ChipScope Pro Integrated Logic Analyzer (v.1.00a、1.01a、1.02a、1.03a)

DS299 2010 年 4 月 19 日

製品仕様

はじめに

ChipScope™ Pro Integrated Logic Analyzer (ILA) コアは、カスタマイズ可能なロジック アナライザ コアで、デザインの内部信号をモニタするために使用されます。ILA コアには、ブール トリガー方程式、トリガー シーケンス、およびストレージ クオリフィケーションなどの近代的なロジック　アナライザのアドバンス機能が多く含まれています。ILA コアはモニタ中のデザインに同期するので、このコア内のコンポーネントにも、デザインに指定したすべてのクロック制約が適用されます。

機能

- ChipScope Pro ICON コアを介して、ChipScope Pro Analyzer ソフトウェアとキャプチャ コアの通信パスを提供
- トリガー幅、データ幅、データ深さをユーザーが選択可能
- 1 つのトリガー条件やシーケンスにまとめることが可能な複数のトリガー ポートを含有
- ある一定の条件が満たされた場合にのみ、コアにサンプルを格納するストレージ クオリフィケーション オプションを含有

ILA　コアの詳細は、『ChipScope　Pro　ソフトウェアおよびコア ユーザー ガイド』を参照してください。

LogiCORE IP に関する情報				
コアの内容				
サポートされるデバイス ファミリ (1)	Spartan®-3、Spartan-3E、Spartan-3A、Spartan-3A DSP、Virtex®-4、Virtex-5			
使用リソース (2)	I/O	LUT	フリップフロップ	ブロック RAM
	0	235	234	2
特別機能	なし			
コアに含有されるもの				
マニュアル	製品仕様			
デザイン ファイル フォーマット	なし			
制約ファイル	なし			
検証	なし			
インスタンシエーション テンプレート	Verilog および VHDL ラッパー			
リファレンス デザイン/アプリケーション ノート	なし			
その他の項目	信号名インポート ファイル (.cdc)			
デザイン ツール要件				
ザイリンクス インプリメンテーション ツール	ISE® 12.1			
検証	ChipScope Pro 12.1			
シミュレーション	シミュレーションでのサポートなし			
合成	XST で合成されたネットリスト			
サポート				
ザイリンクスによるサポートあり				

1. これらの FPGA ファミリの派生デバイスも含みます。
2. これらは、1 つの 8 ビット ベーシック トリガー ポートの深さ 512 サンプルの Virtex-4 デバイス ファミリが使用されたと仮定して概算されています。

アプリケーション

ILA コアは、ChipScope Pro ソフトウェアおよびコアを使用して検証またはデバッグする必要のあるアプリケーションで使用されるように設計されています。

ファンクションの詳細

FPGA デザインの信号は ILA コア入力に接続され、デザイン速度でキャプチャできます。デザインがインプリメントされる前に、キャプチャする信号数およびサンプル数などのコアのパラメータを選択します。ILA コアとの通信は、次の図のように、ICON コアを介した JTAG ポートへの接続を使用して行われます。

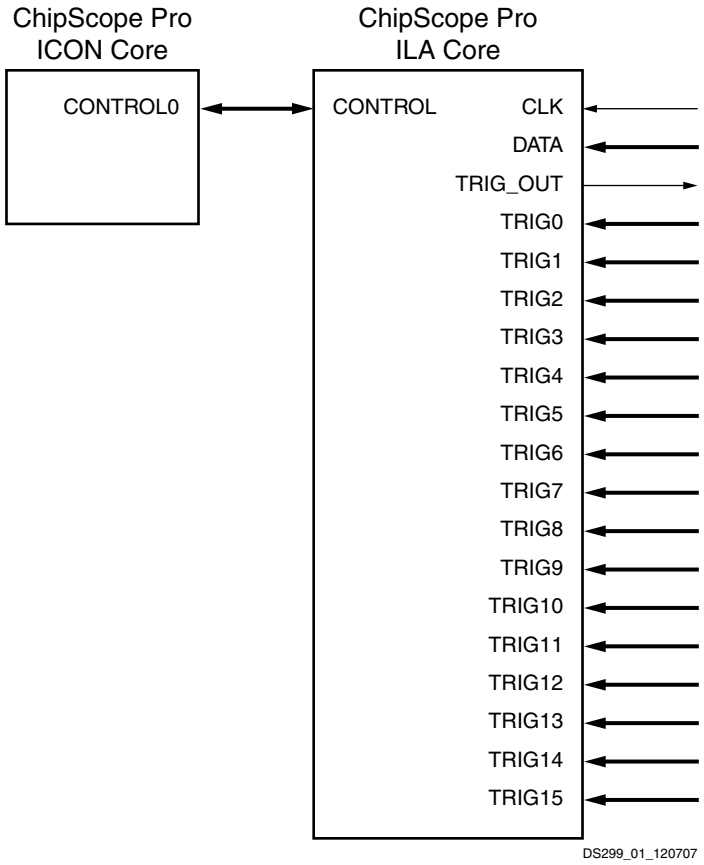

図 1 : ILA コアと ICON コアの接続

デザインを FPGA のボードに読み込むと、ChipScope Pro Analyzer ソフトウェアを使用して ILA コアに接続された信号をいつ、どのようにキャプチャするか定義するトリガー条件を設定できます。トリガーが発生し、サンプル バッファが一杯になったら、データ バッファが ChipScope Pro Analyzer ソフトウェアにアップロードされます。このデータは波形またはリスト形式で表示できます。

マッチ ロジック、キャプチャ コントロール、ステータス機能をインプリメントには、通常の FPGA ロジックが使用されます。オンチップ ブロック RAM メモリには、ソフトウェアでアップロードされるまでデータが格納されます。イベントのトリガー、データのキャプチャ、ILA コアとの通信には、ユーザー入力や出力は必要ありません。

トリガー

ILA　に入力されるデータをキャプチャするタイミングを選択する必要があります。このタイミングは、トリガー イベントと呼ばれます。

トリガー ポート

デザインで異なるタイプの信号またはバスをモニタできるようにするには、複数のトリガ ポートが必要となります。たとえば、デザインで制御、アドレス、およびデータ信号を含むシステム バスを使用している場合、これらに別々のトリガー ポートを割り当てて、各信号グループをモニタできます。異なる信号およびバスを 1 つのトリガー ポートに接続すると、アドレス バスが指定された

範囲内にあるかを確認している間、CE、WE、および OE 信号の各ビット遷移はモニタできません。異なるタイプのマッチ ユニットから選択可能であるため、使用リソースを最低限に維持しながら、必要なトリガー用に ILA コアをカスタマイズできます。

マッチ ロジック

トリガー ポートには、それぞれ 1 つまたは複数のマッチ ユニット (実際の比較を行うロジックのブロック) を含めることができます。マッチ ユニットには複数のタイプがあります。最もシンプルなマッチ ユニットは = (equals) または != (not equals) 比較のみ実行でき、FPGA リソースも最小の容量しか使用しません。最も複雑なマッチ ユニットは、=、!=、>、<、>=、<= および範囲比較などすべてのタイプの比較を実行できます。マッチ ユニットには、立ち上がりエッジ、立ち下がりエッジ、またはどちらかのエッジなど、エッジ検出を含めるように設定もできます。

表 1 : マッチ ユニット タイプ

マッチ ユニット名	説明
Basic Basic with Edges	Basic マッチ ユニットでは、= および != 比較のみが実行できます。Basic with edges では、R (立ち上がり)、F (立ち下がり)、B (いずれか) エッジでマッチできます。
Extended Extended with Edges	Extended マッチ ユニットでは、=、!=、>、<、>=、および <= 比較が実行できます。Extended with edges では、= および != 演算子のエッジ マッチができます。
Range Range with Edges	Range マッチ ユニットでは、Extended マッチ ユニットと同じ比較が実行できるほか、値の範囲や特定範囲以外の値のマッチができます。

マッチ ユニット カウンタ

特定のトリガー ポートのマッチ ユニット グループは、それぞれの出力にマッチ カウンタを付けるようにコンフィギュレーションすることも可能です。マッチ カウンタは、ある数のイベントをシーケンシャルまたはシーケンシャル以外で検出するために使用できます。カウンタのサイズは生成時に決まります。カウントするイベント数とそれらをシーケンシャルにカウントするかどうかは、ランタイム時に決定されます。

トリガー条件とストレージ クオリフィケーション

ILA コアは、トリガーおよびストレージ クオリフィケーション条件ロジックの両方をインプリメントします。トリガー条件は、コアのトリガ ポートに接続されたマッチ ユニット コンパレータで検出されたイベントのブール式またはシーケンシャルな組み合わせです。トリガー条件は、データ キャプチャ ウィンドウで明確な開始点を印付けるために使用され、データ キャプチャ ウィンドウの開始点、終了点、あるいは任意の位置を指定できます。

同様に、ストレージ クオリフィケーション条件も、その後にコアのトリガー ポートに接続されるマッチ ユニット コンパレータで検出されるイベントのブール式組み合わせです。ただし、ストレージ クオリフィケーション条件は、個別のデータ サンプルをキャプチャおよび格納するかを決定するために、トリガー ポートマッチ ユニットのイベントを評価する点でトリガー条件と異なります。トリガー条件およびストレージ クオリフィケーション条件を共に使用すると、キャプチャ プロセスの開始時とキャプチャするデータを決定できます。

ILA コアの例

次を実行すると仮定します。

- アドレス 0xFF0000 への最初の書き込みサイクル (CE = 立ち上がりエッジ、WE = 1、OE = 0) でトリガー
- データ値が 0x00000000 ～ 0x1000FFFF の間の場合に、アドレス 0x23AACC からのメモリ読み出しサイクル (CE = 立ち上がりエッジ、WE = 0、OE = 1) のみをキャプチャ

これらの条件を正しくインプリメントするには、TRIG0 および TRIG1 トリガー ポートの両方にそれぞれ比較ユニット 2 個 (トリガー条件用 1 個とストレージ クオリフィケーション条件用 1 個) が接続されていることを確認する必要があります。上記の条件を満たすトリガーとストレージ クオリフィケーションの式と各マッチ ユニットを設定するには、次を設定する必要があります。

- トリガー条件 = M0 && M2
  - M0[2:0] = CE、WE、OE = R10 (R は立ち上がりエッジ)
  - M2[23:0] = アドレス = 0xFF0000
- ストレージ クオリフィケーション条件 = M1 && M3 && M4
  - M1[2:0] = CE、WE、OE = R10 (R は立ち上がりエッジ)
  - M3[23:0] = アドレス = 0x23AACC
  - M4[31:0] = データ = 範囲は 0x00000000 ～ 0x1000FFFF

ILA コアのトリガー条件およびストレージクオリフィケーション条件により、オンチップ メモリ リソースを無駄にすることなく、必要な情報の位置を正確に把握し、キャプチャできます。

ILA トリガー出力ロジック

ILA コアは TRIG_OUT と呼ばれるトリガ出力ポートをインプリメントします。TRIG_OUT ポートは、ChipScope Pro Analyzer を使用してランタイムに設定されたトリガー条件の出力です。トリガー出力の形 (レベルまたはパルス) とアクティブ エッジ (High または Low) も、ランタイム時に制御できます。入力トリガー ポートに対する TRIG_OUT のレイテンシは、10 クロック サイクルです。

TRIG_OUT ポートは非常に柔軟性があり、多用途に使用できます。このポートをデバイス ピンに接続すると、オシロスコープおよびロジック アナライザなどの外部テスト装置をトリガーすることが可能です。また、デバイスに組み込まれた PowerPC® または MicroBlaze™ プロセッサの割り込みラインに接続すると、ソフトウェア イベントを発生させることができます。

さらに、あるコアの TRIG_OUT ポートを別のコアのトリガ入力ポートに接続すると、オンチップ デバッグ ソリューションのトリガーおよびデータ キャプチャ機能を拡張できます。

ILA データ キャプチャ ロジック

各 ILA コアは、オンチップ ブロック RAM リソースを使用し、デザイン内のその他すべてのコアから独立してデータをキャプチャできます。また、各 ILA コアは、ウィンドウおよび N サンプルのいずれかのキャプチャ モードでデータをキャプチャできます。

ウィンドウ キャプチャ モード

ウィンドウ キャプチャ モードでは、サンプル バッファが 1 つまたは複数の等サイズのサンプル ウィンドウに分割されます。このモードの場合、1 つのトリガー条件イベント (個々のトリガー マッチ ユニット イベントのブール式組み合わせ) を使用して、サンプル ウィンドウを満たすのに十分なデータを収集します。

サンプル ウィンドウのワード数が 131,072 サンプルまでの 2 の累乗の場合、トリガー位置はサンプル ウィンドウの開始点 (最初にトリガーし、データを収集)、終了点 (トリガー イベントまでデータを収集)、あるいはそれら 2 点間の任意の位置に設定できます。一方、ウィンドウのワード数が、2 の累乗以外の場合、トリガー位置はサンプル ウィンドウの開始位置にのみ設定できます。

サンプル ウィンドウが満たされると、トリガ条件が自動的に再設定され、ILA コアはトリガ条件イベントのモニタを継続します。このプロセスは、サンプル バッファのすべてのサンプル ウィンドウが満たされるか、ユーザーが ILA コアを停止するまで繰り返されます。

N サンプル キャプチャ モード

N サンプル キャプチャ モードはウィンドウ キャプチャ モードと類似していますが、次の 2 点が大きく異なります。

- ウィンドウごとのサンプル数は、1 ～ サンプル バッファ サイズから 1 を引いた値の範囲で、任意の整数 N に設定可能
- トリガ位置は、常にウィンドウにおける位置 0 とする

N サンプル キャプチャ モードは、キャプチャ ストレージ リソースを無駄にせず、各トリガで必要とする正確なサンプル数をキャプチャする場合に有効です。

トリガー マーク

トリガー イベントと一致するサンプル ウィンドウ内のデータ サンプルには、トリガー マークが付けられます。このトリガー マークによって、ウィンドウ内のトリガー位置が ChipScope Pro Analyzer に伝えられます。トリガー マークは、サンプル バッファ内の 1 サンプルに対して 1 ビットを使用します。

データ ポート

ILA コアを使用すると、トリガー機能を実行しているトリガー ポートとは別のポート上のデータをキャプチャできます。この機能は、コアのトリガーに使用される情報と同じ情報のキャプチャおよび確認が有用でなく、キャプチャするデータ量を比較的限られた量に制限する際に役立ちます。

ただし、通常は、コアのトリガーに使用されるデータと同一データのキャプチャおよび確認が有用です。このような場合、データが 1 つまたは複数のトリガー ポートで構成されるように選択できます。この機能により、キャプチャに必要となるトリガー情報を選択できる柔軟性を活用しながら、リソースの節約が可能になります。

ワード数

サポートされるアーキテクチャによって使用できるブロック RAM の深さが異なるため、次の表に示すように、データの深さもそれぞれ異なります。

表 2 : 使用可能な ILA のデータの深さ

デバイス	値	デフォルト
Virtex-4、Spartan-3、Spartan-3E、Spartan-3A、Spartan-3A DSP	512、1024、2048、4096、8192、16384	512
Virtex-5	1024、2048、4096、8192、16384、32768、65536、131072	1024

ILA インターフェイス ポート

ILA コアの I/O 信号には、コアが生成されたときのパラメータにしたがって、ICON までのインターフェイスに使用される制御バス、およびその周りのロジックに接続されるその他の 1 つまたは複数のポートが含まれます。

表 3 : ILA インターフェイス ポート

ポート名	方向	説明
CLK	入力	すべてのトリガーおよびストレージ ロジックにクロックを供給するデザイン クロック必須です。
CONTROL[35:0]	入出力	ICON コアへのバス接続を制御します。必須です。 **メモ** : XPS デザインの場合、このポートの方向は IN (入力) です。
DATA[<*m*>-1:0]	入力	データ ポート。データ ポートの幅 (<*m*> の部分) は、Virtex-5 デバイス ファミリの場合は 1 ～ 4096、それ以外のデバイス ファミリの場合は 1 ～ 256 の範囲になります。オプション (data_same_as_trigger パラメータによって異なる) **メモ** : このポートはベクターとして宣言する必要があります。1 ビットのポートの場合、DATA[0:0] を使用します。
TRIG_OUT	出力	トリガー出力ポート。オプション (enable_trigger_output_port パラメータによって異なる)
TRIG0[<*m*>-1:0]	入力	トリガー ポート番号 0。トリガー ポートの幅 (<m> の部分) は、すべてのデバイス ファミリで 1 ～ 256 の範囲になります。必須です。 **メモ** : このポートはベクタとして宣言する必要があります。1 ビットのポートの場合、TRIG0[0:0] を使用します。
TRIG1[<*m*>-1:0]	入力	トリガー ポート番号 1。トリガー ポートの幅 (<*m*> の部分) は、すべてのデバイス ファミリで 1 ～ 256 の範囲になります。 オプション です (number_of_trigger_ports パラメータによって異なる)。 **メモ** : このポートはベクタとして宣言する必要があります。1 ビットのポートの場合、TRIG1[0:0] を使用します。

表 3 : ILA インターフェイス ポート (続き)

ポート名	方向	説明
TRIG2[<*m*>-1:0]	入力	トリガー ポート番号 2。ポート幅 (<*m*> の部分) は、その他すべてのデバイス ファミリで 1 ～ 256 の範囲になります。オプション (number_of_trigger_ports パラメータによって異なる) **メモ**：このポートはベクタとして宣言する必要があります。1 ビットのポートの場合、TRIG2[0:0] を使用します。
TRIG3[<*m*>-1:0]	入力	トリガー ポート番号 3。トリガー ポートの幅 (<*m*> の部分) は、すべてのデバイス ファミリで 1 ～ 256 の範囲になります。 オプションです (number_of_trigger_ports パラメータによって異なる)。 **メモ**：このポートはベクタとして宣言する必要があります。1 ビットのポートの場合、TRIG3[0:0] を使用します。
TRIG4[<*m*>-1:0]	入力	トリガー ポート番号 4。トリガー ポートの幅 (<*m*> の部分) は、すべてのデバイス ファミリで 1 ～ 256 の範囲になります。オプション (number_of_trigger_ports パラメータによって異なる) **メモ**：このポートはベクタとして宣言する必要があります。1 ビットのポートの場合、TRIG4[0:0] を使用します。
TRIG5[<*m*>-1:0]	入力	トリガー ポート番号 5。トリガー ポートの幅 (<*m*> の部分) は、すべてのデバイス ファミリで 1 ～ 256 の範囲になります。 オプションです (number_of_trigger_ports パラメータによって異なる)。 **メモ**：このポートはベクタとして宣言する必要があります。1 ビットのポートの場合、TRIG5[0:0] を使用します。
TRIG6[<*m*>-1:0]	入力	トリガー ポート番号 6。トリガー ポートの幅 (<*m*> の部分) は、すべてのデバイス ファミリで 1 ～ 256 の範囲になります。 オプションです (number_of_trigger_ports パラメータによって異なる)。 **メモ**：このポートはベクタとして宣言する必要があります。1 ビットのポートの場合、TRIG6[0:0] を使用します。
TRIG7[<*m*>-1:0]	入力	トリガー ポート番号 7。トリガー ポートの幅 (<*m*> の部分) は、すべてのデバイス ファミリで 1 ～ 256 の範囲になります。 オプションです (number_of_trigger_ports パラメータによって異なる)。 **メモ**：このポートはベクタとして宣言する必要があります。1 ビットのポートの場合、TRIG7[0:0] を使用します。
TRIG8[<*m*>-1:0]	入力	トリガー ポート番号 8。トリガー ポートの幅 (<*m*> の部分) は、すべてのデバイス ファミリで 1 ～ 256 の範囲になります。 オプションです (number_of_trigger_ports パラメータによって異なる)。 **メモ**：このポートはベクタとして宣言する必要があります。1 ビットのポートの場合、TRIG8[0:0] を使用します。
TRIG9[<*m*>-1:0]	入力	トリガー ポート番号 9。トリガー ポートの幅 (<*m*> の部分) は、すべてのデバイス ファミリで 1 ～ 256 の範囲になります。 オプションです (number_of_trigger_ports パラメータによって異なる)。 **メモ**：このポートはベクタとして宣言する必要があります。1 ビットのポートの場合、TRIG9[0:0] を使用します。
TRIG10[<*m*>-1:0]	入力	トリガー ポート番号 10。トリガー ポートの幅 (<*m*> の部分) は、すべてのデバイス ファミリで 1 ～ 256 の範囲になります。 オプションです (number_of_trigger_ports パラメータによって異なる)。 **メモ**：このポートはベクタとして宣言する必要があります。1 ビットのポートの場合、TRIG10[0:0] を使用します。

表 3 : ILA インターフェイス ポート (続き)

ポート名	方向	説明
TRIG11[<*m*>-1:0]	入力	トリガー ポート番号 11。トリガー ポートの幅 (<*m*> の部分) は、すべてのデバイス ファミリで 1 ～ 256 の範囲になります。 オプションです (number_of_trigger_ports パラメータによって異なる)。 **メモ**：このポートはベクタとして宣言する必要があります。1 ビットのポートの場合、TRIG11[0:0] を使用します。
TRIG12[<*m*>-1:0]	入力	トリガー ポート番号 12。トリガー ポートの幅 (<*m*> の部分) は、すべてのデバイス ファミリで 1 ～ 256 の範囲になります。 オプションです (number_of_trigger_ports パラメータによって異なる)。 **メモ**：このポートはベクタとして宣言する必要があります。1 ビットのポートの場合、TRIG12[0:0] を使用します。
TRIG13[<*m*>-1:0]	入力	トリガー ポート番号 13。トリガー ポートの幅 (<*m*> の部分) は、すべてのデバイス ファミリで 1 ～ 256 の範囲になります。 オプションです (number_of_trigger_ports パラメータによって異なる)。 **メモ**：このポートはベクタとして宣言する必要があります。1 ビットのポートの場合、TRIG13[0:0] を使用します。
TRIG14[<*m*>-1:0]	入力	トリガー ポート番号 14。トリガー ポートの幅 (<*m*> の部分) は、すべてのデバイス ファミリで 1 ～ 256 の範囲になります。 オプションです (number_of_trigger_ports パラメータによって異なる)。 **メモ**：このポートはベクタとして宣言する必要があります。1 ビットのポートの場合、TRIG14[0:0] を使用します。
TRIG15[<*m*>-1:0]	入力	トリガー ポート番号 15。トリガー ポートの幅 (<*m*> の部分) は、すべてのデバイス ファミリで 1 ～ 256 の範囲になります。 オプションです (number_of_trigger_ports パラメータによって異なる)。 **メモ**：このポートはベクタとして宣言する必要があります。1 ビットのポートの場合、TRIG15[0:0] を使用します。

ILA の XCO パラメータ

表 4 : ILA の XCO パラメータ

パラメータ名	使用可能な値	デフォルト値	説明
component_name	A-z、0-9、および _ (アンダースコア) を含む文字列	ila	インスタンシエートされたコンポーネントの名前
counter_width_<*n*>	Disabled または 1-32	Disabled	トリガー ポート <n> に接続された各マッチ ユニットに関連するマッチ ユニット カウンタの幅。値が Disabled の場合は、そのトリガー ポートでマッチ カウンタが使用されないことを示しています。
data_port_width	Virtex-5 の場合 1 ～ 4096、それ以外の場合 1 ～ 256	32	オプションのデータ ポートのサイズ
data_same_as_trigger	true = データ キャプチャ バスとして 1 つまたは複数のトリガー ポートを使用 false = データ キャプチャ バスとしてオプションのデータ ポートを使用	true	トリガー ポートをデータとしてキャプチャするか、オプションのデータ ポートを使用するか指定します。

表 4 : ILA の XCO パラメータ (続き)

パラメータ名	使用可能な値	デフォルト値	説明
enable_storage_qualification	true = ストレージ クオリフィケーション条件をイネーブル false = ストレージ クオリフィケーション条件をディスエーブル	true	オプションのストレージ修飾子をイネーブルにします。
enable_trigger_output_port	true = トリガー出力ポートをイネーブル false = トリガー出力ポートをディスエーブル	false	オプションのトリガー出力ポートを使用
exclude_from_data_storage<*n*>	true = データ キャプチャからトリガー ポート <*n*> を除外 false = データ キャプチャにトリガー ポート <*n*> を含有	false	true の場合、データ ストレージからトリガー ポート <n> を除外します。data_same_as_trigger が true の場合にのみ適用できます。
match_type<*n*>	basic、basic_with_edges、extended、extended_with_edges、range、range_with_edges	basic	トリガー ポート <*n*>に接続されたマッチ ユニットすべてを使用するためのマッチ ユニット タイプ
match_units_<*n*>	1-16	1	トリガー ポート <*n*> のマッチ ユニット数。すべてのトリガー ポートで使用されるマッチ ユニットの総数は 16 以下にする必要があります。
max_sequence_levels	1-16	1	トリガー シーケンサーのレベルまたはステートの数。1 の場合、トリガー シーケンサーは使用されません。
number_of_trigger_ports	1-16	1	トリガー ポート数
sample_data_depth	5 ページの表 2 を参照してください。	表 2 を参照してください。	データ バッファの深さ
sample_on	rising = clk の立ち上がりエッジでサンプリング falling = clk の立ち下がりエッジでサンプリング	rising	キャプチャおよびトリガーする clk ポートのエッジ
trigger_port_width_<*n*>	1-256	8	トリガー ポート <*n*> のサイズ
use_rpms	true = PRM を使用 false= RPM の使用なし	true	相対配置マクロ制約を使用してロジック配置に制約を付けます。

制限

1 つのデザインには、最大で 15 個までコアが使用できます。

エンベデッド開発キット (EDK) での ILA コアの使用

ILA コアは、EDK を使用してエンベデッド プロセッサ デザインに挿入できます。この場合、次の図のように、ILA コアは既にデ

ザインに含まれる ICON および OPB_MDM コンポーネント インスタンスに依存します。

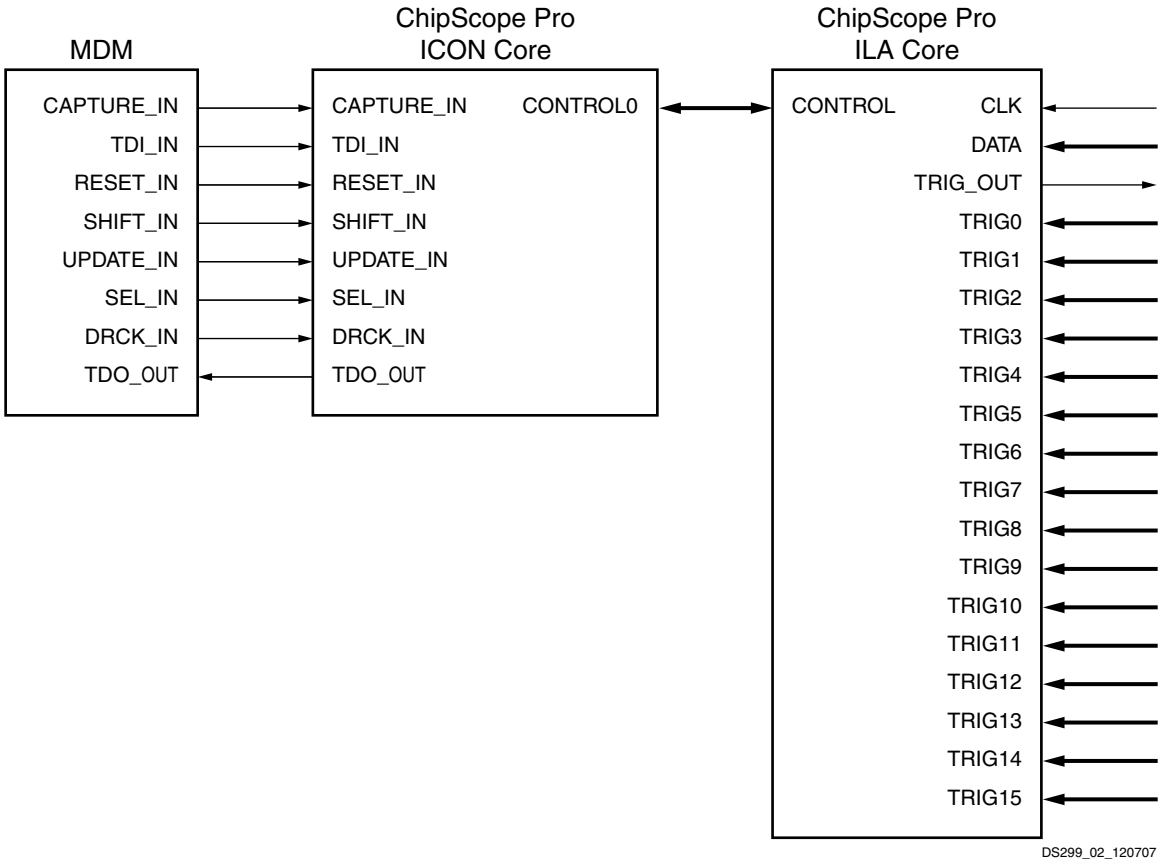

図 2 : EDK デザインの ILA コア コンポーネント

EDK では、ILA コアが Tcl スクリプトを使用してツールに統合されています。EDK の PlatGen ツールが実行されると、Tcl スクリプトが呼び出され、このスクリプトがコマンド ライン モードで CORE Generator を呼び出します。Tcl スクリプトからは CORE Generator にパラメータ ファイル (.xco) が渡され、ILA コアのネットリストが生成されます。また、Tcl スクリプトは次の表のコア パラメータに基づいて ILA ポートと一致するように、HDL ラッパーを生成します。

表 5 : ILA の EDK 用パラメータ

パラメータ名	使用可能な値	デフォルト値	説明
c_data_in_width	1-256	32	使用される場合はデータ ポート幅
c_data_same_as_trigger	0、1	1	1 = トリガと同じデータ 0 = トリガと異なるデータ
c_disable_rpm	0、1	0	0 = RPM をイネーブル 1 = RPM をディスエーブル
c_enable_trigger_out	0、1	0	0 = トリガー出力をディスエーブル 1 = トリガー出力をイネーブル
c_family	virtex4、virtex5、spartan3、spartan3E、spartan3A、spartan3Adsp	virtex5	使用するデバイス ファミリ

表 5 : ILA の EDK 用パラメータ

パラメータ名	使用可能な値	デフォルト値	説明
c_max_sequencer_levels	1-16	16	シーケンサー レベルの数
c_num_data_samples	5 ページの表 2 を参照	表 2 を参照	データ バッファの深さ
c_rising_clock_edge	0、1	1	0 = 立ち下がりエッジ 1 = 立ち上がりエッジ
c_trig<*n*>_counter_width	0-32	0	0 = マッチ カウンタをディスエーブル 1 ～ 32 = トリガー ポート <*n*> に接続されたマッチ ユニットのマッチ カウンターの幅
c_trig<*n*>_match_type	basic、basic_with_edges、extended、extended_with_edges、range、range_with_edges	basic	トリガー ポート <*n*>に接続されたマッチ ユニットすべてのマッチ ユニット タイプ
c_trig<*n*>_trigger_in_width	1-256	8	トリガー ポート <*n*> の幅
c_trig<*n*>_units	0-16	0	0 = ユニットをディスエーブル 1-16 = トリガー ポート <*n*> のマッチ ユニット数

ILA コア用に生成された HDL ラッパー ファイルを合成するには、XST 合成ツールが使用されます。XST および CORE Generator からの NGC ネットリスト出力は、ISE に読み込まれ、実際のデバイスのインプリメンテーションに使用されます。

検証

ILA コアは、ザイリンクス社内で開発されたバス ファンクション モデルを使用し、IP テスト環境で検証されています。

参考資料

- ChipScope Pro ソフトウェアとコアの詳細は、http://japan.xilinx.com/documentation から『ChipScope Pro Software and Cores User Guide』を参照してください。
- EDK での ChipScope Pro を使用したハードウェア検証については、http://japan.xilinx.com/documentationから Platform Studio のオンライン ヘルプを参照してください。
- System Generator for DSP での ChipScope Pro を使用したハードウェア検証については、http://japan.xilinx.com/documentationから『System Generator for DSP ユーザー ガイド』を参照してください。

サポート

ザイリンクスでは、製品マニュアルに記述されているように、この LogiCORE 製品のテクニカル サポートを提供しています。マニュアルで定義されていないデバイスにインプリメントしたり、製品マニュアルで記述されている範囲を超えてカスタマイズしたり、「DO NOT MODIFY」と記述されているセクションに変更を加えたりした場合、タイミング、機能、製品サポートは保証されません。

注文情報

ILA コアは、ザイリンクス エンド ユーザー ライセンス契約書に基づいて提供されており、ザイリンクスの CORE Generator 12.1 またはそれ以降のバージョンを使用して生成できます。CORE

Generator は、ザイリンクスの ISE Design Suite 開発ソフトウェアに含まれています。

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	変更内容
2008 年 3 月 24 日	1.0	リリース 10.1 用 (初期リリース)
2008 年 4 月 25 日	1.1	リリース 10.1 サービス パック 1
2008 年 9 月 19 日	1.2	リリース 10.1 サービス パック 3
2009 年 4 月 7 日	2.0	リリース 11.1
2010 年 4 月 19 日	3.0	リリース 12.1

免責事項