *D を翻訳した日本語版 001- 95839 Rev. ** です。

ワールドワイドな販売と設計サポート

サイプレスは、事業所、ソリューション センター、メーカー代理店および販売代理店の世界的なネットワークを保持しています。お客様の最寄りのオフィスについては、[サイプレスのロケーション](#) ページをご覧ください。

製品

車載用	cypress.com/go/automotive
クロック & バッファ	cypress.com/go/clocks
インターフェース	cypress.com/go/interface
照明 & 電源管理	cypress.com/go/powerpsoc cypress.com/go/plc
メモリ	cypress.com/go/memory
光学式ナビゲーション センサー	cypress.com/go/ons

PSoC	cypress.com/go/psoc
タッチ センシング	cypress.com/go/touch
USB コントローラー	cypress.com/go/usb
ワイヤレス / RF	cypress.com/go/wireless

PSoC®ソリューション

psoc.cypress.com/solutions

PSoC 1 | PSoC 3 | PSoC 5

サイプレス開発者コミュニティ

コミュニティ | フォーラム | ブログ | ビデオ | トレーニング

本書で言及するその他すべての商標または登録商標は、各社の所有物です。



Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com

© Cypress Semiconductor Corporation, 2007-2015. 本文書に記載される情報は予告なく変更される場合があります。Cypress Semiconductor Corporation (サイプレス セミコンダクタ社) は、サイプレス製品に組み込まれた回路以外のいかなる回路を使用することに対して一切の責任を負いません。サイプレス セミコンダクタ社は、特許またはその他の権利に基づくライセンスを譲渡することも、または含意することはありません。サイプレス製品は、サイプレスとの書面による合意に基づくものでない限り、医療、生命維持、救命、重要な管理、または安全の用途のために使用することを保証するものではなく、また使用することを意図したものでもありません。さらにサイプレスは、誤動作や故障によって使用者に重大な傷害をもたらすことが合理的に予想される生命維持システムの重要なコンポーネントとしてサイプレス製品を使用することを許可していません。生命維持システムの用途にサイプレス製品を供することは、製造者がそのような使用におけるあらゆるリスクを負うことを意味し、その結果サイプレスはあらゆる責任を免除されることを意味します。

このソースコード (ソフトウェアおよび/またはファームウェア) はサイプレス セミコンダクタ社 (以下「サイプレス」) が所有し、全世界の特許権保護 (米国およびその他の国)、米国の著作権法ならびに国際協定の条項により保護され、かつそれらに従います。サイプレスが本書面によりライセンシーに付与するライセンスは、個人的、非独占的かつ譲渡不能のライセンスであり、適用される契約で指定されたサイプレスの集積回路と併用されるライセンシーの製品のみをサポートするカスタム ソフトウェアおよび/またはカスタム ファームウェアを作成する目的に限って、サイプレスのソースコードの派生著作物をコピー、使用、変更して作成するためのライセンス、ならびにサイプレスのソース コードおよび派生著作物をコンパイルするためのライセンスです。上記で指定された場合を除き、サイプレスの書面による明示的な許可なくして本ソース コードを複製、変更、変換、コンパイル、または表示することはすべて禁止します。

免責事項: サイプレスは、明示的または黙示的を問わず、本資料に関するいかなる種類の保証も行いません。これには、商品性または特定目的への適合性の黙示的な保証が含まれますが、これに限定されません。サイプレスは、本文書に記載される資料に対して今後予告なく変更を加える権利を留保します。サイプレスは、本文書に記載されるいかなる製品または回路を適用または使用したことによって生ずるいかなる責任も負いません。サイプレスは、誤動作や故障によって使用者に重大な傷害をもたらすことが合理的に予想される生命維持システムの重要なコンポーネントとしてサイプレス製品を使用することを許可していません。生命維持システムの用途にサイプレス製品を供することは、製造者がそのような使用におけるあらゆるリスクを負うことを意味し、その結果サイプレスはあらゆる責任を免除されることを意味します。

ソフトウェアの使用は、適用されるサイプレス ソフトウェア ライセンス契約によって制限され、かつ制約される場合があります。