

SoC 設計におけるソケットの重要性

この文書では、最新のシステムオンチップ(SoC)の設計問題という観点から *Open Core Protocol (OCP)* を紹介します。ここでは競争の激しい SoC 設計にとって業界標準のソケットインターフェースが不可欠である理由を説明し、OCP がどのようにしてこのようなインターフェースの要件を満たしているかを明らかにします。

ここでは、ますます短くなる製品化期間に対する要求に対応するために SoC 設計を加速することの必要性、および開発した IP コアをその後に再利用することの利点について考察します。

最後に、OCP が IP コア設計にもたらす柔軟性を示す 3 つの具体的な実装状況について検討します。

課題

長年にわたって半導体製造は絶え間なく進歩し、また市場からの圧力が上昇しているため、半導体業界においては製品化期間と設計の再利用が常に話題となっています。当然なこととして、SoC の開発期間を短縮すれば製品化期間も同時に短縮することができます。同様に設計の再利用についても簡単に言い表せます。つまり、1 度設計したものを繰り返し再利用できるということです。しかし、SoC 設計期間を短縮して設計の再利用を実際来实现するという事は非常に難しいことがわかっています。

これまで、製造技術の向上によって半導体の集積密度は 18 か月ごとに 2 倍ずつ上昇してきました(これはムーアの法則と呼ばれます)。この事実により、特定サイズの半導体ダイで実現できるゲート規模と機能は、製造コストがぐくわず増加するだけで著しく向上しました。たとえば過去 5 年間、半導体のゲート数は 20 万~50 万ゲートから、1,000 万ゲートを上回り 2,500 万ゲートにまで急増しています。これはゲート数が 50 倍も増加したことになり、設計者が SoC を製造できるようになった大きな理由です。

集積可能なゲート規模の増加と同時に、設計者は、初回の設計とその派生品の設計の両方において設計サイクルの期間短縮を試みてきました。これは競争の激しい市場の圧力に直接的に応えたものであり、設計期間の半減や頻繁な変更(継続的に製品ライフサイクルを短縮して機能を向上する必要があるため)が要求されています(表 1 を参照)。

	1997 年	1998 年	1999 年	2002 年	変化
プロセステクノロジー	0.35m	0.25m	0.18m	0.13m	~7 倍
ゲート数	20~50 万	100~ 200 万	400~ 600 万	1000~ 2500 万	~50 倍
設計サイクル(月)	12~18	10~12	8~10	6~8	~1/2 倍
派生品設計サイクル(月)	6~8	4~6	2~4	2~3	~1/2 倍

表 1. ゲート数の増大と設計サイクル期間の短縮
出典: *Surviving the SoC Revolution*, Chang 他

この数値は非常に簡単に言い表すことができます。つまり、50 倍の回路規模を持つ半導体を従来の半分の設計期間で設計するには、(実現可能ならば)100 倍の生産性があればいいということです。しかし実際には、この目標の達成は極めて困難であることが判明しています。さらに悪いことに、複雑化が絶えず進行し設計サイクルが短縮し続けているため、この命題は日に日に達成困難になっています。実際には、製品化期間とコアの再利用は、製品スケジュールや設計効率とともに常に犠牲となり続けています。

SoC設計期間の短縮

製品化期間の問題について取り組むため、まず SoC に組み込まれる個々のコアと最終的な SoC を並行して設計することの利点について考えてみましょう。並行して設計することにより、SoC シミュレーション(タイミングや性能解析など)を含むすべての設計の局面を並行して行えるため、企業は設計期間を短縮するための大きな可能性が得られることになります。

並行設計により、SoC の設計期間は最も時間のかかる単一要素の設計期間まで短縮することができます。ここでいう要素とは、個々の SoC コアの場合もあれば、SoC の統合作業の場合もあります。どちらの場合でも、開発スケジュールのリスクは限定され、短い開発スケジュールで満足のいく SoC を設計できる確率が高くなります。また、スケジュールを予測しやすくなります。すべての開発は範囲が限定され、すべての設計は並行して行われるため、問題が 1 つずつ順番に解決されるということはありません。つまり、問題はより迅速に発見されて解決されるということです。設計の流れは非常に予測しやすくなります。

ただし、上記のような並行開発を行うには、各コアおよび共有する SoC のリソースに対する責任部門を明確に定義しておく必要があります。なぜなら、コアはコア固有の機能を実行するだけで、システムについては何も知らないからです。たとえば、PCI インターフェースコアや MPEG 復元コアは固有の機能を実行しますが、インターコネク(=SoC 内部におけるコアの相互接続)メカニズムについては具体的に何も知りません。これと同様に、インターコネクメカニズムは、アービトレーション、アドレスマッピング、およびデータ移動などの通信に関する要素を処理しますが、コアが提供する機能については何も知りません。この方法は階層化と呼ばれ、すでに確立されており長年にわたって広く用いられています。

階層化はネットワーク空間でうまく利用されており、各層における責任レベルが定義されています。各層は、独自の機能を備えるとともに対話する相手層との間のインターフェースが明確に定義されています。ソフトウェアにおいても同じことが当てはまります。各関数やタスクは明確に定義された機能とインターフェースを備えています。歴史的に見て、階層化手法はさまざまな分野において素晴らしい成果を残しています。

階層化によるメリット

階層化では当然のこととして、システムが処理する要素は、これらの要素が存在するシステムから分離されます。要素は大規模なソフトウェアプログラム内のソフトウェアモジュールの場合もあれば、SoC 設計者にとってより重要な SoC 内の半導体 IP コアの場合もあります。どちらの場合でも原理はほとんど変わりません。また、階層化のもたらす利点も、以下のようにほとんどの場合で同じです。

- 設計サイクルの短縮
- 検証の単純化
- IP 利用性の向上

階層化によって設計チームは、設計作業を複数の作業に分割することができますが、分割によってこれらの作業は依存度が最小限となって同時処理が可能となります。通常、1 つの作業範囲を狭めることで、実装を正しく行う確率、およびそれを容易に検証できる確率が高まります。これによって最終製品の納期が格段に早まります。

また、階層化によって異なるシステムでの IP コアの再利用が必然的に可能となります。つまり、階層化手法が遵守されていれば、他のシステムリソースは、各 IP コアの機能を知らなくても、残りの要件を処理できます。各 IP コアのインターフェースはシステムから独立(分離)しているため、IP コアのインターフェースが正しく設計されていれば、同一インターフェースをサポートするその後のシステム設計で IP コアを再利用する場合も、インターフェースを変更する必要はありません。すべての IP コアはインターフェースを必要とするので、業界標準のインターフェースを選択することにより、再利用の手法のために余計な時間をかける必要がなくなります。さて、それでは IP コアインターフェースとは何でしょうか。この質問に対する答えは明白で、それはソケットです。

ソケット

何十年もの間、ローカルエリアネットワーク(LAN)は、現在 SoC 設計者が直面している問題と同じ問題に取り組んできました。最終的に LAN 設計者は、物理的接続と、その物理的接続を介して情報を交換するためのプロトコルから構成された明確なインターフェースを作成しました。このインターフェース規格が登場したことにより、その後のコンピュータ業界では、さまざまな機能を備えた独自開発のプラグアンドプレイ製品が製品化され、営利企業はこれらの製品を組み合わせ高度にカスタム化された LAN コンフィギュレーションを構築しました。このように、この手法は明確で実績のある手法です。

理想的なSoCソケットの要件

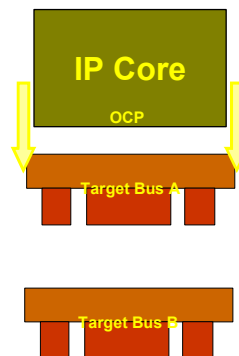
理想的には、SoC ソケットを利用することで、IP コア設計者は担当する IP コアの機能と各 IP コアに関連するインターコネクタ(SoC インターコネクタと USB、802.11b、SDRAM など)に専念することができます。同様に、SoC のシステムインテグレートは SoC のタイミング、IP コアのサービスバンド幅と要求されるレイテンシ、および IP コアの機能とは独立した最終的なフロアプラン設計に専念することができます。したがってソケットは、この明確な階層化の実現に必要な物理的プロトコルとデータ交換プロトコルを提供することになります。

これを達成するため、まず理想的な SoC ソケットとは、必然的に転送の実装(特定のバスやその他のインターコネクタ)に不可知であるべき点に留意してください。つまり、SoC コアは、インターフェースを介して IP コア間転送メカニズムに接続されますが、正確な転送メカニズム(コンピュータスタイルのバス、クロスバー、コンフィギュラブルなオンチップネットワークなど)は IP コアには知らされません。

上記の事項は極めて重要です。これを実施しなければ IP コア設計の内部に転送関連の情報を取り入れてしまい、いろいろな転送メカニズムを使用する SoC 設計での再利用の妨げとなるからです。この転送を意識させない手法を採用することにより実装の独立性を確保でき、システム設計者はシステムのニーズに最適なインターコネクタを選択できるようになります。

最後に、バンド幅要件が異なるため、理想的インターフェースでは、設計者がいろいろなディメンションに合わせてインターフェースコンフィギュレーションできることが必要です。このサイズとしてインターフェースのデータ幅があり、これはバンド幅要件を満たし、ハンドシェイクプロトコル、アクリッジの転送などのために必要となります。これにより、SoC 設計者は IP コアと SoC 設計を調整することが可能となり、複雑さと回路面積を最小限に抑えつつ、IP コアと SoC の要件に対応することができます。

- **OCP is a Core-Centric Protocol Interface:**
 - **Facilitates unrestricted delivery of ALL Core signals and features**
 - **Enables unconstrained interface bridge to ANY bus structure**



SoCソケット

これまで見てきたように、IP コアの再利用の可能性を最大限に引き出すためには、よく練られて規定された**コアセントリック**なプロトコルを固有のコアインターフェースとして採用する必要があります。発効している業界規格を選択することで、IP コア設計者は開発されたコアを所属する企業内で再利用できるだけでなく、IP コアのライセンス契約に基づいて、自社以外でも IP コアを再利用させることができます。この方法を使えば、サードパーティーの IP コアをライセンス取得して自社専用の

SoC 設計に組み込むことができ、自社の能力も最大限に高めることができます。つまり、IP ライセンスを通じて SoC を迅速に設計することが可能となり、利益を創出できるようになるということです。

さらに、IP コアインターフェースの仕様が厳密に定められるとともに、システムのインターコネクトが最適化されるおかげで、IP コア開発者は IP コア機能の開発に専念することができます。これによって接続先のエンドシステムに関する一般的な予備知識が不要になります(このようなエンドシステムは IP コアを利用するとともに、アプリケーションに存在する他の IP コアを利用している可能性もあります)。IP コアに必要なのは、システム要件から IP コアを分離するための実用的なインターフェースだけです。インターフェースはソケットの属性を想定します。ソケットは、業界で広く使用されている強力な費用のかからない接続インターフェースです。

この方法を通じて、システムインテグレータは階層化されたハードウェアを介してコンポーネントを分離することのメリットを実現します。設計者は無数の異なるコアプロトコルやコア間の通信手法に対応する必要はありません。標準の IP コアインターフェースを使用することで、各 SoC 統合の間に各 IP コアを適合させる必要がなくなり、システムインテグレータは SoC 設計に関する問題に専念することができます。また、IP コアは確実にオンチップのインターコネクトから分離され、したがってお互いに分離されるので、IP コアを交換するだけで、進化するシステムと市場の要求を容易に満たすことができます。

要約すると、本当に IP コアを再利用するためには、設計者は、SoC に IP コアを統合するときに IP コアに一切手を加えてはいけません。これを実現できるのは、たとえば、バス幅、バス周波数、またはバスの電氣的負荷が IP コアの修正を必要としない場合に限られます。すなわち、完全なソケットは、SoC インターコネクトメカニズムへの予測のできない変化や変更からコアを遮断してくれるのです。このようなソケットが存在することで、プロトコル、チェッカー、モデル、テストベンチ、およびテストジェネレータのためのツールやこれに対応する開発をサポートできるようになります。これによって、プラグアンドプレイのモジュール性を備えた独立した IP コアの開発が可能となり、コアをインターコネクトするための変更処理を行う必要がなくなります。また、こうすることでシステム設計と並行した IP コア開発が可能になり、貴重な設計の時間を節約することができます。

インターフェースソリューションの要件

IP コアのインターフェース設計要件はさまざまなので、すべての要件にひとつで対処できる特定の実装はありません。標準の IP コアインターフェース仕様は、以下の要件を満たす必要があります。

- 一連の要件にわたって対処できる拡張性がある
- 設計者が、いくつかのディメンション(バス幅、データのハンドシェイクなど)に合わせて特定のインターフェース実装を構成することができる
- データフローの信号以外に以下に対処することができる
 - エラー
 - 割り込み
 - フラグおよびソフトウェアによるフローコントロール
 - コントロールとステータス
 - テスト
- IP コアとシステム間の **すべての**信号を収集する

OCP概要

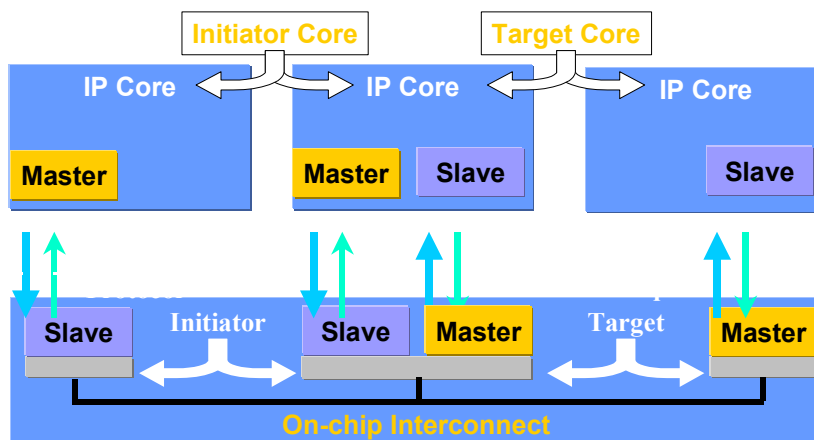
OCP とは、無償で利用できる **バス独立型**のプロトコルであり、すでに説明したすべての **コアセントリックな**要素を満たしています。特に OCP は、IP コアの通信要件を **すべて**完全に満たしています。高度にコンフィギュラブルなインターフェースですが、OCP はあらゆる状況に対応できる **万能**プロトコルではありません。むしろ、OCP は一般的な定義を共有する多くのプロトコルから構成されています。

OCP は、OCP の基本データセットを拡張するオプション機能を介してサイドバンド信号を明確にサポートしています。これらのサイドバンド信号の例を挙げると、リセット、割り込み、エラー、コントロール/ステータス情報などがあります。これに加えて、汎用フラグバスはどのような独特のコア信号のニーズにも対応しています。OCP テストインターフェースのオプションの拡張機

能は、スキャン、JTAG、およびクロックコントロールをサポートしており、SoC に統合するときのコアのデバッグや製造テストを実施できるようになります。

このため、システム設計者は、コア要件に正確に対応できるように特定の OCP コンフィギュレーションを調整することができます。OCP は、簡単なコンフィギュレーション手順を通じて、非常に単純で安価な OCP インターフェースにより、単純で低パフォーマンスの IP コアをサポートしています。その一方で、より複雑なインターフェースを備えた複雑で高パフォーマンスな IP コアもサポートしています。

こうして、IP コア開発者は OCP インターフェースを使用して IP コア設計を**完成**することができます。OCP 以外のエンドアプリケーションに関する知識は必要ないので、多くの場合グローバルな設計チームのメンバーが完全に独立して作業することができます。また、システムインテグレータは、アプリケーションのシステム要件に最適なオンチップインターコネクタを自由に選択し、効率よくそのインターコネクタを「ラップ」することで OCP インターフェースを IP コアに接続できます。



OCP-IP のメンバーには、OCP プロトコル準拠の環境として CoreCreator[®] ツールが提供され、また IP コアの効率的な再利用に必要な事項を説明した「パッケージ」が提供されます。すべての OCP-IP メンバーは無償でこれらを利用できます。

OCP コア インターフェースの例

以下の例では、機能的に全く異なる 3 種類の IP コアがどのように OCP インターフェースを使用しているかを示しています。以下がその 3 つの例です。

1. バスブリッジ
2. プロセッサインターフェース
3. メモリーインターフェース

最初に、各 IP コアの OCP インターフェースについて個別に説明します。次に、その他の共通信号について提示します。

信号名は OCP プロトコルの一部です。信号の完全な説明については『*Open Core Protocol Reference*』マニュアルを参照してください。OCP 規格書は www.ocpip.org にて無償でご利用いただけます。

要約すると、OCP 接続には「マスター」エンティティと「スレーブ」エンティティがあります。以下に OCP の用語とプロトコルについての簡単なコメントを記します。

- マスターは、名前が文字 *M* で始まるすべての信号をドライブする
- スレーブは、名前が文字 *S* で始まるすべての信号をドライブする
- プロトコル用の簡単なハンドシェイクがある

- マスターとスレーブはどちらもフロー制御をアサートできる
- すべての転送と信号は OCP クロックの立ち上がりエッジに同期する

例1-バスブリッジ

バスブリッジは PCI、USB、またはその他のバス規格を OCP にインターコネクトするものです。コントローラは (SoC の) 外部に PCI または USB インターフェースを備えており、SoC 内部インターフェースは OCP になります。

バスブリッジは通常、SoC インターコネクトにおいてマスターとスレーブの両方の役割を果たします。マスターはバストラフィックを所望の場所へ送信し、スレーブはバスブリッジ内部の制御レジスタまたはステータスレジスタをリード/ライトします。

スレーブは簡単な OCP インターフェースを備えているので、多くの場合、少しのサイドバンド信号だけを必要とします。この例で予想されるスレーブコアの OCP 信号セットは、以下のとおりです。

- MCmd マスターのコマンド(リード/ライトなど)
- MAddr マスターのアドレス(最大 32 ビット)
- MData マスターのデータ(データライト: 8、16、32、64、128 ビット幅)
- SCmdAccept スレーブのコマンドのアクセプト
- SResp スレーブのレスポンス
- SData スレーブのデータ(データリード。サイズは MData と同じであること)
- SError スレーブのエラー。ブリッジからのエラー
- SInterrupt スレーブの割り込み。ブリッジからの割り込み
- Control バスブリッジ用のコントロールビット
- Clk クロック信号
- Reset_N リセット信号

この例のインターフェースは 1 つしかスレーブ割り込みを必要としないので、インターフェースはこのスレーブ割り込みを使用します。複数の割り込みが存在する場合は、SFlags はさらに最大 8 つの割り込みを追加で提供することができます。

バスブリッジのマスターは、以下の信号を使用できます。

- MCmd マスターのコマンド
- MAddr マスターのアドレス(最大 32 ビット)
- MData マスターのデータ(データライト、8、16、32、64、128 ビット幅)
- MBurst マスターのバースト
- SCmdAccept スレーブコマンドのアクセプト
- SResp スレーブのレスポンス
- SData スレーブのデータ(データリード。サイズは MData と同じであること)
- Clk クロック信号
- Reset_N リセット信号

PCI では、オプションの OCP スレッド信号によってインターフェースを拡張できます。このおかげで、インターフェースはコンカレンシーおよび転送のアウトオブオーダー処理をサポートすることができます。オプションの OCP コンプレックスエクステンション信号は複数のスレッドをサポートしています。異なるスレッド内のトランザクションについては、順序に関する要件はなく、アウトオブオーダー処理ができます。単一スレッドのデータフロー内では、すべての OCP 転送はその順序を維持する必要があります。PCI インターフェースのスレッドは、異なるメモリーや I/O 領域にアクセスするのに役に立ちます。

- MThreadID マスタースレッド識別子(最大 16 種類のスレッド)
- SThreadID スレーブスレッド識別子(最大 16 種類のスレッド)

例2-プロセッサインターフェース

プロセッサインターフェースは通常、OCP マスターだけを必要とします。信号はバスブリッジのマスターに似ていますが、通常 1 ワード未満の転送ではバイトイネーブル信号を含んでいます。

この例で予想されるコアの OCP 信号セットは、以下のとおりです。

- MCmd マスターのコマンド
- MAddr マスターのアドレス(最大 32 ビット)
- MData マスターのデータ(データライト: 8、16、32、64、128 ビット幅)
- MBurst マスターのバースト
- MByteEn マスターのバイトイネーブル
- SCmdAccept スレーブのコマンドアクセプト
- SResp スレーブのレスポンス
- SData スレーブのデータ(データリード。サイズは MData と同じであること)
- SError スレーブのエラー。プロセッサへの入力
- SInterrupt スレーブの割り込み。通常は NMI ピン
- SFlag スレーブのフラグ。プロセッサへのその他の割り込み(最大 8 フラグ)
- Clk クロック信号
- Reset_N リセット信号

新しく開発されたプロセッサの中にはコンカレント命令やデータキャッシュのフェッチミスをサポートするものもあります。OCP スレッドは直接これらの機能をサポートしています。このため、このようなプロセッサ用にメモリーのコンカレント処理を拡張するには、以下の信号が必要です。

- MThreadID マスタースレッド識別子(最大 16 種類のスレッド)
- SThreadID スレーブスレッド識別子(最大 16 種類のスレッド)
- MThreadBusy マスタースレッドビジー
- SThreadBusy スレーブスレッドビジー

マスターとスレーブのスレッドビジー信号は、各スレッドのフローコントロールを許可します。プロセッサは複数のフェッチを未処理のまま保持できますが、そのすべてを同時に処理するリソースがない場合があります。このため、スレッドビジー信号によって OCP マスタープロセッサまたはターゲットのスレーブは必要に応じて転送フローをコントロールできるようになります。

例3-メモリーサブシステム

メモリーサブシステムは、DRAM、DDR、SRAM、または FLASH に接続されます。高バンド幅の利用率を維持するために OCP 信号は OCP コンプレックスエクステンションをいくつか必要とします。メモリーサブシステムはほとんどの場合マルチスレッドで、複数のメモリーバンクを処理します。またリクエストを効率的に処理するため、メモリーコントローラにはバーストやバイトイネーブルのような OCP シンプルエクステンションも備わっています。

この例で予想されるコアの OCP 信号セットは、以下のとおりです。

- MCmd マスターのコマンド(リード/ライトなど)
- MAddr マスターのアドレス(最大 32 ビット)
- MData マスターのデータ(データライト: 8、16、32、64、または 128 ビット幅)
- MBurst マスターのバースト
- MByteEn マスターのバイトイネーブル
- SCmdAccept スレーブのコマンドアクセプト
- SResp スレーブのレスポンス
- SData スレーブのデータ(データリード。サイズは MData と同じであること)

- MThreadID マスタースレッド識別子 (最大 16 種類のスレッド)
- SThreadID スレーブスレッド識別子 (最大 16 種類のスレッド)
- MThreadBusy マスタースレッドビジー
- SThreadBusy スレーブスレッドビジー
- Clk クロック信号
- Reset_N リセット信号

また、メモリーサブシステムは、Mdata と Sdata 上で、メモリーサブシステムが接続されるメモリーよりも広いビット幅を利用します。こうすることで各システムのインターコネクト転送が最も効率的になります。

各例に共通する信号

上記の例ではそれぞれ、スキャン信号と JTAG 信号を用いることができます。これらのテスト信号とスキャン信号は各 IP コアに共通していますが、スキャンチェーンの数は異なる場合があります。OCP ではテスト構造をインターフェースの一部として認めていますが、個別のエンティティとしては認めていません。これは、ソケットと単なるアドレス指定データフローを対比し、検討する場合の最後の項目になります。

スキャンおよび JTAG で各例のコアにアクセスするには、以下の信号が必要です。

- ScanCtl スキャン制御
- ScanIn スキャンイン (最大 256 のスキャンイン用ポートまたはチェーン)
- ScanOut スキャンアウト (最大 256 のスキャンアウト用ポートまたはチェーン)
- ClkByp クロックバイパス。ノーマルクロックではなくテストクロックを使用
- TestClk テストクロック
- TCK JTAG テストクロック、すべての信号について IEEE 1149.1 の定義を使用
- TDI JTAG テストイン
- TDO JTAG テストアウト
- TMS JTAG テストモード選択
- TRST_N JTAG リセット

結論

ソケットコアの標準プロトコルは、SoC 設計コミュニティにおいて極めて重要です。OCP は、唯一フルサポートされた完全で実績のあるソケットです。今すぐ OCP を採用することで、非互換のソリューションや独自のソリューションが普及することを防ぎ、商用およびレガシーの IP コア用に利用可能な全マーケットを展開していくことができます。

完全でフルサポートされたコアセントリックな OCP は、バスセントリックな旧形式のプロトコルと比較して大幅に上回る明白な利点をもたらします。OCP はコアセントリックでオープンライセンス方式の、ライセンス料が無償のコアインターフェースプロトコルです。OCP はコア固有の機能を制限したり、または妨げたりすることはありません。OCP はスケーラブルでコンフィギュラブルなので、いろいろな IP コア設計と SoC 設計に伴うさまざまな通信要件に対応することができます。

OCP インターフェースの IP コアと OCP インターコネクトシステムは、真のモジュール方式のプラグアンドプレイ統合を実現しているので、システムインテグレータは最適な IP コアと最高のアプリケーションインターコネクトシステムを選択することができます。これにより IP コア設計者とシステム設計者は、並行して作業することができるため、設計期間を短縮することができます。さらに、IP コア内にシステムロジックを持たないため、IP コアを作り直すために余計な時間を費やすことなく IP コアを再利用することができます。

最後に、検証およびテスト一式が OCP 仕様によって記述されていれば、複数の設計にわたってこれらを使用できるため、特定のインターフェースブリッジに合わせた小さな調整すらほとんど必要ありません。

OCP 仕様は、www.ocpip.org で無償でご利用いただけます。